Parte A

Tareas a realizar en el laboratorio

1 Abre el proyecto ***shapes*** con BlueJ

2 Pulsa el botón derecho sobre la caja **`Circle`**, selecciona **`new Circle()`** en el menú desplegable y pulsa ***Ok***. Acabas de crear (***instanciar***) tu primer ***objeto.*** La referencia al objeto se guarda en la variable *`circle1`* y se coloca en el banco de objetos (*object bench*) de BlueJ para su uso posterior.

- Ahora, pulsa el botón derecho del ratón sobre el objeto circulo creado (*`circle1`*, en el banco de objetos) y selecciona ***Inspect*** para invocar al ***Inspector de Objetos (Object Inspector)***. Los campos mostrados son los ***atributos.*** Los atributos describen las características de un objeto. El conjunto de valores que tienen asignados los atributos de un objeto determinan su ***estado***.
- Cuando pulsas el botón derecho del ratón sobre un objeto, además del ***Inspect*** se muestran todos sus ***métodos (comportamientos)***. Los métodos permiten ejecutar acciones (p.ej. mover) y/o modificar el estado del objeto:
  - ¿Qué valor tiene `isVisible`? false
  - Pulsa el botón derecho sobre el objeto y selecciona `makeVisible()`. ¿Qué sucede? Que se ha cambiado el valor del atributo isVisible a true. Ahora podemos ver el objeto en el Canvas.
  - ¿Cuál es la posición (x, y) del círculo? Cambia el valor del atributo isVisible a true. X=20 y=60.
  - Pulsa el botón derecho sobre el objeto y selecciona `moveDown()`.
  - ¿Cuál es la nueva posición (x, y) del círculo? X=20, y=80 .
  - ¿Qué crees que pasará si ***invocas*** el método (es decir, ***pase de mensaje***) `moveDown()` de nuevo? Indica el resultado sin ejecutarlo. Volverá a aumentar el atributo yPosition en 20 unidades (o sea a 100).
  - ¿Has acertado? Sí.
  - Invoca el método `moveRight()` del círculo. ¿Cuál es la nueva posición (x, y) del círculo? X= 40, y=100.

3 Crea un nuevo objeto de la clase **`Circle`** (sigue las instrucciones del ejercicio anterior).

- Pulsa el botón derecho del ratón sobre el objeto y selecciona *Inspect* para invocar al *Inspector de Objetos*.
- ¿Por qué no puedes ver el círculo en el dibujo? Porque el atributo isVisible tiene como valor false.
- Haz que sea visible. He cambiado el valor del atributo isVisible a true.
- Cambia el color del círculo (p.ej. "*red*"). He llamado al método changeColor(String newColor) y le he pasado "red" como argumento.
- ¿Qué ocurre si invocas el método `changeColor`(), y escribes un color *sin* comillas en el cuadro de texto? Que salta un error porque se interpreta el argumento pasado como una variable indefinida en vez de como un String (o una variable tipada como String).

- ¿Qué ocurre si especificas un color desconocido? El color pasa a ser “black”. Esto ocurre porque el método changeColor(), invoca, a su vez, al método draw(). En el método draw() de la clase Circle, a parte de instanciar la clase Canvas, se llama a su método draw(). Este método draw(), entre otras sentencias, invoca al método redraw() (de la clase Canvas también). El método redraw(), invocará al método draw() de la subclase de Canvas, ShapeDescription, y esta, ejecutará el último método setForeground() heredado por la superclase Canvas. En este método se espefica que si el color recibido como parámetro es desconocido (bloque else), se cambiará el color a negro (setColor(color.black)).

- Cambia el tamaño del diámetro a 10. He invocado al método changeSize() y le he pasado como argumento 10.
- Pásale el mensaje de `moveVertical` el valor de 10 pixeles. ¿Qué ha ocurrido con su *estado*, exactamente con el atributo `yPosition`? Que ha aumentado 10 unidades (píxeles), luego yPosition = 70.
- ¿Cómo puedes hacer que el círculo se desplace hacia arriba 25 píxeles? Invocando al método moveVertical() y pasándole como argumento -25.
- Mueve el círculo de manera que quede en la esquina superior izquierda del lienzo (*canvas*). Invoco el método moveVertical() y le paso como argumento el valor del atributo yPosition en negativo. Luego invoco al método moveHorizontal() y le paso como argumento el valor del atributo xPosition en negativo.

4 Crea un nuevo objeto de la clase **`Square`**. Invoca al *Inspector de objetos,* haz visible el objeto y cambia su color a azul

- Muévelo a la esquina superior derecha del lienzo, ¿cómo? Invoco al método moveVertical() y le paso como argumento el valor del atributo yPosition en negativo. Luego invoco al método moveHorizontal() y le paso como argumento la anchura del Canvas menos el valor actual del atributo xPosition menos el size del cuadrado ((300-60) – 30 = 210).
- Pon el cuadrado en el centro del lienzo, ¿cómo? Desde la posición actual (esquina superior derecha), invoco al método moveVertical y le paso como argumento la altura del canvas dividido entre 2 (300/2=150) menos el size del cuadrado dividido entre 2(150-15 = 135). Luego invoco al método moveHorizontal() y le paso como argumento la anchura del canvas en negativo entre dos (-300/2=-150) más el size del cuadrado dividido entre 2(30/2=15)(-150+15 = -135).
- Mueve lentamente el cuadrado 50 píxeles a la derecha con el método `slowMoveHorizontal()`. Invoco el método slowMoveHorizontal y le paso como argumento 50.

5 Crea un objeto de la clase **`Triangle`**. Abre el *Inspector de Objetos* para cada figura.

- Escribe los nombres de los atributos que los 3 tipos de objetos tienen en común:

  xPosition, yPosition, color, isVisible.

- Escribe los nombres de los atributos propios de cada figura

  Circle: diameter.

  Square: size.

  Triangle: height, width.

- ¿Qué métodos distinguen el comportamiento del triángulo de las otras figuras?

  ChangeSize(int newHeight, int newWidth).

6 Utilizando el proyecto ***shapes*** y el menú contextual de *BlueJ* para el **paso de mensajes** a los objetos:

6.a Abre la ventana de terminal de BlueJ (*Ver→Mostrar Terminal*) y activa la opción de registro (Opciones→ Registro de llamadas a método). Si es necesario, limpia lo que tengamos de ejecuciones previas (*Opciones→ Limpiar*)

6.b Crea un **triángulo** y modifícalo de manera que la base cubra el borde inferior completo del lienzo y el vértice superior esté en el borde superior del lienzo. Anota tanto la **instanciación de los objetos** (es decir, creación de las figuras) como el **pase de mensajes** a los objetos (es decir, la invocación de sus métodos)

Objetos Instanciados
Triangle triangle1 = new Triangle();
Pase de Mensajes
triangle1.makeVisible(); *triangle1.changeSize(300, 300);* *triangle1.moveHorizontal(100);* *triangle1.moveVertical(-15);*

6.c Abre la clase **`Demo`** con el editor de código. Copia en el método *exercise6b()* las instrucciones ejecutadas en el ejercicio 6b. Compílalo y pruébalo.

Tareas complementarias:

7 Utilizando el proyecto ***shapes*** y el menú contextual de *BlueJ* para el **paso de mensajes** a los objetos:

7.a Limpia lo que tengas de ejecuciones previas en la ventana de terminal de BlueJ (*Opciones→ Limpiar*)

7.b Pon un cuadrado (**`Square`**) de tamaño 10x10 en el origen de coordenadas y desplázalo a lo largo del borde superior del lienzo. Cuando llegue a la esquina superior derecha, desplázalo a la esquina inferior derecha del lienzo. Anota tanto la **instanciación de los objetos** (es decir, creación de las figuras) como el **pase de mensajes** a los objetos (es decir, la invocación de sus métodos)

Objetos Instanciados
Square square1 = new Square();
Pase de Mensajes

square1.makeVisible();
 	
square1.changeSize(10);

```
square1.moveHorizontal(-60);

square1.moveVertical(-50);

square1.slowMoveHorizontal(290);

square1.slowMoveVertical(290);
```

7.c Abre la clase **`Demo`** con el editor de código. Copia en el método *exercise7b()* las instrucciones ejecutadas en el ejercicio 7b. Pon en el método main una llamada al método *exercise7b()*. Compílalo y pruébalo.