| | |
|---|---|
| **Started on** | Monday, 12 December 2022, 5:56 PM |
| **State** | Finished |
| **Completed on** | Monday, 12 December 2022, 6:03 PM |
| **Time taken** | 7 mins 3 secs |
| **Marks** | 2,00/5,00 |
| **Grade** | **8,00** out of 20,00 (**40**%) |

**Question 1**

Incorrect

Mark 0,00 out of 1,00

Considere o seguinte excerto de código abaixo, que se refere à utilização de um padrão de software.Qual é o padrão que está ilustrado?

```
public class FactoryMain {
    public static void main(String[] args) {
        Pizza p1 = Loja.make("type1", "pp1");
        Pizza p2 = Loja.make("type2", "pp2");
        p1.applyPromotion(5);
        p2.applyPromotion(5);
        System.out.println(p1);
        System.out.println(p2);
    }
}


public class Loja{
    public static Pizza make(String type, String name) {
      switch (type){
          case "type1": return new PizzaNapoles(name);
          case "type2": return new PizzaTuna(name);
          default: throw new IllegalArgumentException();
     }
    }

}
```

Select one:

- a. Simple Factory
- b. Factory Method; ✖
- c. Nenhum dos Anteriores
- d. Abstract Factory

**Question 2**

Correct

Mark 1,00 out of 1,00

Considere o seguinte excerto de código abaixo, que se refere à utilização do padrão Factory Method. Qual a afirmação correta?

```
public class FactoryMethodMain {
    public static void main(String[] args) {
        Loja loja = new LojaAmerica();
        System.out.println(loja.listmenu());
        Pizza p1 = loja.make("type1", "pp1");
        Pizza p2 = loja.make("type2", "pp2");
        p1.applyPromotion(5);
        p2.applyPromotion(5);
        System.out.println(p1);
        System.out.println(p2);
    }
}

public interface Loja {
    Pizza make(String type, String name);
    String listMenu();
}

public class LojaAmerica implements Loja{
    public Pizza make(String type, String name) {
      switch (type){
          case "type1": return new PizzaNapoles(name);
          case "type2": return new PizzaTuna(name);
          default: throw new IllegalArgumentException();
     }
    }

    public String listMenu(){
        return "Pizza Napoles ; PizzaTuna";
    }
}
```

Qual a afirmação correta?

Select one:

a. O factory method tem de ser um método estático.

b. A classe `LojaAmerica` assume o papel de *Concrete Creator* ✔

c. Nenhuma das anteriores

d. O factory method pode ser o método `listMenu`

**Question 3**

Correct

Mark 1,00 out of 1,00

Considere o seguinte excerto de código que se refere ao padrão Memento, onde a classe ShoppingCart assume o papel de Concrete Originator. A classe ShoppingCart simula um carrinho de compras onde se adicionam produtos e se pode obter o seu preço total.

```
public class MainConsola {
    public static void main(String args[]){
        ShoppingCart cart = new ShoppingCart();
        Caretaker caretaker= new Caretaker(cart);
        caretaker.saveState();
        cart.addProduct(new Product("bananas",2));
        cart.addProduct(new Product("peras",2));
        caretaker.saveState();
        cart.addProduct(new Product("maças",5));
        caretaker.restoreState();
        System.out.println(cart.getTotal()); // imprime o valor total dos produtos no carrinho

    }
}
```

Qual o output do programa

Select one:

- a. Nenhum dos Anteriores ✔
- b. 9
- c. 2
- d. 0

**Question 4**

Incorrect

Mark 0,00 out of 1,00

Considere o padrão Memento e o código da classe `X` e da classe `Y` que implementa a interface *Originator*.

```
public class X {
    private int x1,x2;

    public X(int x1, int x2) {
        this.x1 = x1;
        this.x2 = x2;
    }
    //getters e setters
}

public class Y implements Originator {

    private final List<String> at1;
    private final X x;

    public Y() {
        at1 = new LinkedList<>();
        x=new X(0,0);
    }

    // other methods

    private class MyMemento implements Memento {
        private final List<String> stateAt1;
        private final X stateX;

    // code missing
}
```

```
//Snippet A
public MyMemento(List<String> stateAt1ToSave, X stateXtoSave) {
    this.stateAt1 = new LinkedList<>(stateAt1ToSave);
    this.stateX = new X(stateXtoSave.getX1(),stateXtoSave.getX2());
}
```

```
//Snippet B
public MyMemento(Y stateToSave) {
    this.stateAt1 = new LinkedList<>(at1);
    this.stateX = new X(x.getX1(),x.getX2());
}
```

```
//Snippet C
public MyMemento(List<String> stateAt1ToSave, X stateXtoSave) {
    this.stateAt1 = stateAt1ToSave;
    this.stateX = stateXtoSave;
}
```

Relativamente à inner classe `MyMemento`, selecione a opção de código que preenche corretamente o código em falta:

Select one:

- a. Snippet C
- b. Todas estão incorretas
- c. Snippet A
- d. Snippet B ✖

**Question 5**

Incorrect

Mark 0,00 out of 1,00

Relativamente ao padrão MVC podemos afirmar que:

Select one:

- a. Nenhuma das anteriores ✖
- b. O participante Controler conhece o participante Model, mas não conhece o participante View.
- c. O participante Model não conhece o participante View.
- d. O participante View é responsavel por controlar o fluxo da interação da aplicação