OBJETIVOS

- Reforzar el uso de variables, expresiones y condicionales.
- Introducir el concepto de método funcional.
- Introducir el uso de tipos compuestos.
- Introducir el uso de métodos de un tipo.

INTRODUCCIÓN

El objetivo de este segundo boletín de ejercicios es, por una parte, reforzar el uso de variables, expresiones y condicionales, y, por otra, introducir de forma intuitiva los conceptos de métodos funcionales, tipos compuestos, y el esquema de diseño de tipos que iremos usando a lo largo de la asignatura.

EJERCICIOS PARA HACER EN EL AULA

Cree un proyecto Java llamado `B01.2-aula`. Configure el proyecto para trabajar con la librería `FP-B01.2.jar`, que contiene la implementación de los tipos `Punto` y `Circulo`. Añada los paquetes y clases que se indican a continuación para resolver los siguientes ejercicios:

Ejercicio 1 – Métodos funcionales

El boletín B01.1 nos sirvió para introducir algunos elementos de un programa Java. En el ejercicio 4 de la sección de ejercicios para hacer en casa, se pedía implementar un programa que mostrase el perímetro de un triángulo formado por los puntos *A (-2.0, 2.0), B(1.0, 6.0)* y *C(6.0, -6.0)*. Supongamos que ahora queremos calcular el perímetro de dos triángulos más, los formados por los puntos *D(-2.0, 1.0), E(2.0, -1.0)* y *F(0.0, 6.0);* y *G(5.0, 1.0), H(5.0, -3.0)* e *I(2.0, -1.0)*. Una solución sería copiar la misma secuencia de sentencias dos veces más, una vez para cada uno de los nuevos triángulos, lo que nos obliga a repetir mucho código. Para evitar repetir tanto código vamos a usar el siguiente diseño:

a) Cree un paquete llamado `fp.utiles.geometria` y añádale una clase llamada `UtilesGeometria`.
b) Añada un método a la clase `UtilesGeometria` que, dados tres puntos que forman un triángulo, devuelva su perímetro. La cabecera de este método será la siguiente:

```
public static Double calcularPerimetro (Punto p1, Punto p2, Punto p3)
```

c) Cree un paquete llamado `fp.utiles.geometria.test` y añádale una clase llamada `TestUtilesGeometria1`. Asegúrese de que esta clase tiene definido un método `main`, en el que se creen los puntos correspondientes a los tres triángulos, se invoque al método `calcularPerimetro`, y se muestre el perímetro de los tres triángulos por consola. La ejecución del método `main` debe dar un resultado similar al siguiente:

```
El perímetro del triángulo formado por A(-2.0,2.0), B(1.0,6.0) y C(6.0,-6.0)
es 29.31
El perímetro del triángulo formado por D(-2.0,1.0), E(2.0,-1.0) y F(0.0,6.0)
es 17.13
El perímetro del triángulo formado por G(5.0,1.0), H(5.0,-3.0) e I(2.0,-1.0)
es 11.21
```

Ejercicio 2 – Uso de tipos compuestos y expresiones

Haciendo uso del tipo `Circulo` que se le proporciona en el archivo `FP-B01.2.jar`, añada un método a la clase `UtilesGeometria` que devuelva cierto si dados dos círculos, estos tienen puntos de intersección. Tenga en cuenta que dos círculos tienen puntos de intersección si la distancia entre sus centros es menor que la suma de sus radios. La cabecera del método será:

```
public static Boolean tienenPuntosInterseccion(Circulo c1, Circulo c2)
```

En el paquete `fp.utiles.geometria.test` escriba una clase `TestUtilesGeometria2` con un método `main` que permita probar el método. Una ejecución del método `main`, en el que se invoca dos veces a `tienenPuntosInterseccion` con dos parejas de círculos distintas, sería[1]:

```
El círculo con Centro: (6.0,6.0), Radio: 6.0 y el círculo con Centro: (8.0,2.0),
Radio: 4.0 tienen puntos de intersección
El círculo con Centro: (8.0,2.0), Radio: 4.0 y el círculo con Centro: (0.0,0.0),
Radio: 2.0 NO tienen puntos de intersección
```

Para resolver el ejercicio, tenga en cuenta que el tipo `Circulo` dispone de las siguientes operaciones:

```
public interface Circulo {
      Punto getCentro();
      Double getRadio ();
      void setRadio(Double radio);
      Double getArea();
      Double getPerimetro();
}
```

Además, la clase `CirculoImpl` tiene un constructor con la siguiente cabecera:

```
public CirculoImpl (Punto centro, Double radio)
```

Ejercicio 3 – Sentencias condicionales

Añada un nuevo método a la clase `UtilesGeometria` que, dado un punto, devuelva su cuadrante. La cabecera del método será la siguiente:

```
public static Cuadrante obtenerCuadrante (Punto p)
```

Defina `Cuadrante` como un tipo enumerado del paquete `fp.tipos.geometria` que puede tomar los valores `PRIMER_CUADRANTE`, `SEGUNDO_CUADRANTE`, `TERCER_CUADRANTE`, `CUARTO_CUADRANTE`, `EJE`.

Para probar el método añada una clase `TestUtilesGeometria3` al paquete `fp.utiles.geometria.test`. La clase debe contener, al menos un método `main`. Procure hacer un diseño de esta clase en el que pueda reutilizar código. Una ejecución para probar el método sería:

```
El punto (0.0,0.0) pertenece al EJE
El punto (2.0,2.0) pertenece al PRIMER_CUADRANTE
El punto (-2.0,2.0) pertenece al SEGUNDO_CUADRANTE
El punto (-2.0,-2.0) pertenece al TERCER_CUADRANTE
El punto (2.0,-2.0) pertenece al CUARTO_CUADRANTE
El punto (2.0,0.0) pertenece al EJE
```

[1] Estudie la forma de reutilizar el código de la clase `TestUtilesGeometria2`

```
El punto (0.0,-2.0) pertenece al EJE
```

Escriba otro método en la clase `UtilesGeometria`, que dado un cuadrante, devuelva una cadena de texto, con una abreviatura del cuadrante.

```
public static String abreviaturaCuadrante (Cuadrante c)
```

¿Qué tendría que cambiar en `TestUtilesGeometria3` para que el resultado en consola de su prueba sea el siguiente?

```
El punto (0.0,0.0) pertenece al EJE
El punto (2.0,2.0) pertenece al 1ER CUADRANTE
El punto (-2.0,2.0) pertenece al 2° CUADRANTE
El punto (-2.0,-2.0) pertenece al 3ER CUADRANTE
El punto (2.0,-2.0) pertenece al 4° CUADRANTE
El punto (2.0,0.0) pertenece al EJE
```

Ejercicio 4 – Uso de métodos

Añada al paquete `fp.tipos.geometria.test` una clase `TestPunto` en la que se prueben todos los métodos del tipo `Punto`. Un ejemplo de ejecución del método `main` de la clase `TestPunto` es:

```
Test constructor por defecto
============================
Punto 1 (0.0,0.0)

Test constructor con parámetros
===============================
Punto 2 (1.0,-1.0)

Test observadores
=================
Coordenada x de P2 1.0
Coordenada y de P2 -1.0

Test modificadores
==================
Cambiar coordenada X de P1 a 3.0
Punto 1 (3.0, 0.0)
Cambiar coordenada Y de P2 a -3.0
Punto 2 (1.0,-3.0)

Test getDistancia
=================
La distancia de (3.0,0.0) a (1.0,-3.0) es 3.605551275463989
La distancia de (1.0,-3.0) a (1.0,-3.0) es 0.0
```

EJERCICIOS PARA HACER EN CASA

Cree un proyecto Java llamado `B01.2-casa`, añada los paquetes y clases que se indican en los siguientes ejercicios:

1. Usando la misma filosofía del Ejercicio 1 de la sección Ejercicios para el aula, reestructure el ejercicio 5 del B01.1 que permitía calcular el índice de masa corporal. Cree una nueva clase llamada `UtilesSalud` en el paquete `fp.utiles`, añádale un método con la cabecera `public`

`static Float calcularIMC (Float peso, Float altura)` que permita calcular el índice de masa corporal. Finalmente, cree una clase `TestUtilesSalud` con un método `main`, en el que se invoque a este método. El resultado de la ejecución del método `main` es el siguiente:

```
Introduzca su peso (en kilos):
80,0
Introduzca su estatura (en metros):
1,80
Su IMC es: 24.69136
```

2. Cree una clase llamada `Reales` en el paquete `fp.utiles` y añada un método que devuelva cierto si un valor x pertenece al intervalo [2.0,5.0) U (0.0,1.0] U [-5.0,-2.0]. La cabecera del método debe ser:

```
public static Boolean estaEnIntervalo (Double x)
```

   Añada al paquete `fp.utiles.test` una clase `TestReales` en la que se pruebe el método. Use los valores -5.0, -2.0, -1.0, 0.0, 1.0, 1.5, 2.0, 3.0, 5.0 y 10.0.

3. En la clase `UtilesSalud` añada un método que dado un peso (en kilogramos) y una estatura (en metros) devuelva el estado nutricional de la persona, según la Organización Mundial de la salud. La cabecera del método debe ser la siguiente:

```
public static EstadoNutricional obtenerEstadoNutricional (Float peso,
                                                           Float altura)
```

   Defina `EstadoNutricional` en el paquete `fp.tipos.personas` como un tipo enumerado que pueda tomar los valores: `INFRAPESO, NORMAL, SOBREPESO y OBESO.`

   La clasificación de la Organización mundial de la salud, basada en el índice de masa corporal (IMC), determina que:
   - Se tiene `INFRAPESO` si `IMC < 18,5`
   - El peso es `NORMAL` si `18,5 ≤ IMC < 25`
   - Se tiene `SOBREPESO` si `25 ≤ IMC < 30`
   - Se está `OBESO` si `IMC ≥ 30`

   Añada lo necesario al `main` de la clase `TestUtilesSalud` para mostrar también el estado nutricional, de la siguiente forma:

```
Introduzca su peso (en kilos):
80,0
Introduzca su estatura (en metros):
1,80
Su IMC es: 24.69136
Su estado nutricional es: NORMAL
```

   Pruebe también con los siguientes valores de peso y estatura:

Peso	**Estatura**	**IMC**	**Estado nutricional**
45,0	1,70	15.570933	INFRAPESO
100,0	1,80	30.8642	OBESO
90,0	1,80	27.77778	SOBREPESO

4. Añada un método nuevo a la clase `UtilesSalud` que dado un estado nutricional, devuelva una cadena con un mensaje de texto que sirva de consejo. El mensaje de texto a mostrar dependerá del estado nutricional. Los mensajes son los siguientes:

   - Si tiene `INFRAPESO` el mensaje será: `"Está demasiado delgado/a. Consulte a su médico"`.
   - Si el estado es `NORMAL` el mensaje será: `"Está estupendo/a. Manténgase así"`.
   - Si tiene `SOBREPESO` el mensaje será: `"Con un poco de esfuerzo puede conseguir ajustar su peso. Consulte a su médico"`.
   - Si el estado es `OBESO` el mensaje será: `"Bajar de peso, hacer ejercicio y alimentarse sanamente, son metas que puede conseguir. Consulte a su médico"`.

   La cabecera del método a implementar debe ser la siguiente:

   ```
   public static String consejoSalud(EstadoNutricional en)
   ```

   Añada a `TestUtilesSalud` los consejos ofrecidos con este nuevo método.

5. Añada una clase de utilidad llamada `UtilesVentanas` al paquete `fp.ventanas.utiles`, cuyo código es el siguiente:

```
package fp.ventanas.utiles;
import javax.swing.JOptionPane;

public class UtilesVentanas {

	public static String leerDato(String titulo, String mensaje) {
		return JOptionPane.showInputDialog(null,
				mensaje, titulo,
				JOptionPane.PLAIN_MESSAGE);
	}

	public static void mostrarMensaje(String titulo, String mensaje) {
		JOptionPane.showMessageDialog(null, mensaje, titulo,
				JOptionPane.INFORMATION_MESSAGE);
	}
}
```

   Un ejemplo de uso de estas funciones es el siguiente:

```
package fp.ventanas.utiles.test;

import fp.ventanas.utiles.UtilesVentanas;

public class EjemploUtilesVentanas {

   private static final String TITULO = "UtilesVentanas";
   public static void main(String[] args) {

	String realStr = UtilesVentanas.leerDato (TITULO,
					"Introduzca un número real: ");
	Double real = new Double(realStr);
```

```
            String mensaje = "La raiz cuadrada del número introducido es : " +
                                Math.sqrt(real)+"\n";
            UtilesVentanas.mostrarMensaje(TITULO, mensaje);

    }
}
```

Si ejecuta `EjemploUtilesVentanas`, le debe aparecer la secuencia de ventanas de la Figura 1.

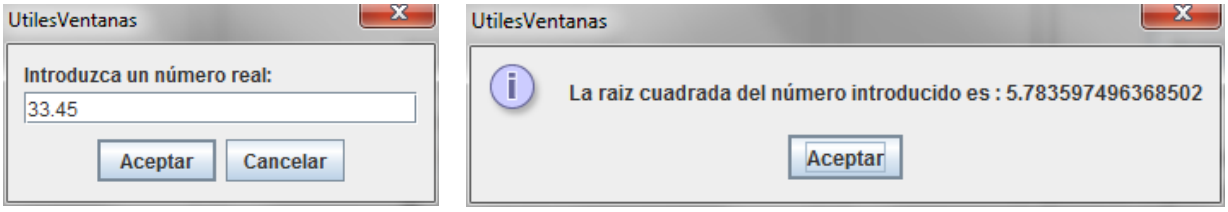

Figura 1. Secuencia de ventanas resultado de la ejecución de `EjemploUtilesVentanas`

6. Cree una clase llamada `TestUtilesSaludVentana` en la que realice las mismas pruebas que en el ejercicio 3, pero usando las ventanas, de forma que vayan apareciendo los diálogos de la Figura 2:

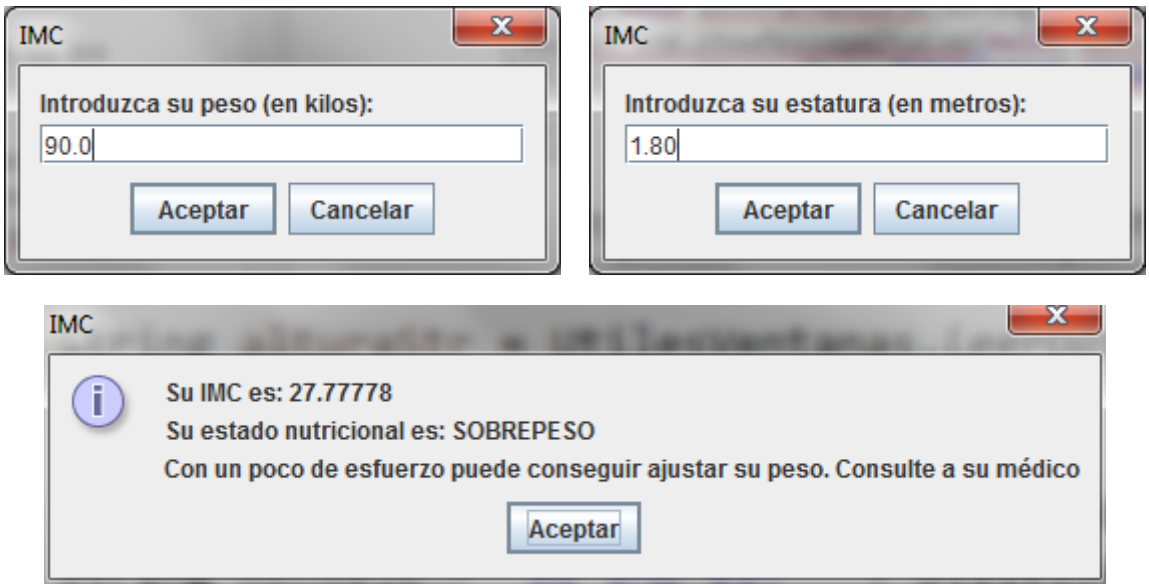

Figura 2. Secuencia de ventanas resultado de la ejecución de `TestUtilesSaludVentana`