EXAMEN DE PROGRAMACIÓN **Enero 2009** **GRADO EN INGENIERÍA INFORMÁTICA** **Leganés y Colmenarejo**	 Universidad Carlos III de Madrid	
Apellidos	***Nombre***	
Firma	***NIA***	***Grupo***

LEA <u>ATENTAMENTE</u> ESTAS INSTRUCCIONES ANTES DE COMENZAR LA PRUEBA:

- **Rellene todas las hojas a <u>bolígrafo,</u> tanto los datos personales como las respuestas**
- **No utilice lápiz ni bolígrafo rojo**
- **No olvide rellenar el NIA y el grupo <u>real</u> al que pertenece**
- **El tiempo máximo de realización es de <u>2 horas y 30 minutos</u>**
- **Se permiten apuntes y/o libros para la realización del examen**
- **Utilice exclusivamente esta hoja de test para las respuestas, use las caras posteriores para contestar si lo necesita. <u>No se recogerá</u> ninguna otra hoja adicional.**

PARTE 1: CUESTIONES

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

1.1. **(0,5 puntos)** En java, podemos decidir en qué momento se ejecuta el recolector de basura.

1.2. **(0,5 puntos)** Nunca se puede acceder a un método de una clase sin haber creado antes un objeto de esa clase.

Falso. Lo único que podemos hacer es recomendar a Java que lo pase usando la instrucción System.gc(), pero eso no garantiza que el recolector de basura vaya a pasar.

Falso. Si el método se ha definido como static será un metodo de clase y por lo tanto se podrá acceder a él sin necesidad de haber creado un objeto de esa clase, usando NombreClase.metodo(). También sería válido decir que cuando estamos creando una clase, desde un método podemos acceder a otro método cualquiera sin necesidad de crear un objeto.

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

2.1. **(0,5 puntos)** "Al final del siguiente programa i tendrá valor igual a 8."

```
public class Pregunta2 {
      public static void main(String[] args) {
            int i = 1;
            while (i<8) {
                  i=i+1;
                  i++;
            }
      }}
```

2.2. **(0,5 puntos)** La siguiente es una declaración válida de una clase

```
public class Pregunta2 {
      public static int a = 0;
      public static void main(String[] args) {
            if(args.length > a) {
                  System.out.println("Existen atributos");
            } else {
                  System.out.println("NO Existen atributos");
            }
      }}
```

Falso. Dentro del bucle el valor de i se incrementa 2 veces, con lo que i va valiendo 3, 5, 7 y 9 momento en que ya no se cumple la condición y por lo tanto sale del bucle.

Verdadero. Es una clase correcta de Java, tiene un atributo static al que se accede desde un método static por lo que no hace falta crear ningún objeto.

Pregunta 3 (1 Punto).- Marcar la respuesta correcta como resultado de ejecutar el método `main` de la clase `Pregunta3` y **explicar** brevemente por qué.

```
public class Pregunta3 {
	public static void main(String[] args){
		int b = 3, a = 4;
		if (compara(a,b) & --a == 3){
			System.out.print(a--);
		} else {
			System.out.print(b++);
		}
		System.out.print (a+b);
	}
	public static boolean compara (int a, int b){
		if(b==3 | ++a/2>2){
			System.out.print(a);
			System.out.print(++b);
			return true;
		}
		else return false;
	}
}
a) 4435
b) 5437
c) 4535
d) 5435
e) 37
f) Ninguna de las anteriores.
```

La respuesta correcta es la d). Declaramos dos variables dentro del método main y les damos valor, a continuación llamamos al método comparar con esas dos variables como parámetros. El método comprueba que b==3 o que ++a/2>2, como es un OR y la primera es verdadera la expresión es verdadera, pero como no es en cortocircuito Java evalúa de todas formas la segunda expresión, por lo que a pasa a valer 5. A continuación imprime a (**5**), imprime b incrementado en uno (**4**) y termina devolviendo true. A continuación comprueba también si -–a == 3 lo que es VERDADERO porque los cambios hechos dentro del método no se reflejan fuera, a continuación imprime a-- (**3** porque el autodecremento va detrás) e imprime a+b que es **5**, porque hay que tener en cuenta el último autodecremento de a.

Pregunta 4 (1 Punto).- Encontrar y explicar los 3 errores de compilación que aparecen en el siguiente código Java. ¿Cómo los resolvería?

```
public class Pregunta4{
      public byte x;
      public boolean y;
      public char z;

         public Pregunta4 (boolean y1, char z1){
               x=12.0;
               y=true;
               z=z1;}

         public Pregunta4 (byte x1, char z1){
               x=x1;
               if (x1>10) y=true;
               else y=false;
               z=z1;}

         public Pregunta4 (byte x1, char z1, boolean y1){
               this(y1,z1);
               x=x1;}

         public Pregunta4 (byte x, char z){
               this.x=x;
               y=false;
               this.z=z;}

         public int getX(){
               return x;}

         public setX(byte x1){
               x=x1;}}
```

1) En el primer constructor asignamos a x que es byte un valor double, se arreglaría con un casting: x = (byte) 12.0
2) El primer y el cuarto constructor tienen los mismos parámetros, hay que eliminar uno de ellos.
3) El método setX no dice lo que devuelve, en este caso sería public void setX (byte x1)

No son errores:

- Que el método getX devuelva un byte cuando en su cabecera pone que debe devolver un int, ya que Java pasa automáticamente de byte a int (casting automático)
- Que el if del segundo constructor no lleve llaves. Las llaves sólo son necesarias cuando se va a poner más de una sentencia.
- Que se reciban parámetros que luego no se usen, como en el primer constructor. Eclipse lanzará un warning avisando, pero el programa compilará y funcionará sin problemas.

PARTE 2: PROBLEMAS

Problema 1 (4 Puntos).- Crear una clase pública denominada `Naranja` con las siguientes características:

- **(0,2 puntos)** Deberá pertenecer al paquete `fruta` e importar la clase `java.lang.Math`
- **(0,2 puntos)** Deberá tener los siguientes atributos públicos:
  - `origen` de tipo `String`
  - `numeroGajos` de tipo `int`
  - `exprimida` de tipo `boolean`
- **(0,2 puntos)** Deberá tener un atributo de tipo `int` común a todos los objetos de la clase, denominado `numeroNaranjas` que en todo momento almacene cuántos objetos `Naranja` se han creado.
- **(0,4 puntos)** Crear un método `getOrigen` que devuelva el valor del atributo `origen` y un método `setOrigen` que reciba un valor y establezca ese valor como nuevo `origen`. Hacer métodos similares para `numeroGajos`. Se debe comprobar que el número de gajos está entre 0 y 15 y en el caso de que sean 0, `exprimida` debe ser verdadero.
- **(0,4 puntos)** Crear un método exprimir que no devuelva nada ni reciba parámetros y al final de su ejecución la naranja está exprimida y el `numeroGajos` es cero.
- **(0,4 puntos)** Crear un método `setGajos` que no reciba parámetros y que establezca el `numerodeGajos` de la `naranja` aleatoriamente entre 0 y 15 (en el caso de que sean 0 `exprimida` debe ser verdadero).
- Crear los siguientes constructores
  - **(0,2 puntos)** Un constructor por defecto, que cree una naranja de Valencia con 10 gajos.
  - **(0,4 puntos)** Un constructor que reciba valores para los tres atributos. Deberá comprobar que el `numeroGajos` está entre 0 y 15. Si se sale de rango deberá usar los valores por defecto para ese atributo.
  - **(0,4 puntos)** Un constructor que sólo reciba valores para el `origen` y el `numeroGajos`. **Debe** usar el anterior constructor.
  - **(0,4 puntos)** Un constructor de copia, que reciba un objeto de tipo `Naranja` y cree otro con los mismos valores. **Debe** usar el segundo constructor creado.
- **(0,4 puntos)** Crear un método denominado `equals` para comparar si dos `naranjas` son iguales. Recibirá como parámetro un objeto `Naranja` y devolverá `true` si la `naranja` recibida tiene los mismos valores de `origen`, `numeroGajos` y `exprimida` que la `Naranja` sobre la que se invoca.
- **(0,4 puntos)** Crear un método denominado, `mostrar`, que devuelva la siguiente cadena de caracteres: `"Tengo una naranja de "+origen+" con "+numeroGajos+" gajos"` si la naranja no está exprimida y en caso contario devuelva `"Soy un zumo de naranja de "+origen.`

Problema 2 (1 Punto).- Crear una clase pública denominada `UsoNaranja` con las siguientes características:

- **(0,1 puntos)** Debe contener un método `main`.
- **(0,3 puntos)** Declarar dentro del método `main` dos variables de tipo `Naranja`. Crearlas usando un constructor distinto para cada una.
- **(0,3 puntos)** Crear un `array` de tipo `Naranja` de 2 posiciones. Crear los elementos del `array` usando los dos constructores que no se usaron en el punto anterior.
- **(0,3 puntos)** Escribir el código que compare 2 objetos `Naranja` de los creados e imprima por pantalla si son iguales (`true`) o no (`false`)

Problema 3 (1 Punto).- Crear un método que reciba como parámetro un array con todas las letras del alfabeto (en minúsculas y mayúsculas) y lo catalogue separándolo en 2 arrays distintos: el primero con las **vocales** y el segundo con las **consonantes**, y los muestre por pantalla. Nota: El array recibido ya está inicializado con: {“a”, “A”, “b”, “B”, “c”, “C”, “d”, “D”, …, “z”, “Z”}

PROBLEMA 1

```
package enero;
import java.lang.Math;
public class Naranja {

	public String origen;
	public int numeroGajos;
	public boolean exprimida;
	public static int numeroNaranjas;

	//Métodos get y set
	public String getOrigen(){
		return origen;
	}
	public void setOrigen(String o){
		origen = o;
	}
	public int getNumeroGajos (){
		return numeroGajos;
	}
	public void setNumeroGajos (int n){
		if (n>=0 && n<=15)
			numeroGajos = n;
		if (n==0) exprimida= true;
	}
	//Método exprimir
	public void exprimir (){
		exprimida = true;
		numeroGajos = 0;
	}
	//Método que pone los gajos de manera aleatoria
	public void setGajos (){
		numeroGajos = (int) Math.random()*16;
		if (numeroGajos == 0) exprimida = true;
	}

	//Constructor por defecto. No hace falta poner exprimida a falso, ya lo
hace Java
	public Naranja (){
		origen = "Valencia";
		numeroGajos = 10;
		numeroNaranjas++;
	}
	//Constructor completo
	public Naranja (String o, int n, boolean e){
		if (n>=0 && n<=15)
			numeroGajos = n;
		else numeroGajos = 10;
		origen = o;
		exprimida = e;
		numeroNaranjas++;
	}
	// Constructor de 2 parámetros
	public Naranja (String o, int n){
		//Como no se dice nada de exprimida la ponemos a false
		this (o,n,false);
		//Más correcto sería: this (o,n, n!=0)
	}
	//Constructor de copia
	public Naranja (Naranja n){
		this (n.origen, n.numeroGajos, n.exprimida);
	}
```

```
	//Método equals
	public boolean equals (Naranja n){
	if (n.origen.equals(origen) && n.exprimida==exprimida && n.numeroGajos==
numeroGajos)
		return true;
	else return false;
	}

	//Método mostrar
	public String mostrar (){
		if (exprimida) return "Soy un zumo de naranja de "+origen;
		else return "Tengo una naranja de "+origen+" con "+numeroGajos+"
gajos";
	}
}
```

PROBLEMA 2

```
//Hay que decir que están en el mismo paquete o importar la clase
package fruta;

public class UsoNaranja {

	public static void main(String[] args) {
		//Declaramos dos objetos naranja
		Naranja n1, n2;
		//Los creamos con los dos primeros constructores
		n1 = new Naranja();
		n2 = new Naranja ("valencia",12,false);
		//Creamos un array de 2 naranjas
		Naranja bolsa [] = new Naranja [2];
		//Y creamos sus elementos con los otros dos constructores
		bolsa [0]= new Naranja ("murcia",14);
		bolsa [1]= new Naranja (n1);
		//Para ver si dos naranjas son iguales usamos el método equals
		System.out.println(n1.equals(n2));

	}

}
```

PROBLEMA 3

```
	public void catalogo (String [] lista){
		//sabemos que hay 10 vocales
		String [] vocales = new String [10];
		// las consonantes serán el resto
		String [] consonantes = new String [lista.length-vocales.length];
		//creamos un para de variables para contar vocales y consonantes
		int numVoc=0, numCon=0;
		//y usamos un bucle para recorrer la lista
		for (int i=0; i<lista.length; i++){
			//si es una vocal la añadimos a su lista
			if (lista[i].equalsIgnoreCase("a") ||
lista[i].equalsIgnoreCase("e")
				||lista[i].equalsIgnoreCase("i")
||lista[i].equalsIgnoreCase("o")
				||lista[i].equalsIgnoreCase("u")){
				vocales[numVoc]=lista[i];
				numVoc++;
			}
			//Si no, es una consonante y hacemos lo propio
			else {
				consonantes[numCon]=lista[i];
				numCon++;
			}
		}
		//Imprimimos las dos listas
		for (int i=0; i<vocales.length;i++)
			System.out.print(vocales[i]+", ");
		System.out.println();
		for (int i=0; i<consonantes.length;i++)
			System.out.print(consonantes[i]+", ");
	}
```