IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Programação Avançada

Ano Letivo de 2020/21

6 de Janeiro de 2021

Teste

- O teste tem a duração de 2H ;
- O teste tem de ser respondido no enunciado nas zonas afetas às respostas;
- Os alunos que desistam só podem sair decorridos 30min e têm de assinar "desisto" no enunciado;
- Todas as implementações solicitadas terão de ser efetuadas na linguagem Java.

Número do aluno: (preencha também em cada folha no rodapé)
Nome (em maiúsculas):

Grelha de Avaliação **(a preencher pelo docente**):

1.1	1.2	1.3	1.4	1.5	1.6		
2.1	2.2	2.3	2.4				
						TOTAL	

GRUPO 1 – Resposta Curta (8 valores)

Responda às questões nas zonas atribuídas.

1. (1val) Complete o algoritmo COUNT_INTERNAL de forma a este calcular o número de nós internos de uma árvore:

```
COUNT_INTERNAL(tree)
    IF IS_EMPTY(tree) THEN
        RETURN 0
    ENDIF
    count <- 0
    IF IS_INTERNAL(tree.root) THEN
        /*A*/
    END IF
    FOR EACH child FROM tree.children
      /*B*/
    END FOR
    RETURN /*C*/
```

R:

A –

B –

C –

Numero:_____

2. (1val) Considere a *travessia de grafos* e em particular a travessia **depth-first**, cujo algoritmo se apresenta de seguida:

```
DFS(Graph,vértice_raiz)
    Marque vértice_raiz como visitado
    Coloque o vértice_raiz na pilha
    Enquanto a pilha não está vazia faça:
        - seja v o vértice que retira da pilha
        - processe v
        Para cada vértice w adjacente a v faça:
             Se w não está marcado como visitado então:
              - marque w como visitado
              - insira w na pilha
```

Complete a seguinte implementação em Java, fornecendo o código das linhas em falta.

```
public void DFS(Graph<V,E> graph, Vertex<V> origin) {
    Stack< /* A */ > stack = new Stack<>();
    List< /* A */ > visited = new ArrayList<>();
    visited.add(origin);
    stack.push(origin);
    while( /* B */ ) {
        /* C */
        process(v);
        for( /* D */ ) {
            if(!visited.contains(v)) {
                visited.add(v);
                /* E */
            }
        }
    }
}
```

R:

A –

B –

C –

D –

E –

3. (2val) Considere um problema de representação de uma rede *peer-to-peer* . Cada nó desta rede consiste num computador do qual se sabe o seu *IP* (texto, e.g., “168.0.0.1”). Cada computador pode ligar-se a outros computadores através de uma ligação *TCP/IP* da qual se conhece a velocidade de transmissão em Mbit (e.g., 100) e o valor de *ping* em milisegundos (e.g., 14). Note que numa ligação *TCP/IP* ambos os computadores podem enviar/receber dados em simultâneo.

   a) Se formos representar este problema utilizando grafos, utilizaria de um **dígrafo** ou um **grafo** (não direcionado) para representar este problema? **Justifique**.

R:

b) Forneça as **assinaturas e atributos** das classes envolvidas para representar;

i. O tipo V a armazenar nos vértices:

ii. O tipo E a armazenar nas arestas:

4. (1val) Considere a *interface* Dao em anexo. Considere também uma aplicação, onde o padrão respetivo será aplicado, que permita manipular/persistir os alunos *alumni* de uma única instituição. Acerca de cada aluno é apenas sabido o seu número, nome e média final de curso.
   Forneça a **assinatura e atributos** da classe `AlunoAlumni` e a *interface* `AlumniDao` que, para além de todas as operações de Dao, deve disponibilizar uma operação para obter todos os alunos cuja média se encontre num intervalo dado fornecido através de dois argumentos `min` e `max`).

   R:

Numero:_____

5. (1val) Considere o padrão **Memento** e a classe `Bingo` cujo estado se pretende guardar ao longo do tempo. Complete o código dos métodos `createMemento` e `setMemento` e da *inner class* `MyMemento`.

```
public class Bingo implements Originator{
    private int numeroSerie;
    private final List<Integer> numeros;

    //...
    @Override
    public Memento createMemento() {


    }
    @Override
    public void setMemento(Memento saved) {
        /* atente ao modificador 'final' */
        MyMemento memento = (MyMemento)saved;




    }

    private class MyMemento implements Memento {









    }
}

public interface Memento {
    /* propositadamente vazia */
}
```

6. Considere o código do Anexo onde as classes A, B e C, implementam o padrão MVC

   a) (1val) Indique o papel assumido por cada uma das classes

   Classe A –

   Classe B –

   Classe C –

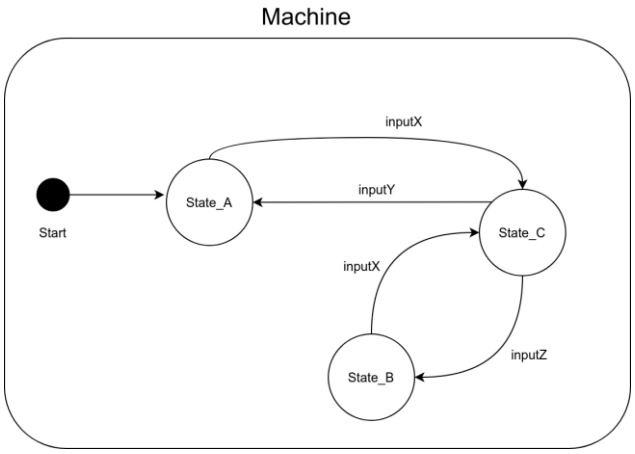
b) (1val) Complete o método main de forma a executar a aplicação MVC

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

}
```

GRUPO 2 – Desenvolvimento (12 valores)

1. (3val) Utilizando o padrão de desenho **State**, implemente a lógica da classe `Machine` ilustrada na imagem seguinte:



Se no diagrama um determinado estado não "responde" a um determinado input, significa que se mantém nesse estado.

Numero:_____

2. (3val) Considere que se pretende implementar um padrão de desenho para criar instâncias de um **Jogo** em função do tipo de jogo selecionado pelo utilizador . Considere que a classe Jogo e as suas 3 subclasses: `JogoGalo`, `JogoSolitario` e `JogoQuem` já se encontram implementadas.

   a) Qual dos padrões `Factory` estudados se aplica melhor a esta situação? **Justifique** a sua resposta.

   b) Para o padrão selecionado identifique cada um dos participantes.

c) **Complete** o código em falta no método `main` e **escreva o código** relativo à(s) classe(s) em falta:

```
public class Main {
    public static void main(String[] args) {
      char op;
      Jogo jogo;
        do {
            System.out.println("menu");
            System.out.println("A - JogoGalo");
            System.out.println("B - JogoSolitario");
            System.out.println("C - JogoQuem");
            System.out.println("Q - Quit");
            System.out.println("Introduz a opção >");
            op = readInput();
            if (op != 'Q') {
              //completar

              jogo=

              jogo.execute();

            }
        } while (op != 'Q');
    }
}
```


Numero:_____

3. (4val) Considere a classe `MapBSTree` em anexo, que é uma implementação parcial do ADT Map usando como estrutura de dados uma árvore binária de pesquisa.

a) Com base no código anterior, forneça a implementação do método `get` :

```
/**
* Returns the value to which the specified key is mapped, or null if this map c
ontains no mapping for the key.
* If this map permits null values, then a return value of null does not necessa
rily indicate that the map contains * no mapping for the key; it's also possibl
e that the map explicitly maps the key to null.
* @param key the key whose associated value is to be returned
* @return the value to which the specified key is mapped, or null if this map c
ontains no mapping for the key
* @throws NullPointerException if the specified key is null and this map does n
ot permit null keys (optional)
*/
V get(K key) throws NullPointerException {









}
```

b) De forma a implementar um teste unitário para testar o método acima, complete a classe abaixo.

```
class MapBSTTest {
    private MapBST<Integer,String> map;

    @BeforeEach
    void setUp() {
        map= new MapBST<Integer, String>();
        map.put(2,"Value2");
        map.put(3,"Value3");
        map.put(4,"Value4");
        map.put(1,"Value1");
    }

    @Test
    void get() {









    }

}
```


Numero:_____

4. (2val) Considere a classe `GraphAdjacencyList` em anexo. Contém uma implementação da *interface* `Graph` utilizando uma estrutura de dados baseada em lista de adjacências.

   Com base no código anterior, forneça a implementação do método `opposite` :

```
/**
* Given vertex v, return the opposite vertex at the other end
* of edge e.
*
* If e is not incident to v, returns null.
*
* @param v          vertex on one end of e
* @param e          edge connected to v
* @return           opposite vertex along e
* @exception InvalidVertexException
            if the vertex is invalid for the graph
* @exception InvalidEdgeException
            if the edge is invalid for the graph
*/
public Vertex<V> opposite(Vertex<V> v, Edge<E, V> e) {









}
```

(fim de enunciado)