EXAMEN DE PROGRAMACIÓN **Enero 2012** **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 permiten apuntes y/o libros para la realización del examen**

PARTE 1: CUESTIONES

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

```
1.  public class Ejecutable {
2.      public static void main(String[] args) {
3.          int a;
4.          float b;
5.          a = 12.0;
6.          b = 14;
7.          float c = suma(a, b);
8.          System.println("Valor: " + c);
9.      }
10.      public float suma(int primero, float segundo) {
11.          return primero + segundo;
12.      }
13.  }
```

1) En la línea 5, no se puede igualar un entero (“a”) a un valor flotante (12.0). Para solucionarlo, o bien se declara la variable “a” como flotante o bien se quita el decimal “.0”.
2) En la línea 8, Falta el “.out” para poder imprimir por pantalla. La línea correcta sería
   System.out.println(“Valor: “ + c);
3) En la línea 10, el método suma debe ser declarado como “static” para poder ser invocado desde un método estático (el main). Por tanto la declaración del método debería ser:

   public static float suma(int primero, float segundo) {...

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

2.1. **(0,25 puntos)** El resultado de ejecutar la siguiente sentencia es que se imprime un salto de línea por pantalla:

```
System.out.println("\n");
```

Falso, esa sentencia imprime dos saltos de línea por pantalla. Para imprimir un único salto de línea valdría cualquiera de las dos siguientes sentencias:

1) System.out.println();
2) System.out.print("\n");

El que las comillas sean simples o dobles (que sea un char o un String) es irrelevante.

2.2. **(0,25 puntos)** El siguiente bucle se ejecuta dos veces:

```
int i = 0;
while(true) {
     if(false) {
          break;
     } else if(i++ == 1) {
          break;
     }
}
```

Verdadero, la primera vez i vale cero (la comprobación se realiza antes de incrementarlo porque el ++ está después) por lo que no sale del bucle, y la segunda vez ya vale 1 por lo que se sale con la sentencia "break".

2.3. **(0,25 puntos)** El siguiente código NO compila:

```
short a, b;
short c;
c = a + b;
```

Verdadero, las variables a y b no están inicializadas, por lo que no se pueden sumar. Además, aunque estuvieran inicializadas, el resultado de sumar dos enteros de tipo short es un entero de tipo int, a no ser que se haga un casting explícito.

2.4. **(0,25 puntos)** Una clase siempre tiene un constructor sin parámetros (constructor por defecto), si no lo tiene Java se encarga de crearlo automáticamente.

Falso, el constructor por defecto sólo se crea si no existe otro constructor.
Por ejemplo, si una clase sólo tiene un constructor con varios parámetros, entonces esta clase no tendrá ningún constructor sin parámetros (Java no lo crea automáticamente si existe otro).

Pregunta 3 (1 Punto).- Explicar cuál sería la salida por consola al ejecutar la siguiente clase:

```
1.  public class Ejecutable {
2.      public static void main(String[] args) {
3.          int a = 5;
4.          float b = 0.9F;
5.          int c = (int) (a * b);
6.          System.out.println(c);
7.          int d, e;
8.          d = 0;
9.          e = cambia(d);
10.         System.out.println(d);
11.         System.out.println(e);
12.     }
13.     public static int cambia(int d) {
14.         return ++d;
15.     }
16.  }
```

1) Se declaran dos variables a y b, una de tipo entero y otra de tipo flotante. Antes de asignarse a la variable “c”, primero se multiplican (el resultado es 4.5), y luego se hace un casting a entero, por lo que se trunca la parte decimal y el resultado es 4. Por tanto **lo primero que se imprime es un 4**.

2) El método “cambia” recibe como parámetro una variable de tipo “int”. Al tratarse de un tipo básico, se pasa por valor y no por referencia, por lo que la variable original “d” no se modifica. Por tanto **lo segundo que se imprime es un 0**.

3) La variable “e” se iguala al resultado del método “cambia”, que lo que hace es devolver el valor de la variable que le han pasado incrementado, por lo que devuelve un 1. Por tanto **lo tercero que se imprime es un 1.**

PARTE 2: PROBLEMAS

Problema 1 (1,5 Puntos).- Crear un método que reciba como parámetro un array de booleanos. El método debe devolver un nuevo array que contenga siete números enteros, en el siguiente orden:

a) **(0,1 puntos)** El número total de elementos del array

b) **(0,3 puntos)** El número de elementos del array que son `true`

c) **(0,3 puntos)** El número de elementos del array que son `false`

d) **(0,1 puntos)** Un número que es cero si el primer elemento del array es `false`, y uno si es `true`.

e) **(0,1 puntos)** Un número que es cero si el último elemento del array es `false`, y uno si es `true`.

f) **(0,3 puntos**) El número de elementos del array en posiciones pares que son `true` (teniendo en cuenta que el primer número del array se considera la posición número uno, y por tanto su posición es impar).

g) **(0,3 puntos)** El número de elementos del array en posiciones impares que son `false` (teniendo en cuenta que el primer número del array se considera la posición número uno, y por tanto su posición es impar).

Por ejemplo, al invocarse al método con el array `{true, false, false, false, false}` el método debería devolver el array {5, 1, 4, 1, 0, 0, 2}, puesto que el array:

a) Tiene 5 elementos

b) Tiene 1 elemento que es `true`

c) Tiene 4 elementos que son `false`

d) El primer elemento es `true`.

e) El último elemento es `false`.

f) No tiene ningún elemento en posiciones pares que valga `true`

g) Tiene dos elementos en posiciones impares que valen `false`

Solución:

```
public static int[] metodo(boolean[] arr) {
        int[] ret = new int[7];
        //El primer número es la longitud del array
        ret[0] = arr.length;
        int verdaderos = 0, falsos = 0;
        int verdaderosPar = 0, falsosImpar = 0;
        for(int i=0; i<arr.length; i++) {
            if(arr[i]) {
                verdaderos++;
                if(i%2 == 1) {
                    verdaderosPar++;
                }
            }
            else {
                falsos++;
                if(i%2 == 0) {
                    falsosImpar++;
                }
            }
        }
        ret[1] = verdaderos;
```

```
        ret[2] = falsos;
        if(arr[0]) {
            ret[3] = 1;
        }
        if(arr[arr.length-1]) {
            ret[4] = 1;
        }
        ret[5] = verdaderosPar;
        ret[6] = falsosImpar;
        return ret;
    }
```

Problema 2 (1,5 puntos).- Crear un nuevo tipo de dato (una clase), denominado `Jarra`, que deberá tener:

a) **(0,1 puntos)** Dos atributos privados: `capacidad` de tipo `int` y `contenido` de tipo `int`

b) **(0,2 puntos)** Métodos `set` y `get` para `capacidad`. El método `set` deberá comprobar que la `capacidad` es mayor que cero.

c) **(0,2 puntos)** Métodos `set` y `get` para `contenido`. El método `set` deberá comprobar que el `contenido` es mayor o igual que cero y menor o igual que `capacidad`.

d) **(0,3 puntos)** Un constructor que reciba valores para los dos atributos y cree el objeto (comprobando que la `capacidad` y el `contenido` son correctos llamando a los métodos `set` anteriores)

e) **(0,3 puntos)** Un constructor por defecto, que cree una jarra vacía (`contenido` igual a 0) de `capacidad` 4.

f) **(0,1 punto)** Un método `llenaJarra` que no devuelve nada y llena la `Jarra`.

g) **(0,1 punto)** Un método `vaciaJarra` que no devuelve nada y vacía la `Jarra`

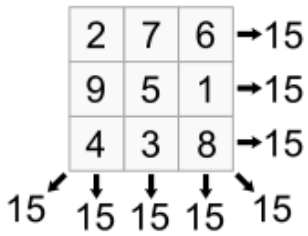
h) **(0,2 puntos)** Crear otra clase denominada `PruebaJarra`, con un método `main` en el que se crea una `Jarra` de 4 litros de `capacidad`, y otra de 3 litros. A continuación se llena la `Jarra` de 4 litros y se imprime por pantalla su `contenido`.

Solución:

```
public class Jarra {
      private int capacidad, contenido;

      //método getCapacidad
      public int getCapacidad (){
            return capacidad;
      }

      //método setCapacidad
      public void setCapacidad (int c){
            if (c>0) capacidad = c;
      }

      //método getContenido
      public int getContenido () {
            return contenido;
      }

      //método setContenido
      public void setContenido (int c){
            if (c>=0 && c<=capacidad) contenido = c;
      }
```

```
	//Constructor con parámetros
	public Jarra (int cap, int cont){
		setCapacidad(cap);
		setContenido(cont);
	}

	//Constructor por defecto
	public Jarra (){
		setCapacidad(4);
		//esto no haría falta
		setContenido(0);
	}

	//método llenaJarra
	public void llenaJarra (){
		contenido = capacidad;
	}

	//método vaciaJarra
	public void vaciaJarra(){
		contenido=0;
	}
}

public class PruebaJarra {
	public static void main (String [] args){
		Jarra jarra1, jarra2;
		jarra1 = new Jarra ();
		jarra2 = new Jarra (3,0);
		jarra1.llenaJarra();
		System.out.println(jarra1.getContenido());
	}
}
```

Problema 3 (3 Puntos).- Un cuadrado mágico es una matriz de dos dimensiones de tamaño NxN tal que al sumar los valores de cada fila, de cada columna y de las diagonales principales resulta el mismo número.



En este ejercicio se desarrollará un programa en Java que, dado un cuadrado de tamaño arbitrario N, imprima en pantalla un mensaje diciendo si se trata de un cuadrado mágico o no. Para ello:

a) **(0,2 puntos)** Crear un nuevo tipo de dato (una clase), denominado `Suma` que tendrá dos atributos públicos: `iguales` de tipo `boolean`, `resultado` de tipo `int`

b) **(0,6 puntos)** En otra clase, crear un método `compruebaFilas` que recibirá como parámetro un array de `int` de dos dimensiones (una matriz cuadrada) y deberá comprobar que todas las filas suman lo mismo. Devolverá un objeto de la clase `Suma`. Si todas las filas suman lo mismo, el atributo `iguales` será `true`, y el atributo `resultado` valdrá el valor de la suma. Si alguna no suma lo mismo que el resto, los

atributos valdrán `false` y cualquier valor respectivamente.

c) **(0,6 puntos)** Crear un método `compruebaColumnas` que se comporte exactamente igual que `compruebaFilas` pero para columnas.

d) **(1 punto)** Crear un método `compruebaDiagonales`, que de forma similar a los anteriores, recibirá como parámetro una matriz cuadrada y devolverá un objeto de la clase Suma. Si la diagonal principal y la diagonal inversa suman lo mismo, el atributo `iguales` será `true` y el atributo `resultado` tendrá el valor de la suma. Si no suman lo mismo, los atributos valdrán `false` y cualquier valor, respectivamente.

e) **(0,6 puntos)** Crear un método `main` que cree la matriz del ejemplo anterior (tipo `int`) y llame a los métodos creados para comprobar si la matriz es un cuadrado mágico o no. Deberá imprimir por pantalla si la matriz es un cuadrado mágico o no.

```
public class Suma {
      public boolean iguales;
      public int resultado;

   }

public class Problema3 {

public static Suma compruebaFilas (int [][] matriz){
      Suma res = new Suma ();
      res.iguales = true;
      int sumaParcial = 0, sumaTotal=0;
    //comprobamos las sumas, si alguna fila no lo cumple paramos
      for (int ii=0;ii<matriz.length && res.iguales;ii++){
            sumaParcial = 0;
            for (int jj=0;jj<matriz.length;jj++){
                  sumaParcial = sumaParcial+matriz[ii][jj];
            }
            //Si es la primera fila, usamos su suma para comparar con el resto
            if (ii==0) sumaTotal = sumaParcial;
            if (sumaParcial!=sumaTotal) res.iguales=false;
      }
      res.resultado = sumaTotal;
      return res;
}

public static Suma compruebaColumnas (int [][] matriz){
      Suma res = new Suma ();
      res.iguales = true;
      int sumaParcial = 0, sumaTotal = 0;
      //comprobamos las columnas
      for (int ii=0; ii<matriz.length && res.iguales; ii++){
            sumaParcial = 0;
            for (int jj=0; jj<matriz.length; jj++){
                  sumaParcial = sumaParcial+matriz[jj][ii];
            }
            //Si es la primera columna, usamos su suma para comparar
      if (ii==0) sumaTotal = sumaParcial;
            if (sumaParcial!=sumaTotal) res.iguales=false;
      }
      res.resultado = sumaTotal;
      return res;
}
```

```
public static Suma compruebaDiagonales (int [][] matriz){
      Suma res = new Suma();
      int sumaDiagonal = 0, sumaInversa = 0;

      for (int ii=0; ii<matriz.length; ii++){
            //los elementos de la diagonal principal tienen fila=columna
            sumaDiagonal = sumaDiagonal + matriz[ii][ii];
            //los elementos de la diagonal inversa tienen fila+columna=length-1
            sumaInversa = sumaInversa + matriz[ii][matriz.length-1-ii];
      }
      res.resultado = sumaDiagonal;
      res.iguales = sumaDiagonal == sumaInversa;
      return res;
}

public static void main(String[] args) {
      //Creamos la matriz
      int [][] m = {{2,6,6},{9,5,1},{4,3,8}};
      Suma filas = new Suma (), columnas = new Suma (), diagonales = new Suma();
      filas = compruebaFilas (m);
      columnas = compruebaColumnas (m);
      diagonales = compruebaDiagonales (m);
      if (filas.iguales && columnas.iguales && diagonales.iguales &&
                  filas.resultado==columnas.resultado &&
                  columnas.resultado==diagonales.resultado)
            System.out.println("Cuadrado mágico");
      else
            System.out.println("No es un cuadrado mágico");
}
}
```

Problema 4 (1 Punto).- Crear un programa que dado un `String s` y un `String w`, proporcionados como argumentos al programa, determine si `w` está incluido en `s`. No se permiten utilizar los métodos de Java para manejo de cadenas, a excepción de `charAt`.

```
public class Substring {
      public static void main(String [] args) {
            //Comprobamos que hay 2 argumentos
            if(args.length != 2) {
                  System.out.println("Número de parámetros incorrectos.");
                  System.exit(-1);
            }

            String s = args[0];
            String w = args[1];
            //Comprobamos si el primer carácter de w está en s
            boolean found = false;
            for(int i=0; i<s.length() && !found; i++) {
                  int j=0;
                  boolean equal = true;
                  while(j<w.length() && i+j<s.length() && equal) {
                        if(s.charAt(i+j)!=w.charAt(j))
                              equal=false;
                        else
                              j++;
                  }
                  if(j==w.length())
                        found=true;
            }
            if(found)
                  System.out.println(w + " está contenido en " + s);
            else
                  System.out.println(w + " no está contenido en " + s);
      }
}
```