# PRÁCTICA 5

## Gráficos 2D

**Parte 1**

El objetivo de esta práctica es adentrarnos en las principales características de la programación de gráficos usando la tecnología Java2D; concretamente, abordaremos los distintos `Shape` que ofrece la tecnología.

Para alcanzar el objetivo anterior, iremos siguiendo el tutorial **"***2D Graphics***"** de Java (http://docs.oracle.com/javase/tutorial/2d/index.html) y, más concretamente, el capítulo "*Working with Geometry*" (http://docs.oracle.com/javase/tutorial/2d/geometry/index.html).

# Formas geométricas: Shape

En este primer bloque probaremos los distintos `Shape` que ofrece el Java2D. Para ello, crearemos una aplicación sencilla e iremos siguiendo el tutorial en su capítulo "*Working with Geometry*". Concretamente, iremos realizando lo siguiente:

- Usando NetBeans, y siguiendo la misma dinámica que en prácticas anteriores, crear una aplicación que incorpore una ventana principal (`JFrame`).
- En la ventana principal, sobrecargar el método `paint` para usar `Graphics2D`:

```
public void paint(Graphics g) {
  super.paint(g);
  Graphics2D g2d=(Graphics2D)g;

  //Código usando g2d
}
```

- Probar todos los `Shape` definidos en Java2D. Para ello, nos guiaremos por lo ejemplos que podemos encontrar en el capítulo "*Working with Geometry*" del tutorial. El código lo iremos incorporando en el método `paint`, incluyendo, para cada forma que probemos:

  1. La creación de la forma (`new`) usando el constructor correspondiente
  2. El dibujo de la forma creada usando las sentencias `draw` y `fill` de la clase `Graphics2D`.

  Por ejemplo, para el caso de la línea:

```
public void paint(Graphics g) {
  super.paint(g);
  Graphics2D g2d=(Graphics2D)g;

  // Línea
  Point2D p1=new Point2D.Float(70,70),
  Point2D p2=new Point2D.Float(200,200);
  Line2D linea = new Line2D.Float(p1,p2);
  g2d.draw(linea);
}
```

Dentro del capítulo "*Working with Geometry*" del tutorial, la sección "Drawing Geometric Primitives" incluye ejemplos correspondientes a los `Shape`:

- `Line2D`
- `Rectangle2D`
- `RoundRectangle2D`
- `Ellipse2D`
- `Arc2D`
- `QuadCurve2D`
- `CubicCurve2D`

Analizar los ejemplos del tutorial e incorporar las formas anteriores en el método `paint` de nuestra aplicación.

- En la sección "Drawing Arbitrary Shapes" del mismo capítulo, se explica el uso de la forma "Trazo libre" (`GeneralPath`). Analizar los ejemplos del tutorial e incorporarlos a nuestra aplicación.

# Ejemplo incorporando eventos

En los ejemplos anteriores hemos usado coordenadas fijas para probar las distintas formas. El objetivo era conocer los objetos `Shape` que nos ofrece Java2D, de ahí que hayamos optado por ejemplos sencillos que no implicasen interacción con el usuario.

En esta parte de la práctica, incorporaremos la gestión de eventos en alguno de los ejemplos anteriores. Concretamente, incluiremos la interacción con el usuario para el dibujo de una línea (es decir, que el usuario seleccione los puntos inicial y final de la línea, visualizándola durante el proceso de "arrastrar y soltar").

Recordar que en esta aplicación, a diferencia de aquellos ejemplos en los que sólo disponíamos de `Graphics`, ya trabajábamos con las posibilidades que ofrece `Graphics2D`; en particular, se usa un objeto `Line2D` como elemento base:

① 
```
Line2D linea;
```

② 
```
public void paint(Graphics g) {
  super.paint(g);
  Graphics2D g2d=(Graphics2D)g;
  if(linea!=null) g2d.draw(linea);
}
```

*

③ 
```
private void formMousePressed(java.awt.event.MouseEvent evt) {
  linea = new Line2D.Float(evt.getPoint(),evt.getPoint());
}

private void formMouseDragged(java.awt.event.MouseEvent evt) {
  linea.setLine(linea.getP1(),evt.getPoint());
  this.repaint();
}

private void formMouseReleased(java.awt.event.MouseEvent evt){
  linea.setLine(linea.getP1(),evt.getPoint());
  this.repaint();
}
```

* Métodos incorporados por NetBeans cuando se gestionan los correspondientes eventos y que son llamados desde la clase manejadora (no se incluye dicha clase ni sus métodos en el código de este guión)

# Mantener las formas: vector de *Shape*

En el ejemplo anterior sólo se mostraba la última línea pintada; el objetivo de este último bloque es conseguir que las líneas que vayamos dibujando se mantengan en el lienzo. Para ello:

1. Creamos un vector de formas:

```
List<Shape> vShape = new ArrayList();
```

2. En el método *paint* dibujamos el contenido del vector

```
public void paint(Graphics g){
  super.paint(g);
  Graphics2D g2d = (Graphics2D)g;
  for(Shape s:vShape) g2d.draw(s);
  if(linea!=null) g2d.draw(linea);
}
```

3. Cada nueva línea se introducirá en el vector:

```
private void formMouseReleased(java.awt.event.MouseEvent evt){
  linea.setLine(linea.getP1(),evt.getPoint());
  vShape.add(linea);
  this.repaint();
}
```