Práctica 2

Objetivos:

- Familiarizarse con los procesos de compilación y ejecución de programas
- Practicar con la creación de objetos y la invocación de sus métodos

Parte 1: compilar y ejecutar un programa desde la línea de comando

Parte 2: compilar y usar una clase desde `bluej`

Parte 3: crear un programa `main` que crea un objeto y usa sus métodos

Parte avanzada: Usar una clase externa para hacer una gráfica

Parte 1: compilar y ejecutar un programa desde la línea de comando

Compilar y ejecutar la clase `Hola` de los apuntes desde el intérprete de órdenes

Hacer una captura de pantalla que muestre la compilación y la ejecución, y meterla en el informe

Parte 2: compilar y usar una clase desde `bluej`

En la parte 2 usaremos una clase ya hecha, `Piedra`, que permite simular el movimiento de una piedra

Por simplicidad se estudia el movimiento sin considerar el rozamiento con el aire

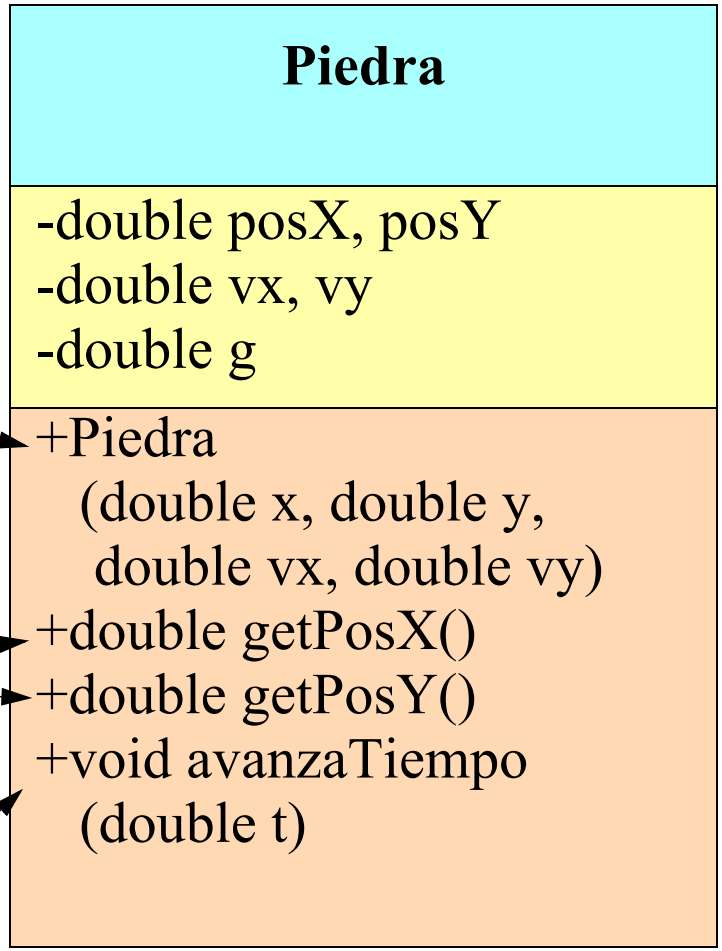
Ver el documento `ejercicio-simulacion-piedra.pdf` para una explicación más detallada

Diseño de la clase Piedra

El método constructor, que pone los valores iniciales a los atributos

Métodos observadores, que retornan el valor de un atributo

Método que modifica los atributos según las ecuaciones



Parte 2: compilar y usar una clase desde `bluej`

Crear en `bluej` un nuevo proyecto:

- Mover a él y compilar la clase `Piedra` que se suministra
- Con el ratón crear un par de objetos de la clase `Piedra` con posición inicial `(100,0)`m (ambos iguales) y velocidades iniciales `(15,20)`m/s para un objeto, y `(-10,10)`m/s para el otro

Hacer dos capturas de pantalla que muestren, cada una, el resultado de usar el método `getPosY()` sobre ambos objetos

- una captura después de crear los objetos
- una segunda captura después de haber invocado el método `avanzaTiempo()` para ambos objetos con `t=0.5` s

Parte 3: crear un programa main que crea un objeto y usa sus métodos

Crear en el proyecto una nueva clase con un método `main` que haga algo similar a lo que hemos hecho con el ratón en `bluej`:

- crea un objeto de la clase `Piedra`
- muestra en pantalla los resultados de invocar `getPosX()` y `getPosY()`
- invoca el método `avanzaTiempo()` para `t=0.5` s
- vuelve a mostrar en pantalla los resultados de invocar `getPosX()` y `getPosY()`

Hacer una captura de pantalla que muestre los resultados

Crear y usar objetos

Crear objetos de una clase:

- Sintaxis

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

- Ejemplo

```
Piedra p=new Piedra(0.0,100.0,30.0,20.0);
// los parámetros indican la pos. y vel. iniciales
```

Invocar un método del objeto:

- Sintaxis

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

- Ejemplo

```
p.avanzaTiempo(0.5);
// avanza el tiempo 0.5 segundos
```

Mostrar resultados en pantalla

Java tiene una operación de *concatenación* que permite unir a un string otro string o un dato primitivo cualquiera

- el operador de concatenación es `'+'`
- el resultado es otro string
- ejemplo, si `x` es una variable numérica que vale 3.0:

```
"valor x="+x // produce "valor x=3.0"
```

Esto nos permite mostrar resultados en pantalla con la instrucción `System.out.println`. Ejemplo:

```
System.out.println("desplazamiento= "+x+" cm");
```

muestra en pantalla:

```
desplazamiento= 3.0 cm
```

Parte avanzada

Crear una gráfica de la evolución de la piedra. Para ello:

Crear una nueva clase con un `main` que haga:

- crea un objeto de la clase `Grafica`
- crea un objeto de la clase `Piedra`
- inserta en la gráfica las posiciones `x` e `y` de la piedra
- repite 10 veces (copiando y pegando instrucciones, pues aún no hemos aprendido los bucles):
  - inserta en la gráfica las posiciones `x` e `y` de la piedra
  - avanza el tiempo 0.5 segundos
- pinta la gráfica

Hacer una captura de la gráfica para el informe

Hacer una gráfica sencilla

Métodos de la clase `Grafica`, del paquete `fundamentos`

new Grafica (String titulo, String tituloX, String tituloY)	Constructor que pone los títulos de la ventana y de los ejes X e Y
void inserta (double x,double y)	Inserta el punto (x,y) en la gráfica actual
void pinta()	Pinta la gráficas

Ejemplo:

```
Grafica g = new Grafica ("Piedra","x","y");
g.inserta(x1,y1);
g.inserta(x2,y2);
...
g.pinta();
```

Nota: acordarse de importar el paquete `fundamentos`:

```
import fundamentos.*; // al principio de la clase
```

Paquete fundamentos

Para usar la clase `Grafica` es necesario instalar el paquete `fundamentos`

Se puede *descargar* de la página moodle

- sección de *Recursos*

Luego hay que instalarlo en *bluej*

- ver apuntes del *capítulo 9*

Entregar

Parte obligatoria

- La captura de pantalla de la parte 1
- Las dos capturas de pantalla de la parte 2
- El código de la clase principal de la parte 3
- La captura de pantalla de la parte 3

Parte avanzada, si se ha hecho

- El código de las clase principal
- Una captura de pantalla con la gráfica