

Número:

Nome:

1º teste de Introdução à Programação

3 de Novembro de 2018

2 horas

Pergunta 1 (4 Valores)

(a) Considere a seguinte classe:

```
public class Enigma {  
    public static final int NEUTRO = 0;  
  
    private int inic;  
    private int valor;  
  
    public Enigma (int valor) {  
        inic = valor;  
        this.valor = NEUTRO;  
    }  
    public void Limpar() {  
        valor = NEUTRO;  
    }  
    public int acumMet() {  
        valor = valor - inic/2;  
        return valor;  
    }  
}
```

Complete nos espaços o que seria escrito na consola após a execução de cada instrução na sequência de instruções dadas. Caso não seja escrito nada, escreva "Não escreve nada".

Enigma x = new Enigma(11);

System.out.println(x.acumMet());

System.out.println(x.acumMet());

x.acumMet();

System.out.println(x.acumMet());

x.Limpar();

System.out.println(x.acumMet());

(b) Considere a seguinte classe:

Número:

Nome:

```
public class PeneiraInt {
    private int num;
    /**
     * @pre: valor > 0
     */
    public PeneiraInt(int valor) {
        num = valor;
    }

    /**
     * @pre: algoritmo >= 0 && algoritmo <= 11
     */
    public int devolveInt(int algoritmo) {
        int i = num;
        int res = 0;
        while (i != 0) {
            if (i % algoritmo == 0) {
                res++;
                i = 0;
            }
            i = i - 2;
        }
        return res;
    }
    public String passaTexto() {
        int i = num;
        String res = "";
        while (i != 0) {
            res = (i % 10) + res;
            i = i/10;
        }
        return res;
    }
}
```

Complete nos espaços o que seria escrito na consola após a execução de cada instrução na sequência de instruções dadas. Caso não seja escrito nada, escreva "Não escreve nada".

PeneiraInt x = **new** PeneiraInt(332);

System.**out**.println(x.passaTexto());

System.**out**.println(x.devolveInt(4));

PeneiraInt x = **new** PeneiraInt(24);

System.**out**.println(x.devolveInt(11));

System.**out**.println(x.devolveInt(5));

Número:

Nome:

Pergunta 2 (8 valores)

Implemente em Java a seguinte classe para simular os movimentos monetários de um portaEuros.

Cada portaEuros tem associado um valor limite de crédito, dado no momento da sua criação. Caso não seja indicado nada no momento da criação, o limite de crédito é zero.

Num portaEuros pode realizar-se depósitos e levantamentos de montantes positivos em euros. O depósito é sempre possível, no entanto o levantamento só pode ser realizado se o montante a levantar é menor ou igual à soma do valor corrente no portaEuros mais o limite de crédito atribuído a esse portaEuros. É sempre possível consultar: (1) valor corrente (denominado de saldo) no portaEuros; (2) valor limite de crédito; (3) valor do maior levantamento efetuado (com sucesso) até ao momento; (4) menor valor já registado como saldo da conta; (5) indica se o saldo corrente é negativo ou zero; e (6) valor médio dos valores registados (considerados saldos) entre depósitos e levantamentos efetuados (com sucesso).

Um portaEuros criado com o limite de crédito de 1000 Euros com saldo inicial 500 Euros, no qual são realizados respetivamente os seguintes movimentos: (1) levantamento de 1000 Euros; (2) depósito de 500 Euros e (3) levantamento de 200 Euros. Após estes movimentos se consultarem o:

- saldo médio, o valor dado é de $(500 + (-500) + 0 + (-200))/4 = -50$ Euros;
- saldo corrente, o valor dado é -200 Euros;
- maior levantamento, o valor dado é 1000 Euros;
- menor saldo, o valor dado é -500 Euros.

Se, para além dos movimentos indicados anteriormente, tentarmos fazer (4) um levantamento de 1200 Euros (levantamento não efectuado já que se excederia o limite de crédito), os valores do saldo médio, saldo corrente, maior levantamento e menor saldo mantêm-se inalterados.

Antes de começar a programar, leia todos os comentários em baixo para compreender o funcionamento da classe, e a poder implementar. Note que existe um espaço para implementar métodos privados, caso necessite.

```
public class PortaEuros {  
    /** Constantes */
```

```
    /** Variáveis de instância */
```

```
    /** Cria portaEuros com o limite de credito dado, e o saldo inicial dado.
```

```
    * @param credito - credito do portaEuros
```

```
    * @param inicial - saldo inicial do portaEuros
```

```
    * @pre: credito >=0 && inicial >0
```

```
    */
```

```
    public PortaEuros(float credito,float inicial) {
```

```
}
```

Número:

Nome:

```
/** Cria portaEuros com o limite de credito com 0 euros, e o saldo inicial dado.  
* @param inicial - saldo inicial do portaEuros  
* @pre: inicial >0  
*/
```

```
public PortaEuros(float inicial) {
```

```
}
```

```
/** Indica o saldo corrente do portaEuros
```

```
* @return saldo em euros
```

```
*/
```

```
public float daSaldo() {
```

```
}
```

```
/** Indica se o saldo do portaEuros é negativo ou zero
```

```
* @return true, caso o saldo seja negativo ou zero; false, caso contrário
```

```
*/
```

```
public boolean eDevedor() {
```

```
}
```

```
/** Indica o limite de crédito do portaEuros
```

```
* @return limite de crédito em euros
```

```
*/
```

```
public float daLimiteCredito() {
```

```
}
```

```
/** Levanta o valor dado no portaEuros, caso o montante seja inferior ou igual
```

```
* à soma do saldo corrente mais o limite de crédito. Neste caso retorna true.
```

```
* Caso não seja possível executar o levantamento retorna false.
```

```
* @param montante - valor em euros a levantar
```

```
* @pre
```

```
* @return true, caso faça o levantamento; false, caso contrário
```

```
*/
```

```
public boolean levantar(float montante) {
```

```
}
```

```
/** Indica o maior levantamento realizado no portaEuros
```

Número:

Nome:

```
* @return valor em euros
```

```
*/
```

```
public float daMaiorLevantamento() {
```

```
}
```

```
/** Deposita o valor dado no portaEuros.
```

```
* @param montante - valor em euros a depositar
```

```
* @pre
```

```
*/
```

```
public void depositar(float montante) {
```

```
}
```

```
/** Indica o menor saldo já registado no portaEuros
```

```
* @return valor em euros
```

```
*/
```

```
public float daMenorSaldo() {
```

```
}
```

```
/** Indica a média do valor do saldo do portaEuros entre levantamentos e/ou depósitos.
```

```
* @return Valor médio do saldo do portaEuros
```

```
*/
```

```
public float saldoMedio() {
```

```
}
```

```
/* Métodos privados */
```

```
}
```

Pergunta 3 (4 valores)

Este programa principal usa a classe PortaEuros referida na pergunta 2 (note que pode resolver esta questão mesmo que não tenha respondido à questão 2). Complete o código de acordo com os pedidos nos comentários colocados no código.

```
public class Main {
```

Número:

Nome:

```
/** Leitura de movimentos a realizar no portaEuros dado.
 * A leitura acaba quando o utilizador indica com a palavra "FIM".*/
private static void lerMovimentos (PortaEuros p){
    Scanner in = new Scanner(System.in);
    String tipoMovimento="DEP";    // tipo de movimento a realizar (levantamento, depósito)
    float montante=0f;              // Valor a levar ou depositar
    // Enquanto tipoMovimento diferente de "FIM"
    while
    {
        System.out.println("Indique o tipo de movimento (LEV, DEP ou FIM):");
        tipoMovimento = in.next().toUpperCase().trim();
        if (tipoMovimento.equals("LEV") || tipoMovimento.equals("DEP")){
            System.out.println("Indique o valor:");
            montante=in.nextFloat();
            // Verifica pré-condições para os movimentos indicados
            if
            {
                processaMovimento(tipoMovimento,montante,p);
            }
            else System.out.println("Montante invalido.");
        } //fim if
        in.nextLine();
    } // fim while
    in.close();
}
/* Processa levantamento ou depósito, escrevendo se o movimento teve sucesso ou não
 * NOTA NÃO PODE USAR A INSTRUÇÃO switch
 * @param tipoMov - tipo de movimento a processar
 * @param valor - valor a levantar ou a depositar
 * @param p - porta euros onde se vai realizar o movimento
 * @pre: (tipoMovimento.equals("LEV") || tipoMovimento.equals("DEP")) && valor > 0
 */
public static void processaMovimento(String tipoMov, float valor, PortaEuros p){
}

/** Programa principal - leitura de movimentos e escrita do saldo do portaEuros */
public static void main (String [] args){
    PortaEuros portaFCT = new PortaEuros(3000f,500f);
    lerMovimentos(portaFCT);
    // Escreve saldo final com duas casas decimais
}
}
```

Número:

Nome:

Pergunta 4 (4 valores)

Para o seguinte enunciado:

Objectivo: Simular um robot.

Descrição e Funcionalidades

Um robot é uma entidade que executa os movimentos "avança" e "recua". Para além dos movimentos, o robot também ajusta a sua velocidade ("acelera").

No momento de criação do robot pode-se indicar o número de casas que o robot avança ou recua em cada movimento. Note que este número é o mesmo para avançar e recuar.

De cada vez que se movimenta, o robot avança ou recua um dado número de casas. Esse número de casas depende do valor dado no momento da criação e das acções de ajuste na velocidade.

É sempre possível saber o número de casas que o robot passa em cada movimento (valor corrente) e o total de avanços e recuos realizados. O robot pode ajustar esse número de casas definido para cada movimento, com a acção "acelera", indicando o incremento (valor inteiro positivo) ou decremento (valor inteiro negativo). Note que no caso de decremento deve verificar que este é inferior ao número de casas corrente.

(a) Que classe seria necessário definir e implementar?

(b) Indique os métodos da interface dessa classe. Não deve implementar, só indicar o seu nome o tipo de dados que devolve e os seus parâmetros (assinatura/cabeçalho do método). Para além disso indique, caso existam, as pré-condições.

(c) Que variáveis de instância teria na classe? Para cada uma delas indique o seu tipo e o que representa.