IPS Instituto Politécnico de Setúbal Escola Superior de Tecnologia de Setúbal	**Teste de Programação Avançada 2021/22** **5 de janeiro de 2022**

Nome:_____ **Número:**_____

Q1(1.5)	Q2(1.5)	Q3(1.5)	Q4(1.5)	Q5(1.5)
Q6(3.0)	Q7(2.0)	Q8(3.0)	Q9(2.5)	Q10(2.0)

TOTAL:

Q1 - Padrão Iterator

Considere o seguinte código parcial de uma implementação de `Stack` e complete o código em falta na classe **MyIterator**, de forma a que o *iterador* fornecido faça uma **travessia do topo para a base da pilha**.

```
public class StackImpl<T> implements Stack<T> {
    private T[] elements;
    private int size;
    //...
    @Override
    public T pop() throws EmptyQueueException {
        //...
        return elements[--size];
    }
    @Override
    public Iterator<T> iterator() {
        return new MyIterator();
    }


    private class MyIterator implements Iterator<T> {
        private int cursor;
        public MyIterator() {



        }
```

```
        @Override
        public boolean hasNext() {




        }
        @Override
        public T next() {




        }
    }
}
```

Q2 - Padrão Memento

Considere o padrão de desenho *Memento*. Complete o código em falta na classe `ListNumbers`, que assume o papel do participante *Originator*:

```
public class ListNumbers implements Originator {
    private int size;
    private List<Integer> numbers;
    //...

    @Override
    public Memento createMemento() {




    }

    public void setMemento(Memento saved) {



    }

    private class MyMemento implements Memento {



        public MyMemento(                    ) {




        }
    }
}
```

Q3 - Padrão MVC

Considere o código relativo à implementação do padrão MVC de uma aplicação que tem como funcionalidade incrementar um contador cada vez que se aciona o botão NEXT, sabendo que esse contador volta a zero sempre que atingir o limite pré-configurado. (ver figura abaixo)

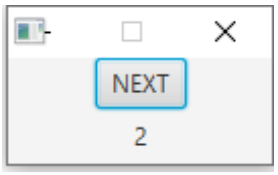

Figura 1

Complete o código em falta. (Completar abaixo do //TODO)

```
public class Main extends Application {

        public static void main(String[] args) {
            launch(args);
        }

        @Override
        public void start(Stage stage) throws Exception {  // TODO 1



            Scene scene = new Scene(panel);
            stage.setTitle("Count");
            stage.setScene(scene);
            stage.setResizable(false);
            stage.show();
        }
}

public class Counter extends Subject {
    private int value;
    private int max;

    public Counter(int max) {
        this.max = max;
    }

    public int getValue() {
        return value;
    }

    public void next(){ //TODO 2




    }}
```

```
public class CounterController {
    public Counter counter;
    public CounterPanel counterPanel;

    public CounterController(Counter counter, CounterPanel counterPanel){        //TODO 3




    }

    public void doNext() {
        counter.next();
    }
}

public class CounterPanel extends BorderPane implements Observer {
    private Counter counter;
    private Button btn1;
    private Label lblCounter;

    public CounterPanel(Counter counter) {
        this.counter = counter;
        createLayout();
    }

    private void createLayout() {
        btn1= new Button("NEXT");
        StackPane btnPane= new StackPane();
        btnPane.getChildren().add(btn1);
        lblCounter= new Label();
        setTop(btnPane);
        setCenter(lblCounter);
        update(null);
    }

    public void setTriggers(CounterController ctrl){
        btn1.setOnAction((ActionEvent event) -> { //TODO 4


        });
    }

    @Override
    public void update(Object obj) {  // TODO 5




    }

}
```

Q4 - Dijkstra

Considere o seguinte resultado do algoritmo *Dijkstra* sobre um grafo não-orientado e valorado:

vertex	cost	predecessor
A	1	D
B	2	D
C	5	F
D	0	
E	1	D
F	4	A
G	∞	

Figura 2

Responda às seguintes questões:

a) Qual o *vértice de origem* na execução do algoritmo? _____

b) Qual o custo do menor caminho até ao vértice C? _____

c) Qual o caminho do *vértice de origem* até ao vértice C? _____

d) O vértice G é garantidamente um vértice isolado do grafo (sim/não)? _____

Q5 - Binary Tree

Considere a seguinte árvore binaria de pesquisa, apresente o resultado de a percorrer em **pós-order**

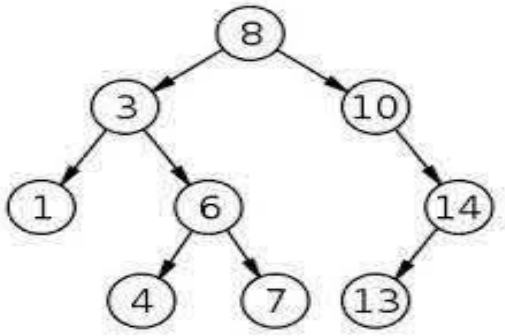

Figura 3

R: _____

Q6 - Grafos

Considere a rede social *Instagram* e a necessidade de modelar utilizadores e relações de "seguir" utilizando grafos. Sobre um utilizador sabe-se o *username* e o *email*; sobre uma relação, sabe-se a data em que foi estabelecida. Atente ao facto de um utilizador poder ter um diferente número de utilizadores que segue e de utilizadores que o seguem.

a) Qual o tipo de grafo mais apropriado para representar esta rede (grafo ou digrafo)? Justifique.

b) Forneça a assinatura e atributos das classes cujas instâncias armazenaria nos vértices e nas arestas:

Tipo em vértice:

Tipo em aresta:

c) Forneça um algoritmo em pseudocódigo que permita calcular o utilizador mais popular (com mais seguidores) da rede.

Q7 - BST

Considere a classe `BST` que armazena *inteiros*:

```
public class BST {
    private Node root;

    //...
    private class Node {
        int elem;
        Node left;
        Node right;
        //...
    }



 }
```

Forneça o código do método `countEvenExternal` que retorna a soma dos valores pares dos *nós externos* da árvore. Recomenda-se uma abordagem recursiva e pode implementar métodos auxiliares, se desejar.

```
public int countEvenExternal() {









}
```

Q8 - Padrão Strategy

Considere a seguinte classe `BagOfNumbers`:

```
public class BagOfNumbers {
    private List<Integer> bag = new ArrayList<>();
    public void add(int num) { bag.add(num); }
    public int min() throws BagException {
        if(bag.isEmpty()) throw new BagException("Empty bag.");
        Collections.sort(bag);
        return bag.get(0);
    }
    public int max() throws BagException {
        if(bag.isEmpty()) throw new BagException("Empty bag.");
        Collections.sort(bag);
        return bag.get(bag.size() - 1);
    }
    public int median() throws BagException {
        if(bag.isEmpty()) throw new BagException("Empty bag.");
        Collections.sort(bag);
        return bag.get(bag.size() / 2);
    }
}
```

a) Aplique o padrão *Strategy* aos vários algoritmos de cálculo. Apresente o código das classes e interfaces resultantes.

b) Para cada uma das classes/interfaces criadas, indique a que participante do padrão correspondem:

Classe/interface	Participante

c) Forneça um método `main` onde ilustre a utilização das classes resultantes (`BagOfNumbers` e as outras criadas por si), fazendo o *output* de pelo menos dois valores (mínimo, máximo ou mediana).

Q9 – Padrão Factory Method

Considere que o código abaixo implementa o padrão FactoryMethod

```
public interface DocStyle {
    public Document create(String type, String ...fields);
}

public class Informal implements DocStyle {
    @Override
    public Document create(String type, String... fields) {
        switch (type) {
            case "journal":
                return new Journal(fields[0]);
            case "letter":
                return new InformalLetter(fields[0], fields[1]);
            default:
                throw new IllegalArgumentException("Not exist : " + type);
        }
    }
}
public abstract class Document {
    private String name;
    private String content;

    public Document(String name) {
        this.name = name;
        this.content="";
    }
    //getters e setters
}
public class Journal extends Document{

    private Date date;

    public Journal(String name) {
        super(name);
        this.date= new Date();
    }
    @Override
    public String toString() {
        return date + "\n" + getName() + "\n" + getContent();
    }
}

public class InformalLetter extends Document{
    private String dst;

    public InformalLetter(String name, String dst) {
        super(name);
        this.dst = dst;
    }

    @Override
    public String toString() {
        return dst + "\n" + getContent()+"\n\t"+getName();
    }
}
```

a) Indique quais os participantes do padrão Factory Method e, para cada um dos participantes, indique qual a classe (ou classes) que no exemplo correspondem a

Classe/interface	Participante

b) **Aplicando o padrão Factory Method** disponibilizado nas classes acima, complete o *main* de forma a obter o seguinte output.

```
Alberto
Feliz Natal
      Patricia
```

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

     Document d=

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

c) Indique que alterações teria que fazer caso quisesse implementar um novo *DocStyle* denominado *FormalStyle*.

Q10 – Padrão do vídeo

Considere o padrão (ou um dos padrões) sobre qual fez o trabalho de vídeo.

a) Indique o Padrão:

b) Qual o problema que esse padrão se propõe resolver?

c) Indique as vantagens do mesmo