1º Teste de "Introdução à Programação" (2010/11)

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 (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     boolean a = true;
3     boolean b = false;
4     String k = "Harry Potter 3";
5     a = a || b;
6     b = a;
7     k = k + 2;
8     a = !b;
9 }
```

b)

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

c)

```
1 {
2     int x1 = 3;
3     int x2 = 5;
4     x1 = x2 % x1;
5     switch (x1) {
6           case 1: x2 = x2 + 2; break;
7           case 2: x2 = x2 * x1;
8           case 3: x2 = x2 / 2; break;
9           default: x2 = x2 + x1; break;
10    }
11 }
```

II – Necessita de conceber uma aplicação para gerir baterias de carros eléctricos. Um carro eléctrico é um automóvel que utiliza propulsão por meio de motores eléctricos. A energia necessária para estes veículos é normalmente disponibilizada por baterias amovíveis, que são trocadas de forma automática em postos de abastecimento. Para a sua aplicação, irá ter que definir uma classe Java `ElectricCarBattery`, cujos objectos fornecerão um conjunto de operações apropriadas para suportar a funcionalidade típica de uma dessas baterias. Considere que há diferentes tipos de baterias: *"chumbo ácido"*, de *"níquel de ferro"*, de *"níquel metal hidreto"* e de *"zinco-ar"*. Cada tipo tem associado uma durabilidade, expressa em kilómetros.

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 `ElectricCarBattery`.

Cada `ElectricCarBattery` caracteriza-se por uma potência (um número real), eficiência (percentagem de energia efectivamente usada na locomoção, face à energia total acumulável na bateria), capacidade de armazenamento de energia, e tipo de bateria.

À medida que o carro eléctrico se desloca, a sua bateria vai-se descarregando, de acordo com as características da bateria e do número de kilómetros percorridos desde o fabrico.

Em cada momento, deve ser possível consultar as características acima referidas (potência, eficiência, capacidade de armazenamento, capacidade actual de energia, total de kilómetros percorridos desde o fabrico, e tipo de bateria).

Além disso, deve ser possível calcular a autonomia do veículo, em kilómetros, dadas a sua velocidade actual, peso do veículo e o seu índice de aerodinâmica (tudo números reais). Note que a fórmula para calcular a autonomia do veículo com a carga actual da bateria não é relevante para este exercício. Pode assumir que os dados da velocidade, peso e índice, combinados com os dados da bateria acima descritos são suficientes.

Finalmente, deve ser possível calcular a percentagem de kilómetros já percorridos, face à durabilidade típica da bateria (esse critério é usado para decidir quando se deve reciclar a bateria, por ter terminado a sua vida útil).

COM BASE NA INFORMAÇÃO ACIMA:
Indique constantes, variáveis de instância construtores e métodos que a classe `ElectricCarBattery` deverá apresentar, para assegurar estas funcionalidades.

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 observadores.

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

```
public class FootballClub {
   private int a, b, c;
   private int x, y;

   public Club() {
      a = 0;
      b = 0;
      c = 0;
      x = 0;
      y = 0;
   }

   public void setNewResult(int goalsFor, int goalsAgainst) {
      if(goalsFor > goalsAgainst)
         a++;
      else if(goalsFor == goalsAgainst)
         c++;
      else b++;
      this.x += goalsFor;
      this.y += goalsAgainst;
   }

   public int getA() { return a; }

   public int getB() { return b; }

   public int getC() { return c; }

   public int getX() { return x; }

   public int getY() { return y; }

   public int getZ() { return x - y; }
}
```

a) Qual poderá ser o objectivo da classe `FootballClub`? 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 `FootballClub`. Indique, para cada chamada que devolva um resultado, que resultado é esse.

```
FootballClub fc = new FootballClub();
fc.getA()
fc.getB()
fc.setNewResult(5, 0);
fc.getA()
fc.getB()
fc.setNewResult(2, 3);
fc.getA()
fc.getB()
fc.setNewResult(0, 1);
fc.getA()
fc.getB()
fc.getX()
fc.getY()
fc.getZ()
```

IV – Implemente em Java uma classe `BookingOffice` cujos objectos representam bilheteiras de espectáculos. Cada espectáculo tem uma capacidade limitada de lugares e um preço associado à venda do bilhete. Na criação de uma nova bilheteira, é indicado o nome do espectáculo, a lotação máxima (capacidade) e o preço do bilhete "normal". Na venda de bilhetes existe um desconto de 20% para bilhetes destinados a espectadores com menos de 18, ou mais do que 65 anos de idade. A sua classe deve permitir construir instâncias de bilheteiras, dados o nome do espectáculo, o preço do bilhete normal e a capacidade para esse espectáculo (quando se cria uma nova bilheteira, o número de bilhetes vendidos à partida é zero e a bilheteira fica "aberta"). Além disso, deve implementar as seguintes operações:

```
/* Abre a bilheteira, isto é, pode-se começar a vender bilhetes para
 * uma nova sessão do espectáculo. Se a bilheteira já estiver aberta
 * não faz nada.
 */
public void open()

/* Indica se a bilheteira está aberta.
 */
public boolean isOpen()

/* Compra de um bilhete (ocupação de um lugar), caso ainda exista
 * lugar. Se existir lugar e a bilheteira estiver aberta devolve
 * o preço do bilhete e ocupa um lugar.
 * Se não existirem lugares ou a bilheteira estiver fechada,
 * este método devolve 0.
 */
public double sellTicket (int age)

/* Fecha a bilheteira e devolve o dinheiro em caixa (recebido pela
 * venda de bilhetes). Esta operação só pode ser realizada se a
 * bilheteira estiver aberta.
 */
public double close()

/* Indica o número de bilhetes vendidos até ao momento
 */
public int occupation()

/* Devolve o nome do espectáculo
 */
public String nameShow()

/* Devolve a capacidade (número de lugares).
 */
public int capacity ()

/* Devolve o preço dum bilhete.
 */
public double ticketPrice()

/* indica o número de bilhetes vendidos a
 * pessoas que tem desconto de 20%
 */
public int occupationWithDiscount();
```

Apresentamos um exemplo de utilização da classe `BookingOffice`:

```
BookingOffice movie = new BookingOffice("O Panda do Kung Fu", 6, 10.0);

movie.nameShow()
"O Panda do Kung Fu" (String)
movie.ticketPrice()
10.0 (double)
movie.capacity()
6 (int)
movie.isOpen()
true (boolean)
movie.occupation()
0 (int)
movie.sellTicket(18);
10.0 (double)
movie.sellTicket(17);
8.0 (double)
movie.sellTicket(19);
10.0 (double)
movie.occupation()
3 (int)
movie.capacity()
6 (int)
movie.sellTicket(21);
10.0 (double)
movie.sellTicket(68);
8.0 (double)
movie.sellTicket(65);
10.0 (double)
movie.occupation()
6 (int)
movie.sellTicket(45);
0.0 (double)
movie.occupation()
6 (int)
movie.capacity()
6 (int)
movie.occupationWithDiscount()
2 (int)
movie.isOpen()
true (boolean)
movie.close()
56.0 (double)
movie.nameShow()
"O Panda do Kung Fu" (String)
movie.ticketPrice()
10.0 (double)
movie.capacity()
6 (int)
movie.isOpen()
false (boolean)
movie.occupation()
0 (int)
movie.open();
movie.nameShow()
"O Panda do Kung Fu" (String)
movie.ticketPrice()
10.0 (double)
movie.capacity()
6 (int)
movie.isOpen()
true (boolean)
movie.occupation()
0 (int)
```

Programe em Java a classe `BookingOffice`.