

**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.

“Un método declarado como `final` en la clase padre puede ser sobrescrito en la clase hija, mientras que si es declarado como `static` no puede ser modificado.”

Falso. Tanto un método declarado como `final` como declarado usando `static` en la clase padre, no puede ser sobrescrito en la clase hija.

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

“El análisis comparativo de la complejidad o eficiencia entre los algoritmos de búsqueda binaria y secuencial, nos asegura que para cualquier caso la binaria es más rápida que la secuencial”.

Falso. En el peor de los casos sí es cierto el enunciado, pero no en cualquier caso. Por ejemplo si el valor que buscamos se encuentra en la primera posición del array tardará menos en encontrarlo la búsqueda secuencial.

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

“El método Quicksort nos permite realizar procesos de búsqueda en listas sin necesidad de que sus elementos estén ordenados”.

Falso. El método Quicksort es de ordenación y no de búsqueda.

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

```
public static int metodo (int valor){  
    if (valor <=1) {  
        return valor+1;  
    }  
    else return = valor+metodo((valor/2)%3);  
}
```

Explicar cuál sería el resultado de la siguiente invocación:

```
System.out.println(metodo (113));
```

Respuesta:

117.

Es un método recursivo, la secuencia de llamadas es la siguiente:

$$\begin{aligned}\text{metodo (113)} &= 113 + \text{metodo (2)} = \\ &= 113 + 2 + \text{metodo (1)} = \\ &= 113 + 2 + 1 + 1 = \\ &= 117\end{aligned}$$

Pues

$$\begin{aligned}113 : 2 &= 56 \\ 56 \% 3 &= 2 \\ 1 \% 3 &= 1\end{aligned}$$

Pregunta 5 (1 Punto).- Dada la siguiente lista: (1, 34, 8, 4, 5, 9, 6, 11, 2)

Explicar paso a paso cómo se cambiaría después de cada pasada completa del método de la burbuja hasta que se den las condiciones para la finalización del algoritmo.

Respuesta:

Lista de partida: (1, 34, 8, 4, 5, 9, 6, 11, 2)

Primera: (1, 8, 4, 5, 9, 6, 11, 2, 34)

Segunda: (1, 4, 5, 8, 6, 9, 2, 11, 34)

Tercera: (1, 4, 5, 6, 8, 2, 9, 11, 34)

Cuarta: (1, 4, 5, 6, 2, 8, 9, 11, 34)

Quinta: (1, 4, 2, 5, 6, 8, 9, 11, 34)

Sexta: (1, 2, 4, 5, 6, 8, 9, 11, 34)

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;
    }
}
```

- a) Crear dos constructores para la clase Tallas:
 - a.1) 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".
 - a.2) Otro por defecto que cree un objeto de la talla "38" utilizando el constructor anterior.
- b) 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.
- c) 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:

- a) Crear dos constructores para la clase Tallas: Uno para dar valor al atributo `talla` y otro por defecto que cree un objeto de la talla "38" utilizando el constructor anterior.

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

- b) 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.

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

- c) Modificar el código del algoritmo de la burbuja para que ordene un array de objetos `Tallas` en lugar de un array de enteros.

```
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++)
        {
            // si el elemento de índice inferior es mayor que el de
            // índice superior cambiamos
            if (lista[j].comparar(lista[j+1])<0)
            {
                //como vamos a hacer un cambio ponemos la variable a
                //verdadero
                cambio = true;

                //hacemos el cambio usando la variable auxiliar
                aux = lista[j+1];
                lista [j+1] = lista [j];
                lista [j] = aux;
            }
        }
    }
}
```

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.1) mangas de tipo String para indicar si son cortas o largas.
 - a.2) 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.

Solución:

- a) Crear la clase `Camiseta`, que hereda de `Ropa` y que tiene los siguientes atributos privados: `mangas` (tipo `String`) y `dibujo` (tipo `boolean`).

```
public class Camiseta extends Ropa
{
    private String mangas;
    private boolean dibujo;
}
```

- 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`.

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

- c) Crear un constructor sin parámetros similar al de la clase `Ropa`.

```
public Camiseta (){
    super();
    mangas ="sin determinar";
    dibujo = false;
}
```

- d) Sobrescribir el método `imprimir`, para que imprima todos los parámetros de la clase `Camiseta`. Utilizar si es posible el heredado.

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