Nome:	Número:

Programação Orientada por Objectos 2009/10

1º Exame
1 de Julho de 2010

Instruções (leia com cuidado):

- Escreva de forma CLARA o seu nome e número em todas as folhas.
- O exame contém 8 páginas dividido em 4 partes. Confirme que tem um exame completo.
- A cotação de cada pergunta é indicada junto à questão e encontra-se resumida no quadro em baixo.
- Tem 2 horas para responder ao exame.
- Para planear melhor o seu tempo leia todos os problemas antes de começar.
- Este exame NÃO permite consulta. Deverá responder às questões no espaço disponível, usando a parte de trás das folhas, se necessário.
- **Sobre a mesa deverá encontrar-se APENAS este exame, uma caneta e o seu cartão de identificação.**
- Desligue o telemóvel. O seu uso anula o exame.

Parte	Problema	Descrição	Pág.	Valores
I	1 a) b)	UML	2	1.5
II	2 a) b) c) d)	Java: desenvolvimento	4	3.0
	3 a) b) c) d) e)	Java: escolha múltipla	5	2.5
	4	Java: miscellaneous	7	1.0
III	5	XML	8	1.0
IV	6	C++	8	1.0
Total				**10.0**

Nome:	Número:

Parte I -- UML (1.5 valores)

1 – Considere um sistema que controla vários dispositivos de uma casa.

Os dispositivos podem ser CD players, TV's e iluminação. A iluminação pode ser de emergência ou regular. A iluminação regular, CD players e TV's devem implementer a interface `ISwitchable`, com métodos `switchOff` e `switchOn` (sem parâmetros e sem retorno), significando que podem ser automaticamente desligados pelo Sistema de Controlo da Casa. Por outro lado, as luzes de emergência nunca são desligadas à noite.

Considere que um dispositivo está sempre associado a uma divisão da casa. Para mais, as divisões contêm: (i) vários dispositivos; e (ii) um nome. Considere que as divisões da casa apenas podem ser halls, salas e quartos. O hall é uma divisão que contém sempre iluminação de emergência, enquanto que as salas e os quartos podem conter qualquer dispositivo.

O Sistema de Controlo da Casa deverá ter um método `main` (como em Java) e quatro outros métodos: (i) `goToSleep`, sem parâmetros; (ii) `wakeUp`, sem parâmetros; (iii) `addHomeDevice`, que adiciona um dispositivo ao Sistema de Controlo; e (iv) `addHouseRoom`, que adiciona uma divisão ao Sistema de Controlo. Os dispositivos devem existir no Sistema de Controlo sempre associados a uma divisão. As divisões da casa são sempre construídas com um nome.

a) [1.0 valores] Defina o diagrama de classes em UML para o problema apresentado.

Nome:	Número:

b) [0.5 valores] Defina um diagrama de objectos em UML tendo em consideração que o Sistema de Controlo da Casa tem quatro dispositivos: uma TV, duas iluminações regulares e uma iluminação de emergência. Os referidos dispositivos encontram-se todos num mesmo quarto. Coloque em todos os atributos valores *dummy*.

Nome: Número:

Parte II -- Java (6.5 valores)

2 – Uma função abstracta é uma entidade matemática que recebe uma entrada, faz um cálculo e produz uma saída. Tal definição é denominada *lambda*. Em POO, podemos modelar funções abtractas como uma interface, chamada `ILambda`, que define um método `apply` que recebe um `Object x` como parâmetro e retorna um `Object`.

a) **[0.5 valores]** Defina a interface `ILambda`.

`ILambda:`

b) **[0.5 valores]** Defina duas funções concretas *lambda*, chamadas `Lambda1` e `Lambda2`, para as quais o método `apply` deverá retornar:
- uma `String` que concatena `"Lambda1 applied to "` com a informação textual (dada pelo método `toString`) do `Object` recebido como parâmetro, no caso de `Lambda1`;
- uma `String` que concatena `"Lambda2 applied to "` com a informação textual (dada pelo método `toString`) do `Object` recebido como parâmetro, no caso de `Lambda2`.

`Lambda1:`

`Lambda2:`

c) **[0.5 valores]** Considere uma interface, chamada `ILogical`, que represeta objectos com a capacidade de aplicar um de dois possíveis *lambdas*. Em POO a interface `ILogical` define um método `select` com três parâmetros: dois `ILambda`'s e um `Object`. O método `select` chama o método `apply` do 1º ou 2º `ILambda` com o `Object` recebido como parâmetro, e retorna o `Object` devolvido na chamada ao método `apply`. Defina a interface `ILogical`.

`ILogical:`

Nome:	Número:

d) **[1.5 marks]** Defina duas classes concretas, chamadas `True` e `False`, que implementam a interface `ILogical` e que satisfazem a especificação seguinte. Quando

- `b.select(new Lambda1(), new Lambda2(), "hello");`

é executada sobre uma variável `b` de tipo `ILogical`, o valor retornado deverá ser

- `"Lambda1 applied to hello"`, quando `b` é de tipo `True`, e
- `"Lambda2 applied to hello"`, quando `b` é de tipo `False`.

O código para `True` e `False` não deverá conter qualquer literal do tipo `String`. Exemplifique a chamada a `select` sobre um objecto de tipo `True` num método `main` dentro duma classe `Main`.

`True:`

`False:`

`Main:`

3 – Preencha as respostas às perguntas de escolha múltipla na seguinte tabela (use apenas maiúsculas). Se quiser corrigir alguma resposta risque a incorrecta e escreva ao lado a resposta correcta. Cada resposta correcta vale 0.5 valores. Uma questão não respondida vale 0 valores, enquanto que uma resposta incorrecta desconta 0.2 valores.

Pergunta	a)	b)	c)	d)	e)
Resposta					

a) **[0.5 valores]** Qual o valor do atributo de classe `x` e do atributo de instância `y` após a construçao de um segundo objecto de tipo `X`?

```
class X {
      static int x=0;
      int y=-1;
      { y=1; }
      public X() { x=x+y; }
}
```

A. `x=1 e y=1`
B. `x=2 e y=1`
C. `x=-1 e y=-1`
D. `x=-2 e y=-1`
E. Nenhuma das anteriores

b) **[0.5 valores]** Assuma que estamos a importar `java.util.Arrays`. O que é imprimido para o terminal?

```
int[][] tiVar1 = new int[3][3];
int[][] tiVar2 = new int[3][3];
System.out.print(tiVar1==tiVar2);
System.out.print(tiVar1.equals(tiVar2));
System.out.print(Arrays.equals(tiVar1,tiVar2));
System.out.print(Arrays.deepEquals(tiVar1,tiVar2));
```

A. `true true true true`
B. `false true true true`
C. `false false true true`
D. `false false false true`
E. Nenhuma das anteriores

c) **[0.5 valores]** O que é imprimido para o terminal?

```
class X{
      public int xpto(){return 5;}
      public static int foo() {return 15;}
      void test(){
            System.out.print(foo());
            System.out.print(xpto());
            System.out.print(((Y)this).foo());
            System.out.print(((X)this).xpto());
      }
}
class Y extends X{
      public int xpto(){return 10;}
      public static int foo() {return 20;}
      public static void main(String[] args){
            new Y().test();
      }
}
```

A. `15 5 20 5`
B. `20 10 20 10`
C. `15 10 20 10`
D. `20 10 20 5`
E. Nenhuma das anteriores

d) **[0.5 valores]** Um `HashSet<?>` denota:

A. `HashSet<Object>`
B. `HashSet<? super Object>`
C. `HashSet<? extends Object>`
E. Nenhuma das anteriores

e) **[0.5 valores]** Que tipo de excepção um `NullPointerException` é?

A. `Error`
B. `Checked exception`
C. `Unchecked exception`
D. Nenhuma das anteriores

Nome:	Número:

4 – [1.0 valores] Considere o programa seguinte, onde a classe `IntPair` representa um par de inteiros.

```
public class IntPair {
      private final int first;
      private final int second;
      public IntPair(int first, int second) {
            this.first = first;
            this.second = second;
      }
      @Override public boolean equals(Object o) {
            if (!(o instanceof IntPair)) return false;
            IntPair ip = (IntPair) o;
            return ip.first==first;
     }
     @Override public int hashCode() {
            return 31*first;
     }
     public static void main(String[] strings){
            Set<IntPair> s = new HashSet<IntPair>();
            for (int i=0; i<10; i++)
                  for (int j=0;j<10;j++)
                        s.add(new IntPair(i,j));
            System.out.println(s.size());
            System.out.println(s.contains(new IntPair(0,100)));
     }
}
```

Era de esperar que o programa anterior imprimisse `100` e `false`, mas após correr o programa descobrimos que não imprime `100` e `false` mas `10` e `true`! Encontre o erro e corrija-o. Justifique a sua resposta. Nota: Os conjuntos não contêm duplicados.

Nome:	Número:

Parte III -- XML (1 valor)

5 – [1.0 valores] Considere o problema apresentado na Parte I e, em particular, o problema 1b), isto é, considere o Sistema de Controlo da Casa com quatro dispositivos associados a um quarto. Informalmente, apresente um documento XML, com elementos e atributos XML, que guarde toda a informção em mãos. Tenha especial atenção à repreentação da relação um-para-muitos entre divisões e dispositivos.

Parte IV -- C++ (1 valor)

6 – [1.0 valores] Implemente em C++ um método `swap`, com parâmetros passados por referência, que troca o valor aos dois parâmetros de tipo `int` (os parâmetros devem ser referências para `int`). Exemplique a sua utilização trocando o valor de duas variáveis de tipo `int`.

`swap:`	
`uso:`	