

**PRUEBA 2 PROGRAMACIÓN I
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 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.

“Una clase que hereda de otra puede sobrescribir cualquiera de sus métodos.”

Respuesta: Falso, los métodos declarados como *final* no pueden ser sobrescritos por una clase hija (ni los métodos de objeto ni los métodos de clase).

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

“Para poder utilizar el método de búsqueda binaria, es necesario que el array en el que se desea buscar esté ordenado”

Respuesta: Verdadero, con la búsqueda secuencial no es necesario que el array en el que buscar se encuentre ordenado, pero para poder realizar búsqueda binaria sí es necesario que lo esté.

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

“El método de ordenación del Montículo o HeapSort es más eficiente ordenando arrays que el método de la Burbuja o BubbleSort”

Respuesta: Verdadero. El HeapSort es un método de ordenación de los denominados avanzados, que es capaz de ordenar un array en un tiempo que es $O(n \log(n))$, mientras que el BubbleSort ordena en un tiempo $O(n^2)$.

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

```
public int metodo1 (int[] lista, int n){
    if (n > 0) {
        int tmp = 0;
        for(int i=0; i<n; i++) {
            tmp += lista[i];
        }
        return tmp + metodo1(lista, n-1);
    }
    else return 0;
}
```

Explicar cuál sería el resultado por pantalla de la siguiente invocación:

```
int[] lista = new int[] {1, 2, 3, 4};
System.out.println(metodo1(lista, 4);
```

Respuesta:

$(4 + 3 + 2 + 1) + (3 + 2 + 1) + (2 + 1) + (1) = 20$

Pregunta 5 (1 Punto).- Dada la siguiente lista: (18, 15, 3, 7, 9, 2, 10)

Escribir cómo evolucionaría dicha lista después de cada pasada completa del método de ordenación de la burbuja, hasta que la lista queda completamente ordenada.

Respuesta:

(15 , 3 , 7, 9 , 2 , 10 , 18)
(3 , 7 , 9 , 2 , 10 , 15 , 18)
(3 , 7 , 2 , 9 , 10 , 15 , 18)
(3 , 2 , 7 , 9 , 10 , 14 , 18)
(2 , 3 , 7 , 9 , 10 , 14 , 18)

PARTE 2: PROBLEMAS

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

```
public class Deporte {
    // Contiene el nombre del deporte
    private String nombre;
    // Contiene el número de equipos que participan en el deporte
    private int numeroDeEquipos;
    // Contiene el número de jugadores de cada equipo
    private int[] numeroDeJugadores;

    public Deporte(String nombre, int numeroDeEquipos,
                   int[] numeroDeJugadores) {
        this.nombre = nombre;
        this.numeroDeEquipos = numeroDeEquipos;
        this.numeroDeJugadores = numeroDeJugadores;
    }
}
```

- a) Crear la clase Futbol, que hereda de Deporte y que tiene el siguiente atributo privado:
 - a. goles de tipo int[] para indicar el número de goles que va marcando cada equipo.
- b) Crear un constructor sin parámetros de la clase Futbol que da valor a todos sus atributos. Inicializará los goles de cada equipo a cero y el número de jugadores de cada equipo a 11. Deberá usar el constructor de la clase Deporte.
- c) Crear un método: public void marcarGol(int equipo) que recibe como parámetro el identificador del equipo que marca el gol (0 ó 1), e incrementa su contador de goles.
- d) Sobrescribir el método toString de la clase Object para que imprima el marcador de goles actual (los goles de cada equipo).

Respuesta:

```
public class Futbol {
    private int[] goles;
    public Futbol() {
        super("Futbol", 2, new int[] {11, 11});
        this.goles = new int[] {0, 0};
    }
    public void marcarGol(int equipo) {
        goles[equipo]++;
    }
    public String toString() {
        String tmp = "El equipo 1 lleva " + goles[0] + "goles.".
        tmp = tmp + ", y el equipo 2 lleva " + goles[1];
    }
}
```

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

```
public class Arbol {
    private float altura;
    public Arbol (float altura) {
        this.altura = altura;
    }

    public float getAltura() {
        return altura;
    }
}
```

- Modificar el código del algoritmo de ordenación de **Selección Directa** para que ordene un array de objetos Arbol por su altura **descendentemente** (los árboles de mayor altura deberán quedar los primeros en el array ordenado).

- o **Nota:** el código del algoritmo de Selección Directa es:

```
public static void seleccionDirecta(int[] vector) {
    for(int i=0; i<vector.length-1; i++) {
        int menor = vector[i];
        int pos = i;
        for(int j=i+1; j<vector.length; j++) {
            if(vector[j] < menor) {
                menor = vector[j];
                pos = j;
            }
        }
        vector[pos] = vector[i];
        vector[i] = menor;
    }
}
```

Respuesta:

```
public static void seleccionDirecta(Arbol[] vector) {
    for(int i=0; i<vector.length-1; i++) {
        Arbol mayor = vector[i];
        int pos = i;
        for(int j=i+1; j<vector.length; j++) {
            if(vector[j].getAltura() > mayor.getAltura()) {
                mayor = vector[j];
                pos = j;
            }
        }
        vector[pos] = vector[i];
        vector[i] = mayor;
    }
}
```