UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO - UFRRJ - SISTEMAS DE INFORMAÇÃO - LINGUAGEM DE PROGRAMAÇÃO III

HELON DE FRANÇA MEDEIROS, PABLO DE OLIVEIRA ARAUJO XAVIER E VANESSA SANTOS DE ANDRADE

TR1 - TRABALHO 1 DE AVALIAÇÃO - RELATÓRIO

Seropédica - RJ
2022

RESOLUÇÃO DOS PROBLEMAS

PROBLEMA 1

RELATÓRIO

PROBLEMA 1

Para fazer a questão 1, foi feita uma revisão sobre Herança utilizando o material fornecido pelo professor e através de pesquisa na internet. Houve dúvida também sobre o uso dos getters e setters, por isso pesquisamos também na internet explicações sobre esses métodos. Desta forma, o grupo entendeu qual seria a estrutura necessária para a construção do código.

O código da classe mãe Navio foi feito sem dificuldades. Foram utilizados dois atributos: nome e ano do tipo String. Também foram criados os respectivos métodos getters e setters. Por último, escrevemos o método toString, que imprime o nome do Navio e ano de fabricação.

No código da classe filha Cruzeiro, tivemos dificuldade para entender como utilizar os atributos presentes na classe mãe. Depois de consultarmos alguns materiais para pesquisa, como o Livro e o site Stack Overflow, concluímos que era necessário usar a palavra super para chamar o construtor da classe mãe. Desta forma, fizemos os getters e setters para o atributo referente ao número máximo de passageiros. E o método toString para retornar o nome do cruzeiro e o máximo de passageiros.

O código da classe filha Cargueiro foi elaborado sem dificuldades, já que seguimos o mesmo raciocínio da classe Cruzeiro. Desta forma, vimos que a diferença seria apenas no atributo, que demos o nome de capacidade (inteiro). E no método toString, onde era necessário retornar o nome do cargueiro e sua capacidade.

Por último, foi criada então a classe TesteNavio. O grupo também não encontrou dificuldades para a criação dessa classe. Nesse código, instanciamos então os objetos das classes navio, cruzeiro e Cargueiro. Tivemos dúvida se deveríamos armazenar os objetos em um vetor ou em ArrayList. Fizemos das duas formas e percebemos que funcionaria das duas maneiras. As duas formas de implementação estão descritas no código. Porém, uma está como comentário.

PROBLEMA 2

Nessa questão do trabalho tivemos algumas dificuldades na parte de analisarmos a classe main VotacaoTeste, nas outras duas classes Candidato e ContagemVotos foi mais fácil de entender, pois foi apenas inserido atributos e métodos para as classes. Agora quando pegamos a main para analisarmos, encontramos algumas ferramentas que ainda não tínhamos utilizados em outros códigos, como por exemplo, as bibliotecas java.io.FileReader, java.io.FileWriter e java.io.PrintWriter. Outra coisa foi a forma de manipular um documento, tivemos que observar bem cada linha de código para entendermos exatamente como era feita a manipulação.

Também ficamos com algumas dúvidas sobre o diagrama UML, mas com ajuda da documentação passada pelo professor e com a análise do código, observamos que só precisaria criar o diagrama para as classe candidato e ContagemVotos, sendo assim fizemos o uso da ferramenta creately e construímos o diagrama.

Da mesma forma foi na questão 4 da 2, já tínhamos observado bastante o código, então na hora de detalharmos como era reconhecido o possível empate dos candidatos mais votados, encontramos uma certa facilidade para fazer o detalhamento.

Na criação do novo método na classe main, na qual foi pedido para encontrar o candidato menos votado, não tivemos dificuldades, pegamos a base do código que era utilizado no outro método, que encontrava o candidato mais votado, e só fizemos algumas substituições.

Quando começamos a fazer essa questão, encontramos essas dificuldades que já foram apontadas, mas após ficarmos dividindo o conhecimento, e analisando bastante o código, conseguimos entender, e absorver bastante em conhecimento.

1) **Analise e estude todos os detalhes das seguintes classes**

- A classe “Candidato” define as variáveis de instância para o nome do candidato, a quantidade de votos do candidato e define um construtor para a classe.

- A classe “ContagemVotos” define as variáveis de instância para armazenar votos válidos e inválidos e define um construtor para a classe.

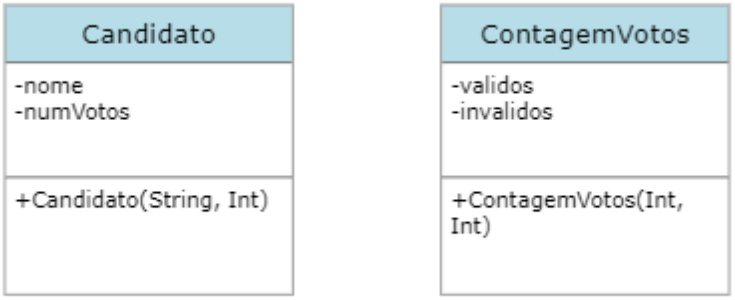
- A classe VotacaoTeste é a classe testadora de candidato e contagemVotos. Nessa classe, foram atribuidos objetos para essas classes.

- O método processarVotos faz parte da classe main. Esse método armazena os atributos da classe ContagemVotos. Assim, irá armazenar a quantidade de candidatos válidos e inválidos.

- O método imprimirResultados também faz parte da classe main. Esse método mostra as informações relativas ao números de votantes, número de votos válidos, inválidos e número total de votos. Também exibe o candidato mais votado e uma lista com todos os candidatos e o número de votos de cada um.

- O método encontrarMaisVotado verifica qual foi o candidato mais votado ou possível empate.

2) Faça a representação das classes usando diagramas UML.



3) Descreva com detalhes tudo o que consiga em relação aos objetos de tipo Candidato. Como são criados? como são organizados/estocados?

Basicamente os objetos do tipo **Candidato** são definidos através de um vetor de tamanho (7+1), onde o atributo **nome** é definido através de um arquivo “votos.txt” em que estão presentes os nomes dos respectivos candidatos. E no método ***processarvotos***, o arquivo é acessado novamente, definindo o atributo **numVotos.**

4) Selecione o segmento de código onde é reconhecido o possível empate dos candidatos mais votados. Explique com detalhes

Se analisarmos o código, veremos que esse método vai encontrar os candidatos mais votados, no método é passado como parâmetro a lista.txt , onde temos os votos de cada pessoa, e também é passado dois valores, o primeiro e último, que recebe o valor um como maior. Depois, é criado uma variável maisVotado que vai atribuir esse primeiro valor. Logo depois é criado um laço que vai percorrer até o MaxCandidatos, e dentro do laço vai existe uma condição, se o número de votos do candidato for maior que o número de votos do mais votado, mais votado vai ser igual ao candidato, assim vai se repetir até percorrer todos os votos e retornar o mais votado.

Problema 3:

Relatório:

Primeiro foi necessário identificar quais seriam os atributos da classe Livro. O grupo concluiu que seria Titulo e Autor (Strings). Também foi observado que seria necessário criar os métodos setTitulo e setAutor para possibilitar a definição ou atualização das variáveis de instância Título e Autor na classe LivroTeste. Depois, vimos que também era necessária a criação dos métodos getTitulo e getAutor para retornar os valores das variáveis e imprimir na tela do usuário. Dessa forma, não haveria erro nas linhas:

“ System.out.println("Titulo : " + livro1.getTitulo()); “

“System.out.println("Autor : " + livro2.getAutor());”

Após verificar essas informações fez-se então o código LivroTeste. O grupo não encontrou dificuldades nem discordâncias para fazer o código.

Problema 4:

Relatório:

O grupo analisou o problema e conseguiu chegar em três tipos de soluções. Primeiramente, definimos a nossa classe **Retangulo** com duas variáveis de instância(largura e comprimento) e os seus respectivos “***gets***” e “***sets****”*. Criamos também, os métodos **perimetro** e **area**, em que aplicamos respectivamente as fórmulas para calcular o perímetro do retângulo e a área do retângulo.

Na nossa classe teste, encontramos três soluções para o mesmo problema. Na primeira solução, definimos duas variáveis (**retangulo1** e **retangulo2**) usando o tipo classe **Retangulo,** logo em seguida “***settamos***” os valores das variáveis

através de um scanner, onde o usuário pode digitar os valores de largura e comprimento para realizar os cálculos definidos pelo problema. Na segunda solução, substituímos as duas variáveis por um vetor do tipo classe **Retangulo**, de tamanho dois, e usamos a mesma lógica da primeira solução. Na terceira solução, "***settamos"*** os valores diretamente no código para as variáveis **retangulo1** e **retangulo2.**

No geral, o grupo não apresentou dificuldades em solucionar o problema proposto. Porém, revisamos alguns conceitos de encapsulamento.

Anexo dos códigos

Link dos códigos:
https://github.com/Pablonilo429/Exercicios-LP3-SI-UFRRJ/tree/main/PrimeiroTrabalhoLP3/src

Problema 1:

Classe Navio:

```
package Exercicio1;

public class Navio {
  public String nome;    //Atributo relacionado ao nome do navio
  private String ano;     //Atributo relacionado ao ano de fabricacao do navio

  public Navio(String nome, String ano) {
     this.nome = nome;
     this.ano = ano;
  }

  public String getNome() {
     return nome;
  }

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

  public String getAno() {
     return ano;
  }

  public void setAno(String ano) {
```

```
        this.ano = ano;
    }

    public String toString() {
        return "Nome do navio:"+ getNome()+"\n"+"Ano de fabricacao:"+getAno();
    }
}
```

Classe Navio:

```
package Exercicio1;

public class Cruzeiro extends Navio {
    private int maxpassageiros; //Atributo referente ao numero maximo de passageiros
    public Cruzeiro(String nome, String ano, int maxpassageiros) {
        super(nome, ano);
        this.maxpassageiros = maxpassageiros;
    }

    public int getMaxpassageiros() {
        return maxpassageiros;
    }

    public void setMaxpassageiros(int maxpassageiros) {
        this.maxpassageiros = maxpassageiros;
    }
    public String toString() {
        return "Nome do cruzeiro: "+ getNome()+"\n"+"Maximo de passageiros: " +
getMaxpassageiros();
    }
}
```

Classe Cargueiro:

```
package Exercicio1;

public class Cargueiro extends Navio {
    int capacidade; //Capacidade em toneladas

    public Cargueiro(String nome, String ano, int capacidade) {
        super(nome, ano);
        this.capacidade = capacidade;
    }

    public int getCapacidade() {
```

```
        return capacidade;
    }

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

    public String toString(){
        return "Nome do cargueiro: "+getNome()+"\n"+"Capaciadade do cargueiro:
"+getCapacidade()+"t";
    }
}
```

Classe teste:

```
package Exercicio1;

import java.util.ArrayList;
public class TesteNavio {
    public static void main(String[] args){
        Navio[] embarcacoes = new Navio[3];

        embarcacoes[0] = new Navio("Mestre Gaivota", "2003");
        embarcacoes[1] = new Cruzeiro("MSC","2015", 400);
        embarcacoes[2] = new Cargueiro("Maersk Line", "2010", 50000);

       for(int i = 0; i < 3; i++){
           System.out.println(embarcacoes[i]); //Percorre o vetor e printa os
determinados atributos
       }

       //Duas formas para o mesmo problema

       ArrayList<Navio> lista = new ArrayList<>();

      //NAVIO

      Navio n1 = new Navio("Mestre Gaivota", "2003");
      lista.add(n1);

      //CRUZEIRO

      Cruzeiro c1 = new Cruzeiro("MSC","2015", 400);
      lista.add(c1);
```

```
    //CARGUEIRO

    Cargueiro CA1 = new Cargueiro("Maersk Line", "2010", 50000);
    lista.add(CA1);

    //IMPRIMINDO NA TELA...

    System.out.print("\n" + lista.toString()); //Sairá no formato de lista


  }
}
```

Problema 2 questão 4:

```
public static int encontrarMaisVotado(Candidato[] lista, int pri, int ult) {
  int maisVotado = pri;
  for (int candidato = pri + 1; candidato <= ult; candidato++) {
      if (lista[candidato].numVotos > lista[maisVotado].numVotos) {
          maisVotado = candidato;
      }
  }
  return maisVotado;
}
```

Problema 2 questão 5:

```
public static int encontrarMenosVotado(Candidato[] lista, int pri, int ult) {
      int menosVotado = pri;
      for (int candidato = pri + 1; candidato <= ult; candidato++) {
          if (lista[candidato].numVotos < lista[menosVotado].numVotos) {
              menosVotado = candidato;
          }
      }
      return menosVotado;
  }
}
```

Problema 3:

Classe teste:

```
package Exercicio3;
```

```
public class LivroTeste {
   public static void main(String[] args) {
      Livro livro1 = new Livro();
      livro1.setTitulo("Tenda dos Milagres");
      livro1.setAutor("Jorge Amado");

      Livro livro2 = new Livro();
      livro2.setTitulo("Dom Casmurro");
      livro2.setAutor("Machado de Assis");

      System.out.println("Titulo : " + livro1.getTitulo());
      System.out.println("Autor : " + livro2.getAutor());
   }
}
```

Problema 4:

Questão 1:

```
package Exercicio4;

public class Retangulo {
   private double comprimento;
   private double largura;
   public double getComprimento() {
      return comprimento;
   }
   public void setComprimento(double comprimento) {
      this.comprimento = comprimento;
   }
   public double getLargura() {
      return largura;
   }
   public void setLargura(double largura) {
      this.largura = largura;
   }
   public double perimetro(){
      return (this.largura+this.comprimento)*2;
   }
   public double area(){
      return (this.largura*this.comprimento);
   }
}
```

Questão 2:

```
package Exercicio4;

import java.util.Scanner;

public class TesteRetangulo {
   public static void main(String[] args) {
      Retangulo[] retangulo = new Retangulo[2];
      Scanner input = new Scanner(System.in);
      /*Opcao com mais linhas de codigo
      Retangulo retangulo1 = new Retangulo;
      Retangulo retangulo2 = new Retangulo;
      System.out.printf("Entre com o comprimento do retangulo 1\n");
      retangulo1.setComprimento(Double.parseDouble(input.nextLine()));
      System.out.printf("Entre com a largura do retangulo 1\n");
      retangulo1.setLargura(Double.parseDouble(input.nextLine()));
      System.out.printf("O perimetro do retangulo1: %.1f\n",retangulo1.perimetro());
      System.out.printf("A area do retangulo1: %.1f\n",retangulo1.area());
      System.out.printf("Entre com o comprimento do retangulo 2\n");
      retangulo2.setComprimento(Double.parseDouble(input.nextLine()));
      System.out.printf("Entre com a largura do retangulo 2\n");
      retangulo2.setLargura(Double.parseDouble(input.nextLine()));
      System.out.printf("O perimetro do retangulo1: %.1f\n",retangulo2.perimetro());
      System.out.printf("A area do retangulo1: %.1f\n",retangulo2.area());
      */

      //Opcao com menos codigo

      for(int i = 0 ; i < 2;i++){
         retangulo[i] = new Retangulo();
         System.out.printf("Entre com o comprimento do retangulo %d \n",i+1);
         retangulo[i].setComprimento(Double.parseDouble(input.nextLine()));
         System.out.printf("Entre com a largura do retangulo %d \n",i+1);
         retangulo[i].setLargura(Double.parseDouble(input.nextLine()));
         System.out.printf("O perimetro do retangulo %d:
%.1f\n",(i+1),retangulo[i].perimetro());
         System.out.printf("A area do retangulo %d: %.1f\n",(i+1),retangulo[i].area());
      }

      /*Outra opção setando os valores diretamente no codigo

      Retangulo R1 = new Retangulo();
    R1.setComprimento(5.3);
    R1.setLargura(2);
```

```
    Retangulo R2 = new Retangulo();
    R2.setComprimento(9.5);
    R2.setLargura(5.2);

    System.out.println("\nPerímetro do Retangulo 1: " + R1.Perimetro() + "\nÁrea do
Retangulo 1: " + R1.Area());
    System.out.println("\nPerímetro do Retangulo 2: " + R2.Perimetro() + "\nÁrea do
Retangulo 2: " + R2.Area());
      */

  }
}
```