



ORACLE

Academy



Java Foundations

9-2

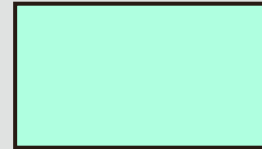
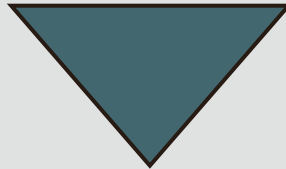
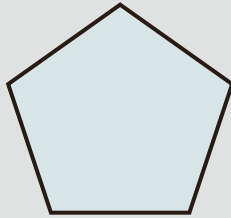
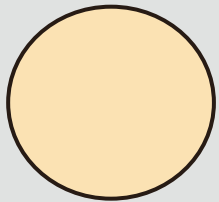
Colores y formas

ORACLE
Academy



Chicos y chicas, ¿Sabéis qué?

- Hoy vamos a aprender los colores y las formas



¡Yupi!



Objetivos

- En esta lección se abordan los siguientes objetivos:
 - Crear y usar campos personalizados
 - Crear formas y explicar sus propiedades y comportamientos
 - Consulte la documentación de la API de JavaFX

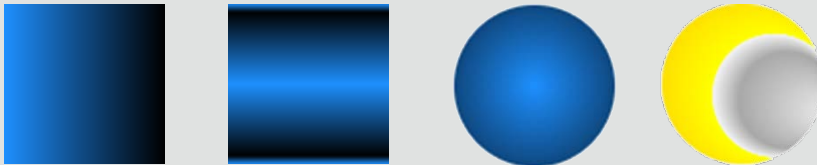


¿Qué puedo hacer con los colores en JavaFX?

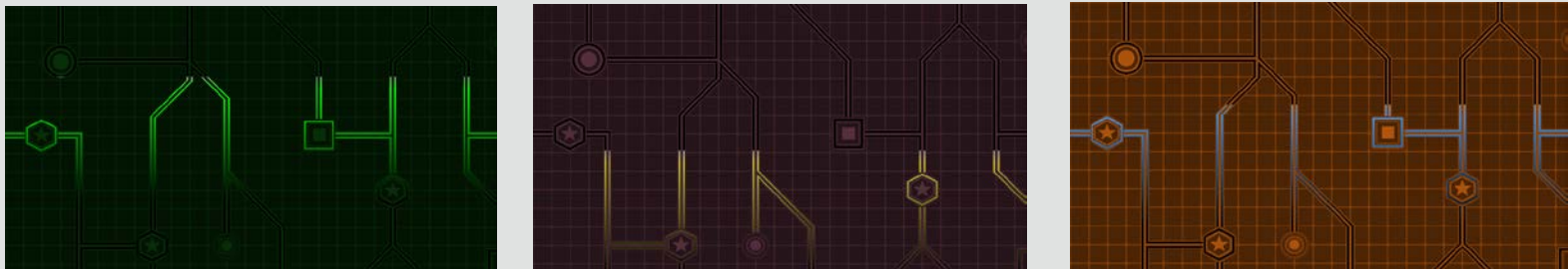
- Colorear formas



- Crear degradados



- Colorear imágenes



JavaFX contiene una clase Color

- Los colores se pueden almacenar como variables:

```
Color color = Color.BLUE;
```

- Los colores se pueden transferir en métodos:

```
Scene scene = new Scene(root, 300, 250, Color.BLACK);
```

- Este ejemplo crea el fondo negro de la escena

- Pero antes de utilizar cualquier color...

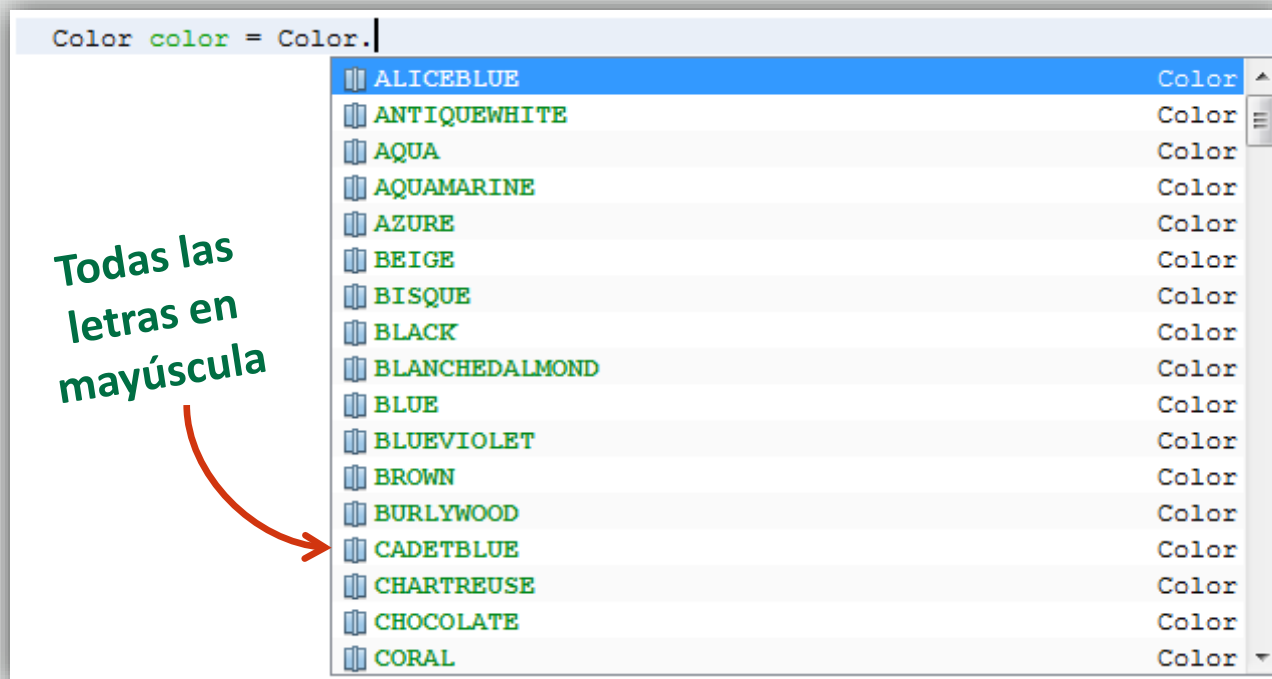
- Deberá importar, en primer lugar, lo siguiente:

```
import javafx.scene.paint.Color;
```

- Ignore las otras sugerencias de importación de Color de su IDE

Referencia a un color

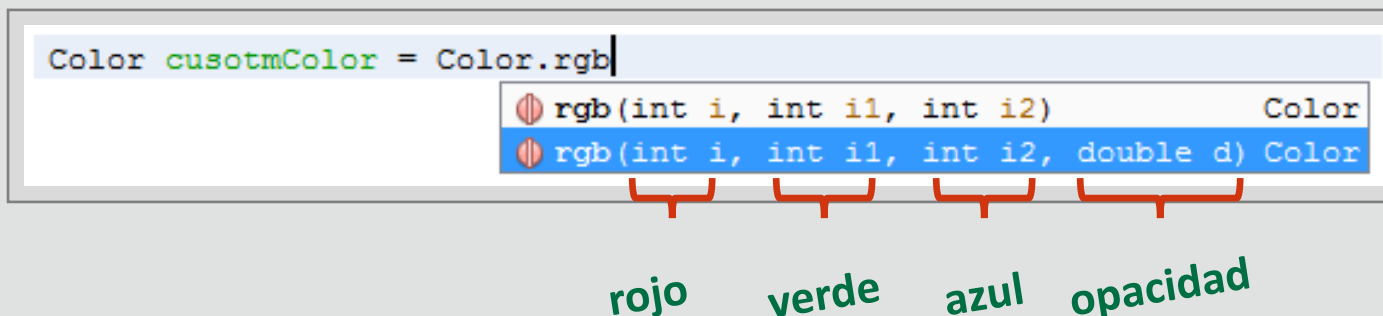
- Hay muchos colores en JavaFX
- Si escribe `Color.` en su IDE, se mostrará toda la lista de colores posibles



Personalización de un color

- Si no está satisfecho con los colores que JavaFX proporciona, hay formas de personalizar su propio color
- La clase Color contiene métodos para realizar esta acción:

```
Color customColor = Color.rgb|
```



rojo verde azul opacidad

- Personalizar un color mediante la mezcla de los componentes rojos, verdes y azules
- También se puede controlar la opacidad

El rango de componentes de color

```
Color customColor = Color.rgb|
```

```
    rgb(int i, int i1, int i2) Color  
    rgb(int i, int i1, int i2, double d) Color
```

rojo verde azul opacidad

Componente	Rango de valores
Rojo	0-255
Verde	0-255
Azul	0-255
Opacidad	0,0-1,0

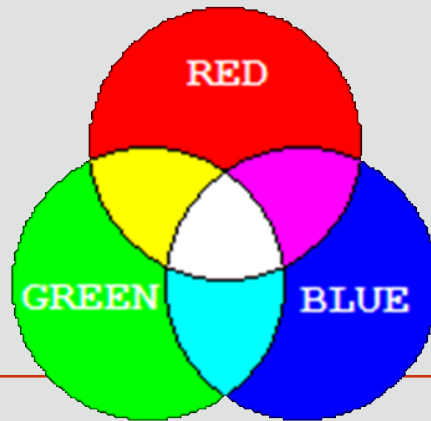
Ejemplo de color

- En este ejemplo, el color resultante contiene...

```
Color color = new Color.rgb(255, 255, 20);
```

- Todo el rojo que sea posible
 - Todo el verde que sea posible
 - Solo un poco de azul
-
- El color resultante se acerca mucho al amarillo
 - Pero, ¿cómo lo sabemos?
 - Para la mayoría, encontrar el color perfecto es cuestión de "prueba y error", pero hay principios

Reglas de mezcla aditiva de colores



Ejemplos:

Código	Color
<code>Color.rgb(255, 0, 0);</code>	rojo
<code>Color.rgb(0, 255, 0);</code>	verde
<code>Color.rgb(0, 0, 255);</code>	azul
<code>Color.rgb(255, 255, 0);</code>	amarillo
<code>Color.rgb(0, 0, 0);</code>	negro
<code>Color.rgb(255, 255, 255);</code>	blanco

Rojo puro
Verde puro
Azul puro
Sin azul
Sin color
Todos los colores

Ejercicio 1

- Cree un nuevo proyecto JavaFX con `JavaFXMainEx1.java`
 - `JavaFXMainEX1.java` es una copia de `JavaFXMain.java`
 - Cambie el nodo raíz a un tipo `Group`
 - Elimine el botón y cualquier otro código innecesario relacionado con el botón
- Practique con colores personalizados
 - Cree algunos colores personalizados
 - Contemple los colores personalizados a través del fondo de la escena, proporcionando un argumento `Color` cuando se instancie `Scene`

Esto es un rectángulo

- Así es cómo se instancia un rectángulo JavaFX:



```
Rectangle rect = new Rectangle(20, 20, 100, 200);
```

posición x posición y ancho altura

- Deberá importar, en primer lugar, lo siguiente:

```
import javafx.scene.shape.Rectangle;
```

- Ignore las otras sugerencias de importación de Rectangle de su IDE

Métodos importantes para rectángulos

- Podemos obtener un concepto de las propiedades del rectángulo a partir del constructor y los métodos siguientes:
 - setX(double d)
 - setY(double d)
 - setWidth(double d)
 - setHeight(double d)
 - setFill(Paint paint)
 - setStroke(Paint paint)
 - setStrokeWidth(double d)
- (Hay muchos más métodos para rectángulos además de estos siete)
- Pero, ¿qué hacen exactamente estos métodos?

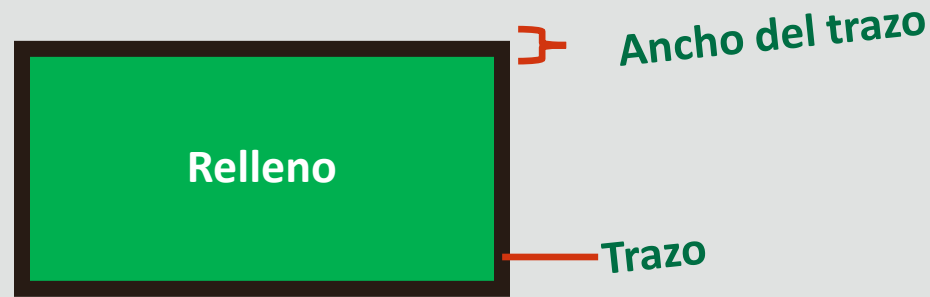
Estos elementos pueden aceptar un color como argumento

Ejercicio 2

- Continúe editando el proyecto JavaFX que ha creado en el ejercicio anterior
- Cree un rectángulo y agréguelo al nodo raíz
- Llame a cada método que se detalla en la diapositiva anterior
- ¿Puede averiguar lo que hace cada método?

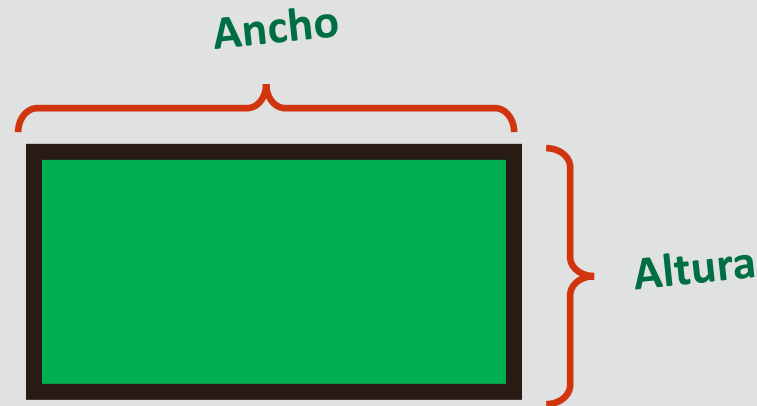
Descripciones de los métodos, parte 1

- `setFill(Paint paint)`
 - Define el color del rectángulo
- `setStroke(Paint paint)`
 - Define el color del contorno del rectángulo
- `setStrokeWidth(double d)`
 - Define el ancho del contorno del rectángulo



Descripciones de los métodos, parte 2

- `setX(double d)`
- `setY(double d)`
 - Define la posición x o y del rectángulo
- `setWidth(double d)`
- `setHeight(double d)`
 - Define el ancho o la altura del rectángulo



Cambio de la posición de un nodo

- Hemos visto varias formas de cambiar la posición de un nodo, pero ¿cuál es la mejor forma de hacerlo?
- `setX(double d)`
- `setY(double d)`
 - Estas son las preferibles en la mayoría de los casos
- `setLayoutX(double d)`
- `setLayoutY(double d)`
 - Utilice estas si el nodo está bloqueado en un panel de diseño, como un objeto `FlowPane`
 - O si `setX()` no está disponible, que es lo que sucede con los elementos de la interfaz de usuario, como los botones

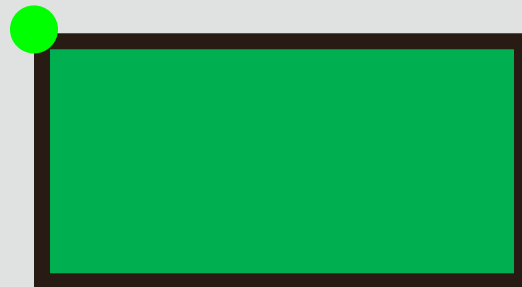
Sin duda, `setX()` no funcionará en este caso



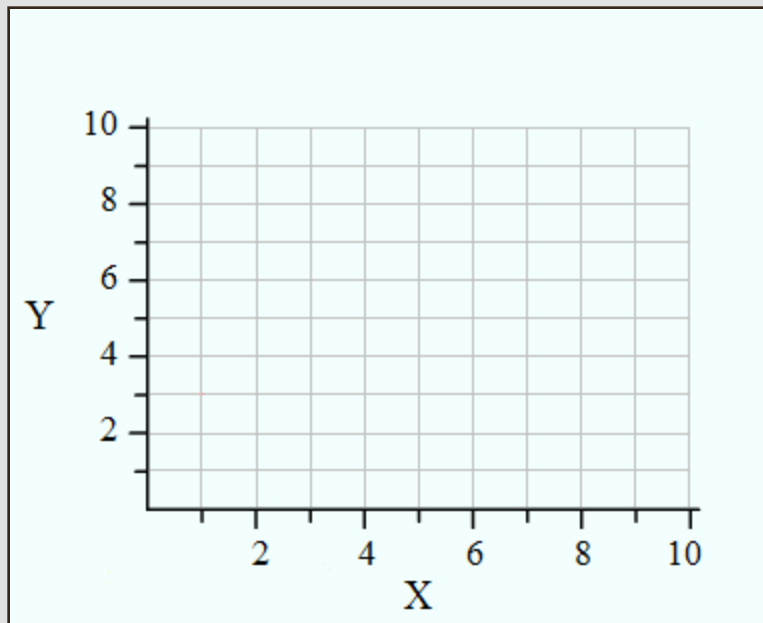
Posicionamiento de un nodo

- La mayoría de los nodos están colocados con respecto a su esquina superior izquierda
 - Y no con respecto a su centro geográfico
- Si llama a `setX(100)` en un nodo...
 - La posición x de la esquina superior izquierda del nodo se establece en 100

(100, 0)

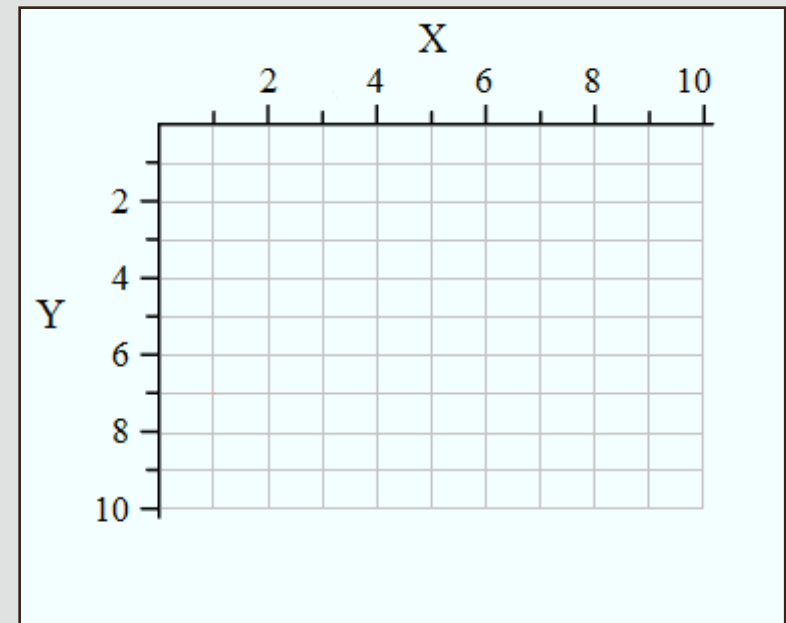


Sistemas de coordenadas



Sistema de coordenadas matemáticas

- El origen se sitúa en la esquina inferior izquierda

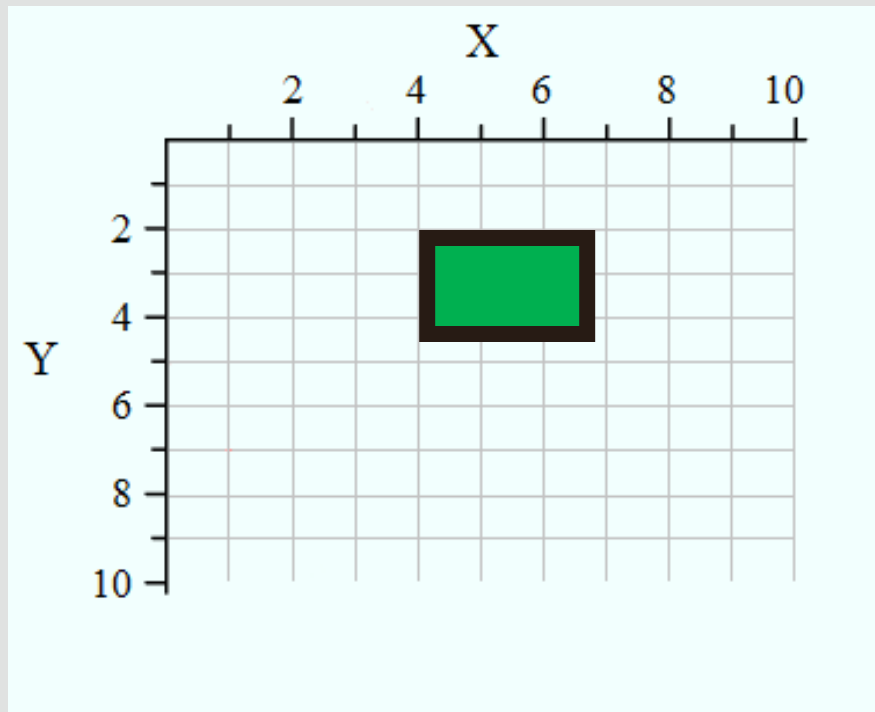


Sistema de coordenadas de JavaFX

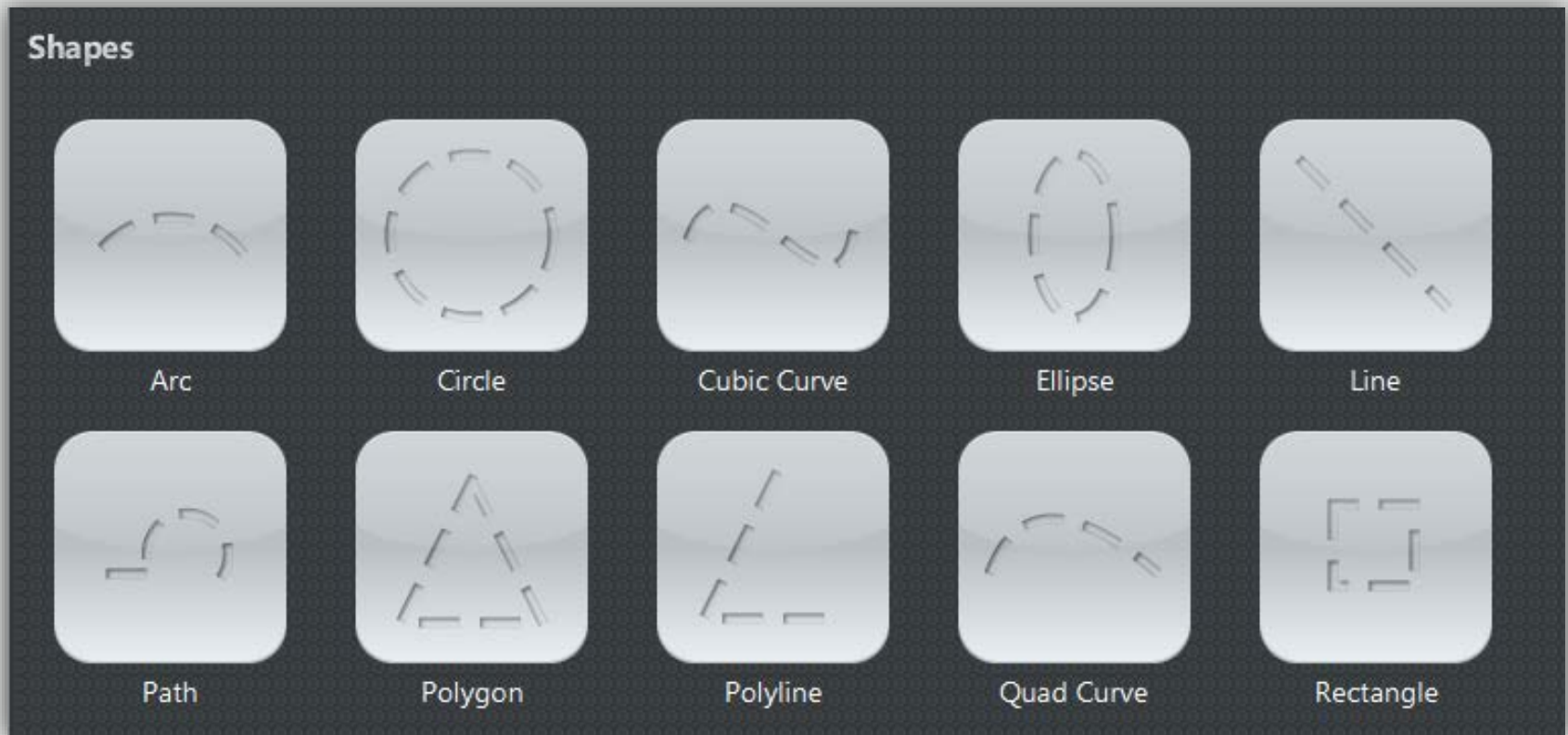
- El origen se sitúa en la esquina superior izquierda
- El eje y va hacia atrás

Ejemplo de posicionamiento

- Este rectángulo se colocará en (4,2) llamando a:
 - `setX(4);`
 - `setY(2);`

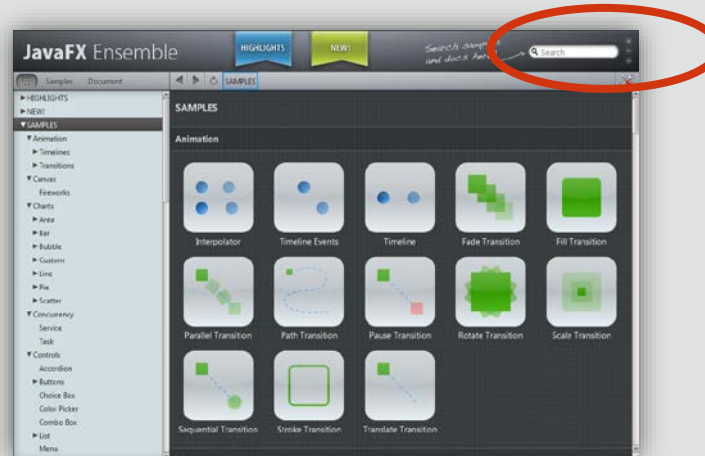


En JavaFX hay disponibles muchas formas



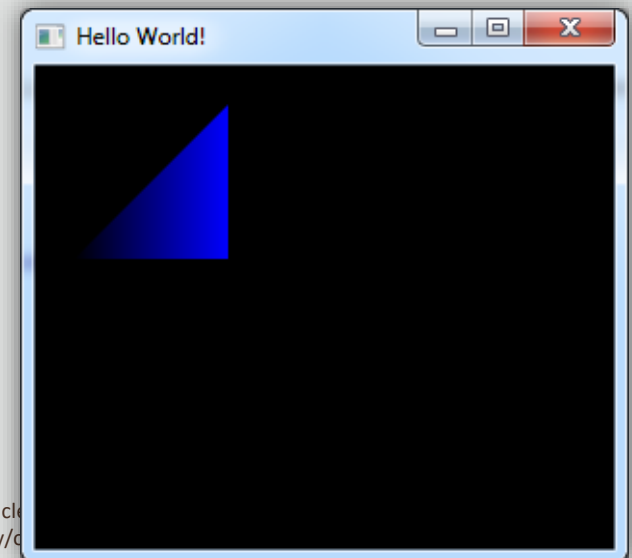
Documentación de la API de JavaFX

- Contiene información de clase y ejemplos de código de las funciones de JavaFX
- Vaya a <https://openjfx.io/javadoc/17/index.html>
- El módulo Graphics es un punto de partida útil
- Hay una función de búsqueda que le permite localizar clases específicas, o bien puede buscar ideas en los paquetes



Ejercicio 3

- Consulte la documentación de la API de JavaFX
- ¿Sabría cómo crear un triángulo rectángulo con un color degradado utilizando el proyecto JavaFX que ha creado en el ejercicio anterior?
- Indicación: Utilice el cuadro de búsqueda para buscar primero el gradiente y, a continuación, el polígono





Consulta de la documentación de la API:

Ejemplo de gradiente lineal

- El ejemplo de degradado lineal nos muestra...
 - Cómo crear un degradado:



```
//create simple linear gradient
LinearGradient gradient1 = new LinearGradient(0, 0, 1, 0, true,
CycleMethod.NO_CYCLE, new Stop[] {
    new Stop(0, Color.DODGERBLUE),
    new Stop(1, Color.BLACK)
});
```

- Cómo colorear una forma con un degradado:

```
//First rectangle
Rectangle rect1 = new Rectangle(0,0,80,80);

//set rectangle fill
rect1.setFill(gradient1);
```

- Recuerde hacer las importaciones adecuadas

Consulta de la documentación de la API:

Ejemplo de polígono



- El ejemplo del polígono nos muestra...
 - Cómo crear un polígono a partir de una arreglo de puntos:

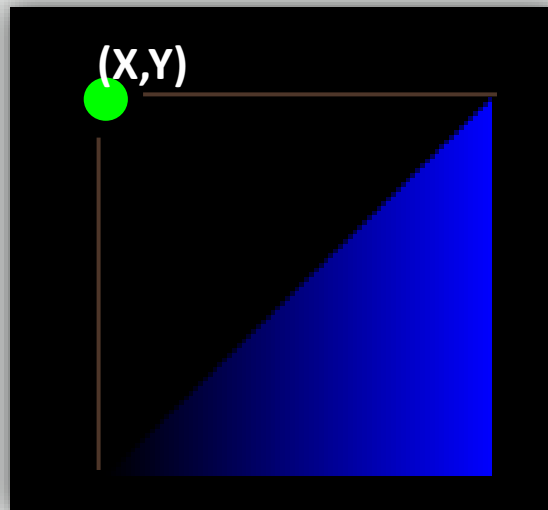
```
//Simple triangle
Polygon polygon1 = new Polygon(new double[] {
    80.0, 10.0,
    80.0, 80.0,
    10.0, 80.0
});
```

- Combine esto junto con el ejemplo del degradado y tendrá la solución
 - Pero lo que es mejor, descubrirá que la documentación de la API es un recurso valioso
 - Esto podría resultar muy útil al resolver el problema planteado

El polígono



- El polígono tiene métodos similares a un rectángulo
 - Los nodos comparten los mismos métodos
- Si experimenta con `setLayoutX()`...
 - Tenga en cuenta que el polígono se coloca con respecto al lugar en el que estaría su esquina superior izquierda



Secretos sobre Java Puzzle Ball

- Trazamos líneas y los polígonos para la detección de colisiones
 - Pero estas líneas están ocultas en la versión más reciente
- Además, también trazamos dos octágonos alrededor de cada deflector
 - Un octágono interno se encarga de la detección de colisiones
 - Un octágono externo detecta si la bola está lo suficientemente lejos para que el deflector rote
- Tenemos que trabajar más para colocar y rotar los nodos de la forma que queremos



Resumen

- En esta lección, debe haber aprendido lo siguiente:
 - Crear y usar campos personalizados
 - Crear formas y explicar sus propiedades y comportamientos
 - Consulte la documentación de la API de JavaFX





ORACLE

Academy

