Nome:	Número:

Programação Orientada por Objectos 2009/10

2º Exame
21 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	6	2.5
	4	Java: miscellaneous	7	1.0
III	5	XML	8	1.0
IV	6	C++	8	1.0
Total				**10.0**

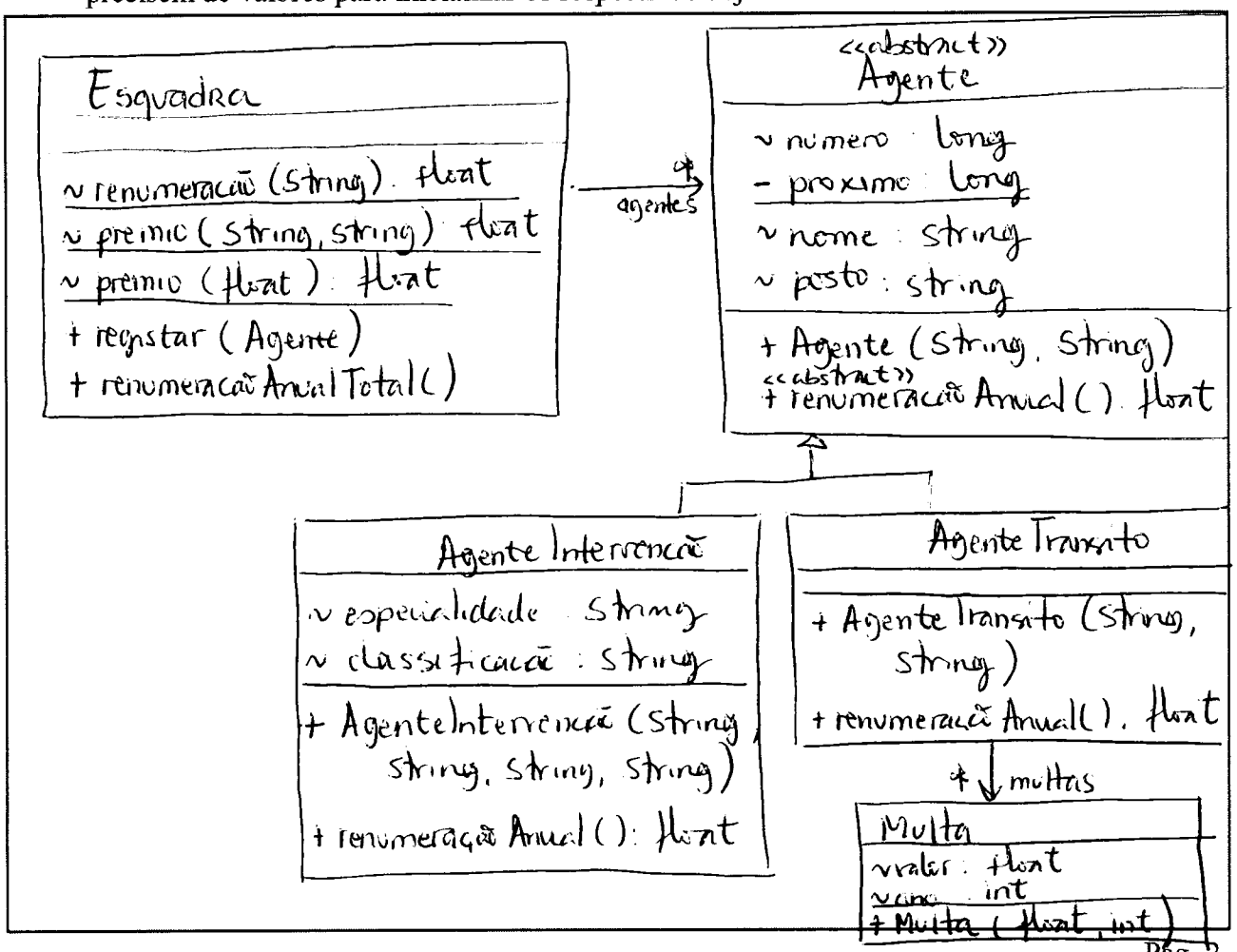
Nome:	Número:

Parte I -- UML (1.5 valores)

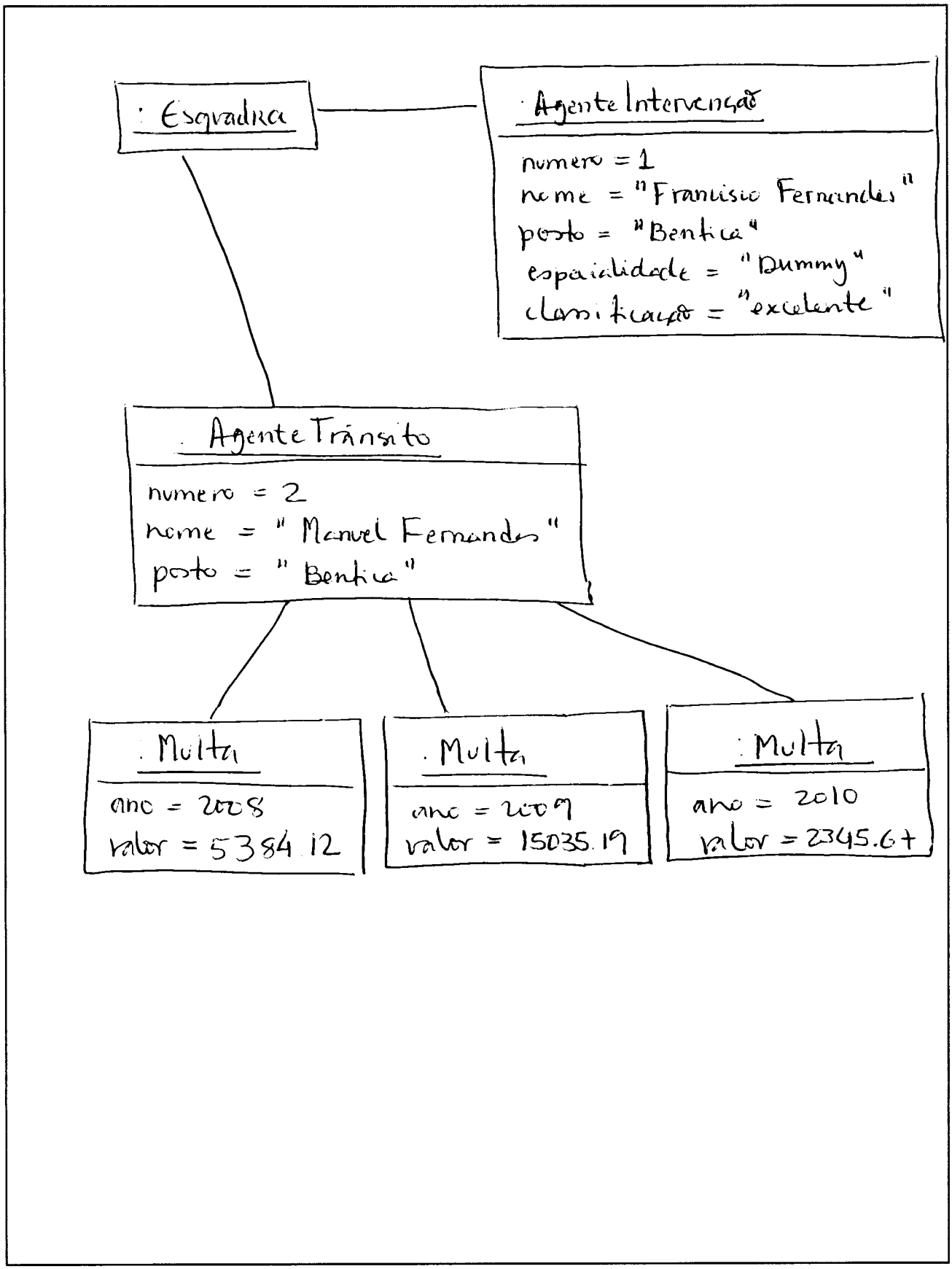
1 – Considere uma esquadra com diversos agentes. A informação relevante sobre os agentes consiste no número, nome e posto. O número de agente deve identificar univocamente o agente em causa e deve ser automaticamente atribuído aos agentes quando estes são registados na esquadra. Os agentes da esquadra podem ser agentes de intervenção ou agentes de trânsito. Relativamente aos agentes de intervenção deve ainda ser guardada a especialidade e classificação anual. No que diz respeito aos agentes de trânsito deve ainda ser guardado um historial do número de multas passadas aos condutores, devendo ser o historial organizado por ano.

A esquadra deverá disponibilizar os seguintes métodos de classe: (i) o método remuneracao que devolve a remuneração mensal de um agente dado o seu posto; (ii) o método premio que devolve o valor de um prémio semestral pago aos agentes de intervenção calculado segundo a respectiva classificação anual e especialidade; (iii) o método premio que devolve o valor de um prémio pago semestralmente aos agentes de trânsito calculado segundo o montante das multas passadas no ano actual. Os agentes deverão implementar um método renumeracaoAnual (sem parâmetros) que devolve a respectiva renumeração anual (remuneração esta que é calculada com base nos métodos de classe disponibilizados pela esquadra). Para mais, a esquadra deverá ainda poder registar um novo agente, através do método registar, e deverá disponibilizar o método renumaracaoAnualTotal que devolve a renumeração total que é necessária pagar aos agentes registados na esquadra.

a) **[1.0 valores]** Defina o diagrama de classes em UML para o problema apresentado. Apresente neste diagrama os construtores das diversas classes do sistema, sempre que estas precisem de valores para inicializar os respectivos objectos.



b) **[0.5 valores]** Defina um diagrama de objectos em UML de uma esquadra com um agente de intervenção e um agente de trânsito com historial de multas desde o ano 2008 até à data de hoje. Coloque em todos os atributos valores *dummy*.



Nome:	Número:

Parte II -- Java (6.5 valores)

2 – Pretende-se implementar a aplicação para apoio à gestão da esquadra (descrita na Parte I) que disponibilize informação sobre os seus agentes. A aplicação de gestão da esquadra baseia-se na classe Esquadra que gere uma única lista - a lista dos seus agentes.

a) **[0.6 valores]** Defina a classe Agente. Ofereça construtores apropriados.

```
public abstract class Agente {

        private static long proximo = 1;
        long numero = proximo++;
        String nome;
        String posto;

        public Agente(String nstr, String pstr){
                nome=nstr;
                posto=pstr;
        }

        abstract float renumeracaoAnual();
}
```

b) **[0.7 valores]** Defina a classe AgenteIntervencao. Ofereça construtores apropriados. Considere que tem disponíveis os métodos de classe da Esquadra.

```
public class AgenteIntervencao extends Agente {

        String especialidade;
        String classificação;

        public AgenteIntervencao(String nstr, String pstr, String estr, String cstr){
                super(nstr,pstr);
                especialidade=estr;
                classificacao=cstr;
        }

        float renumeracaoAnual(){
                return 14*Esquadra.renumeracao(posto)+2*Esquadra.premio(classififcacao,especialidade);
        }
}
```

c) **[0.7 valores]** Defina a classe AgenteTransito. Ofereça construtores apropriados. Considere que tem disponíveis os métodos de classe da Esquadra.

```
public class AgenteTransito extends Agente {

        LinkedList<Long> multas = new LinkedList<Long>();

        public AgenteTransito(String nstr, String pstr){
                super(nstr,pstr);
        }

        float renumeracaoAnual(){
                return 14*Esquadra.renumeracao(posto)+2*Esquadra.premio(multas.getLast());
        }
}
```

d) **[0.5 valores]** Defina a class Esquadra. Relativamente aos métodos de classe, ofereça apenas a assinatura dos métodos, colocando reticências no seu corpo. Implemente toda a restante funcionalidade.

```
public class Esquadra {

        LinkedList<Agente> agentes = new LinkedList<Agente>();

        float renumeracaoAnualTotal() {
                float res=0;
                for (Agente a: agentes)
                        res+=a.renumeracaoAnual();
                return res;
        }

        void registaAgente(Agente a) {
                agentes.add(a);
        }

        static float renumeracao(String posto) {…}
        static float premio(String classificacao, String especialidade){…}
        static float premio(float montante){…}
}
```

e) **[0.5 valores]** Defina a classe `Main` que define um método `main` onde é criada uma esquadra e um agente de trânsito é registado nessa esquadra. Depois calcule a renumeração anual a pagar pela esquadra aos agentes.

```
public class Main {
        public static void main(String[] strings){
                Esquadra esquadra1 = new Esquadra();
                esquadra1.registaAgente(new AgenteTransito("Francisco Fernandes","Benfica"));
                esquadra1.renumeracaoAnualTotal();
        }
}
```

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	D	B	C	C	A

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?

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

A. x=1 e y=0
B. x=0 e y=1
C. x=0 e y=-1
D. x=1 e y=2
E. Nenhuma das anteriores

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

```
String s1="Hello";
String s2="He"+"llo";
String s3=new String("Hello");
System.out.print(s1==s2);
System.out.print(s1==s3);
System.out.print(s2.equals(s3));
```

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

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

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

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

Nome: Número:

d) **[0.5 valores]** Escolha a opção correcta. ArrayList<Number> é uma subclasse de:

A. ArrayList<Object>
B. Number
C. ArrayList<? extends Number>
D. ArrayList<? super Number>
E. Nenhuma das anteriores

e) **[0.5 valores]** De que tipo é um AssertionError?

A. Throwable B. RuntimeException C. Exception D. Nenhum das anteriores

4 – [1.0 valores] Identifique uma importante diferença entre tabelas e contentores genéricos (para além do facto de os contentores genéricos oferecerem estruturas de dados dinâmicas), analisando o que acontece com os seguintes fragmentos de código. Justifique a sua resposta.

```
Object[] tabela = new Long[1];
tabela[0] = "Aqui estou eu!";
```

```
List<Object> lista = new ArrayList<Long>(1);
lista.add("Aqui estou eu!");
```

O fragmento de código relativo às tabelas compila mas lança uma excepção ArrayStoreException em tempo de execução! Por outro lado, o código relativo aos genericos não compila.

É de esperar que não se consiga colocar uma String numa tabela ou contentor de Long, mas com uma tabela este erro só é detectado um tempo de execução. Com uma lista genérica este erro é encontrado em tempo de compilação. Obviamente, é preferível encontrar este erro em tempo de compilação.

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 uma esquadra com um agente de intervenção e um agente de trânsito com historial de multas desde o ano 2008 até à data de hoje. Apresente o DTD para um possível documento XML que contenha atributos e que descreva a informação pedida.

```
<! DOCTYPE esquadra [
        <!ELEMENT esquadra (agentes*)>
        <!ELEMENT agentes (agente*)>

        <!ELEMENT agente (posto,(agenteintervencao|agentetransito))>
        <!ATTLIST agente
                numero ID #REQUIRED
                nome CDATA #REQUIRED>

        <!ELEMENT posto (#PCDATA)>

        <!ELEMENT agenteintervencao (especialidade,classificacao)>
        <!ELEMENT especialidade (#PCDATA)>
        <!ELEMENT classificacao (#PCDATA)>

        <!ELEMENT agentetransito (multas*)>
        <!ELEMENT multas multa>
        <!ELEMENT multa EMPTY>
        <!ATTLIST multa
                ano CDATA #REQUIRED
                valor CDATA #REQUIRED>
]>
```

Parte IV -- C++ (1 valor)

6 – [1.0 valores] Em C++ é possível fazer sobreposição de operadores. Explique em que consiste e dê um breve exemplo.

A sobreposição de operadores permite que seja dado um significado a um operador usual (+,-,etc) num contexto de um novo tipo definido em C++. Por exemplo, permite definir a soma (+) entre dois objectos de tipo Complexo:

```
class Complexo {
        double re, im;
        Complexo(double r, double i) { re=r; im=i; }
        friend Complexo operator+(Complexo,Complexo);
}

Complexo operator+(Complexo c1, Complexo c2) {
        return Complexo(c1.re+c2.re,c1.im+c2.im);
}
```