1º Teste de "Introdução à Programação (A)" 2007-08

Duração: 1:30H

Instruções importantes:

Responda a todos os grupos em folhas separadas.
Identifique todas as folhas com o seu número e nome.
Antes de começar a resolver um exercício, leia o enunciado do princípio até ao fim.
Pode usar caneta ou 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 – Determine o valor **que todas e cada uma** das variáveis declaradas no bloco deverá conter, no fim de cada instrução que for executada (para o ajudar, as linhas do bloco estão numeradas).

a)

```
1.   {   int a;
2.       int b;
3.       int c;
4.
5.       a = 1;
6.       b = a+1;
7.       c = b+1:
8.       b = a;
9.       a = c;
10.      c = b;
11.  }
```

b)

```
1.   {   int d;
2.       int b;
3.       int c;
4.
5.       b = -1;
6.       c = 2;
7.       d = b + c;
8.       if (b>c) {
9.         b--;
10.        if (b < 0) c++;
11.          else b++;
12.      } else if (d < b)
13.                d++;
14.              else b--;
15.  }
```

II – Considere que necessita de implementar uma classe para representar controladores de portas com alarme. Neste exercício, **não terá** que programar, mas sim que definir a lista de operações assim como as variáveis e constantes que achar necessárias para implementar a memória (ou estado) de cada objecto.

Uma porta com alarme só poderá ser aberta se antes tiver sido dada uma combinação secreta de 6 dígitos (combinação de acesso), ou seja, para abrir a porta com sucesso é necessário introduzir uma certa sequência de 6 dígitos. Para introduzir essa sequência e para abrir a porta, encontra-se, junto à entrada da porta, um dispositivo com as seguintes teclas:

• 10 teclas representando, cada uma, um dígito entre 0 e 9;
• Tecla "clear", que permite esquecer todos os números introduzidos até ao momento;
• Tecla "open" para abrir a porta.

Uma sequência de três tentativas sem sucesso para abrir a porta faz soar o alarme. Se o alarme estiver a soar, não é possível abrir a porta. Durante a introdução da combinação, se o utilizador da porta se enganar na introdução de algum dígito, pode sempre pressionar a tecla "clear".

Para desligar o alarme é necessário carregar num botão que existe no lado do dispositivo e que permite fazer "reset" do mesmo. Quando executado o "reset", o dispositivo volta ao estado inicial, o que permite fazer três novas tentativas de abertura da porta.

Além de aceitar uma combinação de acesso, para efeitos de abertura da porta, o dispositivo poderá ainda ser usado para redefinir a combinação. No entanto, isto só poderá ser efectuado se a porta estiver aberta.

Os controladores podem ser criados com uma combinação de acesso pré-definida pelo fabricante, ou com uma combinação de acesso indicada pelo cliente.

a) Para cada operação e para cada construtor que tiver identificado, descreva a sua funcionalidade através de um comentário. Para as operações indique, claramente, o que a operação devolve. Relembre que não tem que programar as operações.

b) Defina as variáveis de instância da classe (memória ou estado dos objectos da classe). Para cada uma delas indique o tipo, nome e escreva uma pequena descrição do que representa.

III – Considere a seguinte classe em Java.

```
class Translation {

int w1;
int w2;

Translation(int p1, int p2) {
  if (p1>p2) {
     w2 = p1;
     w1 = p2;
  } else {
     w1 = p1;
     w2 = p2;
  }
}

int clamp(int w) {
int v;
  v = -1;
  if(w > w2)
    v = w2;
  else if (w < w1)
          v = w1;
       else v = w;
  return v;
}

void right() {
  w2 = w2 + 1;
}

void left() {
  w1 = w1 - 1;
}

}
```

a) Explique qual poderá ser a funcionalidade e objectivo da classe `Translation`.

b) Considere a seguinte sequência de chamada de métodos a dois objectos da classe `Translation`. Indique, para cada chamada que devolva um resultado, que resultado é esse (para o ajudar, as linhas da sequência estão numeradas).

```
1.   Translation t1 = new Translation(0,10);
2.   Translation t2 = new Translation(-1,1);
3.
4.   t1.clamp(5)
5.   t1.clamp(-1)
6.   t1.clamp(12)
7.   t1.clamp(9)
8.   t1.right();
9.   t1.clamp(11)
10.  t1.left();
11.  t1.clamp(-1)
12.  t1.clamp(0)
13.  t1.clamp(-2)
14.  t2.clamp(1)
15.  t2.clamp(t1.clamp(12))
```

IV – Este exercício destina-se a resolver um pequeno problema de programação. Pretende-se desenvolver em Java uma classe `CandyDispenser` cujos objectos representam máquinas de vender rebuçados. Estão disponíveis dois tipos de serviço: básico e especial.

Quando um objecto é criado, é indicado:
a) quantos rebuçados pode conter no máximo o seu depósito de rebuçados;
b) qual é o número de rebuçados a fornecer em cada serviço base e em cada serviço especial, respectivamente;
c) qual é o custo de cada rebuçado, em cêntimos.

Neste problema, todos os valores serão representados em cêntimos.

Cada objecto `CandyDispenser` disponibiliza as seguintes operações:

```
void serveBasic()
```

fornece um serviço básico de rebuçados, se tiver "stock" suficiente. Se não houver "stock" o pedido é ignorado e o pagamento não deve ser considerado (imagine que será devolvido ao utilizador).

```
void serveSpecial()
```

fornece um serviço "especial" de rebuçados, se tiver "stock" suficiente. Se não houver "stock" o pedido é ignorado e o pagamento não deve ser considerado (imagine que será devolvido ao utilizador).

```
int getValueSold()
```

indica o valor total das vendas que a máquina já efectuou, desde a última vez que foi retirado o dinheiro do depósito de moedas.

```
int getValueOnSale()
```

indica o valor de venda total dos rebuçados existentes no depósito.

```
void extractMoney()
```

extrai da máquina todo o dinheiro acumulado no depósito de moedas.

```
void reFill(int units)
```

acrescenta mais `units` rebuçados no depósito de rebuçados. O total de rebuçados no depósito nunca pode exceder o limite indicado quando a máquina foi criada. Se fôr tentado um valor maior, o excesso deverá ser ignorado.

```
int sellingRecord()
```

indica o maior valor em dinheiro que alguma vez a máquina conteve no depósito, desde que foi colocada em operação.

Exemplo

```
CandyDispenser c = new CandyDispenser(20,2,4,2);
c.serveBasic();
c.serveBasic();
c.reFill(30);
c.serveBasic();
c.serveSpecial();
c.serveSpecial();
c.getValueOnSale()
20   (int)
c.getValueSold()
20   (int)
c.serveBasic();
c.getValueOnSale()
16   (int)
c.getValueSold()
24   (int)
c.extractMoney();
c.serveSpecial();
c.serveSpecial();
c.getValueSold()
16   (int)
c.reFill(100);
c.getValueOnSale()
40   (int)
c.sellingRecord()
24   (int)
```

Programe em Java a classe `CandyDispenser`.

****

FIM

****