



UNIVERSIDAD DEL BÍO-BÍO

# Paradigmas de la Programación

## Enum

# Enumeración

Los enums en Java son una especie de clase “especial” que sirve para representar un grupo de constantes de datos. Por ejemplo, los días de la semana, de los cuales serán 7 constantes(Lu/Ma/Mi/Ju/Vi/Sa/Do).

**Los enums se utilizan cuando se conoce todos los valores posibles en tiempo de compilación , como si fuera las opciones en un menú.**

# Ejemplo

Gracias a los enums, se podrá parametrizar posibles valores los cuales podrían representar una acción.

Ejemplo: La clase Proceso contiene un método el cual recibe 2 valores y dependiendo del tercer parámetro es la acción que se realizará (Sumar/Restar).

```
public class Proceso {  
  
    public static int accion(int a, int b, String modo) {  
        if (modo.equals("Sumar"))  
            return a + b;  
  
        if (modo.equals("Restar"))  
            return a - b;  
  
        return 0;  
    }  
}
```

# Ejemplo

Al realizar la ejecución en el Main de nuestro método, obtendremos los siguientes resultados.

```
public static void main(String[] args) {  
  
    System.out.println(Proceso.accion(1, 2, "Sumar"));  
    System.out.println(Proceso.accion(1, 2, "Restar"));  
    System.out.println(Proceso.accion(1, 2, "SuMAR"));  
  
}
```

```
run:  
3  
-1  
0  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pero... ¿Por qué la tercera llamada entregó el valor de 0, si se estaba pidiendo que se efectuara la suma?

**R: Está mal escrita la instrucción.**

# Ejemplo

Entonces ... ¿Cómo se podría eliminar el posible error de escribir mal la instrucción?

R: Creando un conjunto de constantes que representen dichas instrucciones (**Enum**).

Para poder crear un Enum se ocupará el mismo proceso de una Clase normal, con la diferencia que la palabra clave “class” será cambiada por “enum” y dentro de ella se incluirán las palabras que representarán las instrucciones.

```
public enum ProcesoEnum {  
    SUMAR,  
    RESTAR,  
    SIN_ACCION  
}
```

# Ejemplo

Ahora que el enum se encuentra creado, es posible cambiar el antiguo String del método acción en la Clase Proceso, por el nuevo enum de tipo ProcesoEnum.

**Observación 1:** Para comparar los valores de los enum se ocupará el operador de igualdad “==”.

**Observación 2:** Se recomienda usar la estructura switch para los enums.

```
public class Proceso {  
  
    public static int accion(int a, int b, ProcesoEnum modo) {  
        if (modo == ProcesoEnum.SUMAR)  
            return a + b;  
  
        if (modo == ProcesoEnum.RESTAR)  
            return a - b;  
  
        return 0;  
    }  
}
```








# Ejemplo

Ahora al llamar al método “**accion**” solamente serán posible 3 tipos de valores constantes, sin riesgo de cometer errores al ingresar la instrucción.

```
public static void main(String[] args) {

    System.out.println(Proceso.accion(1, 2, ProcesoEnum.SUMAR));
    System.out.println(Proceso.accion(1, 2, ProcesoEnum.RESTAR));
    System.out.println(Proceso.accion(1, 2, ProcesoEnum.));

}
```

	RESTAR	ProcesoEnum
	SIN_ACCION	ProcesoEnum
	SUMAR	ProcesoEnum
	valueOf(String name)	ProcesoEnum
	valueOf(Class<T> type, String string)	T
	values()	ProcesoEnum[]
	class	

run:

3

-1

3

BUILD SUCCESSFUL (total time: 0 seconds)

# Enum

Como ya se mencionó anteriormente, un enum es como una clase “especial”.

## Datos importantes:

1. Un enum puede tener constantes más complejas (Almacenar más de un dato por constante por medio de los atributos).
2. Tener un constructor privado.
3. Tener métodos y atributos.
4. Las constantes son siempre publicas, estáticas y finales (No se pueden cambiar, No se pueden sobre-escribir).
5. Un enum, no puede ser utilizado para crear objetos y tampoco podrá extender de otras clases (Pero puede implementar interfaces).



# Enum – Diseño más complejo

Para este ejemplo, el siguiente enum representará un listado de carreras, en donde se incluye el nombre y la duración.

**Observación:** Recordar que el constructor no puede ser público.

```
public enum TipoCarrera {  
  
    INFORMATICA("Informática", 5),  
    ARQUITECTURA("Arquitectura", 5),  
    INDUSTRIAL("Industrial", 5),  
    MECANICA("Mecánica", 5);  
  
    private String nombre;  
    private int duracion;  
  
    private TipoCarrera(String nombre, int duracion) {  
        this.nombre = nombre;  
        this.duracion = duracion;  
    }  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public int getDuracion() {  
        return duracion;  
    }  
  
    @Override  
    public String toString() {  
        return nombre + " - " + duracion;  
    }  
  
}
```

# Enum – Diseño más complejo

## Ejemplo de uso

```
public class Main {  
  
    public static void main(String[] args) {  
  
        System.out.println(TipoCarrera.ARQUITECTURA);  
        System.out.println(TipoCarrera.INFORMATICA.getNombre());  
        System.out.println(TipoCarrera.ARQUITECTURA == TipoCarrera.INFORMATICA);  
  
    }  
  
}
```

run:

Arquitectura - 5

Informática

false

BUILD SUCCESSFUL (total time: 0 seconds)

# Métodos por defecto

`public T[] values()` – Entrega un array de valores constantes asociados a un Enum.

`public T valueOf(String type)` – Busca y retorna una constante por medio de su nombre en String.

```
public static void main(String[] args) {  
  
    for (TipoCarrera item : TipoCarrera.values())  
        System.out.println(item);  
  
    System.out.println("-----");  
  
    System.out.println(TipoCarrera.valueOf("INFORMATICA"));  
  
}
```

```
run:  
Informática - 5  
Arquitectura - 5  
Industrial - 5  
Mecánica - 5  
-----  
Informática - 5  
BUILD SUCCESSFUL (total time: 0 seconds)
```