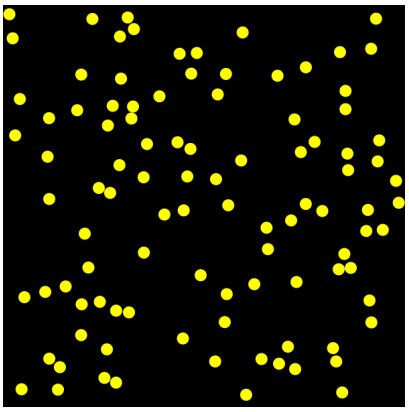
Programación II: Lab7

GII - GII+ADE - ETSIInf - UPM - Curso 23/24

1. Descripción del Problema

El objetivo de este lab es la implementación de una aplicación (`DibujaCirculos`) para generar en un lienzo un conjunto de circulos en posiciones aleatorias de modo que no se toquen entre sí.

- Pueden quedar tangentes a los bordes del lienzo.
- Todos los círculos deben ser del mismo tamaño y del mismo color.
- El fondo debe ser negro.



2. Material necesario

Todo el material está en un archivo `lab07.zip` que contiene los siguientes archivos:

- `NodoCirculo.java`
- `Punto.java`
- `CirculoTest.java` (por completar)
- `Circulo.java` (por completar)
- `stdlib.jar`

3. Material a entregar

- `CirculoTest.java`
- `Circulo.java`
- `GeometriaCirculo.java`
- `ListaDeCirculos.java`
- `DibujaCirculos.java`

4. Compilación y ejecución

Para poder compilar y ejecutar te recomendamos usar las siguientes opciones de `javac` y de `java` (lo vemos con el ejemplo de compilar y ejecutar la aplicación principal):

```
mi_ruta>\lab07> javac -d bin -cp .;bin;stdlib.jar DibujaCirculos.java
mi_ruta>\lab07> java -cp bin;stdlib.jar DibujaCirculos
```

5. La *Abstracción de Datos* `Circulo` y su *tester* `CirculoTest`

Empezaremos completando las pruebas unitarias de la clase `Circulo`: `CirculoTest`. Es importante que añadas todas las pruebas que se te ocurran para intentar probar tu futura implementación de la clase `Circulo`.

Recuerda que para ejecutar las pruebas tendrás que usar la opción `-ea` (*enable assertions*) de `java`.

Ahora ya estamos en disposición de completar la *Abstracción de Datos* `Circulo` definida mediante la siguiente especificación:

```
Circulo

- centro : Punto
- radio : double

+ Circulo (x0, y0, radio0 : double)
    Constructor simple de Circulo

+ trasladar (dx, dy : float)
    Mueve este circulo según (dx, dy)

+ distancia (otro : Circulo) : double
    POST: resultado es la distancia de este Circulo a otro

+ radio () : double
    Retorna el radio de este Circulo

+ centro () : Punto
    Retorna el centro de este Circulo

+ equals (o : Object) : bool
    POST: "este Circulo es igual al objeto o"
```

6. `GeometriaCirculo`

Esta clase va a incluir dos *Abstracciones Funcionales* necesarias para la aplicación.

Su especificación es:

```
GeometriaCirculo

+ static intersectan (c1, c2 : Circulo) : boolean
    POST: "c1 intersecta con c2"

+ static estaDentro (c : Circulo; x1, y1, x2, y2 : double) : boolean
    PRE: x1 <= x2, y1 <= y2
    POST: "c es interno o tangente interior al rectangulo
           cuyo punto inferior izquierdo es (x1, y1) y
           cuyo punto superior derecho es (x2, y2)"
```

7. La *Abstracción de Datos* `ListaDeCirculos`

Esta *Abstracción de Datos* es una lista de `Circulo`(s) implementada en la forma de una cadena enlazada. Su especificación es:

```
ListaDeCirculos

- inicio : NodoCirculo   // Referencia al primer nodo de la cadena enlazada

+ ListaDeCirculos ()
    Constructor de una lista Circulos

+ void insertarPrincipio (c : Circulo)
    Añade c a la lista por el principio

+ void insertarFinal (c : Circulo)
    Añade c a la lista por el final

+ longitud () : int
    POST: resultado es la longitud de la lista

+ obtener (pos : int) : Circulo
    POST: resultado es el Circulo que está en la posición pos en la lista
          (el primer elemento ocupa la posición 0).
```

8. DibujaCirculos

Esta clase va a incluir las *Abstracciones Funcionales* de más alto nivel de la aplicación además del método `main`. Su especificación es:

```
DibujaCirculos

+ static iniciarLienzo ()
    Inicializa el sistema gráfico con un lienzo.

+ static dibujarCirculo (c : Circulo; color : Color)
    Dibuja c con relleno en color

+ static generarCirculo (r : double) : Circulo
    POST: resultado es un Circulo de radio r en una posición aleatoria

+ static generarColor () : Color
    POST: resultado es un Color aleatorio

+ static generarCirculos (n : int) : ListaDeCirculos
    POST: resultado es una lista Circulos formada por n Circulo(s)
          en posiciones aleatorias

+ static dibujarCirculos (circulos : ListaDeCirculos; color : Color)
    Dibuja los Circulo(s) de la lista circulos con color

+ static dibujarCirculos (circulos : ListaDeCirculos)
    Dibuja los Circulo(s) con colores aleatorios

+ static esExterno(c : Circulo; circulos : ListaDeCirculos) : bool
    POST: c es un Circulo externo a todos los que están en la lista circulos

+ static generarValidos (n : int) : ListaDeCirculos
    POST: resultado es una lista formada por n Circulo(s)
          en posiciones aleatorias que son validos, es decir, que
          no se interesectan entre sí y están completamente dentro del
          lienzo o son tangentes interiores al lienzo.
```

Añade un método `main` directamente la clase `DibujaCirculos` para dibujar todos los círculos generados por `generarValidos` y haz las modificaciones necesarias para que aparezca la siguiente información en el lienzo: el número de circulos que se han llegado a generar, el número de círculos no válidos y el porcentaje de éxito (hay varios algoritmos que podrías utilizar usando las abstracciones funcionales anteriores).