IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

# Programação Avançada

**Ano Letivo de 2019/20**

18/12/2019: 18:30

Teste de avaliação

- O teste tem a duração de 1h30 minutos com 15 minutos de tolerância;
- O teste tem de ser respondido no enunciado nas zonas afetas às respostas;
- O teste é composto por um enunciado com 9 páginas e um anexo com 5 figuras (2 páginas)
- Os alunos que desistam têm de assinar "desisto" no enunciado;
- Todas as implementações solicitadas terão de ser efetuadas na linguagem JAVA.

| Número do Aluno: | Nome (em maiúsculas): |
|---|---|
| Assinatura do Aluno: | |

Grelha de Avaliação (a preencher pelo docente):

| 1(2.5) | 2(1.0) | 3(1.5) | 4 (2.0) | 5 (2.0) | 6 (1.0) | 7 (4.0) | 8 (2.0) | 9(4.0) |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | TOTAL | |

1. Considere que se pretende desenvolver uma aplicação para calculo de percursos em caminho de ferro em Portugal. Nessa aplicação é possível obter todos os trajetos possíveis entre a origem e o destino, indicando duas estações de caminho de ferro, (a origem e o destino) e uma data. Para cada trajeto obtido, é indicado o custo total, o tempo total de viagem e a hora de partida, assim, como todos as estações intermedias que compõe o percurso.

   1.1. (1.0) Para tal decidiu-se usar o TAD Digraph. Justifique a razão para tal escolha.

   1.2. (1.5) Para implementar a rede de Caminho de Ferro, usando o DiGraph, é necessário definir as classes que concretizam os tipos genéricos V (elemento do vértice) e E(elemento da aresta).Defina a assinatura e atributos de cada uma das classes

```
//classe que concretiza o tipo genérico V
```



Numero:_____

```
//classe que concretiza o tipo genérico E
```

2. (1.0) Considere o pseudocódigo do algoritmo seguinte referente a uma Binary Search Tree, qual o propósito destes.

```
ALGORITHM FX
Input: bst BSTree
Output: ?
BEGIN
   IF isEmpty(bst) THEN RETURN 0;
    RETURN 1+ MAXIMO( FX(leftTree(bst)),FX(rightTree(bst)))
 END
```

3. (1.5) Considere o código em JAVA da implementação do método recursivo *exists*, que verifica se um determinado elemento existe numa arvore binaria de pesquisa. Complete o código corretamente (tome como referência a estrutura da arvore como especificada na figura 1 do anexo).

```
@Override
    public boolean exists(T elem) {
        return exists(elem, root);
    }


    private boolean exists(T elem, TreeRoot treeRoot) {
        if (treeRoot == null) {
            return false;
        }
        int comparison =









}
```



Numero:_____

4. (2.0) Considere o código da figura abaixo, onde se aplicou o padrão observer. Preencha o código em falta na tabela seguinte, de forma a obtermos o seguinte output quando executamos o método main.

```
Rita
Ja fui notificado 1 vezes!
Joana
Ja fui notificado 2 vezes!
```

| X | | K | |
|---|---|---|---|
| Y | | Z | |

```
public class A                  X          {
    private String name;

    public A(String name) {
        this.name = name;
    }

    public String getName() {
      return name;

    }

    public void setName(String name) {
        this.name = name;
        setChanged();
                    W
    }

}
```

```
public class JavaObserver {

    public static void main(String[] args) {
        A a=new A("Sofia");
        B b=new B();
                k
        a.setName("Rita");
        a.setName("Joana");
    }
```

```
public class B                  Z          {
    public static int count=0;
    @Override
    public void update(Observable o, Object arg) {
                count++;
                System.out.println(((A)o).getName() );
                System.out.println("Ja fui notificado "
                + count + " vezes!");
    }

}
```

5. Considere as classes WPTO e YPTO apresentadas na figura abaixo.

```
public class YPTO {
    private String str;

    public YPTO(String str) {
        this.str = str;
    }
    public void algoritmo(int x){
        System.out.println("INICIO");
        System.out.println("***********");
        System.out.println(x + " " + str);
        System.out.println("FIM");
    }
}
```

```
public class KPTO {

    public void algoritmo(int x) {
        System.out.println("INICIO");
        for (int i = 0; i < x; i++) {
            System.out.println("******" + x + "******");
        }
        System.out.println("FIM");
    }
}
```

De forma a retirar as linhas de código duplicado comum ao método algoritmo em cada uma das classes, decidiu-se aplicar o padrão Template Method.

5.1. (1.0) Implemente a classe abstata resultante após aplicar o padrão Template Method ao exemplo dado.



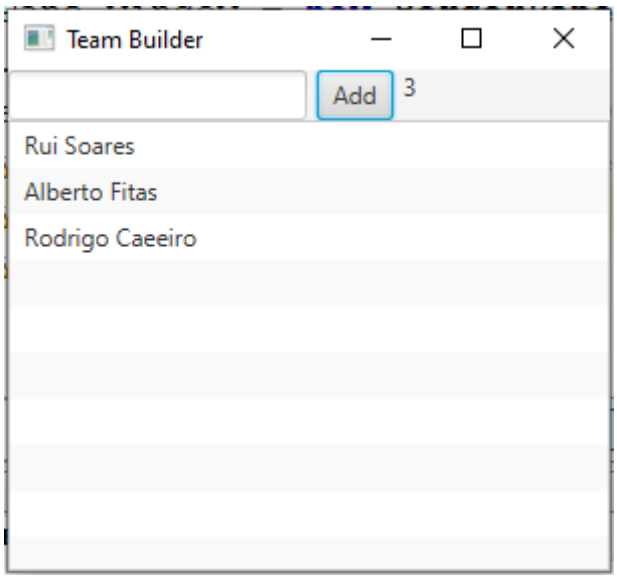
Numero:_____

5.2. (1.0) Preencha a tabela que estabelece uma relação entre o exemplo e os participantes no padrão.

| EXEMPLO | Participante no padrão |
|---|---|
| | Abstract Class |
| YPTO | |
| WPTO | |
| | Template Method |

6. (1.0) Considere o excerto de código apresentado na Figura 2 do Anexo. Implemente a **inner classe** MementoGraph que representa o memento da classe GraphImplementation. Nota: Relembro que as classes do java HashSet, ArrayList, HashMap têm todas construtores de cópia.

7. Pretende-se implementar uma aplicação para simular a criação de uma aplicação para registo dos elementos de uma equipa. Para tal decidiu-se aplicar o padrão MVC, para implementar a sua aplicação, tal como se mostra na figura 5 do Anexo.




.



Numero:_____

7.1. (1.5) A classe que implementa o Controller “perdeu-se”. Implemente a classe que assume o papel de controller de forma a que cada vez que se pressionar o botão Add, seja atualizada a lista com o novo nome introduzido, no campo de texto.

| |
|---|
| |

7.2. (2.5) Indique quais as modificações que teria de realizar em cada uma das classes que implementam os participantes (Model, View e Controller) de forma a:

- Cada vez que um nome repetido é introduzido, apareça uma janela de alerta, com a mensagem: “ Duplicated names not allowed”

Código em Java FX para criar uma janela de alerta de erro:

```
Alert alert = new Alert(Alert.AlertType.ERROR,errMessage,ButtonType.OK)
alert.showAndWait();
```

| | |
|---|---|
| Model | |



Numero:_____

| | |
|---|---|
| View | |
| Controller | |



Numero:_____

8. (2.0) Considere o código da Figura 1. Na classe `BSTImplementation` implemente (não pode adicionar quaisquer novos atributos) o método `ArrayList listOfGreater(int threshold)` que devolve uma lista de todos os elementos da árvore que sejam maiores que `threshold`. Deverá apresentar uma solução recursiva.

9. Considere a interface Graph, e a classe SocialNetwork apresentado na figura 3 e 4. Considere ainda o pseudocódigo do algoritmo BreathFirstSearch

BFS(Graph,vértice)
Marque vértice como visitado
Coloque o vértice na fila
Enquanto a fila não está vazia faça:

- Seja v o elemento retirado da fila.
- Processe v
- Para cada vértice w adjacente a v faça:
  - se w não está marcado
  - então :
    - marque w como visitado
    - insira w na fila



Numero:_____

9.1. (1.5) Complete a implementação do método printAll que usa o algoritmo BFS para imprimir todas os elementos do grafo a partir da do vértice onde se encontra a pessoa p.

```
public String printAll(Person p){

    Vertex<Person> v= getVertex(p);

    String str="";
```

9.2. Pretende-se implementar padrão Strategy para que o método printALL, passe a imprimir a estrutura do grafo usando o algoritmos BFS, mas usando diferentes critérios de paragem.

i. Imprime todas as pessoas da rede que é possível alcançar a partir de um vértice inicial.
ii. Imprime apenas as n primeiras pessoas encontradas
iii. Imprime apenas as pessoas que não distam da pessoa que se encontra no vértice inicial, mais do que n pessoas.

Considere a interface StrategyStopCriterium

```
public interface StrategyStopCriterium {

        public  boolean verify(Collection<Vertex<Person>> visited,int level);
}
```

9.2.1. (1.5) Indique quais as alterações que teria de realizar no método printAll, para integrar o padrão Strategy.



Numero:_____

9.2.2. (1.0) Implemente a estratégia concreta que permita implementar o critério de paragem (ii).

(fim do enunciado)



Numero:_____