Disciplina de Programação em Computadores

Exercícios resolvidos em Java

Cristian Cechinel

Última atualização em Setembro de 2019

Sumário

Capítulo 1

Introdução

1.1 Olá Mundo

Programa em Java que imprime "Olá mundo"na tela:

Classe OlaMundo (código no github):

```
public class OlaMundo{
  public static void main(String[] args) {
    System.out.println("Ola mundo");
  }
}
```

1.2 Soma de números

Programa que recebe quatro números inteiros, calcula e mostra a soma desses números.

Classe SomaNumeros (código no github)::

```
import java.util.Scanner;

public class SomaNumeros {
    public static void main(String[] args) {
        int n1, n2, n3, n4, soma;
        Scanner entrada = new Scanner(System.in);
        System.out.println("Digite 4 numeros inteiros:");
        n1 = entrada.nextInt();
        n2 = entrada.nextInt();
        n3 = entrada.nextInt();
        n4 = entrada.nextInt();
        entrada.close();
```

```
        soma = n1 + n2 + n3 + n4;
        System.out.println("Soma = " + soma);
    }
}
```

1.3 Cálculo de notas

Faça um programa em Java que receba três notas, calcule e mostre a média aritmética entre elas. Ainda, informe se esse aluno está aprovado, em recuperação ou reprovado.

Classe CalculoMedia (código no github):

```
import java.util.Scanner;

public class CalculoMedia {
    public static void main(String[] args) {

        float n1, n2, n3, media;
        Scanner entrada = new Scanner(System.in);

        System.out.println("Digite 3 notas:");
        n1 = entrada.nextInt();
        n2 = entrada.nextInt();
        n3 = entrada.nextInt();
        entrada.close();

        media = (n1 + n2 + n3) / 3;

        System.out.println("Media = " + media);

        if (media >= 6)
            System.out.println("Aluno aprovado!");
        else if (media < 6 && media >= 3)
            System.out.println("Aluno em recuperacao!");
        else
            System.out.println("Aluno reprovado!");

  }
}
```

1.4 Calculadora

Faça um programa que simule um calculadora simples. O usuário deve informar dois números e a operação desejada e o sistema deve apresentar o resultado na

tela.

Classe Calculadora (código no github):

```
import java.util.Scanner;

public class Calculadora {

    public static void main(String[] args) {

        int n1, n2, opcao;
        Scanner entrada = new Scanner(System.in);

        System.out.println("Digite 2 numeros inteiros:");
        n1 = entrada.nextInt();
        n2 = entrada.nextInt();

        System.out.println("Digite a operacao desejada: \n1 -
        ↪  soma \n2 - subtracao \n3 - divisao \n4 -
        ↪  multiplicacao");
        opcao = entrada.nextInt();
        entrada.close();

        switch (opcao) {
            case 1:
              System.out.println("Soma: " + (n1 + n2));
              break;
            case 2:
              System.out.println("Subtracao: " + (n1 - n2));
              break;
            case 3:
              System.out.println("Divisao: " + (n1 / n2));
              break;
            case 4:
              System.out.println("Multiplicacao: " + (n1 * n2));
              break;
            default:
              System.out.println("Operacao invalida!");
              break;
        }
    }
}
```

1.5 N números primos

Faça um programa em Java que dado um número N, imprima os N primeiros números primos.

Classe NumerosPrimos (código no github):

```
import java.util.Scanner;

public class NumerosPrimos {
    public static void main(String[] args) {

        int n;
        boolean primo;
        Scanner entrada = new Scanner(System.in);

        System.out.println("Digite um numero inteiro:");
        n = entrada.nextInt();
        entrada.close();

        for (int i = 2; i <= n; i++) {
            primo = true;
            for (int j = 2; j <= (i / 2); j++) {
                if (i % j == 0) {
                    primo = false;
                    break;
                }
            }
            if (primo)
                System.out.println("Numero primo: " + i);
        }
    }
}
```

1.6 Funções matemáticas

Desenvolva funções e/ou procedimentos para:
- Imprimir a Fibonacci.
- Retornar o quadrado de um número.
- Calcular o fatorial de um número.
- Converter uma temperatura de graus Fahrenheitpara Celsius.
- Converter uma temperatura de graus Celsius para Fahrenheit.

Classe FuncoesMatematicas (código no github):

```
import java.util.Scanner;
```

```java
public class FuncoesMatematicas {

    public static void printFibonacci(int limite) {
        int anterior = 0;
        int atual = 1;
        System.out.print(anterior + " " + atual);

        for (int i = 0; i < limite - 2; i++) {
            int sucessor = anterior + atual;
            System.out.print(" " + sucessor);
            anterior = atual;
            atual = sucessor;
        }

        System.out.println();
    }

    public static float calculaQuadrado(float n) {
        return n * n;
    }

    public static int fatorial(int n) {
        int fat = 1;
        for (int i = 1; i <= n; i++) {
            fat *= i;
        }
        return fat;
    }

    public static double converteTempParaCelsius(int n){
        return ((n - 32) / 1.8);
    }

    public static double converteTempParaFahrenheit(int n) {
        return ((1.8 * n) + 32);
    }

    public static void main(String[] args) {

        Scanner entrada = new Scanner(System.in);

        // testa metodo printFibonacci()
        System.out.println("Sequencia de Fibonacci de 10
        ↪  numeros");
        printFibonacci(10);
```

```
        // testa metodo calculaQuadrado()
        System.out.println("\nDigite um numero para calculo do
        ↪  quadrado:");
        float x1 = entrada.nextFloat();
        float result1 = calculaQuadrado(x1);
        System.out.println("Quadrado: " + result1);

        // testa metodo fatorial()
        System.out.println("\nDigite um numero inteiro positivo
        ↪  para calculo de fatorial:");
        int x = entrada.nextInt();
        entrada.close();
        int result = fatorial(x);
        System.out.println("Fatorial: " + result);

        // testa metodo converteTempParaCelsius()
        double result2 = converteTempParaCelsius(50);
        System.out.println("\n50 graus Fahrenheit equivalente a "
        ↪  + result2 + " graus Celsius");

        // testa metodo converteTempParaCelsius()
        double result3 = converteTempParaFahrenheit(32);
        System.out.println("32 graus Celsius equivalente a " +
        ↪  result3 + " graus Fahrenheit");

    }
}
```

1.7 Função equals

Crie 2 Strings com o mesmo valor e compare a igualdade dos 2 valores através do operador == e da função equals(). Exemplo da função equals: var1.equals(var2), se as 2 variáveis possuírem o mesmo valor, retornará true, senão false.

Classe TestaEquals (código no github):

```
public class TestaEquals {

    public static void main(String[] args){
        String s1 = new String("Ola");
        String s2 = new String("Ola");
        System.out.println(s1 == s2);
        System.out.println(s1.equals(s2));
    }
}
```

Capítulo 2

Vetores

2.1 Vetor de tabuadas

Desenvolva uma classe em Java com um vetor de 10 posições de inteiros e que armazene em cada posição a tabuada de um número informado pelo usuário.

Classe Tabuada (código no github):

```
import java.util.Scanner;

public class Tabuada {

    public static void main(String[] args) {
        int[] tab = new int[11];

        Scanner entrada = new Scanner(System.in);
        System.out.println("Digite um numero para tabuada:");
        int n = entrada.nextInt();
        entrada.close();

        for (int i = 0; i <= 10; i++) {
            tab[i] = i * n;
        }

        for (int i = 0; i < tab.length; i++) {
            System.out.println(tab[i]);
        }
    }
}
```

2.2 Menor valor de reais

Desenvolva uma classe em Java que armazene 10 valores reais informados pelo usuário em um vetor e que informe o menor valor. Modularize a localização do menor por meio de uma função: "public static double menor (double vetor[])".

Classe MenorValorReal (código no github):

```
import java.util.Scanner;

public class MenorValorReal {

    public static double menor(double vetor[]) {
        if(vetor.length < 1)
            return (Double) null;
        double menorValor = vetor[0];
        for (int i = 1; i < vetor.length; i++) {
            if (vetor[i] < menorValor){
                menorValor = vetor[i];
            }
        }
        return menorValor;
    }

    public static void main(String[] args) {

        double[] vetorReais = new double[10];
        Scanner entrada = new Scanner(System.in);

        System.out.println("Digite 10 valores reais:");
        for (int i = 0; i < vetorReais.length; i++)
            vetorReais[i] = entrada.nextDouble();
        entrada.close();

        double menor = menor(vetorReais);
        System.out.println("Menor = " + menor);

  }

}
```

2.3 Vetor de notas

Implemente um algoritmo que leia as notas de 10 alunos armazenando-as em um vetor (matriz) de 10 posições. Ao final escreva na tela somente as notas

maiores que 5.0.

Classe VetorNotas (código no github):

```
import java.util.Scanner;

public class VetorNotas {

    public static void main(String[] args) {

        Scanner entrada = new Scanner(System.in);
        float[] notas = new float[10];

        System.out.println("Digite 10 notas: ");
        for (int x = 0; x < notas.length; x++)
            notas[x] = entrada.nextFloat();

        System.out.println("\nNotas acima de 5.0: ");
        for (int x = 0; x < notas.length; x++)
            if (notas[x] > 5)
                System.out.println(notas[x]);

    }
}
```

2.4 Vetores de nomes e notas

Implemente um algoritmo que leia as notas e os nomes de 5 alunos armazenando os dados em vetores (matriz) de 5 posições, sendo que as notas serão armazenadas em um vetor de reais e os nomes em um outro vetor de String. Ao final o algoritmo deve escrever na tela somente os nomes dos alunos que tiraram nota maior que 5.0.

Classe VetorNomesNotas (código no github):

```
import java.util.Scanner;

public class VetorNomesNotas {

    public static void main(String[] args) {

        Scanner teclado = new Scanner(System.in);
        float[] notas = new float[5];
        String[] nomes = new String[5];

        for (int x = 0; x < 5; x++) {
```

```
            System.out.println("Digite o nome do aluno " + x +
            ↪  ":");
            nomes[x] = teclado.nextLine();

            System.out.println("Digite a nota do aluno " + x +
            ↪  ":");
            notas[x] = teclado.nextFloat();

            teclado.nextLine(); // Consome o resto da linha
            ↪  deixado pelo nextFloat()
        }

        System.out.println("\nNome dos alunos que tiraram nota
        ↪  maior que 5.0: \n");
        for (int x = 0; x < notas.length; x++)
            if (notas[x] > 5)
                System.out.println(nomes[x]);

    }

}
```

2.5 Multiplicação de valores do vetor

Desenvolva um algoritmo que leia um conjunto de 15 números inteiros e armazene-os em um vetor A. Após a leitura dos dados o algoritmo deve multiplicar todos os números do vetor A por 3 e armazenar o resultado em um segundo vetor B.

Classe Multiplicacao (código no github):

```
import java.util.Scanner;

public class Multiplicacao {

    public static void main(String[] args) {

        Scanner teclado = new Scanner(System.in);
        int[] vetorA = new int[15];
        int[] vetorB = new int[15];

        System.out.println("Digite 15 numeros inteiros: ");
        for (int x = 0; x < vetorA.length; x++)
            vetorA[x] = teclado.nextInt();

        System.out.println("\nVetor B: ");
        for (int x = 0; x < vetorB.length; x++) {
```

```
            vetorB[x] = vetorA[x] * 3;
            System.out.println(vetorB[x]);

        }
    }
}
```

2.6 Vetor de sexo de pessoas

Construa um algoritmo que solicite ao usuário o sexo de várias pessoas armazenando os dados em um vetor. Ao final o algoritmo deve imprimir quantas pessoas são do sexo masculino e quantas são do sexo feminino. O algoritmo deve parar de solicitar o sexo quando o número de pessoas chegar a 30.

Classe SexosPessoas (código no github):

```
import java.util.Scanner;

public class SexosPessoas {

    public static void main(String[] args) {

        Scanner teclado = new Scanner(System.in);
        String[] sexos = new String[30];
        int contM = 0;
        int contF = 0;

        System.out.println("Digite o sexo da pessoa: \n- F para
        ↪ feminino \n- M para masculino");

        for (int x = 0; x < sexos.length; x++) {
            sexos[x] = teclado.next();

            if (sexos[x].equalsIgnoreCase("f"))
                contF++;
            else if (sexos[x].equalsIgnoreCase("m"))
                contM++;
        }

        System.out.println("Quantidade de pessoas do sexo
        ↪ feminino: " + contF);
        System.out.println("Quantidade de pessoas do sexo
        ↪ masculino: " + contM);

    }
}
```

2.7 Vetor de idades

Algoritmo para fazer a leitura da idade de dez pessoas e armazená-las em um vetor.

Classe VetorIdades (código no github):

```
import java.util.Scanner;

public class VetorIdades {

    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        int[] idades = new int[10];

        System.out.println("Digite 10 idades: ");
        for (int x = 0; x < idades.length; x++)
            idades[x] = teclado.nextInt();

    }
}
```

2.8 Índice de menor idade do vetor

Algoritmo para localizar a menor idade entre as 10 idades que estão armazenadas em um vetor de idades.

Classe MenorIdade (código no github):

```
import java.util.Scanner;

public class MenorIdade {

    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        int[] idades = new int[10];
        int idadeMenor;

        System.out.println("Digite 10 idades: ");
        for (int x = 0; x < idades.length; x++)
            idades[x] = teclado.nextInt();

        idadeMenor = idades[0];
        for (int x = 0; x < idades.length; x++){
            if (idades[x] < idadeMenor)
                idadeMenor = x;
```

```
        }
        System.out.println("Posicao da menor idade digitada: " +
        ↪  idadeMenor);

    }
}
```

2.9 Vetores de nomes e idades

Algoritmo que leia o nome e a idade de dez pessoas armazenando os dados em vetores.

Classe NomesIdades (código no github):

```
import java.util.Scanner;

public class NomesIdades {

    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        int[] idades = new int[10];
        String[] nomes = new String[10];

        for (int x = 0; x < 10; x++) {
            System.out.println("Digite o nome da pessoa " + x +
            ↪  ":");
            nomes[x] = teclado.nextLine();
            System.out.println("Digite a idade do aluno " + x +
            ↪  ":");
            idades[x] = teclado.nextInt();
            teclado.nextLine(); // Consome o resto da linha
            ↪  deixado pelo nextInt
        }
    }
}
```

2.10 Índice do nome da pessoa com menor idade

Algoritmo que leia o nome e a idade de dez pessoas armazenando os dados em vetores e que localize o nome da pessoa de menor idade.

Classe MenorIdade (código no github):

```
import java.util.Scanner;
```

```
public class MenorIdade {

  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    int[] idades = new int[10];
    String[] nomes = new String[10];
    int menor, index = 0;

    for (int x = 0; x < idades.length; x++) {
      System.out.println("Digite o nome da pessoa " + x + ":");
      nomes[x] = teclado.nextLine();
      System.out.println("Digite a idade da pessoa " + x + ":");
      idades[x] = teclado.nextInt();
      teclado.nextLine(); // Consome o resto da linha deixado
       ↪  pelo nextInt
    }

    menor = idades[0];
    for (int x = 0; x < idades.length; x++) {
      if (menor > idades[x]) {
        menor = idades[x];
        index = x;
      }
    }

    System.out.println("Nome da Pessoa com menor idade: " +
     ↪  nomes[index]);
  }

}
```

2.11 Vetores de nomes, idades e sexos

2.11.1 Parte 1 - leitura e escrita

Algoritmo que leia o nome, a idade, e o sexo de dez pessoas armazenando os dados em vetores.

Classe VetoresNomesIdadesSexosLeituraEscrita (código no github):

```
import java.util.Scanner;

public class VetoresNomesIdadesSexosLeituraEscrita {

  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
```

```
    int[] idades = new int[10];
    String[] nomes = new String[10], sexos = new String[10];

    for (int x = 0; x < idades.length; x++) {
      System.out.println("Digite o nome da pessoa " + x + ":");
      nomes[x] = teclado.nextLine();
      System.out.println("Digite a idade da pessoa " + x + ":");
      idades[x] = teclado.nextInt();
      System.out.println("Digite o sexo da pessoa " + x + ":");
      sexos[x] = teclado.next();
      teclado.nextLine(); // Consome o resto da linha deixado
      ↪ pelo nextInt

    }
  }
}
```

2.11.2 Parte 2 - localização de menor valor

Algoritmo que leia o nome, a idade, e o sexo de dez pessoas armazenando os dados em vetores e que localize o nome e o sexo da pessoa de menor idade.

Classe VetoresNomesIdadesSexosContagem (código no github):

```
import java.util.Scanner;

public class VetoresNomesIdadesSexosContagem {

  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    int length = 10;
    int[] idades = new int[length];
    String[] nomes = new String[length], sexos = new
    ↪ String[length];
    int menor, index = 0;

    for (int x = 0; x < idades.length; x++) {
      System.out.println("Digite o nome da pessoa " + x + ":");
      nomes[x] = teclado.nextLine();
      System.out.println("Digite a idade da pessoa " + x + ":");
      idades[x] = teclado.nextInt();
      System.out.println("Digite o sexo da pessoa " + x + ":");
      sexos[x] = teclado.next();
      teclado.nextLine(); // Consome o resto da linha deixado
      ↪ pelo nextInt
```

```
    }

    menor = idades[0];
    for (int x = 0; x < idades.length; x++) {
      if (menor > idades[x]) {
        menor = idades[x];
        index = x;
      }
    }

    System.out.println("Pessoa com a menor idade: \nNome: " +
    ↪  nomes[index] + "\nSexo: " + sexos[index]);

  }

}
```

2.11.3 Parte 3 - contagem de valores

Algoritmo que leia o nome, a idade, e o sexo de dez pessoas armazenando os dados em vetores e que faca a contagem do numero de pessoas do sexo masculino e do sexo feminino que possuem idade maior que 18.

Classe VetoresNomesIdadesSexosContagem (código no github):

```
import java.util.Scanner;

public class VetoresNomesIdadesSexosContagem {

  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    int length = 10;
    int[] idades = new int[length];
    String[] nomes = new String[length], sexos = new
    ↪  String[length];
    int sexoM = 0, sexoF = 0;

    for (int x = 0; x < idades.length; x++) {
      System.out.println("Digite o nome da pessoa " + x + ":");
      nomes[x] = teclado.nextLine();
      System.out.println("Digite a idade da pessoa " + x + ":");
      idades[x] = teclado.nextInt();
      System.out.println("Digite o sexo da pessoa " + x + ":");
      sexos[x] = teclado.next();
      teclado.nextLine(); // Consome o resto da linha deixado
      ↪  pelo nextInt
```

```
    }

    for (int x = 0; x < idades.length; x++) {
      if (idades[x] > 18 && sexos[x].equalsIgnoreCase("f")) {
        sexoF++;
      }
      else if (idades[x] > 18 && sexos[x].equalsIgnoreCase("m"))
      ↪  {
        sexoM++;
      }
    }

    System.out.println("Quantidade de mulheres com idade maior
    ↪  que 18: " + sexoF);
    System.out.println("Quantidade de homens com idade maior que
    ↪  18: " + sexoM);
  }

}
```

2.11.4 Parte 4 - impressão de valores selecionados

Algoritmo que leia o nome, a idade, e o sexo de dez pessoas armazenando os dados em vetores e que imprima o nome das pessoas do sexo masculino que possuem idade maior que 18.

Classe VetoresNomesIdadesSexosContagem (código no github):

```
import java.util.Scanner;

public class VetoresNomesIdadesSexosContagem {

  public static void main(String[] args) {
    Scanner teclado = new Scanner(System.in);
    int length = 3;
    int[] idades = new int[length];
    String[] nomes = new String[length],
        sexos = new String[length];

    for (int x = 0; x < idades.length; x++) {
      System.out.println("Digite o nome da pessoa " + x + ":");
      nomes[x] = teclado.nextLine();
      System.out.println("Digite a idade da pessoa " + x + ":");
      idades[x] = teclado.nextInt();
      System.out.println("Digite o sexo da pessoa " + x + ":");
```

```
      sexos[x] = teclado.next();
      teclado.nextLine(); // Consome o resto da linha deixado
       ↪  pelo nextInt

    }

    teclado.close();

    System.out.println("Lista de pessoas do sexo masculino com
     ↪  idade maior que 18: ");
    for (int x = 0; x < idades.length; x++) {
      if (idades[x] > 18 && sexos[x].equalsIgnoreCase("m"))
       ↪  //sexo masculino
        System.out.println(nomes[x]);
    }

  }

}
```

2.12 Vetores com notas de 80 alunos

2.12.1 Cálculo de média da turma - Enquanto faça

Elabore um algoritmo que leia a nota de 80 alunos e que imprima ao final a nota de cada aluno e a média da turma (utilize a estrutura de repetição enquanto faca).

Classe MediaTurma (código no github):

```
import java.util.Scanner;

public class MediaTurma {

  public static void main(String[] args) {

    // declaracao de variaveis
    Scanner teclado = new Scanner(System.in);
    int length = 80, index = 0; //mudar length para algo menor ao
     ↪  testar
    float soma = 0, media = 0;
    int[] notas = new int[length];

    // iteracao
    while (index < notas.length) {
      System.out.println("Digite uma nota: ");
```

```
      notas[index] = teclado.nextInt();
      soma += notas[index];
      index++;
    }

    teclado.close();

    //exibe as notas
    System.out.println("Notas dos alunos: ");
    for(int nota: notas)
      System.out.println(nota);

    // calculo da media
    media = soma / notas.length;
    System.out.println("\nMedia das notas da turma: " + media);

  }

}
```

2.12.2 Cálculo de média da turma - Para faça

Elabore um algoritmo que leia a nota de 80 alunos e que imprima ao final a nota de cada aluno e a média da turma (utilize a estrutura de repetição para faca).

Classe MediaTurma (código no github):

```
import java.util.Scanner;

public class MediaTurma {

  public static void main(String[] args) {

    // declaracao de variaveis
    Scanner teclado = new Scanner(System.in);
    int length = 80; //mudar para valor menor ao testar
    float soma = 0, media = 0;
    int[] notas = new int[length];

    // iteracao
    for(int x = 0; x < notas.length; x++) {
      System.out.println("Digite uma nota: ");
      notas[x] = teclado.nextInt();
      soma += notas[x];
    }
```

```
    teclado.close();

    //exibe as notas
    System.out.println("\nNotas dos alunos: ");
    for(int nota: notas)
      System.out.println(nota);

    // calculo da media
    media = soma / notas.length;
    System.out.println("\nMedia das notas da turma: " + media);

  }

}
```

2.13 Nomes e duas notas

Elabore um algoritmo que armazene o nome e duas notas de 5 alunos e imprima uma listagem contendo nome, as duas notas e a média de cada aluno.

Classe NomesDuasNotas (código no github):

```
import java.util.Scanner;

public class NomesDuasNotas {

    public static void main(String[] args) {

        Scanner entrada = new Scanner(System.in);
        String[] nomes = new String[10];
        float[] notas = new float[20];

    for (int x = 0; x < 10; x++) {
            System.out.println("Digite o nome do aluno:");
            nomes[x] = entrada.nextLine();

            System.out.println("Digite a nota 1 do aluno: ");
            notas[2*x] = entrada.nextFloat();

            System.out.println("Digite a nota 2 do aluno: ");
            notas[2*x + 1] = entrada.nextFloat();

            entrada.nextLine();
        }
```

```
        for (int x = 0; x < 10; x++) {
            float media = (notas[2*x] + notas[2*x + 1]) / 2;
            System.out.println("\nNome do aluno: " + nomes[x] +
            ↪  "\nNota 1: " + notas[2*x] + " - Nota 2: " +
            ↪  notas[2*x + 1] + "\nMédia: " + media);
        }
    }
}
```

2.14 Vetores de inteiros e soma

Elabore um algoritmo que armazene números em dois vetores inteiros de cinco elementos cada, gere e imprima o vetor soma.

Classe VetoresInteirosSomas (código no github):

```
import java.util.Scanner;

public class VetoresInteirosSomas {

    public static void main(String[] args) {

        //definicao de variaveis
        Scanner teclado = new Scanner(System.in);
        int length = 5;
        int[] v1 = new int[length], v2 = new int[length], soma =
        ↪  new int[length];

        //leitura do vetor 1
        System.out.println("Digite os valores do vetor 1: ");
        for (int x = 0; x < v1.length; x++) {
            v1[x] = teclado.nextInt();
        }

        //leitura do vetor 2
        System.out.println("\nDigite os valores do vetor 2: ");
        for (int x = 0; x < v2.length; x++) {
            v2[x] = teclado.nextInt();
        }

        teclado.close();

        //soma dos vetores 1 e 2 e impressao
        System.out.println("\nVetor soma: ");
        for (int x = 0; x < v1.length; x++) {
            soma[x] = v1[x] + v2[x];
```

```
            System.out.println(soma[x]);

        }
    }
}
```

2.15 Vetores associados - representando mercadorias de um armazém

Um armazém trabalha com 100 mercadorias diferentes identificadas pelos números inteiros de 1 a 100. O dono do armazém anota a quantidade de cada mercadoria vendida durante o mês. Ele tem uma tabela que indica para cada mercadoria o preço de venda. Elabore um algoritmo para calcular o faturamento mensal do armazém, ou seja: faturamento = soma(quantidade[i] * preço[i]);

Classe Mercadorias (código no github):

```
import java.util.Scanner;

public class Mercadorias {

    public static void main(String[] args) {

        //deficicao e instanciacao de variaveis
        Scanner teclado = new Scanner(System.in);
        int length = 100; //mudar para um valor menor ao testar
        int[] qtdeVendaMercadoria = new int[length];
        float[] precoVendaMercadoria = new float[length];

        float faturamento = 0f;

        //inseri a quantidade de produtos vendidos e preco de
        ↪  venda
        for (int x = 0; x < length; x++) {
            System.out.println("Quantidade vendida do produto " +
            ↪  x + ":");
            qtdeVendaMercadoria[x] = teclado.nextInt();
            System.out.println("Preco de venda do produto " + x +
            ↪  ":");
            precoVendaMercadoria[x] = teclado.nextFloat();

        }

        //calcula o faturamento
        for (int x = 0; x < length; x++) {
```

```
            faturamento += (precoVendaMercadoria[x] *
            ↪  qtdeVendaMercadoria[x]);
        }

        System.out.println("\nFaturamento: " + faturamento);
    }
}
```

2.16 Localização de maior e menor valor em vetores

Elabore um algoritmo que leia um conjunto X com 10 números e calcule a diferença entre o maior e o menor elemento existente.

Classe LocalizaValor (código no github):

```
import java.util.Scanner;

public class LocalizaValor {

    public static void main(String[] args) {

        //deficicao e instanciacao de variaveis
        Scanner teclado = new Scanner(System.in);
        int length = 10;
        int[] vetor = new int[length];
        int maior, menor, diferenca;

        //Le conjunto de inteiros
        System.out.println("Digite 10 numeros");
        for (int x = 0; x < vetor.length; x++) {
            vetor[x] = teclado.nextInt();
        }

        //calcula menor e maior numero do array
        menor = vetor[0];
        maior = vetor[0];
        for (int x = 1; x < vetor.length; x++) {
            if(menor > vetor[x])
                menor = vetor[x];
            if(maior < vetor[x])
                maior = vetor[x];
        }

        //calcula a diferenca entre o maior e menor
```

```
        diferenca = maior - menor;
        System.out.println("Maior: " + maior + "\nMenor: " +
        ↪  menor + "\nDiferenca entre o maior valor e o menor: "
        ↪  + diferenca);
    }
}
```

2.17 Localização das posições do maior e menor valor em vetores

Elabore um algoritmo que leia um conjunto X com 10 números e calcule a diferença entre as posições que maior e o menor elemento existentes ocupam.

Classe LocalizaPosicao (código no github):

```
import java.util.Scanner;

public class LocalizaPosicao {

    public static void main(String[] args) {

        //deficicao e instanciacao de variaveis
        Scanner teclado = new Scanner(System.in);
        int length = 10;
        int[] vetor = new int[length];
        int maior, menor, diferenca, indexMenor = 0, indexMaior =
        ↪  0;

        //Le conjunto de inteiros
        System.out.println("Digite 10 numeros");
        for (int x = 0; x < vetor.length; x++) {
            vetor[x] = teclado.nextInt();
        }

        //encontra o menor, maior e os indices desses valores no
        ↪  array
        menor = vetor[0];
        maior = vetor[0];
        for (int x = 1; x < vetor.length; x++) {
            if (menor > vetor[x]) {
                menor = vetor[x];
                indexMenor = x;
            }
            if (maior < vetor[x]) {
                maior = vetor[x];
```

```
                indexMaior = x;

            }
        }

        //calcula a diferenca entre o index do maior e menor
        diferenca = Math.abs(indexMaior - indexMenor);
        System.out.println("Posicao Maior: " + indexMaior +
        ↪  "\nPosicao Menor: " + indexMenor + "\nDiferenca entre
        ↪  a posicao do maior valor e a posicao do menor: " +
        ↪  diferenca);

    }
}
```

2.18 Localização de um número negativo dentro de vetores

Dada uma coleção de N números, imprimir o índice do primeiro número negativo, se houver.

Classe LocalizaNegativo (código no github):

```
import java.util.Scanner;

public class LocalizaNegativo {

    public static void main(String[] args) {

        //deficicao e instanciacao de variaveis
        Scanner teclado = new Scanner(System.in);
        int length = 10;
        int[] vetor = new int[length];

        //Le conjunto de inteiros
        System.out.println("Digite 10 numeros");
        for (int x = 0; x < vetor.length; x++) {
            vetor[x] = teclado.nextInt();
        }

        //Encontra negativo e mostra o index do menor, se houver
        for (int i = 0; i < vetor.length; i++) {
            if (vetor[i] < 0) {
                System.out.println("\nIndex da menor posicao: " +
                ↪  i + "\nNumero: " + vetor[i]);
```

```
                break;
            }

        }
    }
}
```

2.19 Vetor de nomes e salários

Elabore um algoritmo que leia nome e salário de 20 pessoas. Calcular e armazenar o novo salário sabendo-se que o reajuste foi de 8%. Ao final imprimir uma listagem numerada com nome e novo salário.

Classe NomesSalarios (código no github):

```
import java.util.Scanner;

public class NomesSalarios {

  public static void main(String[] args) {
    // declaracao de variaveis
    Scanner teclado = new Scanner(System.in);
    int length = 5;
    String[] nomes = new String[length];
    double[] salarios = new double[length];

    // iteracao e leitura dos dados
    for (int x = 0; x < length; x++) {
      System.out.println("Digite o nome do funcionario: ");
      nomes[x] = teclado.nextLine();
      System.out.println("Digite o salario do funcionario: ");
      salarios[x] = teclado.nextFloat();
      teclado.nextLine(); // Consome o resto da linha deixado
      ↪  pelo nextInt

    }

    teclado.close();

    // calcula e mostra os novos salarios e nomes
    for (int x = 0; x < salarios.length; x++) {
      salarios[x] = salarios[x] + (salarios[x] * 0.08);
      System.out.println("\nNome do funcionario: " + nomes[x] +
      ↪  "\nNovo Salario: " + salarios[x]);
    }
```

```
  }

}
```

2.20 Vetor de nomes, idades e sexos 2

Elabore um algoritmo que leia o nome, a idade e o sexo de 10 pessoas e imprima (Caso preferir, desenvolva algoritmos separados para cada item):
a) média de idade das pessoas
b) nome da pessoa mais jovem
c) nome da pessoa mais idosa
d) nome e a idade do homem mais jovem
e) nome e idade da mulher mais idosa

Classe VetoresNomesIdadesSexos (código no github):

```
import java.util.Scanner;

public class VetoresNomesIdadesSexos {

  public static void main(String[] args) {
    // instanciacao de variaveis
    Scanner teclado = new Scanner(System.in);
    int length = 10;
    int[] idades = new int[length];
    String[] nomes = new String[length],
        sexos = new String[length];

    // leitura dos dado
    for (int x = 0; x < idades.length; x++) {
      System.out.println("Digite o nome da pessoa " + x + ":");
      nomes[x] = teclado.nextLine();
      System.out.println("Digite a idade da pessoa " + x + ":");
      idades[x] = teclado.nextInt();
      System.out.println("Digite o sexo da pessoa (m/f) " + x +
      ↪  ":");
      sexos[x] = teclado.next();
      teclado.nextLine(); // Consome o resto da linha deixado
      ↪  pelo nextInt

    }

    teclado.close();

    // mÃ©dia de idade das pessoas
    System.out.println(
```

```
        "\nMedia de idades: " + calculaMedia(idades));

    // b) nome da pessoa mais jovem
    System.out.println("\nNome da pessoa mais jovem: " +
    ↪  encontraNomeMaisJovem(idades, nomes));

    // c) nome da pessoa mais idosa
    System.out.println("\nNome da pessoa mais idosa: " +
    ↪  encontraNomeMaisVelho(idades, nomes));

    // d) nome e a idade do homem mais jovem
    int indexMaisJovem = indexHomemMaisJovem(idades, sexos);
    if (indexMaisJovem != -1) {
      System.out.println("\nHomem mais jovem: " +
      ↪  nomes[indexMaisJovem] + "\nIdade: " +
      ↪  idades[indexMaisJovem]);
    } else {
      System.out.println("\nNao ha homens!");
    }

    // e) nome e idade da mulher mais idosa
    int indexMaisIdosa = indexMulherMaisIdosa(idades, sexos);
    if (indexMaisIdosa != -1) {
      System.out.println(
          "\nMulher mais Idosa: " + nomes[indexMaisIdosa] +
          ↪  "\nIdade: " + idades[indexMaisIdosa]);
    } else {
      System.out.println("\nNao ha mulheres!");
    }

  }

  public static double calculaMedia(int[] idades) {
    double soma = 0, media = 0;
    for (int idade : idades)
      soma += idade;
    media = soma / idades.length;
    return media;

  }

  public static String encontraNomeMaisJovem(int[] idades,
  ↪  String[] nomes) {
    int menor = idades[0], indexMaisJovem = 0;
    for (int x = 1; x < idades.length; x++) {
      if (idades[x] < menor) {
```

```
      menor = idades[x];
      indexMaisJovem = x;
    }
  }

  return nomes[indexMaisJovem];
}

public static String encontraNomeMaisVelho(int[] idades,
↪  String[] nomes) {
  int maior = idades[0], indexMaisVelho = 0;
  for (int x = 1; x < idades.length; x++) {
    if (idades[x] > maior) {
      maior = idades[x];
      indexMaisVelho = x;
    }
  }

  return nomes[indexMaisVelho];
}

public static int indexHomemMaisJovem(int[] idades, String[]
↪  sexos) {
  int menor = Integer.MAX_VALUE, indexMaisJovem = -1;
  for (int x = 0; x < idades.length; x++) {
    if (menor > idades[x] && sexos[x].equalsIgnoreCase("m")) {
      menor = idades[x];
      indexMaisJovem = x;
    }
  }

  return indexMaisJovem;
}

public static int indexMulherMaisIdosa(int[] idades, String[]
↪  sexos) {
  int maior = Integer.MIN_VALUE, indexMaisIdosa = -1;
  for (int x = 0; x < idades.length; x++) {
    if (maior < idades[x] && sexos[x].equalsIgnoreCase("f")) {
      maior = idades[x];
      indexMaisIdosa = x;
    }
  }

  return indexMaisIdosa;
}
```

```
}
```

2.21 Inverter vetor

Elabore uma função que inverta os elementos de um vetor

Classe InverterVetor (código no github):

```
public class InverterVetor {

  public static void main(String[] args) {
    // inicializacao de variaveis e chamada do metodo
    ↪  inverteVetor
    int[] vetorTeste = { 1, 2, 3, 4, 5 };
    int[] vetorInvertido = inverteVetor(vetorTeste);

    // teste do novo vetor
    System.out.println("Vetor invertido:");
    for (int item : vetorInvertido)
      System.out.println(item);

  }

  // funcao para inverter vetor de inteiros
  public static int[] inverteVetor(int[] vetor) {
    int[] vetorInvertido = new int[vetor.length];
    int aux = 0;

    for (int x = (vetor.length - 1); x >= 0; x--) {
      vetorInvertido[aux] = vetor[x];
      aux++;
    }

    return vetorInvertido;

  }

}
```

2.22 Imprimir Pirâmide

Imprimir na tela
*
**

```
***
****
*****
******
*******
********
*********
**********
```

Classe Piramide (código no github):

```
public class Piramide {

  public static void main(String[] args) {
    // chamada da funcao no metodo principal
    imprimePiramide(10);

  }

  // funcao para imprimir piramide
  public static void imprimePiramide(int tamanho) {
    String[] piramide = new String[tamanho];
    String asterisco = "*";

    for (int x = 0; x < piramide.length; x++) {
      piramide[x] = asterisco;
      System.out.println(piramide[x]);
      asterisco = asterisco.concat("*");
    }

  }

}
```

Capítulo 3

Orientação a Objetos

3.1 Classe pessoa

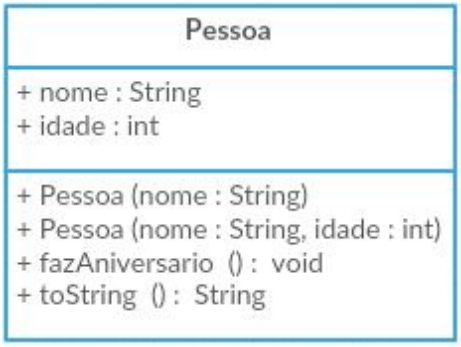

Figura 3.1: Classe Pessoa

Crie uma classe Pessoa com os atributos nome e idade, e com o método fazAniversario(aumentando a idade). Desenvolva também o construtor para essa classe. O mesmo deve receber o nome da pessoa. Desenvolva uma classe TestaPessoa que cria algumas pessoas, utiliza o método fazAniversario e imprime os dados da pessoa na tela.

Classe Pessoa (código no github):

```
public class Pessoa {
    public String nome;
    public int idade;

    Pessoa(String nome) {
        this.nome = nome;
```

```
    }

    Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

    public void fazAniversario() {
        idade++;
    }

    @Override
    public String toString() {
        return this.getClass().getName() + ": nome = " + nome +",
        ↪  idade = " + idade;
    }
}
```

Classe TestaPessoa (código no github):

```
public class TestaPessoa {

    public static void main(String[] args) {
        Pessoa p1 = new Pessoa("Arthur", 20);
        Pessoa p2 = new Pessoa("Natalia", 19);

        p1.fazAniversario();
        p2.fazAniversario();

        System.out.println(p1.toString());
        System.out.println(p2.toString());
    }
}
```

3.2 Classe Quadrado

Desenvolva uma classe em Java para representar um Quadrado, contendo como atributo o seu lado e o método calcularArea(). Desenvolva um construtor para essa classe que deve receber o lado do quadrado. Desenvolva uma classe TestaQuadrado que deve criar alguns quadrados e imprimir a área dos mesmos na tela.

Classe Quadrado (código no github):

```
public class Quadrado {
    float lado;
```

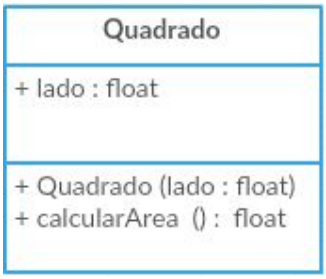

Figura 3.2: Classe Quadrado

```
    Quadrado(float lado) {
        this.lado = lado;
    }

    public float calcularArea() {
        return lado * lado;
    }
}
```

Classe TestaQuadrado (código no github):

```
public class TestaQuadrado {
    public static void main(String[] args) {
        Quadrado q1 = new Quadrado(2);
        Quadrado q2 = new Quadrado(4);

        System.out.println("Area de q1: " + q1.calcularArea());
        System.out.println("Area de q2: " + q2.calcularArea());
    }
}
```

3.3 Classe Contador

Implemente uma classe Contador que possui como atributos uma String contendo o nome do contador, e um inteiro contendo a contagem atual do mesmo. A classe deve possuir um método para incrementar o contador. Crie um construtor específico para a classe. Crie uma classe TestaContador que utilize um objeto da classe Contador. Inclua um novo atributo referente ao valor máximo que o contador pode atingir e um método para resetar o contador. Altere o método incrementa() para garantir que o contador nunca ultrapasse o máximo permitido. Altere o construtor do método para garantir que todo contador tenha um valor máximo na sua inicialização. Altere o construtor para que o contador possa ser nicializado com algum valor. Inclua um novo método incrementaDelta(int d)

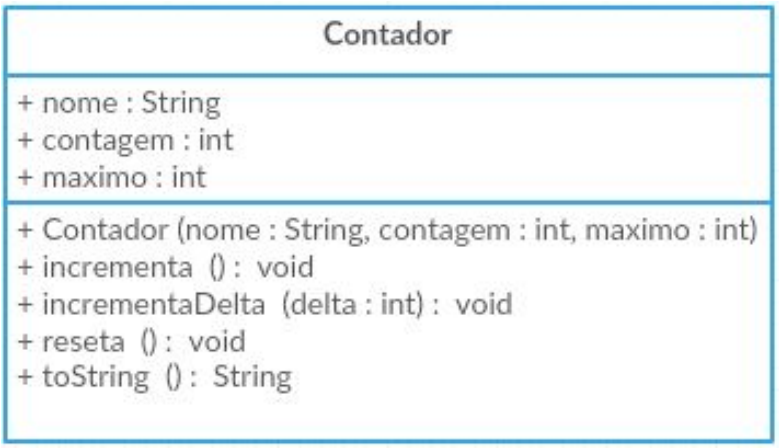

Figura 3.3: Classe Contador

que realiza o incremento de um valor delta enviado.

Classe Contador (código no github):

```
public class Contador {
    String nome;
    int contagem;
    int maximo;

    Contador(String nome, int contagem, int maximo) {
        this.nome = nome;
        this.contagem = contagem;
        this.maximo = maximo;
    }

    void incrementa() {
        if (this.contagem < this.maximo)
            this.contagem += 1;
    }

    void incrementaDelta(int delta) {
        this.contagem += delta;
    }

    void reseta() {
        this.contagem = 0;
    }

    @Override
```

```
    public String toString(){
        return this.getClass().getName() + ": nome =
        ↪  "+this.nome+", contagem = "+this.contagem+", maximo =
        ↪  "+this.maximo;
    }
}
```

Classe TestaContador (código no github):

```
public class TestaContador {
    public static void main(String[] args) {
        Contador c1 = new Contador("pessoas",1,10);
        c1.incrementa();
        c1.incrementaDelta(5);
        System.out.println(c1.toString());
    }
}
```

3.4 Classe Contador Encapsulado

Implemente a classe a seguir juntamente com a classe para a utilização da mesma. Desenvolva o construtor e encapsule os atributos. Desenvolva os métodos de acesso aos atributos.

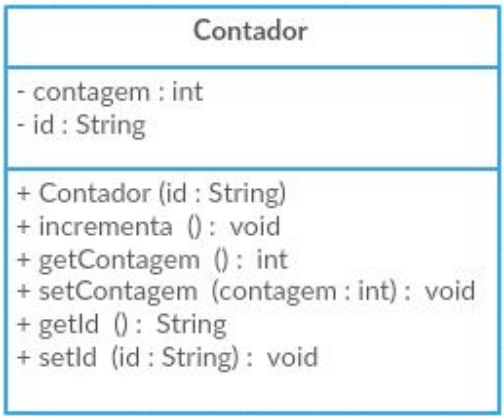

Figura 3.4: Classe Contador

Classe Contador:
– int: contagem (número de incrementos desde a criação)
– String: id (identificador)
– void incrementa (incrementa a contagem em 1)

Uso da classe Classe contador: programa que gera 100 números aleatórios e que possui 2 contadores, um para números pares e outro para números ímpares. O programa imprime a quantidade sorteada de cada tipo e informa qual a contagem venceu.

Classe Contador (código no github):

```
public class Contador {
    private int contagem;
    private String id;

    public Contador(String id) {
        this.id = id;
    }

    public void incrementa() {
        this.contagem += 1;
    }

    public int getContagem() {
        return this.contagem;
    }

    public void setContagem(int contagem) {
        this.contagem = contagem;
    }

    public String getId() {
        return this.id;
    }

    public void setId(String id) {
        this.id = id;
    }
}
```

Classe TestaContador (código no github):

```
import java.util.Random;

public class TestaContador {
    public static void main(String[] args) {
        Random r = new Random();
        Contador contaPar = new Contador("Pares"), contaImpar =
        ↪  new Contador("Impares");

        for (int i = 0; i < 100; i++) {
```

```
            if(r.nextInt() % 2 == 0){
                contaPar.incrementa();
            }else{
                contaImpar.incrementa();
            }
        }

        System.out.println(contaPar.getId() + ": " +
        ↪  contaPar.getContagem());
        System.out.println(contaImpar.getId() + ": " +
        ↪  contaImpar.getContagem());

        if (contaPar.getContagem() == contaImpar.getContagem()){
            System.out.println("Contagens empataram!");
        }else if (contaPar.getContagem() >
        ↪  contaImpar.getContagem()){
            System.out.println("Par venceu!");
        }else{
            System.out.println("Impar venceu!");
        }
    }
}
```

3.5 Classe Hora

Implemente a classe a seguir juntamente com a classe para a utilização da mesma. Desenvolva o construtor e encapsule os atributos. Desenvolva os métodos de acesso aos atributos.

Classe Hora:
– hora, minutos e segundos
– construtor da hora
– método setTime(int h, int m, int s) //garante que não haja valores inválidos nos atributos da hora e coloca em zero sempre que um parâmetro enviado para o atributo for inválido.

Desenvolva métodos separados para incializar a hora, o minuto e o segundo, e reutilize os mesmos no método setTime.

Uso da classe Hora: inicializar alguns objetos do tipo Hora e imprimir a hora na tela. Testar a inserção de valores inválidos nos objetos criados.

Classe Hora (código no github):

```
public class Hora {
    private int hora;
    private int minuto;
    private int segundo;
```

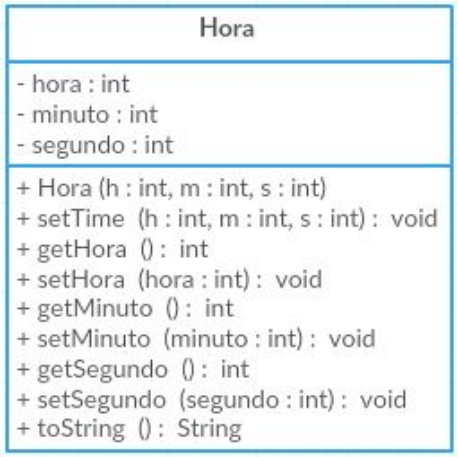

Figura 3.5: Classe Hora

```
Hora(int h, int m, int s){
    this.setTime(h,m,s);
}

public void setTime(int h, int m, int s) {
    this.setHora(h);
    this.setMinuto(m);
    this.setSegundo(s);
}

public int getHora() {
    return this.hora;
}

public void setHora(int hora) {
    if (hora >= 0 && hora < 24)
        this.hora = hora;
    else
        this.hora = 0;
}

public int getMinuto() {
    return this.minuto;
}

public void setMinuto(int minuto) {
    if (minuto >= 0 && minuto < 60)
```

```
            this.minuto = minuto;
        else
            this.minuto = 0;
    }

    public int getSegundo() {
        return this.segundo;
    }

    public void setSegundo(int segundo) {
        if (segundo >= 0 && segundo < 60)
            this.segundo = segundo;
        else
            this.segundo = 0;
    }

    @Override
    public String toString() {
        return this.getClass().getName()+": " + this.hora + ",
        ↪  Minuto = " + this.minuto + ", Segundo = " +
        ↪  this.segundo;
    }
}
```

Classe TestaHora (código no github):

```
public class TestaHora {
    public static void main(String args[]) {
        Hora h1 = new Hora(64, 2, -1);
        Hora h2 = new Hora(0,0,0);
        h2.setTime(17, 2, 80);

        System.out.println(h1.toString());
        System.out.println(h2.toString());
    }
}
```

3.6 Classe Hora 2

Implemente a classe a seguir juntamente com a classe para a utilização da mesma. Desenvolva o construtor e encapsule os atributos. Desenvolva os métodos de acesso aos atributos.

Implemente a classe Hora descrita no diagrama de classe na figura 3.6. Os métodos Set dos atributos devem garantir valores válidos de horas (0 até 23), minutos (0 até 59) e segundos (0 até 59). O método passa1Segundo deve ser responsável por passar 1 minuto caso necessário, assim como o método passa1Minuto

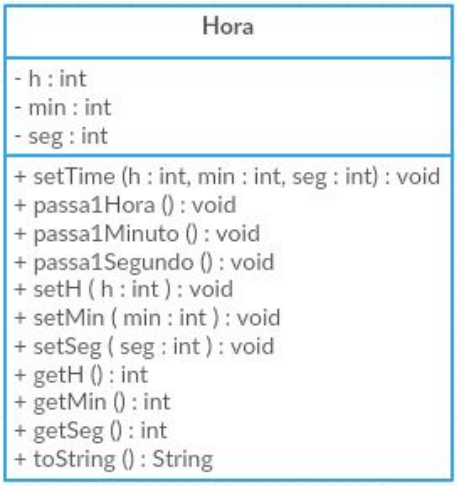

Figura 3.6: Classe Hora

deve fazer com as horas.

Classe Hora (código no github):

```
public class Hora{
    private int h;
    private int min;
    private int seg;

    Hora(int h, int min, int seg){
        setTime(h, min, seg);
    }

    void setTime(int h, int min, int seg){
        setH(h);
        setMin(min);
        setSeg(seg);
    }

    void passa1Segundo(){
        if (this.seg < 59)
            this.seg++;
        else {
            this.seg = 0;
            passa1Minuto();
        }
    }
```

```
void passa1Minuto(){
    if (this.min < 59)
        this.min++;
    else{
        this.min = 0;
        passa1Hora();
    }
}
void passa1Hora(){
    if (this.h < 23)
        this.h++;
    else
        this.h = 0;
}

void setH(int h){
    if (h >= 0 && h <= 23)
        this.h = h;
    else
        this.h = 0;
}
void setMin(int min){
    if (min >= 0 && min <=59)
        this.min = min;
    else
        this.min = 0;
}
void setSeg(int seg){
    if (seg >=0 && seg <= 59)
        this.seg = seg;
    else
        this.seg = 0;
}
int getH(){
    return this.h;
}
int getMin(){
    return this.min;
}
int getSeg(){
    return this.seg;
}

@Override
public String toString(){
```

```
        return this.getClass().getName() + ": h = " + this.h + ",
        ↪  min = " + this.min + ", seg = " + this.seg;
    }
}
```

Classe TestaHora (código no github):

```
public class TestaHora {
    public static void main(String[] args){
        Hora hora = new Hora(23,58,57);
        System.out.println(hora.toString());
        hora.passa1Segundo();
        System.out.println(hora.toString());
        hora.passa1Minuto();
        hora.passa1Segundo();
        System.out.println(hora.toString());
        hora.passa1Segundo();
        System.out.println(hora.toString());
    }
}
```

3.7 Classe Data

Classe Data:
– dia, mês e ano
– método setData(int dia, int mes, int ano) //garante que não haja valores inválidos nos atributos do dia, mês e ano. Um desafio interessante é o de criar um método para validar a data considerando todos os atributos e a possibilidade de existência de anos bissextos.

Classe Data (código no github):

```
public class Data {
  private int dia;
  private int mes;
  private int ano;


  void setData(int d, int m, int a) {
    this.setDia(d);
    this.setMes(m);
    this.setAno(a);
  }

  public int getDia() {
    return dia;
```

```
  }

  private void setDia(int dia) {
    if (dia >= 1 && dia <= 31)
      this.dia = dia;
    else
      dia = 1;
  }

  public int getMes() {
    return mes;
  }

  private void setMes(int mes) {
    if (mes >= 1 && mes <= 12)
      this.mes = mes;
    else
      this.mes = 1;
  }

  public int getAno() {
    return ano;
  }

  private void setAno(int ano) {
    if (ano >= 1900 && ano <= 2018)
      this.ano = ano;
    else
      this.ano = 2018;
  }

  public String getData() {
    return  this.dia +"/"+ this.mes +"/"+this.ano;
  }
}
```

Classe TestaData (código no github):

```
public class TestaData {
    public static void main(String args[]){
        Data data = new Data();
        data.setData(10, 5, 2000);
        System.out.println(data.getData());
    }
}
```

3.8 Classe Calendario

Classe Calendario (código no github):

```
public class Calendario{
    private int dia, mes, ano;

    Calendario(int d, int m, int a){
        this.ano = a;
        if (m == 4 || m == 6 || m==9 || m==11){
            this.mes = m;
            if (d >= 1 && d <= 30){
                this.dia =d;
            }
            else{
                this.dia =1;
            }
        }
        else
            if (m == 1 || m==3 || m==5 || m==7 || m==8 || m==10
            ↪  ||m==12){
                this.mes = m;
                if (d >=1 && d <=31){
                    this.dia = d;
                }
                else {
                    this.dia = 1;
                }
            }
            else {
                if (m == 2) {
                    this.mes = m;
                    if (d >=1 && d<=29){
                        this.dia = d;
                    }
                    else{
                        this.dia = 1;
                    }
                }
                else {
                    this.mes = 1;
                    this.dia = 1;
                }
            }
    }
```

```
    int getDia(){
        return dia;
    }
    int getMes(){
        return mes;
    }
    int getAno(){
        return ano;
    }
    @Override
    public String toString(){
        return "Data: "+this.dia+"/"+ this.mes+"/"+ this.ano;
    }
    @Override
    public boolean equals(Object o){
        if (this == o) return true;
        if (!(o instanceof Calendario)) return false;
        Calendario c = (Calendario) o;
        return this.dia == c.dia && this.mes == c.mes && this.ano
        ↪  == c.ano;
    }
}
```

Classe TestaCalendario (código no github):

```
public class TestaCalendario{

    public static void main(String[] args){
        Calendario novo = new Calendario(02, 01, 2018);
        System.out.println(novo);

        Calendario antigo = new Calendario(02, 01, 2018);
        System.out.println(antigo);

        System.out.println(novo.equals(antigo));

    }

}
```

3.9 Classe Conta Bancária

Classe ContaBancaria (código no github):

```
public class ContaBancaria {
    private float saldo;
```

```
    private float limite;

    ContaBancaria(float lim){
        this.saldo = 0;
        this.limite = lim;
    }


    public void consultaSaldo(){
        System.out.println("Saldo = "+ this.saldo);
        System.out.println("Limite = "+ this.limite);
    }

    public void depositaValor(float valor){
        if (valor > 0)
            this.saldo = this.saldo + valor;
    }

    public boolean saque (float valor){
        if (valor <= this.saldo + this.limite) {
            this.saldo = this.saldo - valor;
            return true;
        }
        else return false;
    }

    public void transferencia(float valor, ContaBancaria
    ↪ destino){
        if (this.saque(valor))
            destino.depositaValor(valor);

    }

}
```

Classe TestaContaBancaria (código no github):

```
import java.util.Scanner;

public class TestaContaBancaria {

    public static void main(String[] args){

        ContaBancaria contaA = new ContaBancaria(500);
        contaA.depositaValor(1000);
        System.out.println("A");
        contaA.consultaSaldo();
```

```
        ContaBancaria contaB = new ContaBancaria(0);
        System.out.println("B");
        contaB.consultaSaldo();

        contaA.transferencia(1300, contaB);

        System.out.println("A");
        contaA.consultaSaldo();

        System.out.println("B");
        contaB.consultaSaldo();

    }

}
```

3.10 Agregação

3.10.1 Estudante, Professor, Faculdade - Endereço

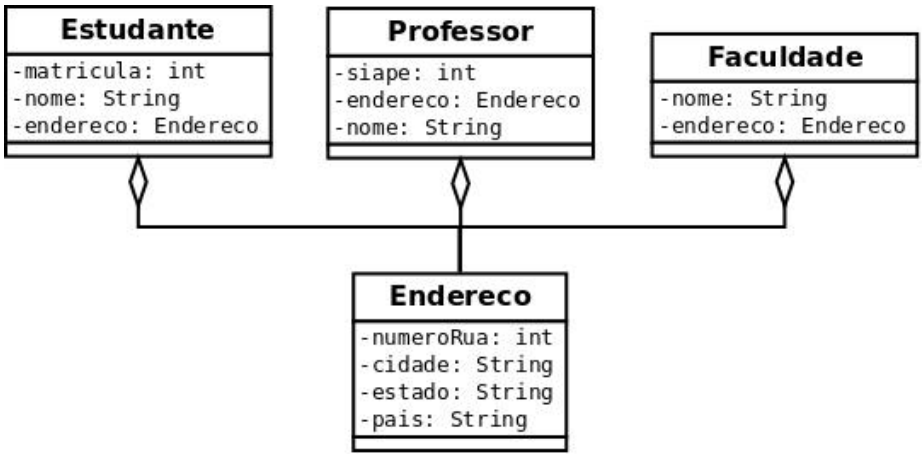

Figura 3.7: Classes Endereco, Estudante, Professor e Faculdade

Desenvolva as classes Endereco, Estudante, Professor e Faculdade descritas na figura 3.7 e uma classe de Teste que instancie objetos dessas classes e que imprima os valores na tela. Em seguida, instancie um mesmo endereco para diferentes objetos, imprima os objetos e altere o valor do endereco. Imprima novamente os objetos e observe o resultado.

Classe Estudante (código no github):

```
public class Estudante {
    private int matricula;
    private String nome;
    private Endereco endereco;

    public Estudante(int matricula, String nome, Endereco
    ↪  endereco) {
        this.matricula = matricula;
        this.nome = nome;
        this.endereco = endereco;
    }

    public int getMatricula() {
        return matricula;
    }

    public void setMatricula(int matricula) {
```

```
        this.matricula = matricula;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Endereco getEndereco() {
        return endereco;
    }

    public void setEndereco(Endereco endereco) {
        this.endereco = endereco;
    }

     @Override
    public String toString() {
        return "Estudante{" + "matricula=" + matricula + ",
        ↪  nome=" + nome + ", endereco=" + endereco + '}';
    }
}
```

Classe Faculdade (código no github):

```
public class Faculdade {
    private int siape;
    private Endereco endereco;
    private String nome;

    public Faculdade(int siape, Endereco endereco, String nome) {
        this.siape = siape;
        this.endereco = endereco;
        this.nome = nome;
    }


    public int getSiape() {
        return siape;
    }

    public void setSiape(int siape) {
        this.siape = siape;
    }
```

```
    public Endereco getEndereco() {
        return endereco;
    }

    public void setEndereco(Endereco endereco) {
        this.endereco = endereco;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
        @Override
    public String toString() {
        return "Faculdade{" + "siape=" + siape + ", endereco=" +
        ↪  endereco + ", nome=" + nome + '}';
    }

}
```

Classe Professor (código no github):

```
public class Professor {
    private String nome;
    private Endereco endereco;

    public Professor(String nome, Endereco endereco) {
        this.nome = nome;
        this.endereco = endereco;
    }


    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Endereco getEndereco() {
        return endereco;
    }
```

```
    public void setEndereco(Endereco endereco) {
        this.endereco = endereco;
    }
        @Override
    public String toString() {
        return "Professor{" + "nome=" + nome + ", endereco=" +
        ↪  endereco + '}';
    }
}
```

Classe Endereco (código no github):

```
public class Endereco {
    private int numeroRua;
    private String cidade;
    private String estado;
    private String pais;

    public Endereco(int numeroRua, String cidade, String estado,
    ↪  String pais) {
        this.numeroRua = numeroRua;
        this.cidade = cidade;
        this.estado = estado;
        this.pais = pais;
    }

    public int getNumeroRua() {
        return numeroRua;
    }

    public void setNumeroRua(int numeroRua) {
        this.numeroRua = numeroRua;
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

    public String getEstado() {
        return estado;
    }
```

```
    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getPais() {
        return pais;
    }

    public void setPais(String pais) {
        this.pais = pais;
    }

    @Override
    public String toString() {
        return "Endereco{" + "numeroRua=" + numeroRua + ",
        ↪  cidade=" + cidade + ", estado=" + estado + ", pais="
        ↪  + pais + '}';
    }

}
```

Classe TestaClasses (código no github):

```
public class TestaClasses {
    public static void main(String[] args) {
        Endereco e = new Endereco(123456, "Ararangua", "SC",
        ↪  "Brasil");
        Endereco e2 = new Endereco(22146, "Bom jardim", "SC",
        ↪  "Brasil");
        Endereco e4 = new Endereco(1462, "Chapeco", "SC",
        ↪  "Brasil");
        Faculdade f = new Faculdade(465789, e4, "UFFS");
        Estudante est = new Estudante(798465465, "Josias", e2);
        Professor prof = new Professor("Cristiano", e);

        System.out.println("--------Teste1---------");
        System.out.println(e);
        System.out.println(e2);
        System.out.println(e4);
        System.out.println(f);
        System.out.println(est);
        System.out.println(prof);

        System.out.println("--------Teste2---------");
        f.setEndereco(e);
        est.setEndereco(e);
        prof.setEndereco(e);
```

```
        System.out.println(f);
        System.out.println(est);
        System.out.println(prof);
        //alterando valores de endereco
        e.setNumeroRua(244455);
        e.setEstado("PR");
        e.setCidade("Curitiba");

        System.out.println("--------Apos alteracao---------");
        System.out.println(f);
        System.out.println(est);
        System.out.println(prof);
    }
}
```

3.10.2 Pessoa - Tempo

Desenvolva uma classe Pessoa (com nome, idade e data de nascimento). Utilize uma classe Tempo (dia, mês e ano) para representar a data de nascimento. Em seguida, desenvolva uma classe de Teste que cria uma pessoa e imprima a mesma.

Classe Tempo (código no github):

```
class Tempo{
    private  int ano;
    private  int mes;
    private  int dia;

  public Tempo(int d, int m, int a){
    setDia(d);
    setMes(m);
    setAno(a);
  }
  public void setDia(int d){
    if((d>0) && (d<=31)){
      this.dia = d;
    }else{
      System.out.println("Dia Invalido");
    }
  }
  public void setMes(int m){
    if((m>0) && (m<=12)){
      this.mes = m;
    }else{
      System.out.println("Mes invalido");
    }
```

```
  }
  public void setAno(int a){
    if((a>0) && (a<2019)){
      this.ano = a;
    }else{
      System.out.println("Ano invalido");
    }
  }
  public String toString() {
        return  dia + " / " + mes + " / " + ano;
    }
}
```

Classe Pessoa (código no github):

```
public class Pessoa{
    private  String name;
    private  int age;
    private  Tempo date;

  public Pessoa(String n, int a, Tempo t){
    this.name = n;
    this.age = a;
    this.date = t;
  }
  public String toString() {
        return "Nome: " + name + ", Idade: " + age + ", Data
        ↪  Nascimento: " + date;
    }
}
```

Classe TestaTempo (código no github):

```
public class TestaTempo{
    public static void main(String[] args){

        Tempo t1 = new Tempo(11,9,1997);
        Pessoa p1 = new Pessoa("Marcelo", 20, t1);
        System.out.println(p1);

  }
}
```

3.10.3 Avaliação - Questão

Desenvolva uma classe Avaliacao que contenha o seu número, e questões (q1, q2 e q3). Utilize uma classe Questao que contenha um atributo chamado "enunciado". Desenvolva uma classe de Teste que crie uma avaliação qualquer e que

imprima suas questões.

Classe Questao (código no github):

```
public class Questao{
    private String enunciado;

    public Questao(String n){
        enunciado = n;
    }
    public String toString(){
        return enunciado+":____________________.";
    }
}
```

Classe Avaliacao (código no github):

```
public class Avaliacao{
    private int numero;
    private Questao q1;
    private Questao q2;
    private Questao q3;

    public Avaliacao(int n, Questao a, Questao b, Questao c){
        this.numero = n;
        this.q1=a;
        this.q2=b;
        this.q3=c;
    }
    public String toString(){
        return "Avaliacao " + numero+ "\n\n"+ q1
        ↪  +"\n"+q2+"\n"+q3;
    }
}
```

Classe TesteAvalicaoQuestao (código no github):

```
public class TesteAvalicaoQuestao{
    public static void main(String[] args){

        Questao calc1 = new Questao("Calcule as raizes: x^2+1");
        Questao calc2 = new Questao("Plot a seguinte equacao:
        ↪  x^2+4");
        Questao calc3 = new Questao("Encontre a serie para
        ↪  x^2+16");

        Avaliacao first = new Avaliacao(1, calc1, calc2, calc3);
```

```
        System.out.println(first);
    }
}
```

3.10.4 Imóvel - Pessoa

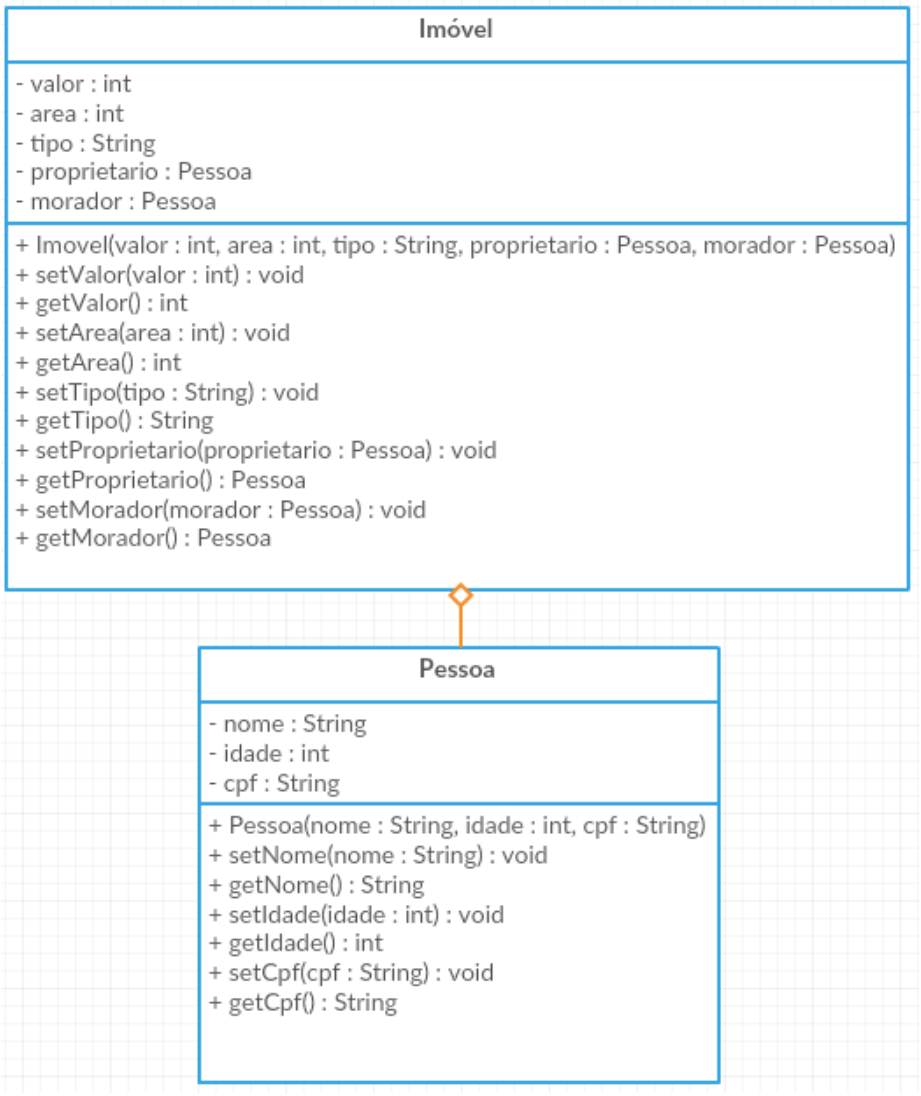

Figura 3.8: Diagrama Imóvel Pessoa

Classe Imovel (código no github):

```
public class Imovel{
    private int valor;
    private int area;
    private String tipo;
    private Pessoa proprietario;
    private Pessoa morador;

    Imovel(int valor, int area, String tipo, Pessoa proprietario,
    ↪  Pessoa morador){
        this.setValor(valor);
        this.setArea(area);
        this.setTipo(tipo);
        this.setProprietario(proprietario);
        this.setMorador(morador);
    };

    public void setValor(int valor){
        this.valor = valor;
    };

    public int getValor(){
        return this.valor;
    };

    public void setArea(int area){
        this.area = area;
    };

    public int getArea(){
        return this.area;
    };

    public void setTipo(String tipo){
        this.tipo = tipo;
    };

    public String getTipo(){
        return this.tipo;
    };

    public void setProprietario(Pessoa proprietario){
        this.proprietario = proprietario;
    };

    public Pessoa getProprietario(){
        return this.proprietario;
```

```
    };

    public void setMorador(Pessoa morador){
        this.morador = morador;
    };

    public Pessoa getMorador(){
        return this.morador;
    };

    @Override
    public String toString(){
        return this.getClass().getName() +": valor =
        ↪  "+this.valor+", area = "+this.area+", tipo =
        ↪  "+this.tipo+",\nproprietario =
        ↪  "+this.proprietario.toString()+",\nmorador =
        ↪  "+this.morador.toString();
    }
}
```

Classe Pessoa (código no github):

```
public class Pessoa{
    private String nome;
    private int idade;
    private String cpf;

    Pessoa(String nome, int idade, String cpf){
        this.setNome(nome);
        this.setIdade(idade);
        this.setCpf(cpf);
    };

    public void setNome(String nome){
        this.nome = nome;
    };

    public String getNome(){
        return this.nome;
    };

    public void setIdade(int idade){
        this.idade = idade;
    };

    public int getIdade(){
        return this.idade;
```

```
    };

    public void setCpf(String cpf){
        this.cpf = cpf;
    };

    public String getCpf(){
        return cpf;
    };

    @Override
    public String toString(){
        return this.getClass().getName() +": nome =
        ↪  "+this.nome+", idade = "+this.idade+", cpf =
        ↪  "+this.cpf;
    }
}
```

Classe TestaImovelPessoa (código no github):

```
public class TestaImovelPessoa {
    public static void main(String args[]){
        Pessoa pessoa1 = new Pessoa("Ana",20,"102.000.006-00");
        Pessoa pessoa2 = new Pessoa("Maria",20,"103.000.006-00");
        Imovel imovel = new
        ↪  Imovel(300000,100,"apartamento",pessoa1,pessoa2);
        System.out.println(imovel.toString());
    }
}
```

3.10.5 Mensagem - Contato

Classe Mensagem (código no github):

```
public class Mensagem{
    private String texto;
    private Contato destinatario;
    private Contato remetente;

    Mensagem(String texto, Contato destinatario, Contato
    ↪  remetente){
        this.setTexto(texto);
        this.setDestinatario(destinatario);
        this.setRemetente(remetente);
    };

    public void setTexto(String texto){
```

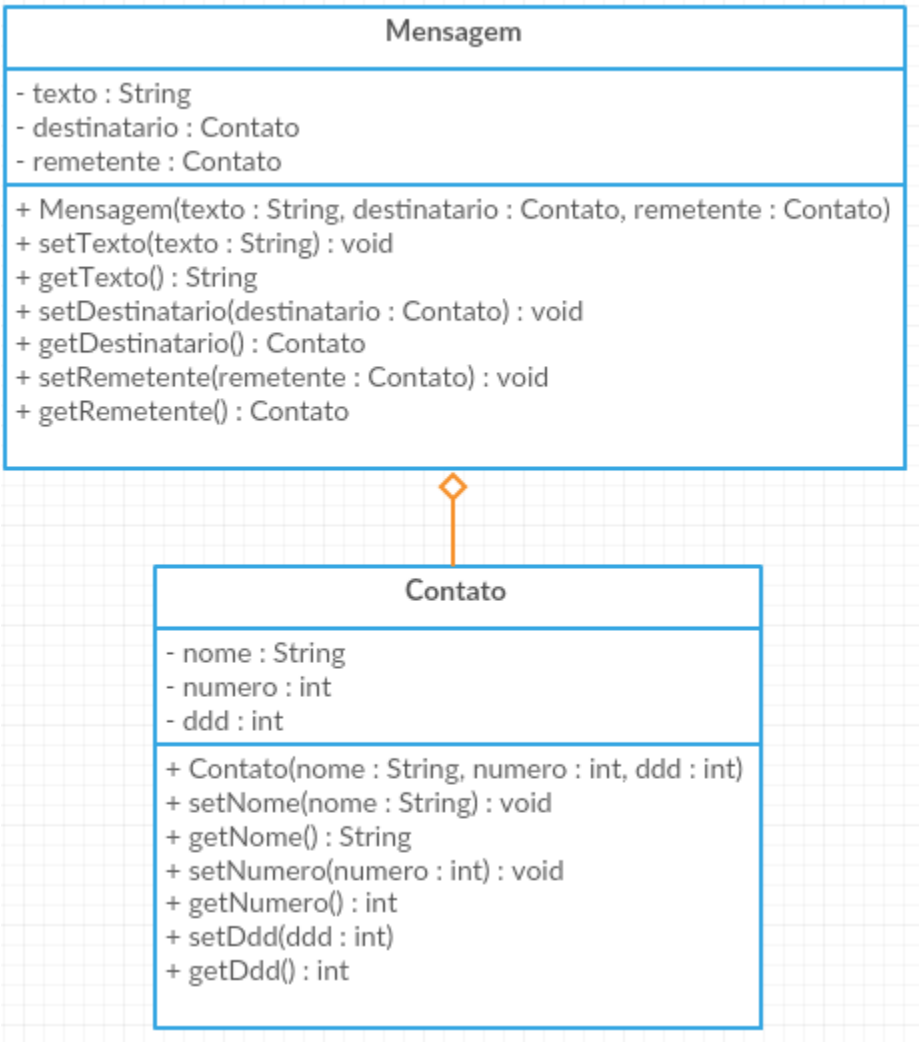

Figura 3.9: Diagrama Mensagem Contato

```
    this.texto = texto;
};

public String getTexto(){
    return this.texto;
};

public void setDestinatario(Contato destinatario){
    this.destinatario = destinatario;
```

```
    };

    public Contato getDestinatario(){
        return this.destinatario;
    };

    public void setRemetente(Contato remetente){
        this.remetente = remetente;
    };

    public Contato getRemetente(){
        return this.remetente;
    };

    @Override
    public String toString(){
        return this.getClass().getName() +": texto =
        ↪  "+this.texto+",\ndestinatario =
        ↪  "+this.destinatario.toString()+",\nremetente =
        ↪  "+this.remetente.toString();
    }
}
```

Classe Contato (código no github):

```
public class Contato{
    private String nome;
    private int numero;
    private int ddd;

    Contato(String nome, int numero, int ddd){
        this.setNome(nome);
        this.setNumero(numero);
        this.setDdd(ddd);
    };

    public void setNome(String nome){
        this.nome = nome;
    };

    public String getNome(){
        return this.nome;
    };

    public void setNumero(int numero){
        this.numero = numero;
    };
```

```
    public int getNumero(){
        return this.numero;
    };

    public void setDdd(int ddd){
        this.ddd = ddd;
    };

    public int getDdd(){
        return this.ddd;
    };

    @Override
    public String toString(){
        return this.getClass().getName() +": nome =
        ↪  "+this.nome+", numero = "+this.numero+", ddd =
        ↪  "+this.ddd;
    }
}
```

Classe TestaMensagemContato (código no github):

```
public class TestaMensagemContato {
    public static void main(String args[]){
        Contato contato1 = new Contato("Anderson",999804321,55);
        Contato contato2 = new Contato("Gabriel",999721234,48);
        Mensagem msg1 = new Mensagem("Bom
        ↪  dia!",contato1,contato2);
        System.out.println(msg1.toString());
    }
}
```

3.10.6 Tinta - Cor

Classe Tinta (código no github):

```
public class Tinta{
    private String nome;
    private int preco;
    private boolean provaDagua;
    private Cor cor;

    Tinta(String nome, int preco, boolean provaDagua, Cor cor){
        this.setNome(nome);
        this.setPreco(preco);
        this.setProvaDagua(provaDagua);
```

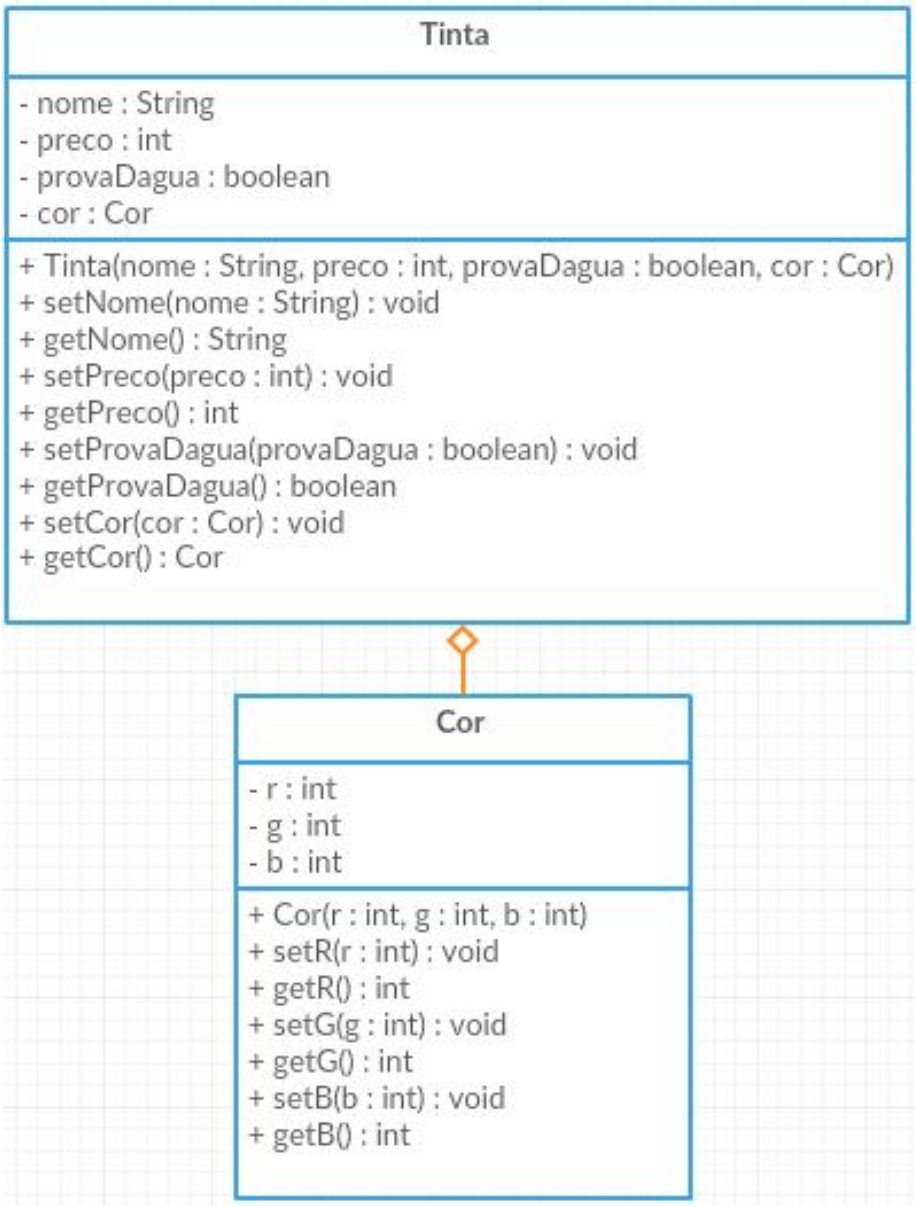

Figura 3.10: Diagrama Tinta Cor

```
        this.setCor(cor);
    };

    public void setNome(String nome){
```

```
        this.nome = nome;
    };

    public String getNome(){
        return this.nome;
    };

    public void setPreco(int preco){
        this.preco = preco;
    };

    public int getPreco(){
        return this.preco;
    };

    public void setProvaDagua(boolean provaDagua){
        this.provaDagua = provaDagua;
    };

    public boolean getProvaDagua(){
        return this.provaDagua;
    };

    public void setCor(Cor cor){
        this.cor = cor;
    };

    public Cor getCor(){
        return this.cor;
    };

    @Override
    public String toString(){
        return this.getClass().getName() + ": nome = " +
        ↪  this.nome + ", preco = " + this.preco + ", provaDagua
        ↪  = " + this.provaDagua + ",\ncor = " +
        ↪  this.cor.toString();
    };
}
```

Classe Cor (código no github):

```
public class Cor{
    private int r;
    private int g;
    private int b;
```

```
    Cor(int r, int g, int b){
        this.setR(r);
        this.setG(g);
        this.setB(b);
    };

    public void setR(int r){
        this.r = r;
    };

    public int getR(){
        return this.r;
    };

    public void setG(int g){
        this.g = g;
    };

    public int getG(){
        return this.g;
    };

    public void setB(int b){
        this.b = b;
    };

    public int getB(){
        return this.b;
    };

    @Override
    public String toString(){
        return this.getClass().getName() + ": r = " + this.r + ",
        ↪  g = " + this.g + ", b = " + this.b;
    };
}
```

Classe TestaTintaCor (código no github):

```
public class TestaTintaCor {
    public static void main(String args[]){
        Cor cor = new Cor(255,255,255); //branco
        Tinta tinta = new Tinta("Tinta branca",20,false,cor);
        System.out.println(tinta.toString());
    }
}
```

3.10.7 Agenda - Contato

Classe Agenda (código no github):

```
import java.util.*;

public class Agenda {
    private ArrayList<Contato> lista;

    Agenda(){
        this.lista = new ArrayList<Contato>();
    }

    void remove2(Contato c){
        this.lista.remove(c);
    }

    void remover(String t){
        int pos = -1;
        for (int i = 0; i < this.lista.size(); i++){
            if (this.lista.get(i).getTelefone() == t) {
                pos = i;
                break;
            }
        }
        if (pos != -1) this.lista.remove(pos);
    }

    void cadastrar(Contato c){
        //boolean adiciona = true;
        /*for (Contato x: this.lista){
            if (x.getTelefone() == c.getTelefone()) {
                adiciona = false;
                break;
            }
        }*/
        //if (adiciona) this.lista.add(c);
        if (!this.lista.contains(c)) this.lista.add(c);

    }

    void imprimirAgenda(){
        for (Contato c: this.lista){
            System.out.println(c);
        }
    }
    @Override
```

```
    public String toString(){
        String result = " ";
        for (Contato c: this.lista) {
            result+= c.toString();
        }
        return result;
    }
}
```

Classe Contato (código no github):

```
public class Contato {
    private String nome;
    private String email;
    private String telefone;

    Contato(String n, String e, String t){
        setNome(n);
        setEmail(e);
        setTelefone(t);
    }
    //sets e gets
    String getNome(){
        return this.nome;
    }
    String getEmail(){
        return this.email;
    }
    String getTelefone(){
        return this.telefone;
    }
    void setNome(String n){
        this.nome = n;
    }
    void setEmail(String e){
        this.email = e;
    }
    void setTelefone(String t){
        this.telefone = t;
    }

    @Override
    public boolean equals(Object o){
        if (this == o) return true;
        if (!(o instanceof Contato)) return false;
        Contato c = (Contato) o;
        return this.email.equals(c.email) &&
```

```
                this.telefone.equals(c.telefone);
    }

    @Override
    public String toString(){
        return "======================\n"
                + "Nome     :"+this.nome +
                "\nTelefone:"+ this.telefone+
                "\nEmail    :"+ this.email+
                "\n======================\n";
    }

}
```

Classe TestaAgenda (código no github):

```
public class TestaAgenda {
    public static void main(String[] args){
        Contato dd = new Contato("jose", "jose@hotmail.com",
        ↪  "333-4444");
        Contato ee = new Contato("ana", "jose@hotmail.com",
        ↪  "333-4444");

        System.out.println(dd.equals(ee));

        Agenda a = new Agenda();

        a.cadastrar(dd);
        a.cadastrar(ee);

        System.out.println(a);

    }
}
```

3.10.8 Biblioteca - Livro

Classe Biblioteca (código no github):

```
import java.util.ArrayList;

public class Biblioteca {
    private ArrayList<Livro> colecao;

    Biblioteca(){
        this.colecao = new ArrayList<Livro>();
    }
```

```
    void adicionaLivro(Livro liv){
        this.colecao.add(liv);
    }

    Livro buscaLivro(int ano){
        for (Livro liv: this.colecao){
            if (liv.getAno() == ano) return liv;
        }
        return null;
    }

    void imprimeLivros(){
        for (Livro liv: this.colecao)
            System.out.println(liv);
    }
}
```

Classe Livro (código no github):

```
public class Livro {
    private String titulo;
    private String autor;
    private int ano;


    Livro (String t, String a, int ano){
        this.titulo = t;
        this.autor = a;
        this.ano = ano;
    }

    int getAno(){
        return this.ano;
    }

    @Override
    public String toString(){
        return "Titulo: "+ this.titulo + "\n"+
                "Autor: "+ this.autor  + "\n"+
                "Ano  : "+ this.ano+"\n";

    }
}
```

Classe TestaBibliotecaLivro (código no github):

```
public class TestaBibliotecaLivro {
```

```
    public static void main(String[] args){

        Biblioteca bib = new Biblioteca();
        Livro liv = new Livro("X", "Y", 1998);
        bib.adicionaLivro(liv);

        Livro liv2 = new Livro("W", "Z", 1977);
        bib.adicionaLivro(liv2);

        bib.imprimeLivros();

        Livro result = bib.buscaLivro(1998);
        System.out.println("=======\n" + result);

    }

}
```

3.10.9 Calendario - Pessoa

Classe Calendario (código no github):

```
public class Calendario{
    private int dia, mes, ano;

    Calendario(int d, int m, int a){
        this.ano = a;
        if (m == 4 || m == 6 || m==9 || m==11){
            this.mes = m;
            if (d >= 1 && d <= 30){
                this.dia =d;
            }
            else{
                this.dia =1;
            }
        }
        else
            if (m == 1 || m==3 || m==5 || m==7 || m==8 || m==10
            ↪  ||m==12){
                this.mes = m;
                if (d >=1 && d <=31){
                    this.dia = d;
                }
                else {
                    this.dia = 1;
```

```
                }
            }
            else {
                if (m == 2) {
                    this.mes = m;
                    if (d >=1 && d<=29){
                        this.dia = d;
                    }
                    else{
                        this.dia = 1;
                    }
                }
                else {
                    this.mes = 1;
                    this.dia = 1;
                }
            }
    }

    int getDia(){
        return dia;
    }
    int getMes(){
        return mes;
    }
    int getAno(){
        return ano;
    }

    @Override
    public String toString(){
        return this.dia + "/" + this.mes + "/" + this.ano;

    }

}
```

Classe Pessoa (código no github):

```
public class Pessoa {
    private String nome;
    private int idade;
    private Calendario dataNascimento;

    Pessoa (String n, int i, Calendario data){
        this.nome = n;
        this.idade = i;
```

```
        this.dataNascimento = data;
    }

    void setNome(String n){
        this.nome = n;
    }
    void setIdade (int i){
        this.idade = i;
    }
    void setDataNascimento(Calendario data){
        this.dataNascimento = data;
    }
    String getNome(){
        return this.nome;
    }
    int getIdade(){
        return this.idade;
    }
    Calendario getDataNascimento(){
        return this.dataNascimento;
    }

    @Override
    public String toString(){
        return this.nome + " tem "+ this.idade + " anos e nasceu
        ↪  em "+ this.dataNascimento.toString();
    }
}
```

Classe TestaCalendarioPessoa (código no github):

```
public class TestaCalendarioPessoa {
    public static void main(String[] args){
        Calendario data = new Calendario(23, 04, 1980);
        Pessoa p = new Pessoa ("joao", 38, data);
        System.out.println(p.toString());

        System.out.println("Nome = "+ p.getNome());
        System.out.println("Idade = "+ p.getIdade());
        System.out.println("Nascimento =
        ↪  "+p.getDataNascimento().getDia()+"/"+
                p.getDataNascimento().getMes()+"/"+
                p.getDataNascimento().getAno());


    }
```

```
}
```

3.10.10 Email - Caixa de Mensagem - Pessoa

Classe Email (código no github):

```
public class Email {
    private String texto;
    private String assunto;
    private Pessoa destinatario;
    private boolean lido; //indica se o email ja foi lido ou nao

    Email(String t, String a, Pessoa d, boolean l){
        setTexto(t);
        setAssunto(a);
        setPessoa(d);
        setLido(l);
    }

    String getTexto(){
        return this.texto;
    }
    Pessoa getPessoa(){
        return this.getPessoa();
    }
    String getAssunto(){
        return this.assunto;
    }

    boolean getLido(){
        return this.lido;
    }
    void setLido(boolean l){
        this.lido = l;
    }
    void setTexto(String t){
        this.texto = t;
    }
    void setAssunto(String a){
        this.assunto = a;
    }
    void setPessoa(Pessoa d){
        this.destinatario = d;
    }

    @Override
```

```
    public boolean equals(Object o){
        if (o instanceof String) return false;
        String aux = (String) o;
        return this.equals(aux);
    }
    @Override
    public String toString(){
        return "destinatario = " + this.destinatario.toString() +
        ↪  ", assunto = " + this.assunto + ", texto: = " +
        ↪  this.texto + ", lido = " + this.lido;
    }
}
```

Classe CaixaDeMensagem (código no github):

```
import java.util.ArrayList;

public class CaixaDeMensagem {
    private ArrayList<Email> listaDeEmails;

    CaixaDeMensagem(){
        listaDeEmails = new ArrayList<Email>();
    }
    void adicionaEmail(Email novo){
        listaDeEmails.add(novo);
    }
    void listaLidos(){
        for (Email e: listaDeEmails){
            if (e.getLido()){
                System.out.println(e);
            }
        }
    }
}
```

Classe Pessoa (código no github):

```
public class Pessoa {
    private String nome;

    Pessoa(String n){
        setNome(n);
    }
    void setNome(String n){
        this.nome = n;
    }
    String getNome(){
```

```
        return this.nome;
    }

    @Override
    public String toString(){
        return nome;
    }
}
```

Classe TestaCaixaDeMensagem (código no github):

```
import java.util.*;

public class TestaCaixaDeMensagem {

    public static void main(String[] args){
        Scanner entrada = new Scanner(System.in);

        Pessoa p = new Pessoa("joao");
        Email e = new Email("oi", "saudacao", p, true);
        CaixaDeMensagem cm = new CaixaDeMensagem();
        cm.adicionaEmail(e);
        cm.listaLidos();

    }
}
```

3.10.11 Computador Desktop

Implemente uma classe que descreva a composição de um computador desktop, criando as classes para o monitor, teclado, mouse e gabinete. Não considere os componentes internos do gabinete no diagrama UML por enquanto.

Classe Monitor:

```
public class Monitor{
  private float polegadas;
  private String marca;

        Monitor(float polegadas, String marca){
            this.setPolegadas(polegadas);
            this.setMarca(marca);
        };

  public void setPolegadas(float polegadas){
    this.polegadas = polegadas;
  };
```

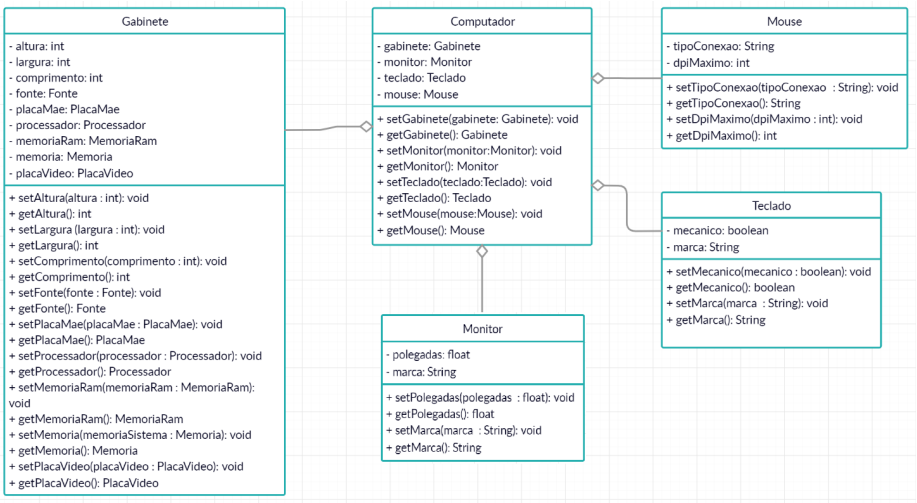

Figura 3.11: Diagrama Computador Desktop

```
  public float getPolegadas(){
    return this.polegadas;
  };

  public void setMarca(String marca){
    this.marca = marca;
  };

  public String getMarca(){
    return this.marca;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " polegadas = " +
     ↪  this.polegadas + ", marca = " + this.marca;
  };
}
```

Classe Teclado:

```
public class Teclado{
  private boolean mecanico;
  private String marca;

        Teclado(boolean mecanico, String marca){
```

```
            this.setMarca(marca);
            this.setMecanico(mecanico);
        }

  public void setMecanico(boolean mecanico){
    this.mecanico = mecanico;
  };

  public boolean getMecanico(){
    return this.mecanico;
  };

  public void setMarca(String marca){
    this.marca = marca;
  };

  public String getMarca(){
    return this.marca;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " mecanico = " +
    ↪  this.mecanico + ", marca = " + this.marca;
  };
}
```

Classe Gabinete:

```
public class Gabinete{
  private int altura;
  private int largura;
  private int comprimento;

        Gabinete(int altura, int largura, int comprimento){
            this.setAltura(altura);
            this.setLargura(largura);
            this.setComprimento(comprimento);
        }

  public void setAltura(int altura){
    this.altura = altura;
  };

  public int getAltura(){
    return this.altura;
  };
```

```
  public void setLargura(int largura){
    this.largura = largura;
  };

  public int getLargura(){
    return this.largura;
  };

  public void setComprimento(int comprimento){
    this.comprimento = comprimento;
  };

  public int getComprimento(){
    return this.comprimento;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " altura = " +
    ↪  this.altura + ", largura = " + this.largura + ",
    ↪  comprimento = " + this.comprimento;
  };
}
```

Classe Mouse:

```
public class Mouse{
  private String tipoConexao;
  private int dpiMaximo;

        Mouse(String tipoConexao, int dpiMaximo){
            this.setTipoConexao(tipoConexao);
            this.setDpiMaximo(dpiMaximo);
        }

  public void setTipoConexao(String tipoConexao){
    this.tipoConexao = tipoConexao;
  };

  public String getTipoConexao(){
    return this.tipoConexao;
  };

  public void setDpiMaximo(int dpiMaximo){
    this.dpiMaximo = dpiMaximo;
  };
```

```
  public int getDpiMaximo(){
    return this.dpiMaximo;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " tipoConexao = " +
    ↪  this.tipoConexao + ", dpiMaximo = " + this.dpiMaximo;
  };
}
```

Classe Computador:

```
public class Computador{
  private Gabinete gabinete;
  private Monitor monitor;
  private Teclado teclado;
  private Mouse mouse;

        Computador(Gabinete gabinete, Monitor monitor, Teclado
        ↪  teclado, Mouse mouse){
            this.setGabinete(gabinete);
            this.setMonitor(monitor);
            this.setTeclado(teclado);
            this.setMouse(mouse);
        };

  public void setGabinete(Gabinete gabinete){
    this.gabinete = gabinete;
  };

  public Gabinete getGabinete(){
    return this.gabinete;
  };

  public void setMonitor(Monitor monitor){
    this.monitor = monitor;
  };

  public Monitor getMonitor(){
    return this.monitor;
  };

  public void setTeclado(Teclado teclado){
    this.teclado = teclado;
  };
```

```
  public Teclado getTeclado(){
    return this.teclado;
  };

  public void setMouse(Mouse mouse){
    this.mouse = mouse;
  };

  public Mouse getMouse(){
    return this.mouse;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + "\n " +
    ↪  this.gabinete.toString() + ",\n " +
    ↪  this.monitor.toString() + ",\n " +
    ↪  this.teclado.toString() + ",\n " + this.mouse.toString();
  };
}
```

Classe TestaComputador:

```
public class TestaComputador {
    public static void main(String[] args) {
        Mouse mouse = new Mouse("USB",3500);
        Teclado teclado = new Teclado(true,"Razer");
        Monitor monitor = new Monitor(24f,"LG");
        Gabinete gabinete = new Gabinete(45,20,75);
        Computador computador = new
        ↪  Computador(gabinete,monitor,teclado,mouse);
        System.out.println(computador.toString());
    }
}
```

3.10.12 Gabinete e componentes

Com base no exercício anterior, elabore classes para os componentes que estão dentro do gabinete (fonte, processador, placa mãe, memória RAM, memória, placa de vídeo).

Classe Monitor:

```
public class Monitor{
  private float polegadas;
  private String marca;
```

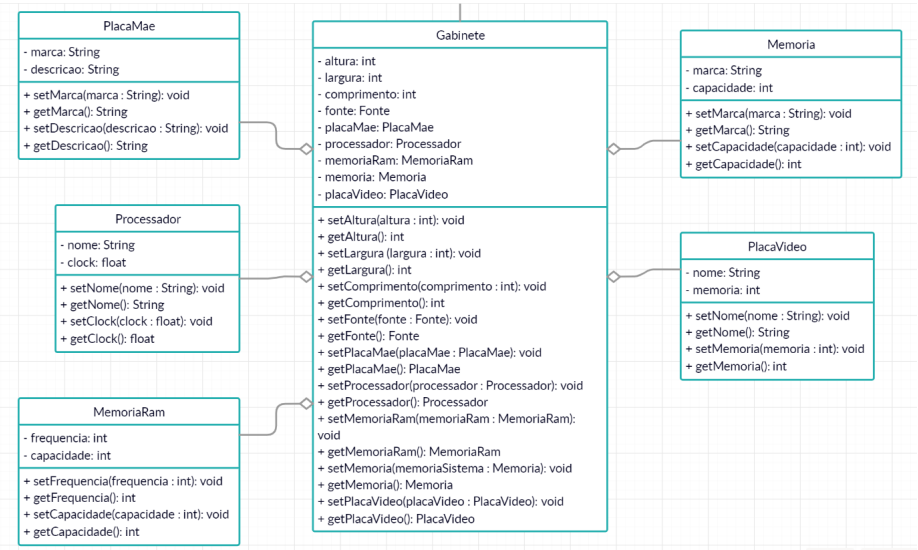

Figura 3.12: Diagrama Gabinete

```
        Monitor(float polegadas, String marca){
            this.setPolegadas(polegadas);
            this.setMarca(marca);
        };

    public void setPolegadas(float polegadas){
      this.polegadas = polegadas;
    };

    public float getPolegadas(){
      return this.polegadas;
    };

    public void setMarca(String marca){
      this.marca = marca;
    };

    public String getMarca(){
      return this.marca;
    };

    @Override
    public String toString(){
```

```
    return this.getClass().getName() + ":" + " polegadas = " +
     ↪  this.polegadas + ", marca = " + this.marca;
  };
}
```

Classe Teclado:

```
public class Teclado{
  private boolean mecanico;
  private String marca;

        Teclado(boolean mecanico, String marca){
            this.setMarca(marca);
            this.setMecanico(mecanico);
        }

  public void setMecanico(boolean mecanico){
    this.mecanico = mecanico;
  };

  public boolean getMecanico(){
    return this.mecanico;
  };

  public void setMarca(String marca){
    this.marca = marca;
  };

  public String getMarca(){
    return this.marca;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " mecanico = " +
     ↪  this.mecanico + ", marca = " + this.marca;
  };
}
```

Classe Gabinete:

```
public class Gabinete{
  private int altura;
  private int largura;
  private int comprimento;

        Gabinete(int altura, int largura, int comprimento){
```

```
            this.setAltura(altura);
            this.setLargura(largura);
            this.setComprimento(comprimento);
        }

  public void setAltura(int altura){
    this.altura = altura;
  };

  public int getAltura(){
    return this.altura;
  };

  public void setLargura(int largura){
    this.largura = largura;
  };

  public int getLargura(){
    return this.largura;
  };

  public void setComprimento(int comprimento){
    this.comprimento = comprimento;
  };

  public int getComprimento(){
    return this.comprimento;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " altura = " +
    ↪  this.altura + ", largura = " + this.largura + ",
    ↪  comprimento = " + this.comprimento;
  };
}
```

Classe Mouse:

```
public class Mouse{
  private String tipoConexao;
  private int dpiMaximo;

        Mouse(String tipoConexao, int dpiMaximo){
            this.setTipoConexao(tipoConexao);
            this.setDpiMaximo(dpiMaximo);
        }
```

```
  public void setTipoConexao(String tipoConexao){
    this.tipoConexao = tipoConexao;
  };

  public String getTipoConexao(){
    return this.tipoConexao;
  };

  public void setDpiMaximo(int dpiMaximo){
    this.dpiMaximo = dpiMaximo;
  };

  public int getDpiMaximo(){
    return this.dpiMaximo;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " tipoConexao = " +
     ↪  this.tipoConexao + ", dpiMaximo = " + this.dpiMaximo;
  };
}
```

Classe Computador:

```
public class Computador{
  private Gabinete gabinete;
  private Monitor monitor;
  private Teclado teclado;
  private Mouse mouse;

        Computador(Gabinete gabinete, Monitor monitor, Teclado
         ↪  teclado, Mouse mouse){
            this.setGabinete(gabinete);
            this.setMonitor(monitor);
            this.setTeclado(teclado);
            this.setMouse(mouse);
        };

  public void setGabinete(Gabinete gabinete){
    this.gabinete = gabinete;
  };

  public Gabinete getGabinete(){
    return this.gabinete;
  };
```

```
  public void setMonitor(Monitor monitor){
    this.monitor = monitor;
  };

  public Monitor getMonitor(){
    return this.monitor;
  };

  public void setTeclado(Teclado teclado){
    this.teclado = teclado;
  };

  public Teclado getTeclado(){
    return this.teclado;
  };

  public void setMouse(Mouse mouse){
    this.mouse = mouse;
  };

  public Mouse getMouse(){
    return this.mouse;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + "\n " +
     ↪  this.gabinete.toString() + ",\n " +
     ↪  this.monitor.toString() + ",\n " +
     ↪  this.teclado.toString() + ",\n " + this.mouse.toString();
  };
}
```

Classe Fonte:

```
public class Fonte{
  private String marca;
  private int potencia;

        Fonte(String marca, int potencia){
            this.setMarca(marca);
            this.setPotencia(potencia);
        }

  public void setMarca(String marca){
    this.marca = marca;
```

```
  };

  public String getMarca(){
    return this.marca;
  };

  public void setPotencia(int potencia){
    this.potencia = potencia;
  };

  public int getPotencia(){
    return this.potencia;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " marca = " +
     ↪  this.marca + ", potencia = " + this.potencia;
  };
}
```

Classe Memoria:

```
public class Memoria{
  private String marca;
  private int capacidade;

        Memoria(String marca, int capacidade){
            this.setMarca(marca);
            this.setCapacidade(capacidade);
        }

  public void setMarca(String marca){
    this.marca = marca;
  };

  public String getMarca(){
    return this.marca;
  };

  public void setCapacidade(int capacidade){
    this.capacidade = capacidade;
  };

  public int getCapacidade(){
    return this.capacidade;
  };
```

```
  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " marca = " +
    ↪  this.marca + ", capacidade = " + this.capacidade;
  };
}
```

Classe MemoriaRam:

```
public class MemoriaRam{
  private int frequencia;
  private int capacidade;

        public MemoriaRam(int frequencia, int capacidade) {
            this.setFrequencia(frequencia);
            this.setCapacidade(capacidade);
        }

  public void setFrequencia(int frequencia){
    this.frequencia = frequencia;
  };

  public int getFrequencia(){
    return this.frequencia;
  };

  public void setCapacidade(int capacidade){
    this.capacidade = capacidade;
  };

  public int getCapacidade(){
    return this.capacidade;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " frequencia = " +
    ↪  this.frequencia + ", capacidade = " + this.capacidade;
  };
}
```

Classe PlacaMae:

```
public class PlacaMae{
  private String marca;
  private String descricao;
```

```
        PlacaMae(String marca, String descricao){
            this.setMarca(marca);
            this.setDescricao(descricao);
        }

  public void setMarca(String marca){
    this.marca = marca;
  };

  public String getMarca(){
    return this.marca;
  };

  public void setDescricao(String descricao){
    this.descricao = descricao;
  };

  public String getDescricao(){
    return this.descricao;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " marca = " +
     ↪  this.marca + ", descricao = " + this.descricao;
  };
}
```

Classe PlacaVideo:

```
public class PlacaVideo{
  private String nome;
  private int memoria;

        PlacaVideo(String nome, int memoria){
            this.setNome(nome);
            this.setMemoria(memoria);
        }

  public void setNome(String nome){
    this.nome = nome;
  };

  public String getNome(){
    return this.nome;
  };
```

```
  public void setMemoria(int memoria){
    this.memoria = memoria;
  };

  public int getMemoria(){
    return this.memoria;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " nome = " +
    ↪  this.nome + ", memoria = " + this.memoria;
  };
}
```

Classe Processador:

```
public class Processador{
  private String nome;
  private float clock;

        Processador(String nome, float clock){
            this.setNome(nome);
            this.setClock(clock);
        }

  public void setNome(String nome){
    this.nome = nome;
  };

  public String getNome(){
    return this.nome;
  };

  public void setClock(float clock){
    this.clock = clock;
  };

  public float getClock(){
    return this.clock;
  };

  @Override
  public String toString(){
    return this.getClass().getName() + ":" + " nome = " +
    ↪  this.nome + ", clock = " + this.clock;
```

```
  };
}
```

Classe TestaComputador:

```
public class TestaComputador {
    public static void main(String[] args) {
        Mouse mouse = new Mouse("USB",3500);
        Teclado teclado = new Teclado(true,"Razer");
        Monitor monitor = new Monitor(24f,"LG");
        Fonte fonte = new Fonte("Corsair",500);
        Memoria memoria = new Memoria("Hitachi",500);
        PlacaVideo placaVideo = new PlacaVideo("gtx 1060", 6);
        PlacaMae placaMae = new PlacaMae("Asus","modelo z170");
        MemoriaRam memoriaRam = new MemoriaRam(2400,8);
        Processador processador = new Processador("i5",2.9f);
        Gabinete gabinete = new
        ↪  Gabinete(45,20,75,fonte,placaMae,processador,memoriaRam,memoria,placaVideo
        Computador computador = new
        ↪  Computador(gabinete,monitor,teclado,mouse);
        System.out.println(computador.toString());
    }
}
```

3.11 Composição

3.11.1 Livro - Capítulo

Classe Livro (código no github):

```
public class Livro{
    private String nome;
    private String autor;
    private Capitulo capitulo1;
    private Capitulo capitulo2;

    Livro(String nome, String autor, String tituloCap1, String
    ↪  textoCap1, String tituloCap2, String textoCap2){
        this.setNome(nome);
        this.setAutor(autor);
        this.setCapitulo1(new Capitulo(tituloCap1, textoCap1));
        this.setCapitulo2(new Capitulo(tituloCap2, textoCap2));
    };

    public String getNome(){
        return this.nome;
    };
```

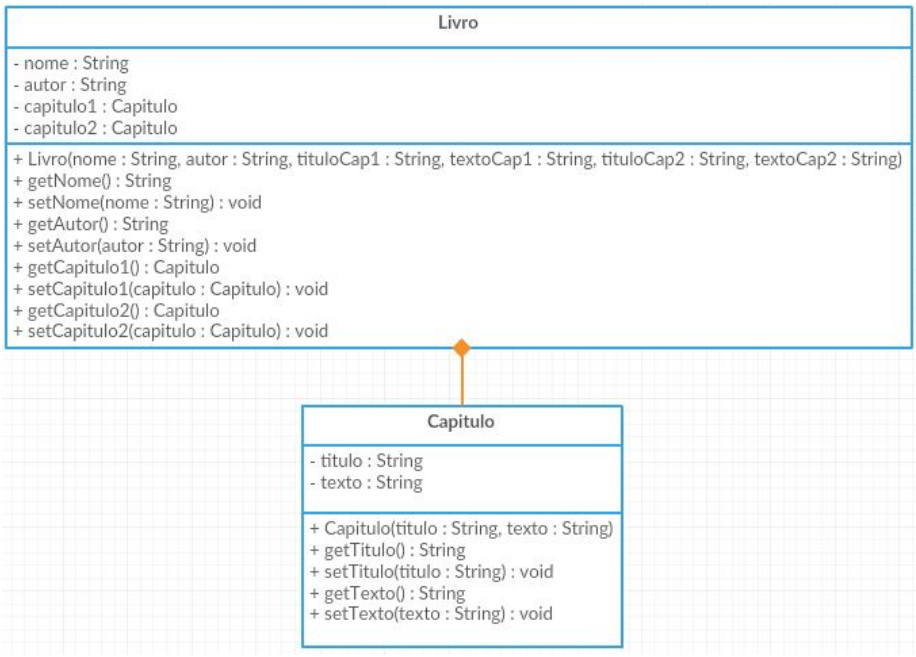

Figura 3.13: Diagrama Livro Capítulo

```
public void setNome(String nome){
    this.nome = nome;
};

public String getAutor(){
    return this.autor;
};

public void setAutor(String autor){
    this.autor = autor;
};

public Capitulo getCapitulo1(){
    return this.capitulo1;
};

public void setCapitulo1(Capitulo capitulo){
    this.capitulo1 = capitulo;
};
```

```
    public Capitulo getCapitulo2(){
        return this.capitulo2;
    };

    public void setCapitulo2(Capitulo capitulo){
        this.capitulo2 = capitulo;
    };

    @Override
    public String toString(){
        return this.getClass().getName() + ": nome =
        ↪  "+this.nome+", autor = "+this.autor+",\ncapitulo 1 =
        ↪  "+this.capitulo1.toString()+",\ncapitulo 2 =
        ↪  "+this.capitulo2.toString();
    }
}
```

Classe Capitulo (código no github):

```
public class Capitulo{
    private String titulo;
    private String texto;

    Capitulo(String titulo, String texto){
        this.setTitulo(titulo);
        this.setTexto(texto);
    };

    public String getTitulo(){
        return this.titulo;
    };

    public void setTitulo(String titulo){
        this.titulo = titulo;
    };

    public String getTexto(){
        return this.texto;
    };

    public void setTexto(String texto){
        this.texto = texto;
    };

    @Override
    public String toString(){
```

```
        return this.getClass().getName() + ": titulo =
        ↪  "+this.titulo+", texto = "+this.texto;
    }
}
```

Classe TestaLivro (código no github):

```
public class TestaLivro {
    public static void main(String args[]){
        Livro livro = new Livro("O guia do mochileiro das
        ↪  galaxias","Douglas Adams","Capitulo 1","Texto super
        ↪  interessante do capitulo 1","Segundo titulo de
        ↪  capitulo","Texto do capitulo 2");
        System.out.println(livro.toString());
    }
}
```

3.11.2 Óculos

Classe Oculos (código no github):

```
public class Oculos{
    private float preco;
    private String cor;
    private Lente lente;

    Oculos(float preco, String cor, String tipoLente, float
    ↪  grauLente){
        this.setPreco(preco);
        this.setCor(cor);
        this.setLente(new Lente(tipoLente, grauLente));
    };

    public void setPreco(float preco){
        this.preco = preco;
    };

    public float getPreco(){
        return this.preco;
    };

    public void setCor(String cor){
        this.cor = cor;
    };

    public String getCor(){
        return this.cor;
```

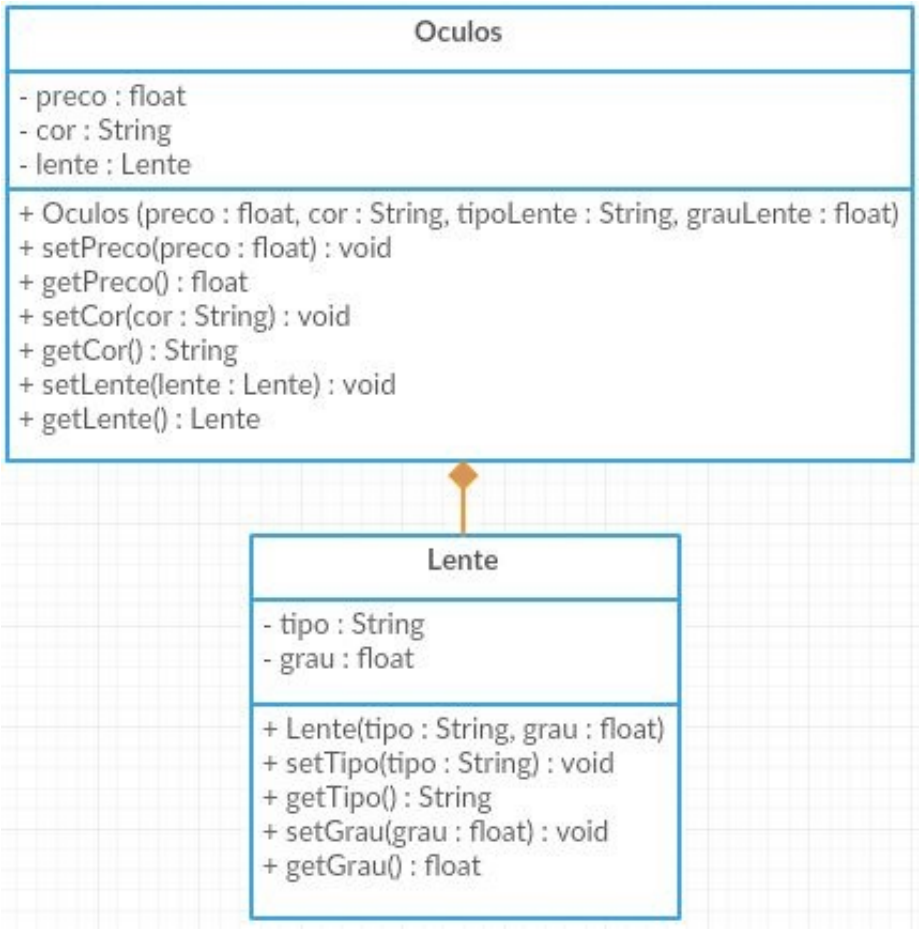

Figura 3.14: Diagrama Óculos Lente

```
};

public void setLente(Lente lente){
    this.lente = lente;
};

public Lente getLente(){
    return this.lente;
};

@Override
public String toString(){
```

```
        return this.getClass().getName() + ": preco = " +
        ↪  this.preco + ", cor = " + this.cor + ",\nlente = " +
        ↪  this.lente.toString();
    };
}
```

Classe Lente (código no github):

```
public class Lente{
    private String tipo;
    private float grau;

    Lente(String tipo, float grau){
        this.setTipo(tipo);
        this.setGrau(grau);
    };

    public void setTipo(String tipo){
        this.tipo = tipo;
    };

    public String getTipo(){
        return this.tipo;
    };

    public void setGrau(float grau){
        this.grau = grau;
    };

    public float getGrau(){
        return this.grau;
    };

    @Override
    public String toString(){
        return this.getClass().getName() + ": tipo = " +
        ↪  this.tipo + ", grau = " + this.grau;
    };
}
```

Classe TestaOculosLente (código no github):

```
public class TestaOculosLente {
    public static void main(String args[]){
        Oculos oculos = new Oculos(100,"preto","descanso",0.5f);
        System.out.println(oculos.toString());
    }
}
```

3.12 Herança

3.12.1 Ponto e Círculo

Implementar as classes Ponto e Circulo e uma classe para testar a utilização dessas classes. Implementar Ponto com atributos private e protected e verificar o acesso dentro da classe filha Circulo.

Classe Ponto (código no github):

```
public class Ponto {
  protected float x;
  protected float y;

  void setPonto(float x, float y) {
    this.x = x;
    this.y = y;
  }

  float getX() {
    return this.x;
  }

  float getY() {
    return this.y;
  }

  Ponto() {
  }

  Ponto(float x, float y) {
    setPonto(x, y);
  }

  void move(float dx, float dy) {
    x += dx;
    y += dy;
  }

  @Override
  public String toString() {
    return this.getClass().getName() + " : x = " + this.x + " y =
    ↪  " + this.y;
  }
}
```

Classe Circulo (código no github):

```
public class Circulo extends Ponto {
  protected double raio;

  Circulo() {
    // aqui existe uma chamada implicita ao construtor de Ponto
    setRaio(0);
  }

  Circulo(double raio, float x, float y) {
    super(x, y);// chamando o construtor em Ponto
    this.raio = raio;
  }

  public double getRaio() {
    return this.raio;
  }

  public void setRaio(double raio) {
    if (raio > 0)
      this.raio = raio;
    else
      this.raio = 0;
  }

  public double area() {
    return Math.PI * this.raio * this.raio;
  }

  @Override
  public String toString() {
    // return "Raio = "+this.raio+ " x:"+this.x + " y: "+this.y;
    return super.toString() + " Raio = " + this.raio;
  }
}
```

Classe TestaPontoCirculo (código no github):

```
public class TestaPontoCirculo {

  public static void main(String[] args) {
    Circulo c = new Circulo(10, 2, 3);

    Circulo c2 = new Circulo();
    System.out.println(c2);
    System.out.println(c);

    Ponto p = new Ponto(4, 5);
```

```
    System.out.println(p.x + " " + p.y);

  }
}
```

3.12.2 Animal

Implementar a seguinte hierarquia:

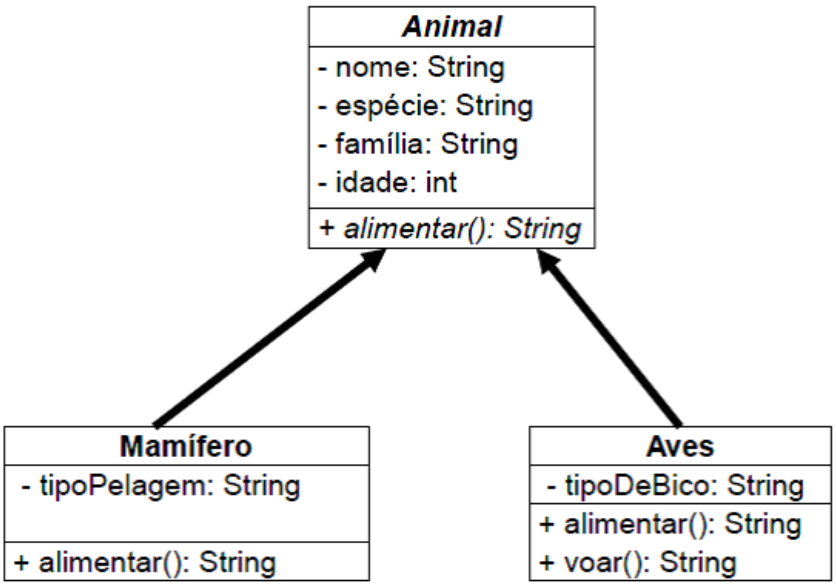

Figura 3.15: Classe Animal, Mamífero e Ave

Classe Animal (código no github):

```
public class Animal {
  protected String nome;
  protected String especie;
  protected String familia;
  protected int idade;

  Animal() {
  }

  Animal(String nome, String especie, String familia,  int idade)
  ↪  {
    this.nome = nome;
```

```
    this.especie = especie;
    this.familia = familia;
    this.idade = idade;
  }

  String alimentar() {
    return "Alimentou na classe Animal";
  }

}
```

Classe Ave (código no github):

```
public class Ave extends Animal {
  protected String tipoBico;

  Ave() {
    this.tipoBico = "";
  }

  Ave(String nome, String especie, String familia, int idade,
  ↪  String bico) {

    super(nome, especie, familia, idade);
    this.tipoBico = bico;
  }

  @Override
  String alimentar() {
    return " Alimentou na classe Ave";
  }

  String voar() {
    return " Classe ave voando";
  }
}
```

Classe Mamifero (código no github):

```
public class Mamifero extends Animal {
  protected String tipoPelagem;

  Mamifero() {
    this.tipoPelagem = "";
  }

  Mamifero(String nome, String especie, String familia, int
  ↪  idade, String pelagem) {
```

```
    super(nome, especie, familia, idade);
    this.tipoPelagem = pelagem;
  }

  @Override
  String alimentar() {
    return " Alimentou na classe Mamifero";
  }
}
```

Classe TestaAnimal (código no github):

```
public class TestaAnimal {
  public static void main(String[] args) {
    Mamifero cachorro = new Mamifero("rex",  "cachorro",
    ↪  "canino", 2, "peludo");

    Ave andorinha = new Ave("and","passaro", "aves", 2, "curto");

    System.out.println(cachorro.alimentar());
    System.out.println(andorinha.voar());
  }

}
```

3.12.3 Conta

Implementar uma classe Conta que deve possuir:

- Atributos: String titular; String numeroConta; String numeroAgencia;float saldo; String status; //positivo ou negativo
- Métodos: Construtor, Saque, AlteraStatus (altera para positivo ou negativo dependendo do saldo), Deposito, toString. Criar um outro método atualizar na classe Conta. Esse método deve incrementar percentualmente o saldo da conta.(this.saldo += this.saldo * percentual;)
- Desenvolva também um método para transferir valores de uma conta para outra. Ex.: transferePara (Conta destino, float valor)
- Crie três classes que serão subclasses de conta: ContaCorrente, ContaPoupanca e ContaInvestimento.
- Essas classes não terão nenhum atributo e deverão ter um método construtor que receba todas as informações dos atributos de conta e deverá invocar o método construtor da superclasse.
- As classes ContaPoupanca e ContaInvestimento deverão sobreescrever o método atualizar que deve aplicar um rendimento diferencial, como segue: this.saldo += this.saldo * 1,005; //Conta Poupança | this.saldo +=

this.saldo * 1,01; //Conta Investimento

- Elabore uma classe TestaConta com um método main() que permita criar instâncias das classes Conta, ContaCorrente, ContaPoupanca e ContaInvestimento. Após isso invoque o método atualizar() para cada instância. Por fim, apresente (imprima) o saldo.

Classe Conta (código no github):

```
public class Conta{
  protected String titular;
  protected String nConta;
  protected String nAgencia;
  protected double saldo;
  protected String status;

  public Conta(String t, String c, String a, double s){
    this.titular = t;
    this.nConta = c;
    this.nAgencia = a;
    deposito(s);
  }
  public void saque(double valor){
    this.saldo-=valor;
    alteraStatus();
  }

  public void alteraStatus(){
      if(this.saldo<0){
      this.status = "negativo";
    }else{
      this.status = "positivo";
    }
  }

  public void deposito(double s){
    this.saldo+=s;
    alteraStatus();
  }
  public void transferePara(Conta destino, double valor){
    destino.deposito(valor);
    saque(valor);
  }
  public void atualizar(){
    this.saldo= this.saldo * 1;
  }
```

```
  public String toString(){
    return "Titular: "+this.titular+"\nConta:
    ↪  "+nConta+"\nAgencia: "+nAgencia+"\nSaldo: "+saldo+" |
    ↪  "+status;
  }

}
```

Classe ContaCorrente (código no github):

```
public class ContaCorrente extends Conta{

  public ContaCorrente(String t, String c, String a, double s){
    super(t,c,a,s);
  }
  public void atualizar(){
    this.saldo= this.saldo * 1;
  }
}
```

Classe ContaPoupanca (código no github):

```
public class ContaPoupanca extends Conta{

  public ContaPoupanca(String t, String c, String a, double s){
    super(t,c,a,s);
  }
  public void atualizar(){
    this.saldo= this.saldo * 1.005;
  }
}
```

Classe ContaInvestimento (código no github):

```
public class ContaInvestimento extends Conta{

  public ContaInvestimento(String t, String c, String a, double
  ↪  s){
    super(t,c,a,s);
  }
  public void atualizar(){
    this.saldo = this.saldo*1.01;
  }
}
```

Classe TestaConta (código no github):

```
public class TestaConta{
    public static void main(String[] args){

        Conta c1 = new Conta("Paulo", "12345","123-1",2000);
        ContaCorrente c2= new ContaCorrente("Claudio", "12345",
        ↪  "123-2", 5000);
        ContaPoupanca c3 = new ContaPoupanca("Caio", "12345",
        ↪  "123-3", 7000);
        ContaInvestimento c4 = new
        ↪  ContaInvestimento("Gustavo","12345","123-4", 1000);

        c1.atualizar();
        c2.atualizar();
        c3.atualizar();
        c4.atualizar();

        System.out.println(c1);
        System.out.println(c2);
        System.out.println(c3);
        System.out.println(c4);

        c1.saque(2000);
        c2.saque(2000);
        c3.saque(2000);
        c4.saque(2000);

        System.out.println(c1);
        System.out.println(c2);
        System.out.println(c3);
        System.out.println(c4);

        c1.deposito(3000);
        c2.deposito(3000);
        c3.deposito(3000);
        c4.deposito(3000);

        System.out.println(c1);
        System.out.println(c2);
        System.out.println(c3);
        System.out.println(c4);

        c1.transferePara(c2, 1000);
        c2.transferePara(c3, 2000);
        c3.transferePara(c4, 3000);
        c4.transferePara(c1, 4000);
```

```
        System.out.println(c1);
        System.out.println(c2);
        System.out.println(c3);
        System.out.println(c4);
    }
}
```

3.12.4 Produto: eletrônico e mobília, parte 1

Uma loja deseja realizar o cadastro de seus produtos de maneira virtual, implemente as classes descritas no diagrama UML para que isto seja possível.

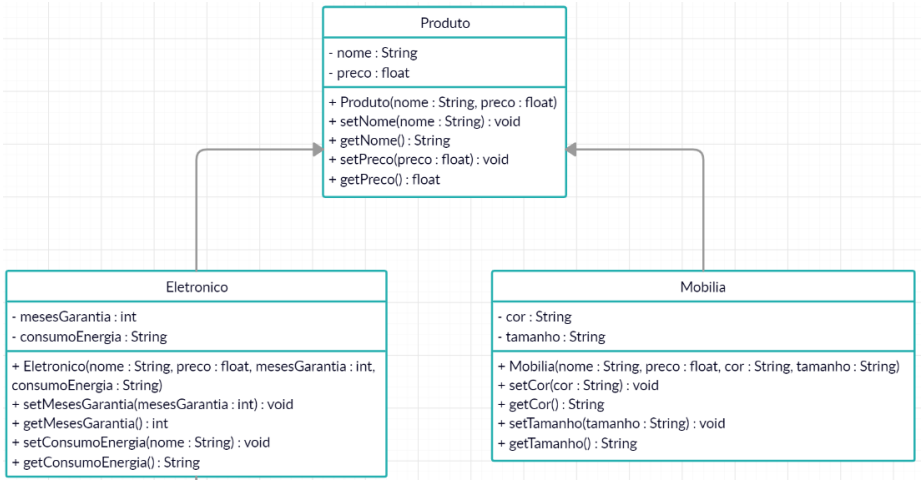

Figura 3.16: Diagrama Produto Eletrônico Mobília

Classe Produto:

```
public class Produto{
    private String nome;
    private float preco;

    Produto(String nome, float preco){
        this.setNome(nome);
        this.setPreco(preco);
    };

    public void setNome(String nome){
        this.nome = nome;
    };
```

```
    public String getNome(){
        return this.nome;
    };

    public void setPreco(float preco){
        this.preco = preco;
    };

    public float getPreco(){
        return this.preco;
    };

    @Override
    public String toString(){
        return this.getClass().getName() + ":" + " nome = " +
        ↪  this.nome + ", preco = " + this.preco;
    };
}
```

Classe Eletronico:

```
public class Eletronico extends Produto{
    private int mesesGarantia;
    private String consumoEnergia;

    Eletronico(String nome, float preco, int mesesGarantia,
    ↪  String consumoEnergia){
        super(nome,preco);
        this.setMesesGarantia(mesesGarantia);
        this.setConsumoEnergia(consumoEnergia);
    };

    public void setMesesGarantia(int mesesGarantia){
        this.mesesGarantia = mesesGarantia;
    };

    public int getMesesGarantia(){
        return this.mesesGarantia;
    };

    public void setConsumoEnergia(String nome){
        this.consumoEnergia = nome;
    };

    public String getConsumoEnergia(){
        return this.consumoEnergia;
    };
```

```
    @Override
    public String toString(){
        return super.toString() + ";" + " mesesGarantia = " +
        ↪  this.mesesGarantia + ", consumoEnergia = " +
        ↪  this.consumoEnergia;
    };
}
```

Classe Mobilia:

```
public class Mobilia extends Produto{
  private String cor;
  private String tamanho;

  Mobilia(String nome, float preco, String cor, String tamanho){
            super(nome,preco);
            this.setCor(cor);
            this.setTamanho(tamanho);
        };

  public void setCor(String cor){
    this.cor = cor;
  };

  public String getCor(){
    return this.cor;
  };

  public void setTamanho(String tamanho){
    this.tamanho = tamanho;
  };

  public String getTamanho(){
    return this.tamanho;
  };

  @Override
  public String toString(){
    return super.toString() + "; " + " cor = " + this.cor + ",
    ↪  tamanho = " + this.tamanho;
  };
}
```

Classe TestaClasses:

```
public class TestaClasses {
```

```
    public static void main (String arg[]){

        Produto p = new Produto("bola",15.5f);
        System.out.println(p.toString());
        Eletronico e = new Eletronico("caixa de
        ↪  som",1799.99f,12,"400 Watts");
        System.out.println(e.toString());
        Mobilia m = new Mobilia("mesa",999.99f,"branca","3m por
        ↪  2m");
        System.out.println(m.toString());
    }
}
```

3.12.5 Produto: eletrônico e mobília, parte 2

Com base no exercício anterior, adicione dois novos tipos de produtos eletrônicos, Televisão e Telefone, de acordo com o diagrama abaixo.

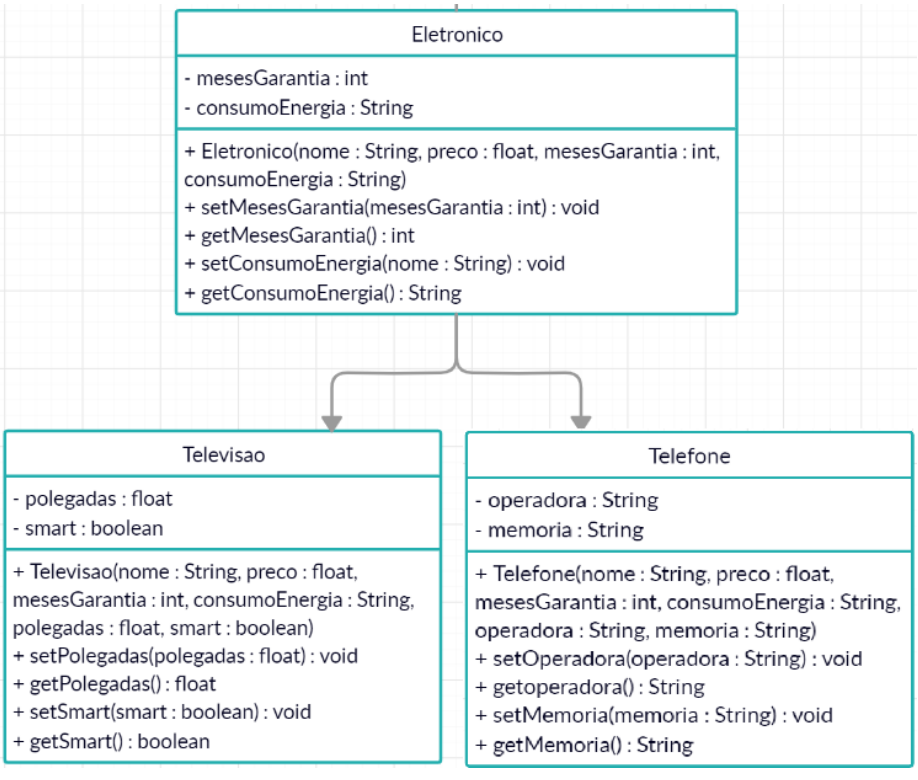

Figura 3.17: Diagrama Eletrônico Telefone Televisão

Classe Produto:

```
public class Produto{
    private String nome;
    private float preco;

    Produto(String nome, float preco){
        this.setNome(nome);
        this.setPreco(preco);
    };

    public void setNome(String nome){
        this.nome = nome;
    };

    public String getNome(){
        return this.nome;
    };

    public void setPreco(float preco){
        this.preco = preco;
    };

    public float getPreco(){
        return this.preco;
    };

    @Override
    public String toString(){
        return this.getClass().getName() + ":" + " nome = " +
        ↪  this.nome + ", preco = " + this.preco;
    };
}
```

Classe Eletronico:

```
public class Eletronico extends Produto{
    private int mesesGarantia;
    private String consumoEnergia;

    Eletronico(String nome, float preco, int mesesGarantia,
    ↪  String consumoEnergia){
        super(nome,preco);
        this.setMesesGarantia(mesesGarantia);
        this.setConsumoEnergia(consumoEnergia);
    };
```

```
    public void setMesesGarantia(int mesesGarantia){
        this.mesesGarantia = mesesGarantia;
    };

    public int getMesesGarantia(){
        return this.mesesGarantia;
    };

    public void setConsumoEnergia(String nome){
        this.consumoEnergia = nome;
    };

    public String getConsumoEnergia(){
        return this.consumoEnergia;
    };

    @Override
    public String toString(){
        return super.toString() + ";" + " mesesGarantia = " +
        ↪  this.mesesGarantia + ", consumoEnergia = " +
        ↪  this.consumoEnergia;
    };
}
```

Classe Mobilia:

```
public class Mobilia extends Produto{
  private String cor;
  private String tamanho;

  Mobilia(String nome, float preco, String cor, String tamanho){
            super(nome,preco);
            this.setCor(cor);
            this.setTamanho(tamanho);
        };

  public void setCor(String cor){
    this.cor = cor;
  };

  public String getCor(){
    return this.cor;
  };

  public void setTamanho(String tamanho){
    this.tamanho = tamanho;
  };
```

```
  public String getTamanho(){
    return this.tamanho;
  };

  @Override
  public String toString(){
    return super.toString() + "; " + " cor = " + this.cor + ",
    ↪  tamanho = " + this.tamanho;
  };
}
```

Classe Telefone:

```
public class Telefone extends Eletronico{
  private String operadora;
  private String memoria;

  Telefone(String nome, float preco, int mesesGarantia, String
  ↪  consumoEnergia, String operadora, String memoria){
          super(nome,preco,mesesGarantia,consumoEnergia);
          this.setOperadora(operadora);
          this.setMemoria(memoria);
      };

  public void setOperadora(String operadora){
    this.operadora = operadora;
  };

  public String getoperadora(){
    return this.operadora;
  };

  public void setMemoria(String memoria){
    this.memoria = memoria;
  };

  public String getMemoria(){
    return this.memoria;
  };

  @Override
  public String toString(){
    return super.toString() + "\n operadora = " + this.operadora
    ↪  + ", memoria = " + this.memoria;
  };
}
```

Classe Televisao:

```
public class Televisao extends Eletronico{
  private float polegadas;
  private boolean smart;

  Televisao(String nome, float preco, int mesesGarantia, String
  ↪  consumoEnergia, float polegadas, boolean smart){
          super(nome,preco,mesesGarantia,consumoEnergia);
          this.setPolegadas(polegadas);
          this.setSmart(smart);
      };

  public void setPolegadas(float polegadas){
    this.polegadas = polegadas;
  };

  public float getPolegadas(){
    return this.polegadas;
  };

  public void setSmart(boolean smart){
    this.smart = smart;
  };

  public boolean getSmart(){
    return this.smart;
  };

  @Override
  public String toString(){
    return super.toString() + "\n polegadas = " + this.polegadas
    ↪  + ", smart = " + this.smart;
  };
}
```

Classe TestaClasses:

```
public class TestaClasses {

    public static void main (String arg[]){

        Produto p = new Produto("bola",15.5f);
        System.out.println(p.toString());
        Eletronico e = new Eletronico("caixa de
        ↪  som",1799.99f,12,"400 Watts");
        System.out.println(e.toString());
```

```
        Mobilia m = new Mobilia("mesa",999.99f,"branca","3m por
        ↪  2m");
        System.out.println(m.toString());
        Telefone telefone = new
        ↪  Telefone("nokia",300,60,"1","vivo","4mb");
        System.out.println(telefone.toString());
        Televisao televisao = new Televisao("lg",3000,24,"30
        ↪  watts",41.5f,true);
        System.out.println(televisao.toString());
    }

}
\end{solution}

\section{Polimorfismo}

\subsection{Funcionário}

%inicio dos slides da aula 8 (Mihael)

Implemente as seguintes subclasses da classe Funcionario
\begin{itemize}
\item TrabalhadorProducao: pago por quantidade de
produção

\begin{itemize}
\item Atributos: quantidade produzida (quantidade) e remuneração
por peça produzida (valorPorPeca)
\end{itemize}
\item TrabalhorHora: pago por horas trabalhadas
\begin{itemize}
\item Atributos: remuneração por hora (valorPorHora) e número de
horas trabalhadas (horas)
\end{itemize}
\end{itemize}

\begin{solution}
Classe Funcionario
↪  \href{https://github.com/cristiancechinel/ExerciciosResolvidosJava/blob/master/Ori
↪  no github)}:
\begin{minted}[tabsize=2,breaklines]{java}
public abstract class Funcionario {
    private String nome;
    private String sobrenome;
```

```
    public abstract double salario();

    Funcionario(){

    }

    Funcionario(String nome, String sobrenome){
      this.setNome(nome);
      this.setSobrenome(sobrenome);
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getSobrenome() {
        return sobrenome;
    }

    public void setSobrenome(String sobrenome) {
        this.sobrenome = sobrenome;
    }

    @Override
    public String toString() {
        return this.getClass().getName() + "nome = " + nome + ",
        ↪  sobrenome = " + sobrenome;
    }
}
```

Classe TrabalhadorProducao (código no github):

```
public class TrabalhadorProducao extends Funcionario {
    private int quantidade;
    private double valorPorPeca;

    TrabalhadorProducao(){

    }

    TrabalhadorProducao(String nome, String sobrenome, int
    ↪  quantidade, double valorPorPeca){
      super(nome,sobrenome);
```

```
      this.setQuantidade(quantidade);
      this.setValorPorPeca(valorPorPeca);
    }
    @Override
    public double salario() {
        return quantidade*valorPorPeca;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    public double getValorPorPeca() {
        return valorPorPeca;
    }

    public void setValorPorPeca(double valorPorPeca) {
        this.valorPorPeca = valorPorPeca;
    }

    @Override
    public String toString() {
    return super.toString() + ", quantidade = " + this.quantidade
    ↪  + " , valorPorPeca = " + this.valorPorPeca;
    }
}
```

Classe TrabalhadorHora (código no github):

```
public class TrabalhadorHora extends Funcionario{
    private double horas;
    private double valorPorHora;

    TrabalhadorHora(){

    }

    TrabalhadorHora(String nome, String sobrenome, double horas,
    ↪  double valorPorHora){
      super(nome,sobrenome);
      this.setHoras(horas);
      this.setValorPorHora(valorPorHora);
    }
```

```
    @Override
    public double salario() {
       return valorPorHora * horas;
    }

    public double getHoras() {
        return horas;
    }

    public void setHoras(double horas) {
        this.horas = horas;
    }

    public double getValorPorHora() {
        return valorPorHora;
    }

    public void setValorPorHora(double valorPorHora) {
        this.valorPorHora = valorPorHora;
    }

    @Override
    public String toString() {
    return super.toString() + ", horas = " + this.horas + " ,
    ↪  valorPorHora = " + this.valorPorHora;
    }
}
```

Classe TestaFuncionario (código no github):

```
public class TestaFuncionario{
    public static void main(String[] args){
       TrabalhadorHora t1 = new
       ↪  TrabalhadorHora("Carlos","Alberto",40,5);
       TrabalhadorProducao t2 = new
       ↪  TrabalhadorProducao("Joao","Silva",20,50);
       System.out.println(t1.toString()+", salario =
       ↪  "+t1.salario());
       System.out.println(t2.toString()+", salario =
       ↪  "+t2.salario());
    }
}
```

3.12.6 Instrumento

Crie uma classe abstrata Instrumento que possua um método público e abstrato chamado tocar(). A partir disto crie:

- Uma subclasse abstrata chamada InstrumentoCorda que herde de Instrumento e que possua um atributo que armazene o número de cordas;
- Uma subclasse que estenda de InstrumentoCorda, por exemplo, Guitarra. Nessa subclasse implemente dois construtores, um padrão (default – sem parâmetros) que inicialize a quantidade de cordas com 6 e um segundo construtor que receberá o número de cordas por parâmetro;
- Na classe Guitarra sobrescreva o método tocar() indicando alguma mensagem;
- Crie uma classe Baixo com as mesmas características da classe Guitarra, incluindo os construtores; No construtor padrão inicialize o atributos de número de cordas com o valor 4;
- Uma classe Viola com as mesmas características das classes anteriores, incluindo os construtores; No construtor padrão inicialize o atributos de número de cordas com o valor 10;
- Implemente uma classe Orquestra que possua um método espetáculo() que recebe como parâmetro um Instrumento e chama o método tocar() desse instrumento.
- Crie uma classe de teste. Nessa classe instancie uma Orquestra e crie um vetor de Instrumento com três posições, cada posição com um tipo de instrumento diferente, Guitarra, Baixo e Viola. Depois faça um laço de repetição para chamar o método espetáculo() da classe Orquestra (para realizar essa atividade é necessário trabalhar com vetores de objetos).

Classe Instrumento (código no github):

```
public abstract class Instrumento {
    public abstract void tocar();
}
```

Classe InstrumentoCorda (código no github):

```
public abstract class InstrumentoCorda extends Instrumento{
    private int numeroCorda;

    public int getNumeroCorda() {
        return numeroCorda;
    }

    public void setNumeroCorda(int numeroCorda) {
```

```
        this.numeroCorda = numeroCorda;
    }
}
```

Classe Viola (código no github):

```
public class Viola extends InstrumentoCorda {

    public Viola() {
        this.setNumeroCorda(10);
    }

    public Viola(int numeroDeCordas) {
        this.setNumeroCorda(numeroDeCordas);
    }

    @Override
    public void tocar() {
        System.out.println("Tocando viola");
    }

}
```

Classe Guitarra (código no github):

```
public class Guitarra extends InstrumentoCorda {

    public Guitarra() {
        this.setNumeroCorda(6);
    }

    public Guitarra(int numeroDeCordas) {
        this.setNumeroCorda(numeroDeCordas);
    }

    @Override
    public void tocar() {
        System.out.println("Tocando guitarra");
    }

}
```

Classe Baixo (código no github):

```
public class Baixo extends InstrumentoCorda {

    public Baixo() {
        this.setNumeroCorda(4);
```

```
    }

    public Baixo(int numeroDeCordas) {
        this.setNumeroCorda(numeroDeCordas);
    }

    @Override
    public void tocar() {
        System.out.println("Tocando baixo");
    }

}
```

Classe Orquestra (código no github):

```
public class Orquestra {
    public void espetaculo(Instrumento i){
        i.tocar();
    }
}
```

Classe TesteInstrumento (código no github):

```
public class TesteInstrumento {
    public static void main(String[] args){
        Orquestra orquestra = new Orquestra();
        Instrumento[] instrumentos = new Instrumento[3];
        instrumentos[0] = new Guitarra();
        instrumentos[1] = new Baixo();
        instrumentos[2] = new Viola();
        for(int i = 0 ; i < instrumentos.length; i++){
            orquestra.espetaculo(instrumentos[i]);
        }
    }
}
```

3.12.7 Forma

Crie uma classe abstrata Forma(), e as classes filhas Circulo() e Quadrado():
Classe Forma (código no github):

```
public abstract class Forma {

    public abstract double getArea();
    public abstract double getPerimetro();
    public abstract String getNome();

}
```

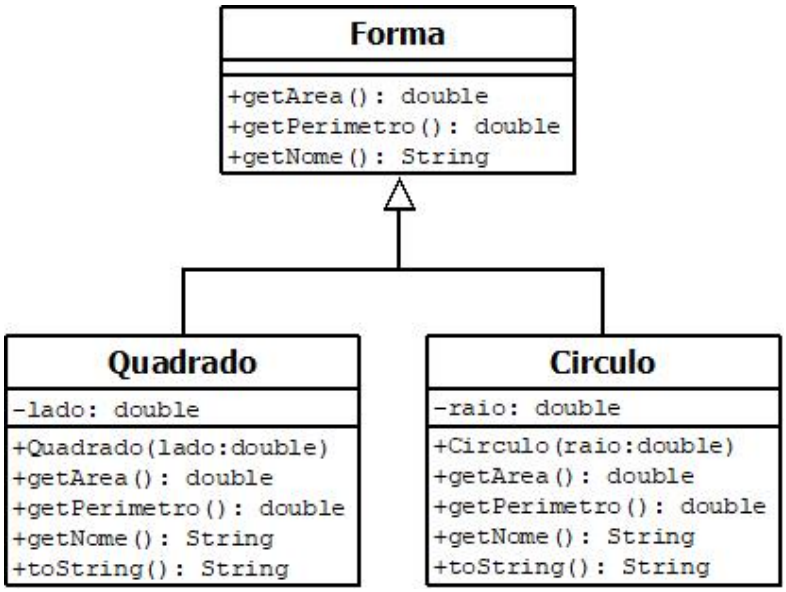

Figura 3.18: Classe Forma, Quadrado e Círculo

Classe Circulo (código no github):

```
import java.lang.Math;

public class Circulo extends Forma{
    private double raio;
    Circulo(double r) {
        this.raio = r;
    }
    @Override
    public double getArea(){
        return Math.PI * this.raio * this.raio;
    }
    @Override
    public double getPerimetro(){
        return Math.PI * 2 * this.raio;
    }
    @Override
    public String getNome(){
        return "Circulo";
    }
    @Override
    public String toString(){
        return this.getClass().getName() + " \n raio = "+ this.raio + " \n area = "+ this.getArea
```

```
    }

}
```

Classe Quadrado (código no github):

```
public class Quadrado extends Forma{
    private double lado;

    public Quadrado(double l){
        this.lado = l;
    }
    @Override
    public double getArea(){
        return this.lado * this.lado;
    }
    @Override
    public double getPerimetro(){
        return this.lado * 4;
    }
    @Override
    public String getNome(){
        return "Quadrado";
    }
    @Override
    public String toString(){
        return this.getClass().getName() + "\n lado = "+ this.lado + "\n area = " + th
    }

}
```

Classe TestaForma (código no github):

```
public class TestaForma{
    public static void main(String[] args){
       Circulo c = new Circulo(3);
       System.out.println(c);
       Quadrado q = new Quadrado(2);
       System.out.println(q);

    }
}
```

Capítulo 4

Estrutura de dados

4.1 Lista de nomes

Escreva um programa que contenha uma lista com 5 nomes précadastrados. O programa deverá dar ao usuário a opção de excluir um único nome da lista, com valores entre 1 e 5, exemplo:

Qual dos nomes abaixo você deseja excluir da lista?
1. Maria da Silva
2. João de Souza
3. James Bond
4. Tony Stark
5. Jack Sparrow

Feita a escolha do usuário pelo número correspondente ao nome, o programa deverá mostrar na tela os quatro nomes que restaram cadastrados na lista.

Classe ListaNomes (código no github):

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class ListaNomes {

    public static List<String> nomes = new ArrayList();
    public static final Scanner in = new Scanner(System.in);

    public static void main(String args[]) {
        nomes.add("Maria da Silva");
        nomes.add("Joao de Souza");
        nomes.add("James Bond");
        nomes.add("Tony Stark");
```

```
        nomes.add("Jack Sparrow");
        int opcao;
        while (nomes.size() > 0) {
          System.out.println("--------------------------------");
          System.out.println("Qual dos nomes abaixo voce gostaria
          ↪  de excluir da lista?");
          for (int i = 0; i < nomes.size(); i++) {
            System.out.println((i + 1) + ". " + nomes.get(i));
          }
          System.out.println("--------------------------------");
          opcao = in.nextInt();
          if (opcao < 1 || opcao > nomes.size()) {
            System.out.println("Opcao invalida");
          } else {
            nomes.remove(opcao-1);

              if (nomes.isEmpty()) {
                System.out.println("Lista vazia");
              }else {
                System.out.println("Nomes restantes: ");
                for(String nome : nomes){
                  System.out.println(nome);
                }
              }
          }
        }
    }
}
```

4.2 Lista de números inteiros

Escreva um programa que permita ao usuário cadastrar 6 números inteiros DIFERENTES em uma lista. O programa não poderá permitir que números já cadastrados sejam inseridos na lista. Utilize o método contains():
– colecao.contains(x)

Classe ListaInteiros (código no github):

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class ListaInteiros {
  public static List<Integer> numeros = new ArrayList();
    public static final Scanner in = new Scanner(System.in);
```

```
    public static void main(String args[]) {
        int opcao;
        int loop = 1;
        while (loop == 1) {
          System.out.println("-------------------------------");
          System.out.println("Digite o numero da opcao que deseja
          ↪  realizar:\n 1. sair \n 2. cadastrar \n 3. listar");
          opcao = in.nextInt();
          switch(opcao) {
            case 1:
              loop = 0;
              break;
            case 2:
              System.out.print("Digite o numero a ser inserido:
              ↪  ");
                  int num = in.nextInt();
                  if(numeros.contains(num)) {
                      System.out.println("Numero ja existe na
                      ↪  lista");
                  }else {
                    numeros.add(num);
                  }
              break;
            case 3:
              if (numeros.isEmpty()) {
                System.out.println("Lista vazia");
                }else {
                for(int i : numeros){
                          System.out.print(i + "\n");
                    }
                }
              break;
            default:
              System.out.println("Opcao invalida.");
              break;
          }
        }
    }
}
```

4.3 Lista de livros

Desenvolva uma classe Livro com os seguintes atributos:
– private String titulo;

– private int ano;
– private String autor;
Desenvolva uma outra classe Biblioteca com as seguintes características:
– lista de livros (ArrayList)
– método de inserção de livros
– impressão dos livros da biblioteca
– busca de um livro pelo ano
Desenvolva uma classe para testar a biblioteca

Classe Biblioteca (código no github):

```
import java.util.ArrayList;
import java.util.List;

public class Biblioteca {

  private List<Livro> listaLivros;

    public Biblioteca(){
        listaLivros = new ArrayList<Livro>();
    }

    public void adicionaLivro(Livro novo){
        listaLivros.add((novo));
    }

    public void listagemLivros(){
        for (Livro item: listaLivros){
            System.out.println(item.getAno());
            System.out.println(item.getTitulo());
        }
    }

    public Livro buscaPorAno(int ano){
        for (Livro item: listaLivros){
            if (item.getAno() == ano){
                return item;
            }
        }
        return null;
    }
}
```

Classe Livro (código no github):

```
public class Livro {
  private String titulo;
```

```
    private int ano;

    void setTitulo(String t){
        this.titulo = t;
    }
    void setAno (int a){
        this.ano = a;
    }
    public String getTitulo(){
        return this.titulo;
    }
    public int getAno(){
        return this.ano;
    }
    public Livro(String t, int a){
        this.titulo = t;
        this.ano = a;
    }

    @Override
    public String toString(){
        return this.titulo + " "+ this.ano;

    }
}
```

Classe TestaLivros (código no github):

```
public class TestaLivros{

    public static void main(String args[]){
        Biblioteca bib = new Biblioteca();

        Livro liv = new Livro("Teste", 2017);
        Livro liv2 = new Livro("Novo", 2000);
        //adicionar os livros na listalivros
        bib.adicionaLivro(liv);
        bib.adicionaLivro(liv2);

        //listar os livros

        bib.listagemLivros();

        Livro retorno = bib.buscaPorAno(2017);
        System.out.println(retorno);

    }
```

```
}
```

4.4 Lista de compras

Classe ListaCompras (código no github):

```
import java.util.*;

public class ListaCompras {

    public static void main(String[] args){

        ArrayList<String> lista = new ArrayList<>();
        int opcao;
        Scanner entrada = new Scanner(System.in);
        do {
            System.out.println("digite opcao");
            System.out.println("1 - cadastrar comida");
            System.out.println("2 - imprimir lista");
            System.out.println("3 - remover comida");
            System.out.println("4 - sair");
            opcao = Integer.parseInt(entrada.nextLine());

            switch (opcao){
                case 1: System.out.println("digite comida");
                        String comida = entrada.nextLine();
                        lista.add(0, comida);
                        break;

                case 2: System.out.println("Lista de compras");
                        for (String s: lista)
                            System.out.println(s);
                        break;

                case 3: System.out.println("remover");
                        if (!lista.isEmpty())
                            lista.remove(0);
                        break;

                case 4: System.out.println("sair");
                        break;

                default: System.out.println("opcao invalida");
            }
        } while (opcao!=4);
```

```
    }
}
```

Capítulo 5

Arquivos

5.1 Salvar nomes

Crie uma classe que leia nomes de pessoas a partir do usuário e grave os dados lidos em um arquivo, repita essa operação até o usuário digitar SAIR. Depois leia o arquivo escrito.

Classe ArquivoNome (código no github):

```
import java.io.*;
import java.util.Scanner;

public class ArquivoNome {

    public static void main(String[] args) throws IOException {
        Scanner in = new Scanner(System.in);
        final String caminho = "nomes.txt";
        File arquivo = new File(caminho);
        FileWriter printer = new FileWriter(arquivo, true);
        PrintWriter pw = new PrintWriter(printer);
        String nome;
        boolean continua;
        continua = true;
        do {
            System.out.println("Digite um nome ou digite sair
            ↪  para finalizar: ");
            nome = in.nextLine();
            if (nome.equalsIgnoreCase("SAIR")) {
                continua = false;
            } else {
                pw.println(nome);
            }
```

```
        } while (continua);
        pw.close();
        System.out.println("Lendo o arquivo: ");
        FileReader reader = new FileReader(caminho);
        BufferedReader leitor = new BufferedReader(reader);
        String line = null;
        while((line = leitor.readLine()) != null){
            System.out.println(line);
        }
    }
}
```

5.2 Copiar dados

Faça um programa que copie em um novo arquivo os nomes armazenados no arquivo criado no exercício anterior.

Classe ArquivoCopia (código no github):

```
import java.io.*;

public class ArquivoCopia {

    public static void main(String[] args) throws IOException {

        final String caminho = "nomes.txt";
        final String copia = "nomesCopia.txt";
        File arquivo = new File(copia);
        FileWriter printer = new FileWriter(arquivo, true);
        PrintWriter pw = new PrintWriter(printer);

        FileReader reader = new FileReader(caminho);
        BufferedReader leitor = new BufferedReader(reader);
        String line = null;
        while((line = leitor.readLine()) != null){
            pw.println(line);
        }
        pw.close();

    }
}
```

5.3 Verificar nome entre arquivos

Faça um programa que receba um nome através do usuário e verifique se está no arquivo criado no exercício 1. Caso o nome se encontra no arquivo mostrar uma mensagem na tela "Nome já cadastrado". Se não, armazenar no arquivo.

Classe ArquivoVerifica (código no github):

```
import java.io.*;
import java.util.Scanner;

public class ArquivoVerifica {

    public static void main(String[] args) throws IOException {

        final String caminho = "nomesCopia.txt";

        Scanner in = new Scanner(System.in);

        System.out.println("Digite o nome a ser pesquisado:");
        String nome = in.nextLine();
        FileReader reader = new FileReader(caminho);
        BufferedReader leitor = new BufferedReader(reader);
        String line = null;
        while((line = leitor.readLine()) != null){
            if(line.equalsIgnoreCase(nome)){
                System.out.println("Nome já cadastrado");
                return;
            }
        }
        leitor.close();
        reader.close();

        File arquivo = new File(caminho);
        FileWriter printer = new FileWriter(arquivo, true);
        PrintWriter pw = new PrintWriter(printer);
        pw.println(nome);
        pw.close();

    }
}
```

5.4 Verificar vários nomes entre arquivos

Altere o programa anterior e faça que o mesmo execute várias vezes, até que o usuário digite SAIR.

Classe ArquivoVerifica (código no github):

```
import java.io.*;
import java.util.Scanner;

public class ArquivoVerifica {

    public static void main(String[] args) throws IOException {

        final String caminho = "nomesCopia.txt";

        Scanner in = new Scanner(System.in);
        String nome = "";

        while (!nome.equalsIgnoreCase("sair")) {
            boolean encontrado = false;
            System.out.println("Digite o nome a ser
            ↪  pesquisado:");
            nome = in.nextLine();
            FileReader reader = new FileReader(caminho);
            BufferedReader leitor = new BufferedReader(reader);
            String line = null;
            while ((line = leitor.readLine()) != null) {
                if (line.equalsIgnoreCase(nome)) {
                    System.out.println("Nome já cadastrado");
                    encontrado = true;
                    break;
                }
            }
            leitor.close();
            reader.close();
            if (!encontrado) {
                File arquivo = new File(caminho);
                FileWriter printer = new FileWriter(arquivo,
                ↪  true);
                PrintWriter pw = new PrintWriter(printer);
                pw.println(nome);
                pw.close();
            }
        }
    }
}
```

Capítulo 6

Exceções

6.1 Numeros

6.1.1 Inteiro

Classe InteiroErro (código no github):

```
import java.util.Scanner;
public class InteiroErro {

    public static void main(String[] args){
        Scanner entrada = new Scanner(System.in);
        System.out.println("Informe o primeiro valor");
        int x = entrada.nextInt();
        System.out.println("Informe o segundo valor");
        int y = entrada.nextInt();
        int r = x/y;
        System.out.println("resultado = "+ r);
    }
}
```

Classe Inteiro (código no github):

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class Inteiro {

    public static void main(String[] args){
        Scanner entrada = new Scanner(System.in);
        System.out.println("Informe o primeiro valor");
        try{
            int x = entrada.nextInt();
```

```
            System.out.println("Informe o segundo valor");
            int y = entrada.nextInt();
            int r = x/y;
            System.out.println("resultado = "+ r);
        }
        catch (InputMismatchException e){
            System.out.println("valor digitado deve ser inteiro
            ↪  "+ e.getMessage());
        }
        catch (ArithmeticException e){
            System.out.println("divisao por zero "+ e);
        }
    }
}
```

6.1.2 Entrada de Inteiro

Classe EntradaInteiro (código no github):

```
import java.io.*;

public class EntradaInteiro {

    public static void main(String[] args){
        int v1 = obtemNumero();
        int v2 = obtemNumero();

        System.out.println("v1 "+ v1);
        System.out.println("v2 "+ v2);
    }

    static int obtemNumero() {
        BufferedReader bw = new BufferedReader(new
        ↪  InputStreamReader(System.in));
        System.out.println("Digite um numero");
        int x;
        try {
            x = Integer.parseInt(bw.readLine());
            return x;
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
            return 0;
        }
    }
}
```

6.1.3 Loop de entrada de inteiros

Classe InteiroLoop (código no github):

```
import java.util.*;
public class InteiroLoop {

    public static void main(String[] args){

        boolean outravez = true;
        Scanner entrada = new Scanner(System.in);
        int numero = 0;
        while (outravez == true) {
            System.out.println("Digite um numero inteiro");
            try {
                numero = Integer.parseInt(entrada.nextLine());
                outravez = false;
            }
            catch (NumberFormatException e){
                System.out.println(e);
            }
        }
    }
}
```

6.1.4 Numeros positivos

Classe Positivo (código no github):

```
import java.util.*;
public class Positivo {
    public static void main(String[] args){

        System.out.println("digite um numero positivo");
        Scanner entrada = new Scanner(System.in);

        int x = entrada.nextInt();
        if (x < 0) {
            throw new ArithmeticException("valor negativo");
        }

    }
}
```

Classe PositivoTryCatch (código no github):

```
import java.util.*;
public class PositivoTryCatch {
```

```
    public static void main(String[] args){

        System.out.println("digite um numero positivo");
        Scanner entrada = new Scanner(System.in);
        try {
            int x = entrada.nextInt();
            if (x < 0) throw new ArithmeticException("valor
            ↪  negativo");
        }
        catch (ArithmeticException e) {
            System.out.println(e);
        }
    }
}
```

6.2 Posição de Vetor

Classe PosicaoVetorErro (código no github):

```
public class PosicaoVetorErro {

    public static void main(String args[]){
        int a[] = new int[2];
         System.out.println("Acessando a posicao 3 :" + a[2]);
         System.out.println("Termino do programa");
  }

}
```

Classe PosicaoVetor (código no github):

```
public class PosicaoVetor {

  public static void main(String args[]){
    try{
      int a[] = new int[2];
      System.out.println("Acessando a posicao 3 :" + a[2]);
    }
    catch(ArrayIndexOutOfBoundsException e){
      System.out.println("Acesso a posicao inexistente:" + e);
    }
    finally {
      System.out.println("Termino do programa");
    }
  }
}
```

6.3 Execução de método

Classe MetodoErro (código no github):

```
public class MetodoErro {

    public static void metodo1(){
        System.out.println("inicio do metodo 2");
        int a[] = new int[10];
        for (int i = 0; i < 15; i++){
            a[i] = i;
            System.out.println(i);
        }
        System.out.println("fim do metodo 2");

    }

    public static void main(String[] args){
        System.out.println("inicio do principal");
        metodo1();
    }
}
```

Classe Metodo (código no github):

```
public class Metodo {

    public static void metodo1(){
        System.out.println("inicio do metodo 2");
        int a[] = new int[10];
        for (int i = 0; i < 15; i++){
            a[i] = i;
            System.out.println(i);
        }
        System.out.println("fim do metodo 2");

    }

    public static void main(String[] args){
        System.out.println("inicio do principal");
        try{
            metodo1();
        }
        catch(ArrayIndexOutOfBoundsException e){
            System.out.println("acesso a posicao inexistente");
        }
    }
```

```
}
```

Classe Metodo2 (código no github):

```
public class Metodo2 {

    public static void metodo1(){
        System.out.println("inicio do metodo 1");
        int a[] = new int[10];
        try{
            for (int i = 0; i < 15; i++){
                a[i] = i;
                System.out.println(i);
            }
        }
        catch (Exception e){
            System.out.println("posicao inexistente");
        }
        System.out.println("fim do metodo 1");

    }

    public static void main(String[] args){
        System.out.println("inicio do principal");
        metodo1();
    }
}
```

6.4 Arquivo

Classe LerArquivo (código no github):

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class LerArquivo {
    public static void main(String[] args) {
      String caminho = "teste.txt";
      String linha = null;
      try{
         FileReader fr = new FileReader(caminho);
         BufferedReader br = new BufferedReader(fr);
         while((linha = br.readLine()) != null){
```

```
            System.out.println(linha);
         }
         br.close();
      }
      catch(IOException e){
         System.out.println("Erro arquivo " + caminho + " nao
          ↪  encontrado!");
      }

   }
}
```

Classe LerArquivoFinally (código no github):

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class LerArquivoFinally {
    public static void main(String[] args) throws IOException {
      String caminho = "nomes1.txt";
      String linha = null;
      FileReader fr = null;
      BufferedReader br = null;
      try{
         fr = new FileReader(caminho);
         br = new BufferedReader(fr);
         while((linha = br.readLine()) != null){
            System.out.println(linha);
         }
      }
      catch(FileNotFoundException e){
         System.out.println("Erro arquivo " + caminho + " nao
          ↪  encontrado!");
      }
      catch(IOException e){
          System.out.println(e);
      }
      finally {
            if (br != null) br.close();
            if (fr != null) fr.close();
            System.out.println("fechou br");
            System.out.println("fechou fr");

      }
    }
```

```
}
```