EXAMEN DE PROGRAMACIÓN
Junio de 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**

Pregunta 1 (1 punto).- Encontrar los **4 errores** de compilación que aparecen en el siguiente código y **explicar** cómo se podrían resolver, en caso de que tengan solución (se han numerado las líneas por comodidad)

```
1. public class Cuestion1 {
2.    public static void metodo1 (int a){
3.          a=a+2;
4.          return a;
5.    }
6.    public int metodo2 (int a){
7.          a=a+5;
8.          return a;
9.    }
10.   public static void main(String[] args) {
11.         int a=3,b,c;
12.         b = metodo2(a);
13.         c=b+a;
14.         for (int b=0; b<10; b=b+2){
15.               float d=0;
16.               d=d+3;
17.         }
18.         double d=44.2F;
19.         if (d>22)
20.               a= b/d;
21.         else
22.               a=b;
23.         }
24.   }
```

El primer error está en la línea 4, el método `metodo1` se ha definido como `void`, por lo que no puede devolver nada mediante el `return`. La solución sería o cambiar `void` por `int` o quitar `return a`.

El segundo error está en la línea 12, `metodo2` no se ha definido como `static` por lo que no se le puede llamar. La solución es declararlo como `static`.

El tercer error está en la línea 14, no podemos volver a declarar b porque ya lo habíamos declarado en la línea 11. La solución es cambiar el nombre de la variable del `for`.

El cuarto error está en la línea 20, a es `int` pero `b/d` al ser d `double` es `double`. Se soluciona con un casting: `a = (int) (b/d)`

Pregunta 2 (1 punto).- Dada la lista {6,2,4,9,5} explicar los pasos que habría que realizar para ordenarla de **mayor a menor** según el algoritmo de **selección directa.**

Para ordenar de mayor a menor, el algoritmo de selección directa busca el elemento mayor y lo coloca el primero, a continuación busca el mayor de los que quedan y lo coloca el segundo, etc. En la versión que se implementa normalmente, lo que se hace es intercambiar la posición del primer elemento y el mayor, la del segundo elemento con el segundo mayor y así sucesivamente. Si lo hacemos así los pasos serían:

Cambiamos el 9 (mayor) por el 6 (primero) → (9,2,4,6,5)
Cambiamos el 6 (mayor) por el 2 (segundo) → (9,6,4,2,5)
Cambiamos el 5 (mayor) por el 4 (tercero) → (9,6,5,2,4)
Cambiamos el 4 (mayor) por el 2 (cuarto) → (9,6,5,4,2)

Pregunta 3 (1,5 puntos).- Crear:

a) Un método estático que reciba como parámetro un `array` de enteros y haga lo siguiente:

- Leer el valor del primer elemento del `array` y saltar tantas posiciones en el `array` como ese valor.
- Leer el elemento al que se ha saltado y volver a saltar tantas posiciones como el valor de ese elemento.
- Continuar este procedimiento hasta que no se pueda seguir saltando y devolver el número de veces que se ha saltado.

  Ejemplo: si el `array` recibido es `{1,3,4,4,6,7,2,1}`, leerá el primer elemento, `{1}` y saltará una posición, leerá el elemento de esa posición (el `{3}`) y saltará tres posiciones con lo que llegará al `{6}`, como no puede saltar 6 posiciones, terminará y devolverá un 2.

  Nota: El método debe funcionar para cualquier longitud del array. Se puede asumir que todos los valores del array son > 0.

b) Un método `main` que cree dos `arrays` de 100 elementos, los rellene de forma aleatoria con números entre 1 y 9, y llame al método anterior. Se deberá imprimir por pantalla el número de saltos de cada `array` y cuál de los dos ha necesitado de más saltos.

```
public class Problema1 {
	public static int salta (int [] a){
		int saltos=0;
		boolean puedeSaltar=true;
		int posicion=0;
		while (puedeSaltar){
			posicion= posicion+a[posicion];
			if (posicion<a.length)
				saltos++;
			else
				puedeSaltar=false;
		}
		return saltos;
	}
	public static void main(String[] args) {
		int [] arr1 = new int[10], arr2 = new int[10];
		for (int ii=0; ii<arr1.length; ii++){
			arr1[ii]= (int) (Math.random()*9+1);
			arr2[ii]= (int) (Math.random()*9+1);
		}
```

```
            //Imprimimos los arrays aunque el enunciado no lo pide
            for (int ii=0; ii<arr1.length; ii++){
                  System.out.print(arr1[ii]+" ");
            }
            System.out.println();
            for (int ii=0; ii<arr2.length; ii++){
                  System.out.print(arr2[ii]+" ");
            }
            System.out.println();
            int a = salta(arr1);
            int b = salta(arr2);
            System.out.println("El primer array tiene "+a+" saltos");
            System.out.println("El segundo array tiene "+b+" saltos");
            //Como no se dice en el enunciado, ignoramos el caso en que sean
            iguales
            if (a>b)
                  System.out.println("El primer array tiene más saltos que el
                  segundo");
            else
                  System.out.println("El segundo array tiene más saltos que el
                  primero");
      }
}
```

Pregunta 4 (2 puntos).- Crear los siguientes métodos estáticos:

a) **(0,3 puntos)** Un método que reciba como parámetro una matriz de enteros de cualquier tamaño y un número entero y devuelva cuántas veces se ha encontrado dicho número entero en la matriz.

b) **(0,5 puntos)** Un método que reciba como parámetro una matriz de cualquier tamaño y dos números enteros y devuelva 1 si el primer número está contenido más veces en la matriz que el segundo, 2 en caso contrario y 0 si ambos números aparecen las mismas veces en la matriz.

c) **(0,5 puntos)** Un método que reciba un `array` de enteros y una matriz y devuelva el elemento del `array` que aparece más veces en la matriz.

Nota: Los métodos deben funcionar para cualquier longitud de los arrays.

d) **(0,4 puntos)** Un método `main` que cree una matriz de 6x9 y la rellene aleatoriamente con números entre 0 y 9, a continuación pida al usuario que introduzca por teclado 5 números entre 0 y 9, e imprima cuál de los números se repite más en la matriz.

e) **(0,3 puntos)** Cambiar el método `main` anterior para que el programa compruebe que cada número introducido esté entre 0 y 9 y en caso contrario lo vuelva a pedir (es suficiente cambiar la parte de introducción por teclado).

```
import java.util.Scanner;

public class Problema2 {
      public static int metodo1(int [][] matriz, int num){
            int veces=0;
            for (int [] fila:matriz)
                  for (int ele:fila)
                  if (ele==num)
                        veces++;
            return veces;
      }

      public static int metodo2 (int [][] matriz, int n1, int n2){
            int veces1 = metodo1(matriz,n1);
            int veces2 = metodo1(matriz,n2);
            if (veces1>veces2) return 1;
            else if (veces1<veces2) return 2;
```

```
            else return 0;
      }

      public static int metodo3 (int [][] matriz, int[] arr){
            int resultado= arr[0];
            for (int ii=1; ii<arr.length; ii++)
                  if (metodo2(matriz, arr[ii], resultado)==1)
                        resultado=arr[ii];
            return resultado;
      }

      public static void main(String[] args) {
            //Creamos la matriz, la rellenamos y la imprimimos (aunque el
            //enunciado no pide imprimir)
            Scanner sc = new Scanner(System.in);
            int [][] matriz = new int [6][9];
            for (int filas=0; filas<matriz.length; filas++){
                  for (int col=0; col<matriz[filas].length; col++){
                        matriz[filas][col]= (int) (Math.random()*9+1);
                        System.out.print(matriz[filas][col]+" ");
                  }
                  System.out.println();
            }
            //Creamos el array y pedimos los valores
            int a [] = new int [5];
            System.out.println("Introduce 5 números entre 0 y 9");
            for (int ii=0; ii<5;ii++){
                  a[ii]= sc.nextInt();
            //Descomentar si queremos ver cuántas veces se repite ese número
//                System.out.println(metodo1(matriz,a[ii]));
            }

            //Para comprobar que están entre 0 y 9 (apartado e)
//          System.out.println("Introduce 5 números entre 0 y 9");
//          int ii=0;
//          while (ii<5){
//                a[ii]=sc.nextInt();
//                if (a[ii]>=0 && a[ii]<10) ii++;
//                else
//                      System.out.println("¡Entre 0 y 9!");
//          }

            //Calculamos qué número se repite más
            System.out.println("El número que más se repite es el
            "+metodo3(matriz,a));
      }
}
```

Pregunta 5 (1,5 puntos).- Crear un programa que pida al usuario cuánto es un euro en cinco divisas distintas (por ejemplo dólares, reales, coronas, rublos y pesos). A continuación le pedirá al usuario que escriba qué cantidad y moneda quiere convertir en euros e imprimirá el resultado. Ejemplo de ejecución:

```
Introduce cuánto es un euro en dolares
1,3
Introduce cuánto es un euro en reales
2,8
Introduce cuánto es un euro en coronas
8,7
Introduce cuánto es un euro en rublos
43,1
Introduce cuánto es un euro en pesos
17,1
Introduce la cantidad a convertir a euros y la moneda (Ejemplo: 3 dolares)
```

```
45 coronas
45.0 coronas son 5.172413793103448 euros

public class Problema3 {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		String divisas [] = {"dolares","reales","coronas","rublos","pesos"};
		double [] cambios = new double [divisas.length];
		for (int ii=0; ii<divisas.length; ii++){
		System.out.println("Introduce cuánto es un euro en "+divisas[ii]);
			cambios[ii]=sc.nextDouble();
	}
		System.out.print("Introduce la cantidad a convertir a euros y la
		moneda ");
		System.out.println("(Ejemplo: 3 dolares)");
		double cantidad = sc.nextDouble();
		String divisa = sc.next();
		int pos;
		switch (divisa){
		case "dolares": pos=0; break;
		case "reales": pos= 1; break;
		case "coronas": pos= 2; break;
		case "rublos": pos= 3; break;
		case "pesos": pos= 4; break;
		default: System.out.println ("Esa divisa no la conozco");
				pos=-1;
		}
		if (pos>-1)
			System.out.println(cantidad+" "+divisa+" son "+
cantidad/cambios[pos] +" euros");

	}
}
```

Pregunta 6 (3 puntos).- Crear:

- **(0,1 punto)** El tipo enumerado `Color = {ROJO, NEGRO}`
- La clase `Carta` con las siguientes características:
  a. **(0,1 punto**) Atributos privados: `color` de tipo `Color`, `valor` de tipo `int` y `oculta` de tipo `boolean`. El atributo `valor` valdrá 1 por defecto.
  b. **(0,2 puntos)** Un método `setColor` y otro `getColor`
  c. **(0,2 puntos)** Un método `setValor` y otro `getValor`. El método `setValor` debe comprobar que el `valor` que se le da a la `Carta` está entre 1 y 12. En caso contrario no hará nada.

Para el resto del programa asumir que existen los métodos `setOculta` y `getOculta`.

  d. **(0,2 puntos)** Un constructor por defecto que cree un uno `NEGRO oculto`.
  e. **(0,2 puntos)** Un constructor que reciba el `color` y el `valor` y si está `oculta` o no. Deberá comprobar que el `valor` es correcto (está entre 1 y 12).
  f. **(0,1 punto)** Un método `toString` que devuelva "<valor> <color> <oculta/visible>". Por ejemplo si la `carta` es el `4 ROJO` y no está `oculta`, devolverá "`4 ROJO visible`".

- La clase `Baraja` con las siguientes características:
  a. **(0,1 punto)** Atributo público: `cartas` de tipo `array` de `Carta`

b. **(0,3 puntos)** Un método privado `crearPalo`, que reciba un `Color` y devuelva un `array` de `cartas ocultas` con valores ordenados del 1 al 12 de ese color.

c. **(0,3 puntos)** Un constructor por defecto que cree una `baraja` con 24 cartas, la mitad rojas y la mitad negras, todas ocultas, de forma que la primera carta sea el 1 rojo y la última el 12 negro.

d. **(0,4 puntos)** Un constructor que reciba el número de palos rojos y el número de palos negros y cree una baraja de cartas ocultas con esos palos. Por ejemplo si recibe 3 y 2, creará una baraja de 60 cartas (3 * 12 + 2* 12), en la que los tres primeros palos serán rojos y los dos últimos negros. De esta forma habrá 3 cartas rojas de cada valor y 2 negras de cada valor.

e. **(0,3 puntos)** Un método `barajar` que reciba como parámetro el número de cartas a barajar e intercambie aleatoriamente de lugar tantas cartas como el doble del número recibido. Si recibe un 8, cambiará aleatoriamente 16 cartas de sitio, intercambiándolas dos a dos. No es necesario comprobar que las dos cartas intercambiadas sean distintas.

f. **(0,2 puntos)** Un método `darLaVuelta` que reciba un número n y dé la vuelta (ponga como no `ocultas`) las primeras n cartas de la baraja.

g. **(0,1 punto)** Un método `imprimir` que imprima por pantalla las cartas visibles (las no `ocultas`)

- **(0,2 puntos)** Crear una clase denominada `Prueba` con un método `main` que cree una `Baraja` con 3 palos rojos y 2 negros, baraje 12 cartas, dé la vuelta a las 15 primeras y las imprima.

```
enum Color {ROJO, NEGRO}
public class Carta {
	private Color color;
	private int valor=1;
	private boolean oculta;

	public void setColor (Color c){
		color = c;
	}
	public Color getColor(){
		return color;
	}
	public void setValor (int v){
		if (v>0 && v<13)
			valor = v;
	}
	public int getValor(){
		return valor;
	}
	public void setOculta (boolean o){
		oculta=o;
	}
	public boolean getOculta(){
		return oculta;
	}

	public Carta(){
		valor = 1;
		color = Color.NEGRO;
		oculta = true;
	}
```

```
	public Carta (Color c, int v, boolean o){
		setColor(c);
		setValor(v);
		setOculta(o);
	}
	public String toString (){
		String oc;
		if (oculta) oc= "oculta";
		else oc ="visible";
		return valor+" "+color+" "+oc;
	}
}
```

Clase Baraja:

```
public class Baraja {
	public Carta [] cartas;

	private Carta [] crearPalo (Color c){
		Carta [] resultado = new Carta [12];
		for (int ii=0; ii<resultado.length; ii++){
			resultado[ii]= new Carta (c,ii+1,true);
		}
		return resultado;
	}
	public Baraja (){
		Carta [] palo1 = crearPalo(Color.ROJO);
		Carta [] palo2 = crearPalo(Color.NEGRO);
		cartas = new Carta [palo1.length+palo2.length];
		System.arraycopy(palo1, 0, cartas, 0, palo1.length);
		System.arraycopy(palo2, 0, cartas, palo1.length, palo2.length);
	}
	public Baraja (int r, int n){
		Carta [] palo1 = crearPalo(Color.ROJO);
		Carta [] palo2 = crearPalo(Color.NEGRO);
		cartas = new Carta [palo1.length*r+palo2.length*n];
		for (int ii=0; ii<r*palo1.length; ii=ii+palo1.length)
			System.arraycopy(palo1, 0, cartas, ii, palo1.length);
		for (int ii=palo1.length*r; ii<r*palo1.length+n*palo2.length;
		ii=ii+palo2.length)
			System.arraycopy(palo2, 0, cartas, ii, palo2.length);
	}
	public void barajar (int c){
		Carta aux;
		for (int ii=0; ii<c; ii++){
			int pos1 = (int)(Math.random()*cartas.length);
			int pos2 = (int)(Math.random()*cartas.length);
			aux = cartas[pos1];
			cartas[pos1]=cartas[pos2];
			cartas[pos2]=aux;
		}
	}
	public void darLaVuelta (int n){
		for (int ii=0; ii<n && ii<cartas.length; ii++)
			cartas[ii].setOculta(false);
	}
	public void imprimir (){
		for (Carta c:cartas)
			if (!c.getOculta())
				System.out.println(c);
	}
}
```

Clase Prueba

```
public class Prueba {
	public static void main (String args[]){
		Baraja b = new Baraja (3,2);
		b.barajar(0);
		b.darLaVuelta(b.cartas.length);
		b.imprimir();
	}
}
```