Práctica 3: Clases y objetos

El objetivo de esta práctica es la utilización de los conceptos de clase y objeto.

Ejercicio 3.1: Utilización de la clase Círculo.

La clase Circulo es la que se muestra a continuación.

```
// fichero Circulo.java

public class Circulo {
    static int numCirculos = 0;
    public static final double PI=3.14159265358979323846;
    public double x, y, r;

    public Circulo(double x, double y, double r) {
        this.x=x; this.y=y; this.r=r;
        numCirculos++;
    }

    public Circulo(double r) { this(0.0, 0.0, r); }
    public Circulo(Circulo c) { this(c.x, c.y, c.r); }
    public Circulo() { this(0.0, 0.0, 1.0); }

    public double perimetro() { return 2.0 * PI * r; }
    public double area() { return PI * r * r; }

    // método de objeto para comparar círculos
    public Circulo elMayor(Circulo c) {
        if (this.r>=c.r) return this; else return c;
    }

    // método de clase para comparar círculos
    public static Circulo elMayor(Circulo c, Circulo d) {
        if (c.r>=d.r) return c; else return d;
    }

    // método de para imprimir coordenadas y radio
    public String toString() {
        return "x: " + x + " y: " + y + " r: " + r;
    }
} // fin de la clase Circulo
```

Se debe realizar un programa que use los métodos de la clase Circulo. Se puede utilizar como programa inicial el siguiente:

```
class pruebaCirculos {
    public static void main(String[] args) {
      System.out.println(new Circulo());
      System.out.println(new Circulo(5.0));
      System.out.println(new Circulo(3,5,2));
      Circulo c1 = new Circulo(7);
      Circulo c2 = new Circulo(12, 10, 8);
      Circulo c3 = new Circulo(c1);
      Circulo c4 = c1;
      System.out.println("Circulo1: " + c1);
      //...
      System.out.println("Perimetro de Circulo1: " + c1.perimetro());
      //...
      System.out.println("El mayor de: " +c1+ ", " +c2+ " es " +  c1.elMayor(c2));
      //...
      System.out.println("Cambio de radio de c3 a 10 y c4 a 15");
```

```
      c3.r = 10;
      //...
    }
}
```

El Circulo3 se crea con el constructor Circulo (Circulo c) utilizando como argumento el Circulo1 y el Circulo4, no se crea con new(), si no que se inicializa con Circulo1. Se debe completar el programa para que obtenga la siguiente salida con las funciones de la clase Circulo:

```
x: 0.0 y: 0.0 r: 1.0
x: 0.0 y: 0.0 r: 5.0
x: 3.0 y: 5.0 r: 2.0
Circulo1: x: 0.0 y: 0.0 r: 7.0
Circulo2: x: 12.0 y: 10.0 r: 8.0
Circulo3: x: 0.0 y: 0.0 r: 7.0
Circulo4: x: 0.0 y: 0.0 r: 7.0
Perimetro de Circulo1: 43.982297150257104
Area de Circulo1: 153.93804002589985
Perimetro de Circulo2: 50.26548245743669
El mayor de: x: 0.0 y: 0.0 r: 7.0, x: 12.0 y: 10.0 r: 8.0 es x: 12.0 y: 10.0 r: 8.0
El mayor de: x: 0.0 y: 0.0 r: 7.0, x: 12.0 y: 10.0 r: 8.0 es x: 12.0 y: 10.0 r: 8.0
Cambio de radio de c3 a 10 y c4 a 15
Circulo1: x: 0.0 y: 0.0 r: 15.0
Circulo2: x: 12.0 y: 10.0 r: 8.0
Circulo3: x: 0.0 y: 0.0 r: 10.0
Circulo4: x: 0.0 y: 0.0 r: 15.0X
```

La primera comparación de mayor está realizada con la función:

```
Circulo elMayor(Circulo c)
```

La segunda debe realizarse con el método static:

```
static Circulo elMayor(Circulo c, Circulo d)
```

Ejercicio 3.2: Aplicación con objetos.

En el ejercicio 2.6, del capítulo anterior se creaba una aplicación con un menú y las operaciones de introducir, mostrar y multiplicar dos matrices. En este ejercicio se deben realizar las mismas funciones pero utilizando la clase Matriz, que contiene tres variables miembro:

- Una variable entera (int) para el número de filas.
- Una variable entera (int) para el número de columnas.
- Un vector de dos dimensiones para contener los valores de cada celda.

De esta manera cada matriz puede tener sus propias dimensiones y la lectura de la matriz recoge todos los valores de dicha matriz.

Una definición inicial de la clase matriz es:

```
import java.util.*;

class Matrix {
    int filas;
    int columnas;
    float [][] value;

    public Matrix() {}

    public Matrix(int filas, int columnas) {
        this.filas = filas;
        this.columnas = columnas;
        value = new float[filas][columnas];
    }

    public String toString() {
        return "Matriz de: " + filas + " filas y " + columnas + " columnas.";
    }
}
```

Completar la clase Matrix para que tenga las funciones de lectura, escritura y producto, esta última función definida como estática (static). Y modificar el programa para que utilice la clase Matrix.