EXAMEN DE PROGRAMACIÓN 17 de Enero 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 = 2;
		System.out.println(a++);
		a = modificar(a);
		System.out.println(++a);
	}
	public static int modificar(int a) {
		a = a + 5;
		return 9;

	}
}
```

Respuesta: Inicialmente se da a la variable “a” el valor 2. Al imprimir el valor con “a++”, como el operador está después de la variable, primero se imprime (2) y luego se incrementa su valor. Posteriormente se iguala el valor de la variable “a” a lo que devuelve el método “modificar”, es decir, “a” vale 9. En la siguiente impresión por pantalla, como el operador “++” está antes de la variable, primero se incrementa y luego se imprime (10). Por tanto al ejecutar el programa se imprime por pantalla:

2

10

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

1.1. **(0,25 puntos)** El uso de `break` está recomendado únicamente en la sentencia de control `switch`, en cualquier otro caso da error.

Falso. Además de su uso dentro de una sentencia "switch", se puede utilizar "break" dentro de cualquier bucle (for, while, do-while). Esto no da error, lo que hace break dentro de un bucle es salir de éste.

1.2. **(0,25 puntos)** Los parámetros o argumentos de tipo array que reciben los métodos se pasan siempre por referencia, esto es, si dentro del método se modifica un elemento del array estamos modificando también el array original.

Verdadero. En Java, mientras que los tipos básicos se pasan por valor, tanto los arrays como los objetos se pasan por referencia.

1.3. **(0,25 puntos)** En una sentencia de control `switch` podemos hacer que dos valores posibles (dos `case`) ejecuten el mismo código sin necesidad de duplicar dicho código.

Verdadero. No poniendo "break" en el primer "case", se puede hacer que dos valores posibles de la variable ejecuten el mismo código. Ejemplo:

```
Case 1:
Case 2:
        System.out.println("Esto se imprime tanto si vale 1 como 2");
        break;
```

1.4. **(0,25 puntos)** Siempre que queramos buscar un número dentro de un array de enteros, podemos hacerlo utilizando búsqueda binaria para ahorrarnos operaciones.

Falso. Para poder utilizar búsqueda binaria es necesario que el array en el que deseamos buscar el valor esté ordenado.

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

```
public static void main(String[] args) {
      int a=8;
      boolean b=false;
      double c=0;
      c = metodo1 (a,b);
      System.out.println(c);
}
```

Sabiendo que imprime `12.0` por pantalla, indicar y **explicar** para cada uno de los métodos siguientes si podría ser el método al que se está llamando o no:

a)

```
public static void metodo1 (int a, boolean b){
      if (b== false)
            return 12;
      return 4;
}
```

NO puede ser. Este método no compila. El método es declarado como "void", por tanto no puede devolver ningún valor.

b)

```
public static double metodo1 (boolean b, int d){
      double c=0;
      if (b== false){
            c= c+4;
            }
      return d+c;
}
```

NO puede ser. El método que se invoca desde el main recibe dos parámetros, el primero de ellos es un int y el segundo un boolean. En este método los parámetros están cambiados de orden.

c)

```
public static double metodo1 (int x, boolean w){
      double y;
      if (!w==true) {
            x=x+4;
      }
      y=x;
      return x;
}
```

SI puede ser. El método se invoca con "8, false", por lo que entra dentro del if, y por tanto se incrementa el valor de la variable que se devuelve, pasa a valer 12.

d)

```
public static int metodo1 (int x, boolean y){
      return 12;
}
```

SI puede ser. El método se invoca con "8, false", y devuelve un int que vale 12. El main obtiene el valor de vuelta y hace un casting automático a double para meterlo en la variable c, que luego imprime.

e)

```
public static double metodo1 (int a, boolean z){
      return 12.0;
}
```

SI puede ser. El método se invoca con "8, false", y devuelve un double que vale 12.

PARTE 2: PROBLEMAS

Problema 1 (2,5 Puntos).- Dado el siguiente código en Java:

```
public static void main(String[] args) {
      int[][] matrizCuadrada = new int[][] {
                  { 2,  3,   9,  4},
                  { 4,  1,  3,  5},
                  { 1,  9,  9,  9},
                  {-7,  8,  0,  1}};
      int[][] matrizNOCuadrada = new int[][] {
                  { 3,  3,  8,  2},
                  { 1,  1,  4,  4},
                  { 1,  9,  9,  9, 2, 2},
                  { 7,  8,  0,  2}};

      boolean tmp = tieneNegativos(matrizCuadrada);
      System.out.println("Primer Valor: " + tmp);
      tmp = tieneNegativos(matrizNOCuadrada);
      System.out.println("Segundo Valor: " + tmp);

      int suma = sumaDiagonal(matrizCuadrada);
      System.out.println("Tercer Valor: " + suma);
      suma = sumaDiagonal(matrizNOCuadrada);
      System.out.println("Cuarto Valor: " + suma);

      suma = sumaMitadInferior(matrizCuadrada);
      System.out.println("Quinto Valor: " + suma);
      suma = sumaMitadInferior(matrizNOCuadrada);
      System.out.println("Sexto Valor: " + suma);
}
```

Y sabiendo que al ejecutarse imprime por pantalla:

```
Primer Valor: true
Segundo Valor: false
Tercer Valor: 13
Cuarto Valor: 0
Quinto Valor: 30
Sexto Valor: 49
```

Implementar los siguientes métodos:

1) **(0,75 Puntos)** Método `tieneNegativos`: Dado un array bidimensional de números enteros (puede no ser una matriz cuadrada), devuelve `true` si contiene algún número negativo, o `false` de lo contrario.

```
public static boolean tieneNegativos(int[][] m) {
      for(int i=0; i<m.length; i++) {
            for(int j=0; j<m[i].length; j++) {
                  if(m[i][j] < 0) {
                        return true;
                  }
            }
      }
      return false;
}
```

2) **(1 Punto)** Método `sumaDiagonal`: Dada una matriz cuadrada de números enteros, devuelve la suma de los elementos de su diagonal. Si la matriz no es cuadrada, devuelve cero.

```
public static int sumaDiagonal(int[][] m) {
      int suma = 0;
      for(int i=0; i<m.length; i++) {
            // Si no es cuadrada devuelve cero
            if(m[i].length != m.length) {
                  return 0;
            }
            suma += m[i][i];
      }
      return suma;
}
```

3) **(0,75 Puntos)** Método `sumaMitadInferior`: Dada una matriz de números enteros, devuelve la suma de los elementos de su segunda mitad (de la mitad inferior de la matriz).

Nota: Si la matriz contiene un número impar de filas, la fila del medio **SÍ** se tendrá en cuenta en la suma.

```
public static int sumaMitadInferior(int[][] m) {
      int suma = 0;
      for(int i=m.length/2; i<m.length; i++) {
            for(int j=0; j<m[i].length; j++) {
                  suma += m[i][j];
            }
      }
      return suma;
}
```

Problema 2 (1 Punto).- Escribir el código de un método `imprimePrimos` que recibe un número como parámetro e imprime todos los números primos comprendidos entre dos y el número recibido.

Ejemplo: si se invoca `imprimePrimos(30)`, el método deberá imprimir por pantalla:

Los números primos entre 2 y 30 son: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29

* Nota: Un número primo es aquel que sólo es divisible por sí mismo y por 1.

```
public static void imprimePrimos(int hasta) {
  System.out.print("Los números primos entre 2 y " + hasta + " son: ");
  boolean bPrimero = true;
  boolean bDivide = false;
  for(int i=2; i<=hasta; i++) {
      for(int j=2; j<i & !bDivide; j++) {
          if(i%j == 0) {
              bDivide = true;
          }
      }
      if(!bDivide) {
          if(!bPrimero) {
              System.out.print(",");
          }
          System.out.print(i);
          bPrimero = false;
      }
      bDivide=false;
  }
}
```

Problema 3 (1,5 Puntos).- Crear una clase denominada `Problema3`, que contenga dos métodos con las siguientes especificaciones:

1) **(1 Punto)** Método `convertir`:
   - Recibe como parámetro un array de enteros (denominado `lista`).
   - Crea **un nuevo array** de la misma dimensión que `lista`. El nuevo array tendrá los mismos valores que `lista`, excepto que los números impares serán transformados en pares (para hacerlo, se sumará 1 a los números impares). Cada vez que convierta un número impar en par, debe mostrar por pantalla "Número impar encontrado: *X*" donde *X* es el número impar.
   - **Devolverá** el nuevo array, sin modificar el original

   NOTA: El método debe funcionar con arrays de cualquier longitud.

2) **(0,5 Puntos)** Método `main`:
   - Inicializa un array de 10 números enteros con valores aleatorios entre 0 y 99 (ambos inclusive).
   - Llama al método `convertir`
   - Imprime el array original y el convertido

EJEMPLO DE SALIDA POR PANTALLA DEL PROGRAMA:

```
Número impar encontrado: 65
Número impar encontrado: 17
Número impar encontrado: 45
Número impar encontrado: 15
Array Inicial:
65 16 17 86 80 45 88 15 50 52
Array Convertido:
66 16 18 86 80 46 88 16 50 52
```

```
public class Problema3 {
	public static int[] convertir(int[] original) {
		int[] resultado = new int[original.length];
		for(int i=0; i<original.length; i++) {
			if(original[i]%2 == 0) {
				resultado[i] = original[i];
			}
			else {
				System.out.println("Número impar encontrado: " +
original[i]);
				resultado[i] = original[i] + 1;
			}
		}
		return resultado;
	}

	public static void main(String[] args) {
		int[] array = new int[10];
		for(int i=0; i<10; i++) {
			array[i] = (int) (Math.random() * 100);
		}
		int[] array2 = convertir(array);
		System.out.println("Array Original:");
		for(int i=0; i<10; i++) {
			if(i>0) {
				System.out.print(",");
			}
			System.out.print(array[i]);
		}
		System.out.println();
		System.out.println("Array Convertido:");
		for(int i=0; i<10; i++) {
			if(i>0) {
				System.out.print(",");
			}
			System.out.print(array2[i]);
```

```
            }
      }
}
```

Problema 4 (2,5 Puntos).- Crear una clase pública denominada `Asiento` que representará un asiento en el vagón de un tren. La clase tendrá las siguientes características:

- **(0,2 puntos)** Deberá tener los siguientes atributos privados:
  - `identificador` (Cadena de texto con el identificador único del asiento)
  - `ventana` (Indicador de si el asiento está junto a la ventana del vagón o no)
- **(0,2 puntos)** Deberá tener los siguientes atributos públicos:
  - `fila` (Número que indica la fila del asiento)
  - `columna` (Número que indica la columna del asiento)

- Crear los siguientes métodos públicos:
  - **(0,5 puntos)** `setIdentificador`: Calcula y asigna el identificador del asiento, según la fila y la columna recibidas como parámetro. El método no devuelve nada.
    Por ejemplo, si se invoca al método como `setIdentificador(2, 4)` [fila 2 y `columna 4`], el identificador del asiento será "2D".
    NOTA: Reutilizar el siguiente fragmento de código para calcular la letra correspondiente a la columna:
    `String letra = String.valueOf( (char) (c + 'A' - 1) );`
  - **(0,2 puntos)** `getIdentificador`: Devuelve la cadena de texto correspondiente al identificador del asiento (asumir que el valor se ha asignado previamente llamando a `setIdentificador`).
  - **(0,5 puntos)** `setVentana`: Recibe un parámetro de tipo entero (llamado `asientosPorFila`), que representa el número de asientos por fila. El método debe calcular si el asiento está junto a la ventana o no, y asignar el valor correspondiente al atributo `ventana`. Un asiento está junto a la ventana si está en la primera columna (columna 1), o en la última de la fila. El método no devuelve ningún valor.
  - **(0,2 puntos)** `getVentana`: Devuelve el valor correspondiente al atributo `ventana` del asiento.

```
public class Asiento {
      private String identificador;
      private boolean ventana;
      public int fila;
      public int columna;
      public void setIdentificador(int fila, int columna) {
            identificador = String.valueOf(fila);
            String letra = String.valueOf( (char) (columna + 'A' - 1) );
            identificador += letra;
      }
      public String getIdentificador() {
            return identificador;
      }
      public void setVentana(int asientosPorFila) {
            if(columna == 1 || columna == asientosPorFila) {
                  ventana = true;
            } else {
                  ventana = false;
            }
      }
      public boolean getVentana() {
            return ventana;
      }
}
```

- Crear una clase denominada UsarAsiento y dentro de ella un método main, que deberá:
  - (**0,2 puntos**) Crear un objeto asiento, en la fila 8 columna 9. Dar valor adecuadamente a los atributos identificador y ventana.
  - (**0,5 puntos**) Crear un array de 40 objetos asiento. Utilizar un bucle para crear cada uno de los elementos del array (suponer que hay 10 filas y 4 columnas).

```
public class UsarAsiento {
	public static void main(String[] args) {
		Asiento s1 = new Asiento();
		s1.fila = 8;
		s1.columna = 9;
		s1.setIdentificador(8, 9);
		s1.setVentana(9);

		Asiento[] todos = new Asiento[40];
		for(int i=0; i<todos.length; i++) {
			todos[i] = new Asiento();
			todos[i].fila = (i%10) + 1;
			todos[i].columna = (i / 10) + 1;
			todos[i].setIdentificador(todos[i].fila, todos[i].columna);
			todos[i].setVentana(4);
		}
	}
}
```