PA2021/22 - Perguntas Tipo/Teste/Exame

Grafos

Considere o código da Figura 1. Na classe `GraphImplementation` implemente (ignore quaisquer validações de argumentos, i.e., assuma que são válidos):

1. O método `boolean containsEdge(Vertex<V> v, Edge<E,V> e)`, que, dado um vértice e uma aresta, verifica se o vértice pertence à aresta;

2. O método `boolean isIsolated(Vertex<V> v)` que verifica se o vértice `v` é isolado;

3. O método `List<Vertex<V>> adjacentVerticex(Vertex<V> v)`, que, dado um vértice, devolve uma lista contendo todo os vértices adjacentes:

4. O método `Vertex<V> getMaxDegreeVertex()` que devolve o vértice com o maior grau no grafo. Assuma que possui implementado corretamente o método `List<Vertex<V>> adjacentVerticex(Vertex<V> v)`.

5. O método `int sumOfActiveEdges()` que devolve a soma dos elementos ativos de todas as arestas do grafo;

6. O método `int countEvenValuedEdges()` que devolve a contagem do número de arestas que contêm valores pares;

7. O método `int sumOfActiveAdjacentEdges(Vertex<V> v)` que, dado um vértice, devolve a soma dos elementos ativos de todas as arestas adjacentes a `v`.

8. O método `void removeVertex(Vertex<V> v)` que remove o vértice `v` do grafo, removendo também todas as suas arestas adjacentes;

```
public interface Vertex<V> {
    V element();
}
public interface Edge<E, V> {
    E element();
    Vertex<V>[] vertices();
}
public interface ValuedEdge {
    int value();
    boolean isActive();
}
public class GraphImplementation<V, E extends ValuedEdge> {
    //Estrutura de dados: lista de adjacências
    private Map<V, Vertex<V>> vertices = new HashMap<>();
    private Map<E, Edge<E, V>> edges = new HashMap<>();

      //...
}
```

Figura 1

Árvores BST

Considere o código da Figura 2. Na classe `BSTImplementation` implemente (não pode adicionar quaisquer novos atributos):

1. O método `boolean isEmpty()` que verifica se a àrvore está vazia.
2. O método `boolean isExternal(TreeNode node)` que verifica se `node` é um nó externo;
3. O método `boolean isInternal(TreeNode node)` que verifica se `node` é um nó interno;
4. O método `boolean exists(T element)` que verifica se existe na àrvore um elemento com o mesmo valor (i.e., `value()`) que `element`.
5. O método `int size()` que devolve o número de elementos da àrvore. A única solução que existe é uma solução recursiva.
6. O método `int sumOfTree()` que devolve a soma de todos os elementos da àrvore. A única solução que existe é uma solução recursiva.
7. O método `int sumOfLessThan(int threshold)` que devolve a soma de todos os elementos da àrvore que sejam menores que `threshold`. A única solução que existe é uma solução recursiva.
8. O método `int countExternal()` que devolve o número de nós externos presentes na àrvore. A única solução que existe é uma solução recursiva. Assuma que possui implementado o método `boolean isExternal(TreeNode node)`.

```
public interface IntegerElement {
    int value();
}
public class BSTImplementation<T extends IntegerElement> {
    private TreeRoot root;

    //...

    private class TreeRoot {
        T element;
        TreeRoot left, right;
    }
}
```

Figura 2

PADRÕES

Responda com resposta curta às seguintes questões:

1. Pretende-se que seja possível disponibilizar para o TAD Tree uma forma de percorrer todos os seus elementos. Qual o padrão mais adequado, para implementar essa funcionalidade?

   _____

2. Pretende-se que a aplicação para gestão dos rankings passe a apresentar o ranking dos 10 melhores alunos. No entanto pretende-se que este o critério de seriação possa variar de ano para ano, e sendo assim pretende-se contruir uma classe Ranking suficientemente flexível para suportar diferentes critérios de seriação.
   Qual o padrão mais adequado, para implementar essa funcionalidade?

   _____

3. Explique, porque podemos afirmar que se classe X<E> implementar a interface Iterable<E>, isto significa que é possível executar o seguinte código:

```
X<E> objects= new X();
for(E elem: objects)
   System.out.println(elem);
```

   _____

4. Para cada uma das 4 situações indique qual o padrão mais adequado a aplicar, justifique a resposta (justificação curta)
   a. Implementação do registo de informação dos alunos da turma, sendo possivel criar três relatorios distintos - relatórioSimples (só com as notas finais, e média da turma), relatórioNotasDetalhadas, relatórioIntermedio(só com as notas obtidas até à data atual).
   b. Implementação do mecanismo de undo, no jogo do solitário
   c. Implementação do mecanismo de percorrer os elementos de uma arvore binária por niveis usando um ciclo foreach.

5. Considere o seguinte Código em Java referente ao padrão Memento

```
public interface Memento {
    //vazia
}

public class MyTime {
    int hour, minute, second;

    public MyTime(int hour, int minute, int second) {
        this.hour = hour;
        this.minute = minute;
        this.second = second;
    }
```

```
    //acessores e modificadores
}

public class Message {
    private MyTime timeStamp;
    private String content;

    //construtor e outros métodos; não interessam quais.

    //a) inner class MessageMemento

    public void setMemento(Memento savedState) {
        //b)
    }

    public Memento createMemento() {
        //c)
    }
}
```

a. Forneça a implementação da inner class MessageMemento responsável por representar um memento da classe Message.
b. Forneça a implementação do método setMemento.
c. Forneça a implementação do método createMemento.

6. Considere o padrão Memento e as classes definidas na Figura seguinte.
A) Se a classe Linha for considerada como o Originator.
Implemente o construtor da classe LinhaMemento que implementa a interface Memento.

B) Considerando que tem o padrão Memento implementado para a classe Linha a a classe CareTaker, implementada. Implemente no main, um conjunto de instruções que mostre como pode salvar uma Linha, altera-la e recupera-la posteriormente.

```
public class Ponto {

  private int x, y;

  public Ponto(int x, int y) {
    this.x = x;
    this.y = y;
  }

  public int getX() {
    return x;
  }

  public int getY() {
    return y;
  }
```

```
  void moveX(int dx) {
    x+=dx;
  }

  void moveY(int dy) {
    y+=dy;
  }

  boolean equalsGeo(Ponto p1) {
    return p1.x == x && p1.y == y;
}
}
```

```
public class Linha implements Originator{
```

```
  public Ponto getP2() {
```

```
private Ponto p1, p2;

public Linha(Ponto p1, Ponto p2) {
   this.p1 = p1;
   this.p2 = p2;
}

public Ponto getP1() {
   return p1;
}
```

```
   return p2;
}

void moveX(int dx) {
   p1.moveX(dx);
   p2.moveX(dx);

}

void moveY(int dy) {
   p1.moveY(dy);
   p2.moveY(dy);
}
```

---

7. **Considere o seguinte problema. Pretende-se implementar uma aplicação para gestão da admissão dos alunos numa escola de artes. Os alunos são admitidos em função de um conjunto de critérios que podem variar consoante o curso, e que são decididos anualmente. Presentemente a escola definiu 4 perfis para Admissão dos alunos:**
   - Criterio Idade – Onde é establecido como critério de admissão uma idade mínima e máxima.
   - Critério Habilitação – Onde é estabelecido como critério de admissão o nível mínimo de Habilitação (6ºano, 9ºano,12ºano,licenciatura).
   - Critério Área de Formação–– Onde é estabelecido como critério de admissão o aluno ter como formação base numas das áreas especializadas: Exemplo: o aluno terá que ter formação em Desenho ou Escultura.
      a) Para implementar a classe que gere a admissão de alunos aos cursos, decidiu-se usar o padrão Strategic. E definiu-se a que a classe gestor de aluno deveria disponibilizar as seguintes operações :
         - `boolean admiteAluno(Aluno aluno) //caso o aluno satisfaça o critério associado a este gestor de aluno, adiciona o aluno na lista de alunos do curso e retorna true.`
         - `boolean removeAluno(Aluno aluno) // caso o aluno exista na lista de alunos remove-o da lista e retorna true.`
         - `int numeroAlunos()// devolve o numero de alunos inscritos.`
         - `String getNomeCurso()// devolve o nome do curso.`

Assuma que a interface Aluno está definida da seguinte forma.

```
public interface Aluno {
    public Aluno createAluno(String Habilitacao, int nivel, int idade);
    public void setFormacao(String Formacao);
    public String getFormacao ();
    public void setIdade(int idade);
    public int getIdade();
    public void setNivel(int nivel);
    public int getNivel();
}
```

Figura 3.1

   a. Identifique os participantes desse padrão.
   b. Desenhe o Diagrama de classes em UML, que evidencie a utilização do padrão estratégia para a resolução do problema indicado.

c. Defina a assinatura da classe GestorAlunos, os seus atributos e o método construtor.
d. Segundo a especificação efetuada, em B. Elabore um pequeno programa de teste (MainClass), que teste as duas situações seguintes. Assuma que as classes definidas no Diagram UM estão implementadas.
    - Criação do cursoA que admite alunos com idade superior a 23 anos.
    - Criação do cursoB que admite alunos com uma das seguintes formações: Escultura ou Desenho
e. De acordo com o que foi proposto nas alíneas b e c, **indique** como implementaria o método , `boolean admiteAluno(Aluno aluno),na implementação da interface GestorCurso.`

---

8. **Considere** o código da Figura seguinte que implementa uma aplicação com interface em modo consola usando o padrão MVC.

```
public class ClassC implements Observer {
    String name;
    ClassA a;
    ClassB  b;

    public ClassC(String name, ClassA a, ClassB b) {
        this.name = name;
        this.a = a;
        this.b = b;
    }

    public void triggerEvent(int value) {
        this.a.newValue(value);
    }
    @Override
    public void update(Observable o, Object arg) {
        System.out.println(String.format("View %s got value: %d", name, arg));
    }
    void setDone(String done) {
        System.out.println("-----------" + done + "----------------");

    }
    private void inicializa() {
     System.out.println(String.format("****View ***** value %d",  b.getValue()));
    }
}
```

```
public class ClassA {

    private ClassB b;
    private ClassC c;

    public ClassA(ClassB b) {
        this.b = b;
    }
    public void setC(ClassC c)    {
       this.c=c;
    }
    public void newValue(int value) {
            b.incValue(value);
            c.setDone("DONE");
    }
}
public class ClassB extends Observable {
private int value = 0;

    public void incValue(int value) {
        this.value += value;
        setChanged();
        notifyObservers(this.value);
    }
    public int getValue() {
        return value;
    }
}
```

```
class Factory {

    private static ClassB b = new ClassB();

    public static ClassC getObjC(String name)
{
        ClassB b = new ClassB();
        ClassA a = new ClassA(b);
        ClassC c = new ClassC(name,a,b);
        a.setC(c);
        b.addObserver(c);
        return c;
    }
}


public class Test {
    public static void main(String[] args) {
        ClassC c = Factory.getObjC("Teste”);
        c.triggerEvent(2);
        c.triggerEvent(5);



      }

}
```

a) Relacione as classes ClassA, ClassB, ClassC, com os participantes no padrão. Preencha a tabela abaixo.

Participante	Classe
M	
V	
C	

b) Indique qual o output da execução do main de teste apresentado.

c) A implementação do padrão MVC apresentada recorre ao uso do padrão Observer. Explique com que finalidade o padrão Observer é aqui implementado.

d) A implementação apresentada recorre também o uso do padrão SimpleFactory. Explique com que finalidade este padrão foi implementado.

e) **Indique quais as modificações** que teria que realizar em cada uma das classes, caso se pretendesse que a aplicação respondesse a um novo tipo de evento – denominado **triggerClearEvent –** com a finalidade de limpar o valor acumulado.
Nota: Após este evento criado, deverá ser possível executar a seguinte instrução no método main apresentado: **c.triggerClearEvent().**