EXAMEN DE PROGRAMACIÓN
Enero 2013
GRADO EN INGENIERÍA INFORMÁTICA
Leganés

LEA ATENTAMENTE ESTAS INSTRUCCIONES ANTES DE COMENZAR LA PRUEBA:

- **Rellene todas las hojas a bolígrafo, no utilice lápiz ni bolígrafo rojo**
- **El tiempo máximo de realización es de 3 horas**
- **Se permiten apuntes y/o libros para la realización del examen**

PARTE 1: CUESTIONES

Pregunta 1 (1 Punto).- Encontrar y **explicar** los 5 errores de compilación que aparecen en el siguiente código Java. ¿Cómo los resolvería? (por comodidad se ha numerado cada línea)

```
00. import java.util.Scanner;
01.
02. public class Pregunta1 {
03.
04.   public static void main(String [] args) {
05.         int a=1, b=2, c, d;
06.         double x, y;
07.         Scanner sc;
08.
09.         x = sc.nextDouble();
10.         y = 3.1;
11.         System.out.println("y"+y);
12.         for(int i=0; i<y; i++) {
13.               double z;
14.               if(i % 2 == 0) {
15.                     z =  1;
16.               } else {
17.                     z = -1;
18.               }
19.               c = test(a, b);
20.               d = i/2 + z;
21.         }
22.         System.out.println("z"+z);
23.   }
24.   public boolean test(int n1, int n2) {
25.         return (n1 > n2);
26.   }
27.}
```

El primer error aparece en la línea 9 y en realidad es un error de ejecución, no de compilación. Es debido a que en la línea 7 no hemos creado el objeto `Scanner`. Se arreglaría poniendo `Scanner sc = new Scanner (System.in)` en la línea 7.
El segundo y tercer error aparecen en la línea 19. Por un lado el método `test` devuelve un `boolean` y lo intentamos guardar en una variable que es de tipo `int`. Se arreglaría cambiando el tipo de la variable `c`. Además, el método `test` no es `static` por lo que no se le puede llamar desde el método `main`. Se arreglaría haciéndolo `static`.
El cuarto error se produce en la línea 20. La variable `z` es de tipo `double`, por lo que al sumarla a `i` el resultado es `double` y no se puede guardar en `d` que es `int`. Se arreglaría cambiando el tipo de `d` o de `z`.

El quinto error aparece en la línea 22. La variable `z` se ha declarado dentro del bloque del `for` y por lo tanto no se puede usar una vez se ha cerrado ese bloque. Se arreglaría declarándola antes del `for`.

Pregunta 2 (1 punto).- Dada la lista {1,6,2,8,9} explicar los pasos que habría que realizar para ordenarla de **mayor a menor** según el algoritmo de **inserción directa.**

El algoritmo de inserción directa crea una sublista con el primer elemento que al ser una lista de un solo elemento ya es una lista ordenada. A continuación toma el segundo elemento y lo coloca en la posición adecuada dentro de esta sublista, desplazando hacia la derecha los elementos hasta llegar a su posición natural. De esta forma tenemos una sublista ordenada de 2 elementos. Hacemos lo mismo con el tercero y así sucesivamente. En nuestro caso las sublistas serían:

{6,1,2,8,9} --> {6,2,1,8,9} --> {8,6,2,1,9} --> {9,8,6,2,1}

PARTE 2: PROBLEMAS

Problema 1 (1,5 puntos).- Crear un programa en Java que calcule el precio de un producto según el mes del año en que nos encontremos. Para ello se debe crear un tipo enumerado `Meses` con los meses del año. A continuación dentro del método `main` se deberá pedir al usuario el precio de un `producto`, un `día` y un `mes` (no hace falta comprobar que el día o el mes son correctos). Si estamos fuera del periodo de rebajas el `precio` será el introducido por el teclado, si estamos dentro del periodo de rebajas el `precio` se reducirá un 10% cada semana desde el inicio de las rebajas (en la primera semana valdrá un 10% menos, en la segunda un 20% menos, etc.). Considerar como periodos de rebajas los comprendidos entre el 1 de Enero y el 28 de Febrero y el 1 de Julio y el 31 de Agosto.

Ejemplo:

```
Introduce el mes
AGOSTO
Introduce el día
31
Introduce el precio inicial
100.0
El precio final es 10.0 euros
```

Solución:

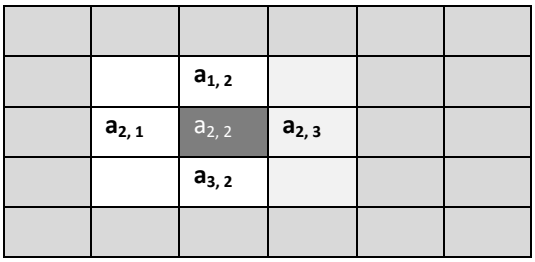
```
import java.util.Scanner;
enum Meses {ENERO, FEBRERO, MARZO, ABRIL, MAYO, JUNIO, JULIO, AGOSTO,
SEPTIEMBRE, OCTUBRE, NOVIEMBRE, DICIEMBRE}
public class Problema1 {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Introduce el mes");
		Meses mes = Meses.valueOf(sc.next());
		System.out.println("Introduce el día");
		int dia = sc.nextInt();
		System.out.println("Introduce el precio inicial");
		double precio= sc.nextDouble();
		switch (mes){
		//Enero y Julio son iguales, dividimos el número de días entre 7
		//para obtener el número de semanas que han pasado, le sumamos 1
		// para que la primera semana no sea la semana 0 y multiplicamos el
		//número de semanas por 0,1
```

```
            case JULIO:
            case ENERO: precio = precio - precio * 0.1*((dia/7)+1); break;
            //En Agosto y Febrero hay que sumar los días que llevemos de Julio y
            //Enero
            case AGOSTO:
            case FEBRERO: precio = precio - precio * 0.1*(((dia+31)/7)+1);break;
            }
            System.out.println("El precio final es "+precio+" euros");
            sc.close();
      }
}
```

Problema 2 (1,5 puntos).- Crear:

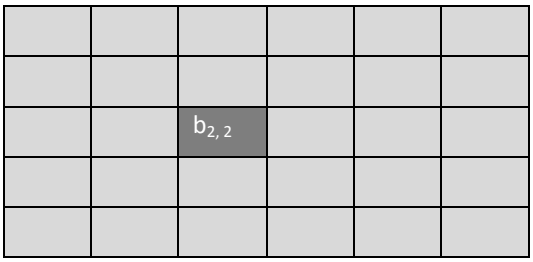
- un método que reciba como parámetro un array `A` de `n` filas y `m` columnas y devuelva un array `B` del mismo tamaño en el que cada elemento $b_{i,j}$ sea la suma de todos los elementos que rodean en horizontal o vertical al elemento $a_{i,j}$ equivalente del array `A`. Ejemplo, el valor del elemento $b_{2,2}$ será la suma: $a_{1,2}$, $a_{2,1}$, $a_{2,3}$, $a_{3,2}$.

A



$a_{1,2} + a_{2,1} + a_{2,} + a_{3,2}$

B



Para los elementos del borde del array hay que tener en cuenta que hay elementos que no existen. Por ejemplo para $a_{1,0}$ solo hay que considerar $a_{0,0}$, $a_{1,1}$ y $a_{2,0}$.

Nota: el método debe funcionar para arrays de cualquier tamaño.

- Crear un método `main` que cree un array de 5x6, lo inicialice con valores aleatorios entre 0 y 9, llame al método anterior e imprima el array resultado por pantalla.

Ejemplo:

Array Inicial
```
1 0 2 8
0 1 0 0
1 7 1 3
```

Resultado
```
0 4  8  2
3 7  4 11
7 3 10  1
```

Solución:

```
public class Problema2 {

public static int[][] metodo (int [][] array){
      //Creamos el nuevo array
      int res [][] = new int [array.length][array[0].length];
      //recorremos el array con un par de bucles
      for (int ii=0;ii<array.length;ii++){
            for (int jj=0;jj<array[ii].length;jj++){
                  //Para todas las filas excepto la primera sumamos
                  //el elemento de arriba
                  if(ii>0)
                        res[ii][jj]= res [ii][jj]+array[ii-1][jj];
                  //Para todas las filas excepto la última sumamos
                  //el de la abajo
                  if (ii<array.length-1)
```

```
                        res[ii][jj]= res [ii][jj]+array[ii+1][jj];
                  //Para todas las columnas excepto la primera
                  //sumamos el elemento de la izquierda
                  if (jj>0)
                        res[ii][jj]= res [ii][jj]+array[ii][jj-1];
                  //Para todas las columnas excepto la última
                  //sumamos el elemento de la derecha
                  if (jj<array[ii].length-1)
                        res[ii][jj]= res [ii][jj]+array[ii][jj+1];
            }
      }
      return res;
}
      public static void main(String[] args) {
            //Creamos el array
            int [][] arr = new int [5][6];
            //Lo rellenamos aleatoriamente
            for (int ii=0; ii<arr.length;ii++)
                  for (int jj=0;jj<arr[ii].length;jj++)
                        arr[ii][jj]= (int)(Math.random()*10);
            //Creamos el segundo array y llamamos al método
            int [][]arr2 = metodo(arr);
            //Imprimimos el primer array (el enunciado no lo
            //pedía pero para comprobar que la suma se hace bien)
            for (int ii=0; ii<arr.length;ii++){
                  for (int jj=0;jj<arr[ii].length;jj++){
                        System.out.print(arr[ii][jj]+" ");
                  }
                  System.out.println();
      }
            System.out.println();
            //Imprimimos el array resultado
            for (int ii=0; ii<arr2.length;ii++){
                  for (int jj=0;jj<arr2[ii].length;jj++){
                        System.out.print(arr2[ii][jj]+" ");
                  }
                  System.out.println();
      }
      }

}
```

Problema 3 (3,5 puntos).- Crear las siguientes clases:

1. **(1,5 puntos)** Clase `Persona`, con las siguientes características:
   - **(0,2 puntos)** Atributos privados: `nombre` de tipo `String`, `dinero` de tipo `double`, `edad` de tipo `int`, y `carnet`[1] de tipo `boolean`.
   - **(0,2 puntos)** Crear un método `get` y otro `set` para `edad`. Se debe comprobar que la `edad` está entre 0 y 150 años.
   - **(0,2 puntos)** Crear un método `get` y otro `set` para `carnet`. Si se va a poner `carnet` a `true` se debe comprobar que la `Persona` sea mayor de edad. En caso contrario no se cambiará el valor de `carnet` y se imprimirá por pantalla: `"no puede tener carnet porque tiene X años"`. Donde `X` debe ser la edad.
   - Suponer que existen métodos `set` y `get` para el resto de atributos.

[1] El atributo `carnet` representa si la persona tiene o no carnet de conducir.

- **(0,3 puntos)** Crear un constructor que reciba valores para todos los parámetros. Deberá comprobar que los valores pasados para `edad` y `carnet` tienen sentido.
- **(0,2 puntos)** Crear un constructor por defecto que no reciba parámetros y que cree personas "`sin nombre`" de `0` años de `edad` con `0` euros y sin carnet.
- **(0,2 puntos)** Crear un método `toString` que devuelva "`A tiene B años, C euros y tiene/no tiene carnet de conducir`" donde `A`, `B` y `C` deben cambiarse por los valores adecuados para cada objeto `Persona`.
- **(0,2 puntos)** Crear un método `equals` que compare dos `Personas`. Dos personas son iguales si su `nombre`, `edad`, `dinero` y `carnet` son iguales.

Solución:

```
public class Persona {
	private String nombre;
	private double dinero;
	private int edad;
	private boolean carnet;

	public int getEdad (){
		return edad;
	}
	public void setEdad (int e){
		if (e>=0 && e<=150)
			edad=e;
	}
	public boolean getCarnet(){
		return carnet;
	}
	public void setCarnet(boolean c){
		//Comprobamos si se va a poner el carnet a true
		if (c){
			if (edad>=18)
				carnet = c;
			else
				System.out.println("No puede tener carnet porque tiene
				"+edad+" años");
		}
		else
			carnet=c;
	}
	public Persona (String n, double d, int e, boolean c){
		//Aunque se supone que hay métodos setNombre y setDinero
		//también se pueden asignar los valores directamente
		nombre=n;
		dinero=d;
		//Es importante poner la edad antes que el carnet
		setEdad(e);
		setCarnet(c);
	}
	public Persona (){
		nombre="sin nombre";
		//El resto de atributos ya tienen por defecto los valores
		//que nos piden
	}
	public String toString(){
		String res;
		if (carnet)
			res="y tiene carnet";
```

```
		else
			res="y no tiene carnet";
		return nombre+" tiene "+edad+" años, "+dinero+" euros "+res;
	}
	public boolean equals (Persona otra){
		return nombre.equals(otra.nombre) && edad==otra.edad &&
		dinero==otra.dinero && carnet==otra.carnet;
	}
	//Métodos que se supone que existían
	public String getNombre (){
		return nombre;
	}
	public double getDinero (){
		return dinero;
	}
	public void setNombre(String n){
		nombre=n;
	}
	public void setDinero (double d){
		dinero=d;
	}
		}
```

2. **(1,4 puntos)** Clase `Vehiculo`, con las siguientes características:
   - **(0,3 puntos)** Atributos privados, `matricula` de tipo `String`, `dueño` de tipo `Persona` y `precio` de tipo `double`.
   - **(0,2 puntos)** Crear un método `set` para `matricula`. Debe comprobar que la cadena que se recibe tiene exactamente 7 caracteres.
   - **(0,3 puntos)** Crear un método `set` para `dueño`. El método deberá comprobar que la `Persona` es mayor de `edad` y tiene `dinero` suficiente para comprar el coche. En caso contrario no hará nada. Se deberá restar el `precio` del `Vehiculo` del `dinero` del `dueño`.
   - Suponer que existen métodos `set` y `get` para el resto de atributos.
   - **(0,3 puntos)** Crear un constructor que reciba valores para todos los atributos. Debe comprobar que el `dueño` es mayor de edad y tiene `dinero` y que la `matricula` es correcta. En caso contrario no dará valor a estos atributos.
   - **(0,3 puntos)** Crear un constructor por defecto que cree un `Vehiculo` con `matricula` `"0000BBB"`, con un `dueño` creado con el constructor por defecto de la clase `Persona` y `precio` 10.000 euros.

Solución:

```
package enero;
public class Vehiculo {
	private String matricula;
	private Persona dueño;
	private double precio;


	public void setMatricula (String m){
		if (m.length()==7)
			matricula = m;
	}
	public void setDueño (Persona d){
		//Como los atributos de Persona son privados
		//hay que usar los métodos get para acceder a ellos
```

```
            if (d.getEdad()>=18 && d.getDinero()>=precio){
                  //De igual forma para cambiar el dinero usamos setDinero
                  d.setDinero(d.getDinero()-precio);
                  //Una vez comprobado que puede comprar el Vehiculo ponemos
                  //esa persona como dueño
                  dueño = d;
            }
      }
      public Vehiculo (String m, Persona d, double p){
            precio=p;
            //Para comprobar que la matrícula y el dueño son correctos
            //lo más fácil es llamar a los métodos set que ya lo hacen
            setMatricula(m);
            setDueño(d);
      }
      public Vehiculo(){
            matricula = "000BBB";
            //Si no hiciésemos el new, dueño estaría a null que es su
            //valor por defecto
            dueño = new Persona();
            precio = 10000;
      }
      public boolean equals (Vehiculo otro){
            return matricula.equals(otro.matricula);
      }

      //Metodos get y set que se supone existian (solo los que necesitamos)
      public Persona getDueño(){
            return dueño;
      }
}
```

3. **(0,6 puntos)** Clase `Prueba` con un método `main` en el que:
   - **(0,2 puntos)** Se cree una `Persona` usando el primer constructor y otra usando el segundo.
   - **(0,2 puntos)** Se creen dos `Vehiculos` que tengan a cada una de esas `Personas` como `dueños`.
   - **(0,2 puntos)** Se cree un `array` de `Vehiculo` con los dos `Vehiculos` creados y se imprima por pantalla el `dueño` del primer elemento del `array`.

Solución:

```
public class Prueba {

      public static void main(String[] args) {
            Persona p = new Persona ("Juan sin Miedo",15000,21,true);
            Persona p2 = new Persona ();
            Vehiculo v = new Vehiculo ("0213JJJ",p,10000);
            Vehiculo v2 = new Vehiculo ("0213HHE",p2,15000);
            Vehiculo arr [] = new Vehiculo[]{v,v2};
            //getDueño nos da el dueño del Vehiculo y al imprimirlo
            //se llama automáticamente al método toString
            System.out.println(arr[0].getDueño());
      }
}
```

Problema 4 (1,5 puntos).- Crear un método estático denominado `cruz` que reciba por parámetro un entero n y dibuje una cruz de asteriscos de tamaño n en la pantalla. Si n es par la cruz debe ser de tamaño n+1. Crear un método `main` que reciba un número por

argumento y llame al método `cruz`.

Ejemplo:

Si el argumento recibido en el método `main` es 6, dibujará:

```
  *
  *
  *
*******
  *
  *
  *
```

Solución:

```
public class Problema4 {
	public static void cruz (int n){
		//Si es par sumamos uno
		if (n%2==0) n=n+1;
		//Hacemos un bucle para filas y otro para columnas
		for (int ii=0; ii<n; ii++){
			for (int jj=0;jj<n;jj++){
				//Si no estamos en la fila del medio
				if (ii!=n/2){
					//Si no estamos en la columna central
					if (jj!=n/2)
						System.out.print(" ");
					else
						System.out.print("*");
				}
				//Si estamos en la fila del medio
				else
					System.out.print("*");
			}
			System.out.println();
		}
	}

	public static void main(String[] args) {
		int tamaño = Integer.parseInt(args[0]);
		cruz(tamaño);
	}
}
```