1º Teste de “Introdução à Programação” (2011/12 Semestre Par)

Duração 2H

Instruções importantes:

- Responda a cada grupo em **folhas separadas**.
- Verifique que **todas** as folhas estão identificadas com o número de caderno.
- Antes de começar a resolver, leia o enunciado do **princípio até ao fim**.
- **Pode** usar caneta ou lápis.
- Não é permitido consultar quaisquer elementos para além deste enunciado.
- Qualquer tentativa de fraude comprovada acarretará a reprovação na disciplina.
- Não é permitido sair da sala antes que o exame termine.

I – Para cada um dos fragmentos de programas Java apresentados, assinale **que linhas são executadas**, e indique qual o valor que **cada uma** das variáveis contém após **cada linha** ser executada (sugestão: para cada alínea, faça uma tabela cujas linhas são as linhas do programa e cujas colunas são as variáveis do programa; assinale as linhas não executadas com “NE”).

a)

```
1 {
2     int a = 14;
3     int b = 5;
4     boolean c = false;
5     a = a/b;
6     b = a % b;
7     c = a == b;
8 }
```

b)

```
1 {
2     String filmSeries = "Tin Tin 201";
3     int inc = 0;
4     boolean cond = true;
5     filmSeries = filmSeries + inc++;
6     if (inc >= 1 || !cond)
7         inc--;
8     filmSeries = filmSeries + inc;
9 }
```

c)

```
1 {
2     int x1 = 3;
3     int x2 = 5;
4     int x3 = 7;
5     x1 = x1 % x2;
6     x2 = x2 % x3;
7     x3 = x3 % x1;
8     if (x1 > x3 && x2 < x1)
9         x1 = (x1 + x2) / x3;
10    else if (x2 > x3)
11        x1 = x3++;
12 }
```

II – Necessita de conceber uma aplicação – MyVideoOnDemand - para gerir o aluguer de filmes através de um operador de fibra óptica. O objectivo desta aplicação é permitir ao utilizador alugar filmes (tal como num clube de vídeo) sem sair de casa. O pagamento deste serviço é similar ao pagamento dos telemóveis pré-pagos: antes de fazer o aluguer de filmes é necessário fazer um carregamento/pagamento através do multibanco, após o carregamento o cliente pode alugar os filmes que pretender. O custo do aluguer será debitado ao saldo da conta do MyVideoOnDemand do cliente. Considere que o carregamento mínimo é 10 euros.

Para a sua aplicação, irá ter que definir uma classe Java `ManageVideoRental`, cujos objectos fornecerão um conjunto de operações apropriadas para suportar as funcionalidades associadas ao aluguer de filmes. Considere que há diferentes tipos de filmes: “*novidades*”, “*clássicos*” e “*em exibição*”. Cada tipo de filme tem associado um preço diferente, sendo que “*novidades*” têm um custo de 5 euros, os “*clássicos*” 3 euros e “*em exibição*” 4 euros. Considere também que o MyVideoOnDemand divide os clientes em dois grupos: um cliente é “*standard*” ou “*premium*”. Aos clientes “*premium*” é permitido alugarem, mesmo sem saldo disponível, um filme. Ou seja, é permitido que um cliente “*premium*” com saldo insuficiente possa alugar no máximo um filme, ficando então com um saldo final negativo. Após o carregamento da conta, esse cliente recupera o direito a alugar um filme para além do seu saldo.

Note que neste exercício **não terá** que programar, mas apenas que definir o conjunto de operações (métodos e construtores), assim como as variáveis de instância e constantes que achar necessárias para representar o estado de cada objecto da classe `ManageVideoRental`.

A classe `ManageVideoRental` deve permitir consultar a seguinte informação relativa a cada conta do MyVideoOnDemand: nome do cliente, se é um cliente “*standard*” ou “*premium*”, o saldo da conta, o valor gasto nos alugueres de cada um dos vários tipos de filmes, o valor total e o valor médio dos carregamentos efectuados. Deve ser também possível determinar se o cliente pode ou não fazer um aluguer de um determinado tipo de filme.

COM BASE NA INFORMAÇÃO ACIMA:

Indique constantes, variáveis de instância construtores e métodos que a classe `ManageVideoRental` deverá apresentar, para assegurar estas funcionalidades. Não se esqueça de descrever as pré-condições para cada método, sempre que tal se justifique.

Para cada constante, indique o seu tipo e explique a sua finalidade.

Para cada variável indique o seu tipo e explique a sua finalidade.

Para cada método indique o tipo do seu resultado e o tipo dos seus parâmetros (se existirem). Explique ainda a sua finalidade (para que serve) de forma clara e intuitiva.

Indique ainda que métodos são modificadores e que métodos são de consulta.

III – Considere a classe Amplitudes programada em Java.

```
public class Amplitudes {
    private int i,j;
    private float sum;

    public Amplitudes(){
        this.i = 0;
        this.j = 0;
        this.sum = 0;
    }

    // pre: dois>=0
    // pre: um<0
    public Amplitudes(float um, float dois){
        this.i = 1;
        this.j = 1;
        this.sum = dois-um;
    }

    public int getI(){
        return i;
    }

    public int getJ(){
        return j;
    }

    public int getN() {
        return (i+j)/2;
    }

    //pre: getN()>0
    public float getM() {
        return sum/getN();
    }

    // pre: dois >= um
    public void read(float um, float dois)
    {
        int neg = 0;
        int pos = 0;

        if (um<0)
            if (dois<0)
                neg=2;
            else {
                neg=1;
                pos=1;
            }
        else
            if (dois>=0)
                pos=2;
            else {
                neg=1;
                pos=1;
            }
        i = i + neg;
        j = j + pos;
        sum = sum + (dois - um);
    }
}
```

a) Qual poderá ser o objectivo da classe `Amplitude`? Explique de forma clara a finalidade de cada um dos seus métodos e o que representa cada variável de instância. Tipicamente, deve conseguir explicar cada operação ou variável numa frase curta (1 linha).

b) Considere a seguinte sequência de chamadas de métodos a objectos da classe `Amplitude`. A primeira coluna indica o número de linha, para mais fácil referência. A segunda coluna, tem uma instrução em java. Indique, para cada chamada que devolva algum valor, qual é esse valor e qual o tipo desse valor (tal como costuma ver no traço de execução da consola do BlueJ – tem um exemplo de traço no final do grupo IV).

N.	Comando
1.	<code>Amplitudes amp = new Amplitudes (-2.0f, 8.0f);</code>
2.	<code>amp.getI()</code>
3.	<code>amp.getJ()</code>
4.	<code>amp.getM()</code>
5.	<code>amp.read(10.0f, 30.0f);</code>
6.	<code>amp.getI()</code>
7.	<code>amp.getJ()</code>
8.	<code>amp.getM()</code>
9.	<code>amp.read(0.5f, 5.0f);</code>
10.	<code>amp.getM()</code>
11.	<code>amp.getJ()</code>
12.	<code>amp.getI()</code>

Por favor, apresente a sua resposta com o seguinte formato (1 linha para cada valor retornado, em que *n* representa o número da linha no traço acima reproduzido, *xxx* o valor retornado e *yyy* o tipo de dados do valor retornado):

Linha *Valor retornado* *Tipo*
n *xxx* (*yyy*)

IV – Este exercício visa a construção de uma classe `ScoreCardGas` que permite criar objectos que representam um cartão de pontos numa gasolinheira. Cada cartão de pontos tem associado o número de pontos acumulados (valor inteiro).

Com um cartão de pontos é possível realizar compras na loja da gasolinheira, sempre e quando os pontos acumulados sejam iguais ou superiores ao valor (em pontos) da compra. Note que cada produto na loja tem sempre associado um valor inteiro que indica o seu custo em pontos.

Os pontos são acumulados no cartão cada vez que se realiza um abastecimento de gasolina e se apresenta o cartão. Por cada abastecimento serão agregados pontos no cartão, que resultam do arredondamento para as unidades do valor correspondente a 20% do total em litros de gasolina ou gásóleo colocados num veículo.

De modo a incentivar os abastecimentos na gasolinheira, após cada 5 abastecimentos serão agregados pontos no cartão. O número de pontos agregados resulta do arredondamento para as unidades do valor correspondente a 50% da média dos litros colocados nos últimos 5 abastecimentos.

A criação de cartões de pontos pode ser realizada de duas formas diferentes:

- Não indicando nada, ficando o cartão com zero pontos;
- Indicando o número de pontos iniciais do cartão.

Cada objecto da classe `ScoreCardGas` possui as seguintes operações:

```
/* Indica o número de pontos do cartão
 */
public int score();

/* Indica o número de abastecimentos realizados com o cartão
 */
public int suppliesNumber();

/* Abastecimento de gasolina, correspondente a amountLiters litros.
 * O abastecimento implica sempre a adição de pontos no cartão (valor
 * inteiro resultante do arredondamento de 20% do amountLiters).
 * O abastecimento poderá ainda implicar a adição de pontos no cartão
 * (valor inteiro resultante do arredondamento de 50% da média dos
 * últimos 5 abastecimentos), após cada 5 abastecimentos realizados.
 */
public void supplyGas(float amountLiters);

/* Compra na loja da gasolinheira no valor de amount pontos.
 * Pre: amount <= score()
 */
public void purchase(int amount);
```

Exemplo de traço de execução:

```
ScoreCardGas myCard = new ScoreCardGas();
myCard.score()
0 (int)
myCard.suppliesNumber()
0 (int)
myCard.supplyGas(20.5f);
myCard.score()
4 (int)
myCard.suppliesNumber()
1 (int)
myCard.supplyGas(20.5f);
myCard.supplyGas(20.5f);
myCard.supplyGas(20f);
myCard.score()
16 (int)
myCard.suppliesNumber()
4 (int)
myCard.supplyGas(21f);
myCard.suppliesNumber()
5 (int)
myCard.score()
30 (int)
myCard.supplyGas(60.5f);
myCard.score()
42 (int)
myCard.suppliesNumber()
6 (int)
myCard.supplyGas(60.5f);
myCard.supplyGas(60.5f);
myCard.supplyGas(60.5f);
myCard.suppliesNumber()
9 (int)
myCard.supplyGas(60.5f);
myCard.score()
120 (int)
myCard.purchase(80);
myCard.score()
40 (int)
myCard.supplyGas(6.9f);
myCard.score()
41 (int)
myCard.suppliesNumber()
11 (int)
```

Programme em Java a classe ScoreCardGas.