1º Teste de "Introdução à Programação" (2009/10)

Duração 2H

Instruções importantes:
Responda a todos os grupos em folhas separadas.
Identifique a folha de rosto do caderno de exame com o seu nome e número.
Antes de começar a resolver um exercício, leia o enunciado do princípio até ao fim.
Pode usar caneta ou lápis. Recomendamos que use o lápis.
Não é permitido consultar elementos para além deste enunciado.
Qualquer tentativa de fraude comprovada acarretará a reprovação na disciplina.
Não é permitido sair da sala antes que o teste 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 (faça uma tabela com as linhas do programa nas linhas e variáveis nas colunas).

a)
```
1 {
2   String a;
3   String b;
4   int k = 0;
5   a = ”Ben 10”;
6   b = a;
7   k = k + 2;
8   a = a + 1;
9 }
```

b)
```
1 {
2   int a;
3   int b;
4   a = 10;
5   b = a + a;
6   if (b > a + 5)
7     a = a + 1;
8   else
9     a = a - 1;
10 }
```

c)
```
1 {
2   int x1;
3   int x2;
4   boolean b;
5   x1 = 1;
6   x2 = 1;
7   x1 = x2 * x1;
8   x2 = x2 + 2;
9   x1 = x2 * x1;
10  b = (x2 > x1);
11  if (b && (x1 > 5)) {
12        x1 = 5 + x1;
13  }
14  else x2 = x1;
15 }
```

II – Considere que necessita de programar uma aplicação para gerir as notas das disciplinas da LEI. Para isso irá ter que definir uma classe Java `GradeBook`, cujos objectos fornecerão um conjunto de operações apropriadas para fornecer essa funcionalidade.

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

Cada `GradeBook` deve poder informar a média ponderada corrente do curso. Para isso, deverá poder receber a classificação, o número de créditos ECTS obtidos e a área científica de cada uma das cadeiras em que o aluno já tenha sido aprovado. Não é necessário guardar as notas individuais, ou o número de ECTS obtidos em cada uma das disciplinas em particular, mas estes valores são importantes para o cálculo da média ponderada.

As áreas científicas são: "Informática" (mínimo 98 ECTS), "Matemática" (mínimo 31 ECTS), "Ciências Humanas e Sociais" (mínimo 9 ECTS), "Engenharia Electrotécnica" (mínimo 6 ECTS), "Física" (mínimo 6 ECTS), "Outra" (mínimo 0 ECTS).

Em cada momento, deve ser possível consultar o número de créditos ECTS já realizados em cada uma das áreas científicas, assim como o número de créditos total já realizado, além da já referida média ponderada corrente do curso.

Deve ainda ser possível consultar se já estão ou não realizados o número mínimo de créditos ECTS em cada uma das áreas.

COM BASE NA INFORMAÇÃO ACIMA:

Indique variáveis de objecto, métodos e construtores que a classe `GradeBook` deverá apresentar, para assegurar as funcionalidades pretendidas.

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 **observadores**.

Nota: por exemplo, um aluno que tenha concluído IP (8 ECTS), ISRC (6 ECTS) e CHS (3 ECTS) acumulou 14 ECTS em "Informática" e 3 ECTS em "Ciências Humanas e Sociais". Assim, para efeitos de cálculo de média, a nota de IP tem peso 8, a de ISRC tem peso 6, e a de CHS tem peso 3. Ou seja, a média seria (notaIP*8+notaISRC*6+notaCHS*3)/(8+6+3).

III – Considere a classe `Toll` programada em Java.

```
class Toll {
   static final int CANCEL = -1;
   static final float BASIC_RATE = 1.0f;
   static final float EXTRA_RATE = 0.5f;

  int numberVehicles;
  float totalFee;
  String local;

  Toll(String place) {
     local = place;
     totalFee = 0.0f;
     numberVehicles = 0;
   }

  float pass(boolean vehicleClass, float payment) {
     float toPay;
     float result;
     toPay = 0.0f;
     if(vehicleClass)
        toPay = toPay + BASIC_RATE;
     else {
        toPay = toPay + BASIC_RATE + EXTRA_RATE;
     }
     if(toPay > payment)
        result = CANCEL;
     else {
        result = payment-toPay;
        totalFee = totalFee + toPay;
        numberVehicles = numberVehicles + 1;
     }
     return result;
   }

   int traffic() {
     return numberVehicles;
   }
   float totalFee() {
     return totalFee;
   }
   String getLocal() {
     return local;
   }
}
```

a) Qual poderá ser o objectivo da classe `Toll`? Explique de forma clara a finalidade de cada um dos seus métodos.

b) Considere a seguinte sequência de chamada de métodos a objectos da classe `Toll`. Indique, para cada chamada que devolva um resultado, que resultado é esse.

```
Toll t1 = new Toll("Ponte 25 Abril");
t1.pass(false,2.0f)
t1.pass(true,0.5f)
t1.pass(true,3.0f)
t1.traffic()
t1.totalFee()
t1.getLocal()
```

IV – Este exercício visa a construção de uma classe Java `Game31` para implementar o famoso jogo tradicional português *"Trinta e Um de Boca"*, em vias de internacionalização sob a desginação *"31 out of your Mouth"*.
O jogo é para dois jogadores e o objectivo é determinar qual é o jogador que diz primeiro *"31"*! As regras são as seguintes:

Regra 1. O primeiro jogador diz *"1"*, *"2"*, ou *"3"*.
Regra 2. Cada jogador só pode dizer um número maior que o último número dito, mas que não o exceda em mais do que três unidades. Por exemplo, se a primeira jogada for *"2"*, a segunda só pode ser *"3"*, *"4"* ou *"5"*.
Regra 3. Os dois jogadores continuam a jogar alternadamente, mas nenhum pode dizer um número maior ou igual a *"31"*. Perde o jogador que for obrigado a dizer *"31"* (de boca) ou superior.

Cada objecto da classe `Game31` possui as seguintes operações.

> **`void`** `startGame()`
> permite começar (ou recomeçar) um novo jogo.
>
> **`int`** `getPlay()`
> indica o último número válido jogado (ou 0 se não começou o jogo).
>
> **`int`** `nextToPlay()`
> indica quem deve ser o próximo jogador a jogar (1,2, ou 0 se o jogo acabou).
>
> **`boolean`** `validPlay(`**`int`** `move)`
> indica se o próximo a jogar pode ou não jogar `move`, de acordo com a **Regra 2**.
>
> **`int`** `playerOne(`**`int`** `move)`
> joga com o primeiro jogador.
>
> O resultado devolvido dever ser:
>
> -1 (menos um) se a jogada é inválida (e tem que ser repetida)
> 0 (zero) se a jogada foi aceite.
> 1 (um) se o jogo acabou com a vitória do jogador 1.
> 2 (dois) se o jogo acabou com a vitória do jogador 2.
>
> **`int`** `playerTwo(`**`int`** `move)`
> joga com o segundo jogador. O formato do resultado é semelhante ao de `playerOne`.

Apresentamos um exemplo de utilização da classe `Game31`:

```
Game31 g = new Game31();
g.getPlay()
0   (int)
g.nextToPlay()
1   (int)
g.playerOne(3)
0   (int)
g.validPlay(4)
true   (boolean)
g.validPlay(8)
false   (boolean)
g.playerOne(4)
-1   (int)
g.getPlay()
3   (int)
g.playerTwo(3)
-1   (int)
g.playerTwo(8)
-1   (int)
g.nextToPlay()
2   (int)
g.playerTwo(6)
0   (int)
g.playerOne(9)
0   (int)
g.playerTwo(11)
0   (int)
g.playerOne(14)
0   (int)
g.playerTwo(17)
0   (int)
g.playerOne(20)
0   (int)
g.playerTwo(21)
0   (int)
g.playerOne(24)
0   (int)
g.playerTwo(27)
0   (int)
g.playerOne(30)
0   (int)
g.playerTwo(30)
-1   (int)
g.nextToPlay()
2   (int)
g.playerTwo(31)
1   (int)
g.nextToPlay()
0   (int)
g.getPlay()
31   (int)
g.startGame();
g.nextToPlay()
1   (int)
g.getPlay()
0   (int)
```

Programe em Java a classe `Game31`.

****

****