Nome: Nº

2º Teste

Programação Orientada por objetos
2014/15
Aulas Teórica de 4 de Junho de 2015
Departamento de Informática
Universidade Nova de Lisboa
(duração 2h)

– Sem consulta –

Leia com atenção a informação constante desta página, enquanto espera a indicação do docente para começar a resolução do teste.

Este enunciado é composto por:
Uma página de Rosto (esta)
16 páginas de enunciado.
1 boletim/folha de respostas

O teste é composto por perguntas de resposta múltipla.

Atenção:

Resposta múltipla para seleção de apenas uma alínea
Uma resposta errada desconta um quarto do valor dessa pergunta na cotação total do teste
(VALOR DA PERGUNTA) / 4

Todas as perguntas de resposta múltipla devem ser respondidas no boletim de resposta fornecido em conjunto com este enunciado assinalando com uma bola a cheio a opção pretendida. (para evitar enganos, sugere-se o uso de lápis e/ou a folha do enunciado, assinalando-se no fim a resposta final a caneta preta ou azul).

Não serão fornecidos boletins de resposta substitutos. Perguntas com respostas rasuradas não serão consideradas.

Se o boletim não tiver o número de aluno nem estiver assinado pelo próprio não será considerado para avaliação.

A resolução final no boletim tem de ser feita a caneta azul ou preta (qualquer outra cor levará a anulação).

No fim de 1h30 minutos de teste o docente **recolherá o enunciado e boletim de respostas**.

Boa Sorte!

1- Admitindo que o ficheiro "exam-123.lab" tem o seguinte conteúdo, (sombreado são apenas comentários não existentes no ficheiro)):

```
2213    34
João Rafael Nunes
23/4/2010
M N N
EOG EOG EOG EOG
23.2 33.3 23.2 12.1      (linha 1 dos dados)
23.2 33.3 23.2 12.1      (linha 2 dos dados)

... (mais dados iguais nas restantes linhas)

23.2 33.3 23.2 12.1      (linha 120 dos dados)
```

Considere agora o seguinte trecho de código Java:

```
import java.util.Scanner;
import java.io.FileReader;
import java.io.IOException;
public static void main(String[] args) throws IOException {
    Scanner in = new Scanner(new FileReader("exam-1234.lab"));
    String[] partesDaLinha;
    String linhaDoFicheiro;
    int valor;
    while (true) {
          for(int i =0; i <5; i++){
                  linhaDoFicheiro = in.nextLine();
                  if  (i == 0){
                       partesDaLinha = linhaDoFicheiro.split("\t");
                       valor = Integer.parseInt(partesDaLinha[1]);
                  }
          }
          for (int i = 0; i <120; i++){
               linhaDoFicheiro = in.nextLine();
               partesDaLinha = linhaDoFicheiro.split("\t");
          }
          System.out.println(partesDaLinha[0]);
          break;
     }
     in.close();
}
```

Ao executar o código anterior o resultado na consola será:

[A] dá erro

[B] mostra "23.2" 120 vezes, um por cada linha

[C] mostra "23.2 33.3 23.2 12.1" 120 vezes, um por cada linha

[D] mostra os dados todos até à data (i.e., de "2213 34" a "EOG EOG EOG EOG", inclusivé)

[E] mostra uma vez "23.2"

2- Admitindo que no momento em que começa a execução deste trecho de código a variável linhaDeFicheiro está carregada com o valor:

```
23.2 33.3 23.1 12.1
```

Qual o resultado da execução do seguinte trecho de código:

```
partesDaLinha = linhaDoFicheiro.split("\t");
valor = Integer.parseInt(partesDaLinha[1]);
valor += Integer.parseInt(partesDaLinha[2]);

System.out.println(valor);
```

[A] 23.2

[B] 33.3

[C] 56.5

[D] 23.1

[C] 56.4

3- Qual a resposta correta com respeito ao conceito de interface?

[A] As interfaces fornecem uma implementação.

[B] O conceito de interface não está associado a princípios de abstração.

[C] Interface representa todos os objetos que suportam um dado protocolo público e que sejam instâncias de classes que declarem explicitamente implementar essa interface.

[D] No caso dos objetos, a interface normalmente consiste num conjunto de métodos privados que manipulam dados ocultos da implementação.

[E] Nenhuma das opções.

4- Qual é o resultado (output) deste programa?

```
import java.io.*;
class serialization {
    public static void main(String[] args) {
        try {
            Myclass object1 = new Myclass("Hello", -7, 2.1e10);
            FileOutputStream fos = new FileOutputStream("serial");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(object1);
            oos.flush();
            oos.close();
        }
        catch(Exception e) {
            System.out.println("Serialization" + e);
            System.exit(0);
        }
        try {
            int x;
            FileInputStream fis = new FileInputStream("serial");
            ObjectInputStream ois = new ObjectInputStream(fis);
            x = ois.readInt();
            ois.close();
            System.out.println(x);
        }
        catch (Exception e) {
            System.out.print("deserialization");
            System.exit(0);
        }
    }
}
class Myclass implements Serializable {
    String s;
    int i;
    double d;
    Myclass(String s, int i, double d){
        this.d = d;
        this.i = i;
        this.s = s;
    }
}
```

[A] -7

[B] Hello

[C] 2.1E10

[D] deserialization

[E] Nenhuma das opções

5- Com respeito ao conceito de polimorfismo, qual a resposta correta?

[A] Denota o princípio de que o comportamento não depende do tipo real de um objeto.

[B] O polimorfismo é a propriedade que permite que o tipo real do objecto seja usado para decidir qual a implementação do método a escolher, em vez de o tipo declarado.

[C] O polimorfismo permite que objetos de classes que foram definidas sem qualquer relação entre si, ou algo em comum (não usando, por exemplo, implements e extends), sejam tratadas exatamente da mesma forma.

[D] Denomina-se polimorfismo em Java ao mecanismo de herança múltipla.

[E] Nenhuma das opções..

6- Atendendo à interface Iterator apresentada nas aulas da disciplina e considerando que o método getCoffeeType retorna uma String com o tipo de café, considere o seguinte iterador que implementa a interface:

```
public class TypeIterator implements Iterator {
        private CoffeeCapsule[] capsules;
        private int counter;
        private int current;
        private String type;

        public TypeIterator(CoffeeCapsule[] capsules, int counter) { (1) }

        public TypeIterator(String type, CoffeeCapsule[] capsules, int counter) {
(2) }

        public void init() {
                current = 0;
                if (type != null)
                        while ( (current < counter) &&
!capsules[current].getCoffeeType().equals(type))
                                current++;
        }

        public boolean hasNext() {
                return (current < counter);
        }

        public CoffeeCapsule next() {
                CoffeeCapsule res = capsules[current++];
                if (type != null)
                        (3)
                return res;
        }
}
```

Na classe TypeIterator apresentada, pretendemos implementar dois iteradores e por isso temos dois construtores diferentes, um que permite iterar sobre todos os tipos de cápsulas de café e outro onde se pretende iterar sobre o tipo de café. Qual a opção que implemente corretamente um dos construtores?

[A] (1) terá:

```
                this.capsules = capsules;
                this.counter = counter;
                this.type = 0;
                init();
```

[B] (2) terá:

```
                this.capsules = capsules;
                this.counter = counter;
                this.type = null;
                init();
```

[C] (1) terá:

```
                this.capsules = capsules;
                this.counter = counter;
                this.type = null;
```

[D] (2) terá:

```
            this.capsules = capsules;
            this.counter = counter;
            this.type = null;
```

[E] Nenhuma das opções.

7- Ainda relativamente ao código apresentado na pergunta anterior (5). Na classe TypeIterator apresentada, pretendemos implementar o método next, que retorna a cápsula seguinte. Qual a opção que implementa corretamente (3) com o intuito de se retornar a próxima cápsula de um determinado tipo e que da próxima invocação a mesmo também funcione corretamente (assuma que o construtor está bem declarado)?

[A]

```
       do {
              current++;
       } while ((current < counter) &&
!capsules[current].getCoffeeType().equalsIgnoreCase(type));
```

[B]

```
       if( (current < counter) &&
!capsules[current].getCoffeeType().equalsIgnoreCase(type))
              current++;
```

[C]

```
       while (!capsules[current].getCoffeeType().equalsIgnoreCase(type))
              current++;
```

[D]

```
       while ( hasNext() &&
!capsules[current].getCoffeeType().equalsIgnoreCase(type))
              current++;
```

[E] Nenhuma das opções.

8- Qual das seguintes não é uma afirmação verdadeira?

[A] todas as classes que contenham métodos abstratos devem de ser declaradas como abstratas

[B] Um classe abstrata define define apenas a estrutura da classe e não a sua implementação

[C] Uma classe abstrata pode ser iniciada (instanciar um objecto) com um operador new.

[D] uma classe abstrata pode ser herdada

[E] Nenhuma das anteriores

9- Qual das seguintes instruções é usada para se referir a um membro (exemplo atributo) de uma case base a partir de uma sub-classe?

[A] upper

[B] super

[C] this

[D] other

[E] Nenhuma das opções

10- Qual das seguintes afirmações está correta?

[A] os membros públicos de uma classse podem ser acedidos por qualquer código no programa.

[B] os membros privados de uma classe só podem ser acedidos por outros membros da classe.

[C] Os membros privados de uma classe podem ser herdados por uma subclasse e tornar-se membros protegidos na subclasse.

[D] Membros protegidos de uma classe podem ser herdados por uma subclasse, e tornar-se membros privados dessa subclasse.

[E] Nenhuma das opções

11 - Um membro de uma classe tornado protected torna-se membro de uma subclasse de tipo:

[A] public

[B] private

[C] protected

[D] static

[E] Nenhuma das opções

12- Qual é a maneira correta da classe B herdar da class A ?

[A] class B + class A {}

[B] **class** B inherits class A {}

[C] class B extends A {}

[D] class B extends class A {}

[E] Nenhuma das opções

13- Considere o seguinte código:

```
public interface C {
     public int getValue1();
     public int getValue2();
}
public class C1 implements C {
     private int x;
     protected int y;

     public C1(int x){
     this.x=x;
     }
     public C1(int x,int y){
          this.x=x;
          this.y=y;
     }
     public int getValue1(){
               return x;
     }
     public abstract int getValue2();
 }

public class C2 extends C1{
     protected int z;
     public C2(int x, int y, int z){
          super(x);
          super.y=y;
          this.z=z;
     }

     public int getValue1(){
            return super.getValue1()+ 2*z;
     }
}

(1) public class Main {
(2)
(3)      public static void main(String[] args) {
(4)           C vec[] = new C[3];
(5)           vec[0]= new C1(3);
(6)           vec[1]= new C1(2,3);
(7)           vec[2]= new C2(1,2,3);
(8)           if (vec[0] instanceof C1)
(9)                System.out.println(vec[0].getValue1());
(10)         System.out.println(vec[1].getValue1());
(11)         System.out.println(vec[2].getValue1());
(12)      }
(13) }
```

Relativamente à classe C1, podemos afirmar que:

a) Está bem implementada

b) Vai dar erro pois o construtor vazio não está implementado

c) Vai dar erro pois os membros da classe têm de ser todos privados

d) Vai dar erro pois o método getValue2() não pode ser abstrato, já que a classe não é abstrata

e) Vai dar um warning relativo ao método getValue2() mas vai compilar

14- Ainda relativamente ao código apresentado em 13. relativamente ao construtor da classe C2 podemos afirmar que:

a) Está bem implementado

b) Vai dar erro pois não se pode aceder à variável y da classe C1.

c) Vai dar erro pois o construtor da classe C1 tem de ser chamado com 2 parâmetros e não 1 apenas.

d) Vai dar erro pois o construtor da classe C2 só pode receber 2 parâmetros e não 3

e) As alíneas b) e c) estão corretas.

15- Ainda relativamente ao código apresentado em 13. Relativamente à linha 4 da classe Main podemos afirmar que:

a) Vai dar erro pois eu não posso criar objetos do tipo C

b) Vai dar erro pois o tamanho do vetor está mal definido

c) Está correta

d) Vai dar erro pois a interface C não tem definido nenhum construtor

e) Vai dar um warning mas o programa compila

16- Ainda relativamente ao código apresentado em 13. Relativamente à linha 8 da classe Main

a) Vai dar erro pois o método instanceof não é invocado desta forma

b) Está correta e indica se a posição 0 do vetor contem um objeto do tipo C1

c) Vai dar erro pois o java não possui nenhum método instanceof

d) Vai dar erro pois o método instanceof só pode ser invocado para um vetor e não uma posição do vetor.

e) Vai dar erro pois o método instanceof não pode ser invocado para vetores.

17- Ainda relativamente ao código apresentado em 13. Relativamente à linha 9 do programa o valor imprimido é:

a) 3

b) 2

c) Nada é imprimido

d) 0

e) O valor imprimido é indefinido

18- Ainda relativamente ao código apresentado em 13. Relativamente às linhas 10 e 11 os valores imprimidos serão:

a) 3 e 7

b) 1 e 5

c) 2 e 3

d) 2 e 7

e) 2 e 5

19- Qual o resultado da execução do código seguinte?

```
class overload {
       int x;
       double y;
       void add(int a , int b) {
           x = a + b;
       }
       void add(double a , double b){
           y = a + b;
       }
       overload() {
           this.x = 0;
           this.y = 0;
       }
   }
   class Overload_methods {
       public static void main(String args[])
       {
           overload obj = new overload();
           int a = 2;
           double b = 3.2;
           obj.add(a, a);
           obj.add(b, b);
           System.out.println(obj.x + " " + obj.y);
       }
  }
```

[A] 5.2 5.2

[B] 5.2 5

[C] 6.4 6

[D] 4 6.4

[E] Nenhuma das opções

20- Qual é o resultado (output) deste programa?

```
   class A {
       int i;
       void display() {
           System.out.println(i);
       }
   }
   class B extends A {
       int j;
       void display() {
           System.out.println(j);
       }
   }
   class inheritance_demo {
       public static void main(String args[])
       {
           B obj = new B();
           obj.i=1;
           obj.j=2;
           obj.display();
       }
  }
```

[A] 0

[B] 1

[C] 2

[D] Compilation Error

[E] Nenhuma das opções

21- Qual é o resultado (output) deste programa?

```
class A {
       int i;
   }
   class B extends A {
       int j;
       void display() {
           super.i = j + 1;
           System.out.println(j + " " + i);
       }
   }
   class inheritance {
       public static void main(String args[])
       {
           B obj = new B();
           obj.i=1;
           obj.j=2;
           obj.display();
       }
  }
```

[A] 2 2

[B] 3 3

[C] 2 3

[D] 3 2

[E] Nenhuma das opções

22- Qual é o resultado (output) deste programa?

```
class A {
       public int i;
       private int j;
   }
   class B extends A {
       void display() {
           super.j = super.i + 1;
           System.out.println(super.i + " " + super.j);
       }
   }
   class inheritance {
       public static void main(String args[])
       {
           B obj = new B();
           obj.i=1;
           obj.j=2;
           obj.display();
       }
  }
```

[A] 2 2

[B] 3 3

[C] Erro de execução (Runtime Error)

[D] Erro de Compilação (Compilation Error)

[E] Nenhuma das opções

23- Qual é o resultado (output) deste programa?

```
class A {
    public int i;
    public int j;
    A() {
        i = 1;
        j = 2;
    }
}
class B extends A {
    int a;
    B() {
        super();
    }
}
class super_use {
    public static void main(String args[])
    {
        B obj = new B();
        System.out.println(obj.i + " " + obj.j)
    }
}
```

[A] 1 2

[B] 2 1

[C] Erro durante a execução (Runtime Error)

[D] Erro durante a compilação (Compilation Error)

[E] Nenhuma das opções

24- Qual é o resultado (output) deste programa?

```
class A {
    public int i;
    protected int j;
}
class B extends A {
    int j;
    void display() {
        super.j = 3;
        System.out.println(i + " " + j);
    }
}
class Output {
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
        obj.display();
    }
}
```

[A] 1 2

[B] 2 1

[C] 1 3

[D] 3 1

[E] Nenhuma das opções

25 - Qual é o resultado (output) deste programa?

```
class A {
       public int i;
       private int j;
   }
   class B extends A {
       void display() {
           super.j = super.i + 1;
           System.out.println(super.i + " " + super.j);
       }
   }
   class inheritance {
       public static void main(String args[])
       {
           B obj = new B();
           obj.i=1;
           obj.j=2;
           obj.display();
       }
   }
```

[A] 2 2

[B] 3 3

[C] Erro de execução (Runtime Error)

[D] Erro de compilação (Compilation Error)

[E] Nenhuma das opções

26- Qual é o resultado (output) deste programa?

```
class A {
       public int i;
       public int j;
       A() {
           i = 1;
           j = 2;
        }
   }
   class B extends A {
       int a;
       B() {
           super();
       }
   }
   class super_use {
       public static void main(String args[])
       {
           B obj = new B();
           System.out.println(obj.i + " " + obj.j)
       }
  }
```

[A] 1 2

[B] 2 1

[C] Erro de execução (Runtime Error)

[D] Erro de compilação (Compilation Error)

[E] Nenhuma das opções

27- Qual é o resultado (output) deste programa?

```
abstract class A {
       int i;
       abstract void display();
   }
   class B extends A {
       int j;
       void display() {
           System.out.println(j);
       }
   }
   class Abstract_demo {
       public static void main(String args[])
       {
           B obj = new B();
           obj.j=2;
           obj.display();
       }
  }
```

[A] 0

[B] 2

[C] Erro de execução (Runtime Error)

[D] Erro de compilação (Compilation Error)

[E] Nenhuma das opções

28- Qual é o resultado (output) deste programa?

```
class A {
       int i;
       void display() {
           System.out.println(i);
       }
   }
   class B extends A {
       int j;
       void display() {
           System.out.println(j);
       }
   }
   class method_overriding {
       public static void main(String args[])
       {
           B obj = new B();
           obj.i=1;
           obj.j=2;
           obj.display();
       }
  }
```

[A] 0

[B] 1

[C] 2

[D] Erro de compilação (Compilation Error)

[E] Nenhuma das opções

29- Qual é o resultado (output) deste programa?

```
class A {
       public int i;
       protected int j;
   }
   class B extends A {
       int j;
       void display() {
           super.j = 3;
           System.out.println(i + " " + j);
       }
   }
   class Output {
       public static void main(String args[])
       {
           B obj = new B();
           obj.i=1;
           obj.j=2;
           obj.display();
       }
  }
```

[A] 1 2

[B] 2 1

[C] 1 3

[D] 3 1

[E] Nenhuma das opções

30 - Considere o código seguinte:

```
public abstract class SensorElectrico {

   private static final int TIME=120;

   private String name;
   private static int time;
   private float [] data;

   protected SensorElectrico(String name, float [] leituras){
      this.name=name;
      this.data=leituras;
   }

   public String getName(){
      return this.name;
   }

   public float media() {
      int i=0;
      float sum=0;
      while(i<TIME) {
         sum += data[i++];
      }
      return sum/TIME;
   }

   public abstract boolean misterio();
}

public class ECG extends SensorElectrico {
   private static final String NAME="ECG";
   private static final int MAX=200;
   private static final int MIN=120;

   public ECG(float leituras[]){
      super(ECG.NAME, leituras);
   }

   public boolean misterio() {
      return (super.media()>MIN && super.media() < MAX);
   }
}
```

O que faz o código anterior?(escolha a afirmação falsa):

[A] O método “misterio” na classe SensorElectrico determina no vector de floats se a sua média está dentro de um intervalo MAX e MIN, retornando true caso positivo

[B] O construtor de ECG chama o constutor de Sensor Electrico

[C] Quando uma instância de SensorElectrico chama o método getName consegue aceder ao atributo name que não consegue aceder de outro modo diretamente.

[D] Mais nenhuma instância de classe consegue aceder à constante NAME a não ser se for do tipo SensorElectrico (mesmo que que uma classe herde desta)

[E] Nenhuma das opções