Práctica 3

Objetivo: Practicar con la creación de objetos y la invocación de sus métodos, para hacer un dibujo animado

Software suministrado: Un conjunto de clases para dibujar figuras en una ventana:

- `Circulo`: representa un círculo
- `Rectangulo`: representa un rectángulo
- `Linea`: representa una línea
- `Imagen`: representa una imagen obtenida de un archivo en formato ***jpg*** o ***png***
- `Figura`: clase auxiliar que representa una figura abstracta
  - es una clase abstracta, lo que impide crear objetos de ella
  - no la usamos directamente

Información útil

Coordenadas: las coordenadas de las figuras se describen en ***píxeles*** (puntos de pantalla)

- el sistema de coordenadas se sitúa en la esquina superior izquierda de la ventana
  - las coordenadas `x` avanzan hacia la *derecha* en el dibujo
  - las coordenadas `y` avanzan hacia ***abajo*** en el dibujo
- la ventana mide 900x900 píxeles

Documentación:

- Para comprender el uso de las clases suministradas puede consultarse la documentación desde un navegador de Internet
  - generar la documentación desde ***Bluej*** con `Tools->Project Documentation`
  - esta documentación se guarda en el directorio `doc` dentro del proyecto

Repaso: Crear y usar objetos

Crear objetos de una clase:

- Sintaxis

```
Clase nombreObjeto=new Clase(parámetros);
// los parámetros son los del constructor
```

- Ejemplo

```
Rectangulo caja=new Rectangulo(10,10,30,30);
```

Invocar un método `void` del objeto

- Sintaxis

```
nombreObjeto.nombreMétodo(parámetros);
// los parámetros son los que requiera el método
```

- Ejemplo

```
caja.mueve(20,30);
// mueve la caja 20 píxeles a la derecha
// y 30 hacia abajo
```

Descripción

Se pide hacer un programa principal (una nueva clase con método `main`) que haga una animación del vuelo de un dron que intenta posarse en la azotea de un edificio, pero se estrella contra él

Ejecutar los siguientes pasos:

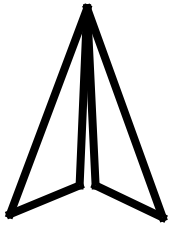
- Crear un objeto de la clase `Imagen` hacia la derecha del dibujo
  - esto será el edificio
  - puede usarse la foto `edificio.jpg`
- Crear un objeto de la clase `Imagen` situado en el suelo a la izquierda del edificio
  - usar la imagen `dron.jpg`
  - esto será el dron
- Esperar un segundo con el método estático `espera` definido en la clase `Figura`
  - ejemplo: `Figura.espera(1000); // espera 1000 ms`

Descripción (cont.)

- Mover el dron hacia arriba 70 píxeles y esperar 200 milisegundos
- Reproducir las acciones del paso anterior tantas veces como sea necesario para que el dron quede casi en la parte de arriba del edificio
  - ahora que aún no hemos visto los bucles, esto se puede hacer copiando y pegando las instrucciones tantas veces como sea necesario
- Mover el dron hacia la derecha 70 píxeles y esperar 200 milisegundos
- Reproducir las acciones del paso anterior tantas veces como sea necesario para que el dron choque con el edificio
- Usando el fichero `boom.png` crear un objeto de la clase `Imagen` colocado sobre la última posición del dron
  - esto representará el impacto contra el edificio

Parte avanzada

Queremos crear un objeto que hace una trayectoria como la del dron, pero compuesto por varias líneas con forma de avión de papel



Para ello crearemos una nueva clase, llamada `AvionPapel` con:

- atributos:
  - las líneas que forman el avión de papel (objetos de la clase `Linea`)
- métodos:
  - Constructor que crea las figuras del avión de papel; se le pasan como parámetros las coordenadas de la punta
  - `mueve`(): método que desplaza el avión en la cantidad de píxeles que se indican como parámetros; lo hace a base de desplazar cada figura individualmente

Parte avanzada (cont.)

Crear una nueva clase principal, copia de la anterior que haga una animación similar a la de la parte obligatoria, pero con el nuevo avión de papel en lugar del dron original

- solo debería cambiar la creación del objeto proyectil
- el resto debe ser igual

Entregar

Informe:

- Parte obligatoria
  - El código de la clase creada
  - Una captura de pantalla con el dibujo tal como queda hacia el principio de la ejecución del programa
  - Una captura de pantalla con el dibujo tal como queda al final de la ejecución
- Parte avanzada
  - El código de las clases creadas en la parte opcional
  - Una captura de pantalla con el dibujo que queda hacia el principio de la ejecución

El proyecto `bluej` en un archivo comprimido

(Total, dos archivos: informe y proyecto)