EXAMEN DE PROGRAMACIÓN **4 de Julio de 2011** **GRADO EN INGENIERÍA INFORMÁTICA** **Leganés**	Universidad Carlos III de Madrid	
Apellidos	***Nombre***	
Firma	***NIA***	***Grupo***

LEA ATENTAMENTE ESTAS INSTRUCCIONES ANTES DE COMENZAR LA PRUEBA:

- **Rellene todas las hojas a bolígrafo, tanto los datos personales como las respuestas**
- **No utilice lápiz ni bolígrafo rojo**
- **No olvide rellenar el NIA y el grupo real al que pertenece**
- **El tiempo máximo de realización es de 3 horas**
- **Se permite cualquier tipo de material escrito para la realización del examen. No se permiten dispositivos electrónicos (portátiles, móviles, etc.)**

PARTE 1: CUESTIONES

Pregunta 1 (0,5 Puntos).- Indicar y explicar cuál sería la salida por consola al ejecutar la siguiente clase:

```
public class Pregunta1 {
	public static void main(String[] args) {
		int a = 3;
		int b = 2;
		b *= a++;
		System.out.println(a);
		System.out.println(b);
	}
}
```

Respuesta: Inicialmente se da a la variable “a” el valor 3 y a la variable “b” el valor 2. En la sentencia siguiente, se indica que “b” es igual a sí misma multiplicada por “a++”. Al estar el operador de incremento después de la variable, primero se opera con “a” y luego se incrementa. Por tanto el valor de “b” será el de sí misma (2) multiplicado por el de “a” (3), es decir **6**. Posteriormente se incrementará el valor de “a”, por tanto valdrá 4. Por tanto al ejecutar el programa se imprime por pantalla:

4

6

Pregunta 2 (1 Punto).- Indicar si las siguientes afirmaciones son o no ciertas, y **explicar** brevemente por qué.

1.1. **(0,25 puntos)** Dentro de un método de Java siempre debe haber al menos una sentencia `return`.

Falso. Si el método es declarado de tipo "void" no es necesario que contenga una sentencia "return".

1.2. **(0,25 puntos)** El siguiente bloque de código nunca terminará de ejecutarse, independientemente del código que pongamos dentro del bucle:

```
while(true) {
      … cualquier código …
}
```

Falso. Dentro del código del bucle podría existir una sentencia "break" que haga que la ejecución del bucle termine.

1.3. **(0,25 puntos)** En java, se puede dar valor a los elementos de un array en la misma sentencia en la que se crea el array, no es necesario utilizar dos sentencias.

Verdadero. A la vez que creamos el array podemos dar valor a sus elementos.

Ejemplo: int[] miArray = new int[] {1, 2, 3, 4};

1.4. **(0,25 puntos)** El algoritmo de ordenación de la burbuja es uno de los más eficientes que existen.

Falso. Los algoritmos de ordenación más eficientes son aquellos que realizan O(n log(n)) operaciones. El algoritmo de la burbuja es O(n^2).

Pregunta 3 (1 Punto).- Dado el siguiente método:

```
public static int metodo1(int[] valores, char caracter) {
            if(valores.length > 5) {
                  System.out.println("Valor del array demasiado largo");
            } else {
                  for(int i=0; i< valores.length-1; i++) {
                        if(i != 0) {
                              System.out.print(",");
                        }
                        System.out.print(valores[i]);
                  }
                  return -1;
            }
            System.out.println("Caracter: " + caracter);
            return 0;

      }
```

Indicar cuál sería el resultado de cada una de las siguientes invocaciones:

a) `metodo1(new int[] {1, 2, 3, 4}, 'a');`

1, 2, 3

b) `metodo1(new int[] {1, 2, 3, 4, 5, 6}, 'b');`

Valor del array demasiado largo
Caracter: b

c) `System.out.println(metodo1(new int[]{1}, 'd'));`

-1

d) `System.out.println(metodo1(new int[]{1, 2, 2}, 'e'));`

1,2-1

Problema 1 (2,5 puntos).- Crear una clase denominada `Problema1` que contendrá los siguientes métodos estáticos:

- **(0,5 puntos)** Método `porcentaje`
  - Parámetros: array de `double` de una dimensión
  - Acción: deberá contar cuántos elementos del array son mayores o iguales que el primer elemento (sin contar éste).
  - Devuelve: un `double` que será el resultado del cálculo anterior dividido por el número de elementos del array .

  Ejemplo: si el array es {3.1, 5.2, 6.5, 2.3, 1.0} devolverá 2/5 = 0.4
- **(0,5 puntos)** Método `mayores`
  - Parámetros: array de `double` de una dimensión, y un `double`
  - Acción: deberá encontrar los elementos del array que son mayores o iguales que el segundo parámetro.
  - Devuelve: un array de `double` con esos elementos.

  Ejemplo: para {3.1, 5.2, 6.5, 2.3, 1.0} y el elemento 2.1 devuelve {3.1, 5.2, 6.5, 2.3}
- **(1 punto)** Método `encontrarPicos`
  - Parámetros: array de `double` de una dimensión
  - Acción: deberá encontrar los "picos" del array: elementos que sean mayores que el elemento anterior y que el posterior (el primer elemento sólo se debe comparar con el posterior, y el último sólo con el anterior).
  - Devuelve: un array de `double` con esos elementos

  Ejemplo: si el array es {3.1, 5.2, 6.5, 1.0, 2.3} devolverá {6.5, 2.3}
- **(0,5 puntos)** Método `main`
  - Acción: crea un array de `double` de 50 elementos y los rellena aleatoriamente de valores entre 1 y 100. Imprime el array. Llama al método `encontrarPicos`. Imprime el array resultante. Con el array resultado y un número cualquiera (elegido por el alumno) llama al método `mayores` e imprime el array resultante. Con el array resultado llama al método `porcentaje` e imprime el resultado.

```
public static double porcentaje (double [] array){
      double resultado = 0;
      //no comparamos el primero
      for (int j=1; j<array.length; j++)
            if (array[j]>=array[0]) resultado++;
      return resultado/array.length;
}

public static double [] mayores (double [] array, double elemento){
//como no sabemos la longitud del array solución, lo creamos
//tan grande como el original
double [] resultado = new double [array.length];
//para llevar la cuenta de los elementos encontrados
int contador = 0;
//recorremos el array buscando los elementos
for (int j=0; j<array.length; j++)
      if (array[j]>= elemento)
            resultado[contador++]= array[j];
//creamos un nuevo array
double res2 [] = new double [contador];
//copiamos los elementos no nulos del array resultado al nuevo array
System.arraycopy(resultado, 0, res2, 0, contador);
return res2;      }
```

```
public static double [] encontrarPicos (double [] array){
	//como no sabemos la longitud del array solución, lo creamos
	//tan grande como el original
	double [] resultado = new double [array.length];
	// para llevar la cuenta de los picos encontrados
	int contador = 0;
	//comprobamos el primer elemento
	if (array[0]>array[1]) resultado[contador++]= array[0];
	//comprobamos el resto
	for (int j=1; j<array.length-1; j++) {
		//si el elemento es mayor que el anterior y que el posterior
		if (array[j]>array[j-1] && array[j]>array[j+1])
			//lo colocamos en el resultado
			resultado[contador++] = array[j];
	}
	//comprobamos el último
	if (array[array.length-1]>array[array.length-2])
		resultado[contador++]= array[array.length-1];
	//creamos un nuevo array
	double res2 [] = new double [contador];
	//copiamos los elementos no nulos del array resultado al nuevo array
	System.arraycopy(resultado, 0, res2, 0, contador);
	return res2;
}

public static void main(String[] args) {
	// TODO Auto-generated method stub
	double [] prueba = new double [50];
	for (int i=0; i<50; i++) {
		prueba[i] = Math.random()*100;
		System.out.print(prueba[i]+"\t");
	}
	double [] resul = encontrarPicos (prueba);
	for (int a=0; a<resul.length;a++)
		System.out.print(resul[a]+"\t");
	System.out.println();
	double resul2 [] = mayores (resul, 50);
	for (int a=0; a<resul2.length;a++)
		System.out.print(resul2[a]+"\t");
	System.out.println();
	System.out.println(porcentaje(resul2));
}
```

Problema 2 (2,5 puntos).- Crear una clase denominada `Carta` con las siguientes características:

- **(0,4 puntos)** Crear los siguientes atributos:
  - `Palos` de tipo array de `String` con los siguientes elementos {"bastos", "oros", "espadas", "copas"}
  - `Nombres` de tipo array de `String` con los siguientes elementos {"as", "dos", "tres", "cuatro", "cinco", "seis", "siete", "sota", "caballo", "rey"}
- **(0,2 puntos)** Crear los siguientes atributos:
  - `palo` de tipo `String`
  - `nombre` de tipo `String`
- **(0,6 puntos)** Crear el método estaEn
  - Parámetro: un array de `String` (denominado `array`) y un `String` (denominado `valor`)

  - Acción: comprueba si el `String` es alguno de los elementos del `array` (si el `array` contiene al `String`)
  - Devuelve: un `String` que será igual a `valor` si está en el `array`, o el primer elemento del `array` si no lo está.
- **(0,4 puntos)** Crear el método `setPalo`
  - Parámetro: Un `Stringq` que será el `palo` de la carta
  - Acción: cambia el atributo `palo` al valor recibido, deberá comprobar que el `String` recibido es un valor válido (un palo) para ello llamará al método `estaEn` anterior con el parámetro recibido y el atributo `Palos`.
  - Devuelve: nada.
- **(0,4 puntos)** Crear el método `setNombre`
  - Parámetro: Un `string` que será el `nombre` de la carta
  - Acción: cambia el atributo `nombre` al valor recibido, deberá comprobar que el `String` recibido es un valor válido (un nombre de carta) para ello llamará al método `estaEn` anterior con el parámetro recibido y el atributo `Nombres`.
  - Devuelve: nada.
- **(0,3 puntos)** Crear el método `getPalo`
  - Parámetro: ninguno
  - Devuelve: el valor del atributo `palo`
- **(0,2 puntos)** Crear el método `getNombre`
  - Parámetro: ninguno
  - Devuelve: el valor del atributo `nombre`

```
public class Carta {
	String [] Palos = new String [] {"bastos", "oros", "espadas", "copas"};
	String [] Nombres = new String []
				{"as", "dos", "tres", "cuatro", "cinco", "seis",
"siete", "sota", "caballo", "rey"};
	String palo;
	String nombre;

	public String estaEn (String [] array, String valor){
		for (int j=0; j<array.length; j++)
			if (array[j].equals(valor)) return valor;
		return array[0];
	}
	public void setPalo (String p){
		palo = estaEn(Palos,p);
	}

	public void setNombre (String n){
		nombre = estaEn(Nombres,n);
	}

	public String getPalo (){
		return palo;
	}

	public String getNombre(){
		return nombre;
	}
}
```

Problema 3 (2 puntos).- Crear la clase `Baraja` con las siguientes características:

- **(0,2 puntos)** La clase tendrá los siguientes atributos
  - cartas: array de 40 objetos de tipo `Carta`
- **(0,3 puntos)** Crear el método `crearPalo`
  - Parámetro: ninguno
  - Acción: crea aleatoriamente un `palo` de entre los permitidos
  - Devuelve: el `palo` creado (tipo `String`)
- **(0,3 puntos)** Crear el método `crearNombre`
  - Parámetro: ninguno
  - Acción: crea aleatoriamente un `nombre` de entre los permitidos
  - Devuelve: el `nombre` creado (tipo `String`)
- **(0,5 puntos)** Crear el método `crearCarta`
  - Parámetro: un `nombre` y un `palo` (tipo `String`)
  - Acción: crea una carta con ese `nombre` y ese `palo`
  - Devuelve: el objeto `carta` creado
- **(0,5 puntos)** Crear el método `creaBaraja`
  - Parámetro: ninguno
  - Acción: crea aleatoriamente 40 `cartas` y las inserta en el atributo `cartas`.
  - Devuelve: nada
- **(0,2 puntos)** Crear el método `imprimeBaraja`
  - Parámetro: ninguno
  - Acción: imprime las cartas de la baraja, separadas por comas: "`as de oros, tres de bastos, rey de copas`" etc.
  - Devuelve: nada

```
public class Baraja {
	Carta [] cartas = new Carta [40];

	public String crearPalo(){
		//creamos un random entre 0 y 3
		int p = (int) (Math.random()*4);
		switch (p) {
		case 0: return "bastos";
		case 1: return "oros";
		case 2: return "espadas";
		case 3: return "copas";
		//este nunca se ejecutará, pero hay que ponerlo
		default: return "oros";
		}
	}

	public String crearNombre(){
		//creamos un random entre 0 y 9
		int p = (int) (Math.random()*9);
		switch (p) {
		case 0: return "as";
		case 1: return "dos";
		case 2: return "tres";
		case 3: return "cuatro";
		case 4: return "cinco";
		case 5: return "seis";
		case 6: return "siete";
		case 7: return "sota";
```

```
            case 8: return "caballo";
            case 9: return "rey";
            //este nunca se ejecutará, pero hay que ponerlo
            default: return "as";
            }
      }

      public Carta crearCarta (String n, String p){
            Carta miCarta = new Carta();
            miCarta.setNombre(n);
            miCarta.setPalo(p);
            return miCarta;
      }

      public void creaBaraja (){
            for (int j=0; j<40; j++){
                  String nom = crearNombre();
                  String pal = crearPalo();
                  cartas [j] = crearCarta(nom, pal);
            }
      }

      public void imprimeBaraja(){
            for (int j=0; j<40; j++)
               System.out.print(cartas[j].nombre+" de "+ cartas[j].palo+", ");
      }

}
```

Problema 4 (0,5 puntos).- Crear la clase `MiBaraja` con las siguientes características:

- tendrá un método `main`, que hará lo siguiente: crear un objeto `Baraja`, crear las `cartas` del objeto e imprimir la baraja por pantalla.

```
public class MiBaraja {

      public static void main(String[] args) {
            Baraja b = new Baraja();
            b.creaBaraja();
            b.imprimeBaraja();

      }

            }
```