Nome:	Número:

Programação Orientada por Objectos 2010/11

2º Exame
28 de Junho de 2011

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) e)	Java: desenvolvimento	3	3.0
	3 a) b) c) d) e)	Java: escolha múltipla	6	2.5
	4	Java: outros	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 que se pretende oferecer um Sistema de Gestão para um Hospital (SGH). O SGH pretende modelar os seus trabalhadores e pacientes. Nesse sentido, sobre qualquer pessoa, trabalhador ou paciente, deve guardar o nome, título, sexo, data de nascimento, endereço e telefone. Um trabalhador pertence a um determinado departamento do Hospital e deve guardar ainda a data de admissão e os diversos graus de educação. Por sua vez, um paciente deve conter um historial dos sues problemas de saúde, incluindo para cada ida ao hospital a data, sintomas, diagnóstico, prescrições e todo o pessoal operacional (médico/enfermeiro) que o assistiu. Os médicos devem incluir as suas especialidades. Para além do pessoal operacional o SGH pretende guardar informação sobre o pessoal administrativo, incluindo recepcionistas, e o pessoal técnico. Os recepcionistas devem conter informação sobre o local de trabalho assim como a sua extensão. Por sua vez o pessoal técnico e operacional devem conter um número de telemóvel. Todos os trabalhadores devem conter um número mecanográfico atribuído sequencialmente. O mesmo deve acontecer para os pacientes (o número de paciente deve ser atribuído independentemente do número de trabalhador).

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

b) [0.5 valores] Defina um diagrama de objectos em UML de um paciente que foi assistido por dois médicos e um enfermeiro. Preencha os atributos com valores *dummy*.

Parte II -- Java (6.5 valores)

2 – Pretende-se oferecer um pequeno subconjunto de um jogo de computador de caça ao pato. Neste contexto, considere um conjunto de interfaces e classes que permitam implementar patos com diferentes comportamentos no que diz respeito a grasnar (uns patos fazem "quack", outros "quick", e outros, como o pato Donald, fazem "fa-fa fa-fa-fa") e voar (uns patos voam, outros não; o pato Donald não voa). Para isso considere que tem disponíveis as seguintes interfaces:

```
interface IGrasnar {
      void grasnar();
}
```

```
interface IVoar {
      void voar();
}
```

a) **[0.4 valores]** Defina duas concretizações de `IGrasnar`, chamadas `GrasnarQuack` e `GrasnarFafafafafa`. A operação correspondente ao grasnar deve imprimir para o terminal o som emitido durante o dito.

Nome: Número:

b) **[0.4 valores]** Defina duas concretizações de `IVoar`, chamadas `VoarBemAlto` e `VoarNãoVoa`. A operação correspondente ao voar deve imprimir para o terminal uma mensagem descritiva (tais como "A voar e bem alto" e "Voar não sei").

c) **[0.7 valores]** Defina uma class abstracta `Pato` com três métodos de instância, `realizarGrasnar`, `realizarVoar` e `nadar`. Os dois primeiros métodos devem delegar o grasnar e o voar em atributos de instância de subtipos das interfaces `IGrasnar` e `IVoar`, respectivamente. O método de `nadar` deve imprimir para o terminal "Splash splash".

Nome: Número:

d) **[0.75 valores]** Defina uma classe `PatoDonald` que estenda a classe `Pato` com o comportamento desejado (grasna "Fa-fa fa-fa-fa" e não voa). Defina ainda uma classe `PatoApenas` que estenda a classe `Pato` e que grasna "quack" e voa bem alto.

e) **[0.75 marks**] Defina uma classe `JogoCaçaPato` que no método `main` defina uma estrutura de dados (à sua escolha) que cotenha um `PatoDonald` e um `PatoApenas`. Percorra os elementos dessa estrutura por forma a que sejam realizadas as operações de grasnar, voar e nadar de forma polimórfica.

Nome:	Número:

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]** O que é imprimido para o terminal?

```
String v1="Hello";
String v2="Hell";
String v3="o";
System.out.println(v1==v2+v3);
System.out.println(v1=="Hell"+"o");
System.out.println(v1=="Hello");
```

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

b) **[0.5 valores]** Considere as seguintes classes e identifique, método a método, quais são sobreposições, redefinições ou erros de compilação. A ordem dos métodos nas alternativas (A, B, C e D) é a seguinte: `q`; `r`; `v`; `t`.

```
class X {
      Number q(){…}
      void r(Number x) {…}
      String v() {…}
      <T extends X> T t(){…}
}
```

```
class Y extends X {
      Integer q(){…}
      void r(Integer y) {…}
      <T extends String> T v() {…}
      Y t() {…}
}
```

A. sobreposição; sobreposição; erro de compilação; erro de compilação
B. redefinição; sobreposição; erro de compilação; redefinição
C. redefinição; redefinição; erro de compilação; erro de compilação
D. redefinição; sobreposição; redefinição; erro de compilação
E. Nenhuma das anteriores

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

```
package p1;
import p2.Y;
public class X{
      public void xpto(){System.out.println(5);}
      void ypto(){System.out.println(15);}
      public void test() {
            Y y = (Y) this;
            xpto(); ypto();
            y.xpto(); y.ypto();
      }
}
package p2;
import p1.X;
public class Y extends X{
      public void xpto(){System.out.println(10);}
      public void ypto(){System.ou.println(20);}
      public static void main(String[] args){
            new Y().test();
      }
}
```

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

Nome: Número:

d) **[0.5 valores]** Quantos objectos `Integer` são criados quando fazemos

```
Integer i = new Integer(1); i=i+2;
```

A. Nenhum, trata-se de um tipo primitivo B. 1 C. 2 D. 3 E. Nenhuma das anteriores

e) **[0.5 valores]** Que método de `Object` é chamado quando o `garbage collector` determina que não existem mais referências para o objecto em causa.

A. Não existe tal método B. `delete` C. `destroy` D. `finalize` E. Nenhuma das anteriores

4 – [1.0 valores] De uma maneira geral é utilizada uma tabela no retorno de um método sempre que é necessário devolver zero ou mais objectos de um determinado tipo. Na eventualidade de o método querer devolver zero objectos existem duas formas distintas de o fazer. Identifique-as e exemplifique ambas. Quais as vantagens e as desvantagens? Qual usaria? Justifique.

Nome:	Número:

Parte III -- XML (1 valor)

5 – [1.0 valores] Considere um elemento `Ypto` com conteúdo vazio, que tem um atributo `y` com valor por omissão "ypto". Apresente o respectivo DTD e dois documentos XML, um válido e outro inválido.

Parte IV -- C++ (1 valor)

6 – [1.0 valores] Um método de uma classe em C++ pode ser implementado dentro da própria declaração da classe ou fora (usualmente, num .cpp). Qual é o significado desta diferença? Identifique as vantagens e desvantagens.