

**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**

<b>Apellidos</b>	<b>Nombre</b>	
<b>Firma</b>	<b>NIA</b>	<b>Grupo</b>

**PARTE 1: CUESTIONES**

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

*“Cuando heredamos un método y lo sobrescribimos en la clase hija, podemos cambiar su visibilidad de public a private.”*

Falso.

Podemos cambiar la visibilidad de los métodos sobrescritos siempre y cuando se aumente la visibilidad, no si se disminuye.

---

**Pregunta 2 (1 Punto).**- ¿Qué significa el calificador *final* en un atributo, en un método y en una clase?

Un atributo final es una constante, es decir una vez se le ha dado un valor ya no se puede cambiar.

Un método final no se puede sobrescribir.

De una clase final no se puede heredar, es decir, no puede tener clases hijas.

---

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

Desde un constructor de una clase hija solo se puede llamar al constructor sin argumentos de la clase Madre.

Falso.

Por defecto desde un constructor de una clase hija se llama al constructor sin argumentos de la clase madre, pero si se desea llamar a otro constructor de la clase madre basta con utilizar la palabra clave “super ( )”.

**Pregunta 4 (1 Punto).**- Dados los siguientes métodos:

```

public void recursivo(int a, int b){
    int m=a+b;
    if(m<=10){
        recursivo(m,b+1);
        recursivo(m+1,b+1);
    }
    sumar(m,a);
}

public void sumar(int a,int b){
    System.out.println("suma:"+(a+b));
}

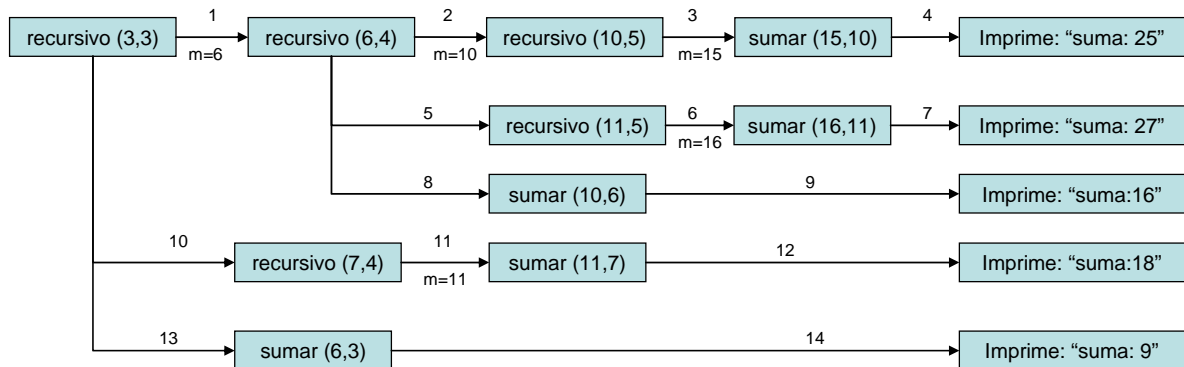
```

**Explicar** cuál sería el resultado de la siguiente invocación:

```
recursivo(3,3);
```

Es un método recursivo en el que el caso base es que la suma de sus dos parámetros sea mayor que diez. Si no estamos en el caso base, se llama recursivamente 2 veces. Una vez que se han acabado todas las llamadas recursivas, se llama al método suma.

La traza del método es por lo tanto:



Y por pantalla imprime:

```

suma:25
suma:27
suma:16
suma:18
suma:9

```

**Pregunta 5 (1 Punto).**- Dada la siguiente lista: (3, 1, 6, 2, 4, 5)

**Explicar** paso a paso cómo se ordenaría dicha lista mediante el método de la burbuja.

(1,3,6,2,4,5) cambio  
 (1,3,6,2,4,5) no cambio  
 (1,3,2,6,4,5) cambio  
 (1,3,2,4,6,5) cambio  
 (1,3,2,4,5,6) cambio

(1,3,2,4,5,6) no cambio  
 (1,2,3,4,5,6) cambio

De aquí en adelante ya no habría más cambios.

**PARTE 2: PROBLEMAS**

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

```
public class Alumno{
    // fecha de nacimiento
    int anio;
    int mes;
    int dia;
    // identificador
    int id;
    static int num_Alumnos = 0;
}
```

Ampliar la clase Alumno con los siguientes métodos:

- Un constructor que asigne la fecha de nacimiento de un alumno recibida como parámetro y le asigne un identificador único al alumno.
- Un constructor sin parámetros que cree un Alumno nacido el 1/1/1980 y que utilice el anterior constructor.
- Un método menor que determine si un alumno es menor que otro en función de la fecha de nacimiento. Si ambas fechas son iguales, un alumno es menor que otro si su identificador es menor.
- Implementar el método de ordenación por selección directa para una lista de Alumnos, ordenándolos de mayor a menor utilizando la función menor.

- Nota: El código del algoritmo de ordenación por selección directa para un array de enteros es el siguiente:

```
public 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;
    }
}
```

**Solución:**

```
a) public Alumno(int a, int m, int d){
    anio = a;
    mes = m;
    dia = d;
    id = ++num_Alumnos;
}
```

```
b) public Alumno () {
    this(1980,1,1);
}
```

```
c) public boolean menor(Alumno a){  
    if (anio<a.anio) return true;  
    else if ((anio==a.anio) && (mes<a.mes)) return true;  
    else if ((anio==a.anio) && (mes==a.mes) && (dia<a.dia)) return true;  
    else if ((anio==a.anio) && (mes==a.mes) && (dia==a.dia) && (id<a.id))  
        return true;  
    else return false;  
}  
  
c) public void seleccionDirecta(Alumno[] vector) {  
    for (int i=0; i<vector.length-1; i++){  
        Alumno menor = vector[i];  
        int pos = i;  
        for (int j=i+1; j<vector.length; j++){  
            if (vector[j].menor(menor)) {  
                menor = vector[j];  
                pos = j;  
            }  
        }  
        vector[pos] = vector[i];  
        vector[i] = menor;  
    }  
}
```

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

```
public class Jugador {
    private String nombre;
    private String demarcacion;
    private int partidosJugados;
    private int golesMarcados;
    private int tarjetasAmarillas;
    private int tarjetasRojas;
    protected static int identificador;

    public Jugador (String n, String d, int p, int g, int a, int r){
        nombre = n;
        demarcacion = d;
        partidosJugados = p;
        golesMarcados = g;
        tarjetasAmarillas = a;
        tarjetasRojas = r;
        identificador ++;
    }

    public Jugador (){
        this("sin nombre", "sin demarcacion", 0, 0, 0, 0);
    }

    public void imprimir () {
        System.out.println("Nombre: "+nombre);
        System.out.println("Demarcación: "+demarcacion);
        System.out.println("Partidos Jugados: "+partidosJugados);
        System.out.println("Goles Marcados: "+goles);
        System.out.println("Tarjetas amarillas:"+tarjetasAmarillas);
        System.out.println("Tarjetas Rojas: "+tarjetasRojas);
        System.out.println("Identificador: "+identificador);
    }
}
```

- a) Crear la clase `Portero`, que hereda de `Jugador` y que tiene los siguientes atributos privados:
  - a. `golesRecibidos` de tipo `int` para indicar el número de goles encajados.
  - b. `penaltisParados` de tipo `int` para indicar el número de penaltis parados.
- b) Crear un constructor para dar valor a todos los atributos de la clase `Portero`, incluidos los heredados. Deberá usar el constructor de la clase `Jugador`.
- c) Crear un constructor por defecto similar al de la clase `Jugador`.
- d) Sobrescribir el método `imprimir`, para que imprima todos los parámetros. Utilizar si es posible el heredado.

```
public class Portero extends Jugador {
    private int golesRecibidos;
    private int penaltisParados;

    public Portero (String n, int p, int g, int a, int r, int enc, int pen){
        super(n, "Portero", p, g, a, r)
        golesRecibidos = enc;
        penaltisParados = pen;
    }

    public Portero (){
```

```
        this("sin nombre", 0, 0, 0, 0, 0, 0);  
    }  
  
    public void imprimir () {  
        super.imprimir();  
        System.out.println("Goles Encajados: "+golesEncajados);  
        System.out.println("Penaltis Parados: "+penaltisParados);  
    }  
}
```