

**PRUEBA 2 PROGRAMACIÓN
Mayo 2007
INGENIERÍA INFORMÁTICA**

UNIVERSIDAD CARLOS III DE MADRID

LEA ATENTAMENTE ESTAS INSTRUCCIONES ANTES DE COMENZAR LA PRUEBA:

- Rellene todas las hojas a bolígrafo, tanto los datos personales como las respuestas. No use bolígrafo rojo.
- No olvide rellenar el NIA y el grupo real al que pertenece.
- El tiempo máximo de realización es de 50 minutos.
- El único material permitido sobre la mesa es la hoja de test y un bolígrafo

NO PASE DE ESTA HOJA, hasta que se le indique

Apellidos	Nombre	
Firma	NIA	Grupo

PARTE 1: CUESTIONES

Pregunta 1 (1 Punto).- Indicar si la siguiente afirmación es cierta, y explicar brevemente por qué.

"En Java se permite la herencia múltiple, es decir, que una clase A herede a la vez de B y C (class A extends B,C)"

FALSO.

En Java SOLAMENTE se permite heredar de una clase. Lo que sí puede hacerse es establecer múltiples niveles de herencia, por ejemplo:

class A extends B y class B extends C, de manera que A hereda de C a través de B (C sería la "abuela" de A y B sería la "madre" de A)

Pregunta 2 (1 Punto).- Indicar si la siguiente afirmación es cierta, y explicar brevemente por qué.

"Si queremos ordenar una lista, el mejor método será siempre el método rápido, ya que es el que menos tiempo tarda y el que menos recursos consume."

FALSO. La complejidad computacional del Método Rápido nos indica que en el peor caso este método es más lento que el método del montículo.

Normalmente es el método más rápido, pero si buscamos seguridad en la rapidez de la ordenación sería aconsejable utilizar el Método del Montículo.

Pregunta 3 (1 Punto).- Indicar si la siguiente afirmación es cierta, y explicar brevemente por qué.

Dado un método recursivo, las sucesivas llamadas recursivas deben realizarse cada vez con parámetros de inferior valor que los anteriores.

FALSO.

Lo único que importa en cuanto a la llamada recursiva es que exista un caso base, para el cual se terminan las llamadas recursivas, y al que deben converger las sucesivas llamadas recursivas de forma que lleguemos a él en algún momento. También es de interés indicar que no es necesario que un método recursivo reciba parámetros estrictamente numéricos, para los que no tendría sentido hablar de valores "inferiores".

Pregunta 4 (1 Punto).- Dado el siguiente método:

```
public float metodo1 (float [] lista){
    float [] lista2;
    if (lista.length>1) {
        lista2 = new float [lista.length-1];
        System.arraycopy(lista,1,lista2,0,lista2.length);
        return lista[0] + metodo1(lista2);
    }
    else return  lista[0];
}
```

Explicar cuál sería el resultado de la siguiente invocación:

```
float miLista [] = {1.0F,3.0F,3.2F};
System.out.println(metodo1(miLista));
```

Respuesta:

Es un método recursivo, en el caso base, que se produce cuando la lista la forma un solo elemento (lista.length no es >1), devuelve el primer elemento (y único de la lista). En los casos recursivos crea una nueva lista con un elemento menos que la original y copia todos los elementos menos el primero, devolviendo el elemento cero de la lista más el resultado de llamar al mismo método con la nueva lista. El resultado final es la suma de los elementos de la lista original.

La traza sería la siguiente:

```
metodo1({1.0F,3.0F,3.2F}) → 1F + metodo1(3.0F,3.2F) →
1F + 3F + metodo1(3.2F) → 1F+3F+3.2F → 7.2F
```

Y por lo tanto por pantalla se imprime 7.2F

Pregunta 5 (1 Punto).- Dada la siguiente lista: (12, 8, 4, 5, 9, 6)

Explicar paso a paso cómo se ordenaría dicha lista mediante el método de inserción directa.

Respuesta:

El método de inserción directa coloca cada elemento en su sitio en la lista formada por los anteriores, por lo tanto el proceso quedaría así:

(12,8,4,5,9,6) situación inicial
(8,12,4,5,9,6) colocamos el 8
(4,8,12,5,9,6) colocamos el 4
(4,5,8,12,9,6) colocamos el 5
(4,5,8,9,12,6) colocamos el 9
(4,5,6,8,9,12) todos colocados

PARTE 2: PROBLEMAS**Problema 1 (2,5 Puntos).-** Dado el siguiente código java:

```
public class Tallas {
    private int talla;
    private final int [] listaTallas =
        {32,34,36,38,40,42,44,46,48,50,52,54,56,58,60};
    //Comprueba que la talla está entre las anteriores
    public boolean tallaValida (int talla){
        boolean resultado = false;
        for (int i=0; i<listaTallas.length;i++){
            if (talla==listaTallas[i]) resultado = true;
        }
        return resultado;
    }
}
```

- Crear dos constructores para la clase Tallas:
 - Uno para dar valor al atributo talla. Deberá comprobar que el dato recibido es una talla válida usando el método tallaValida. Si no lo es creará un objeto de la talla “32”.
 - Otro por defecto que cree un objeto de la talla “38” utilizando el constructor anterior.
- Escribir el método public int comparar (Tallas otraTalla) que devuelve 1 si la talla del objeto que se le pasa como parámetro es mayor que la de nuestro objeto, -1 si es menor, y 0 si son iguales.
- Modificar el código del algoritmo de la burbuja para que ordene un array de objetos Tallas en lugar de un array de enteros.

o **Nota:** el código del algoritmo de la burbuja es:

```
public static void burbuja (int [] lista, int ultimo){
    int aux = 0;
    boolean cambio = true;
    for (int i=1; i<=ultimo && cambio; i++){
        cambio = false;
        for (int j=0; j<=ultimo-i; j++){
            if (lista[j]>lista[j+1]){
                cambio = true;
                aux = lista [j+1];
                lista [j+1] = lista [j];
                lista [j] = aux;
            }
        }
    }
}
```

Solución

```
public Tallas (int t){
    //comprueba que la talla es válida
    if (tallaValida(t)) talla=t;
    else talla=32;
}
public Tallas (){
    this (38); }
}
```

```
public int comparaTallas(Tallas otraTalla){
    if (otraTalla.talla>talla) return 1;
    else if (otraTalla.talla<talla) return -1;
    else return 0;
}

public static void burbuja (Tallas [] lista, int ultimo){
    Tallas aux;
    boolean cambio = true;
    for (int i=1; i<=ultimo && cambio; i++){
        cambio = false;
        for (int j=0; j<=ultimo-i; j++){
            if (lista[j].comparaTallas(lista[j+1])<0){
                cambio = true;
                aux = lista[j+1];
                lista [j+1] = lista [j];
                lista [j] = aux;
            }//fin if
        }//fin for j
    }//fin for i
} // fin burbuja
```

Problema 2 (2,5 Puntos).- Dado el siguiente código java:

```
public class Ropa {
    private String color;
    private String fabricante;
    private float precio;
    private String tejido;
    protected static int identificador;

    public Ropa (String c, String f, float p, String t){
        color = c;
        fabricante = f;
        precio = p;
        tejido = t;
        identificador ++;
    }
    public Ropa (){
        this("sin color", "sin fabricante", 0, "sin tejido");
    }
    public void imprimir () {
        System.out.println("Color: "+color);
        System.out.println("Fabricante: "+fabricante);
        System.out.println("Precio: "+precio);
        System.out.println("Tejido: "+tejido);
        System.out.println("Identificador: "+identificador);
    }
}
```

- a) Crear la clase Camiseta, que hereda de Ropa y que tiene los siguientes atributos privados:
 - a. mangas de tipo String para indicar si son cortas o largas.
 - b. dibujo de tipo boolean que indique si tiene dibujo o no.
- b) Crear un constructor que reciba parámetros para dar valor a todos los atributos de la clase Camiseta, incluidos los heredados. Deberá usar el constructor de la clase Ropa.
- c) Crear un constructor sin parámetros similar al de la clase Ropa.
- d) Sobrescribir el método imprimir, para que imprima todos los parámetros de la clase Camiseta. Utilizar si es posible el heredado.

```
public class Camiseta extends Ropa{

    private String mangas;
    private boolean dibujo;

    public Camiseta (String c, String f, float p, String t, String
m, boolean d){
        super (c,f,p,t);
        mangas = m;
        dibujo = d;
    }

    public Camiseta (){
        this("sin color","sin fabricante",0,"sin tejido","sin
determinar",false);
    }

    public void imprimir (){
        super.imprimir ();
        System.out.println("Mangas: "+mangas);
        System.out.println("Dibujo: "+dibujo);
    }
}
```