Enums en Java

- 1. ¿Qué son los Enums?
  - Un tipo especial de clase que define un conjunto finito de nombres simbólicos y sus valores.
  - Estos nombres simbólicos se denominan generalmente constantes de enumeración o constantes con nombre.
  - Representa un conjunto de elementos cuyos tipos se conocen en tiempo de compilación.
  - Ejemplos comunes incluyen:
    - Direcciones de la brújula
    - Meses del año
    - Planetas del sistema solar
    - Cartas en una baraja
    - Estados de una máquina (BUSY, IDLE, BLOCKED)
- 2. Ventajas de los Enums
  - Seguridad de tipos:
    - Proporcionan comprobación de tipos segura.
    - Evitan pasar valores inválidos sin descubrirlo hasta el tiempo de ejecución.
    - Los valores inválidos generan un error de compilación.
    - Mejoran la seguridad y robustez del código.
  - Orientación a Objetos:
    - Permiten tener atributos y métodos junto con valores estáticos.
    - Acercan el código a la programación orientada a objetos.
    - Permite definir métodos abstract, los valores del Enum deben proveer la implementación
    - Se puede definir el método main()
    - Implementan la interface Comparable, respeta el orden como se hayan definido los valores.
  - Facilidad de uso:
    - Facilitan la escritura de sentencias switch exhaustivas.
    - Se pueden iterar fácilmente utilizando el método values().
    - Se pueden obtener por su nombre utilizando el método valueOf().
  - Eficiencia:
    - Implementan la interfaz java.io.Serializable, lo que permite una serialización eficiente.
- 3. Uso de Enums
  - Acceso a las constantes:
    - Se acceden utilizando el nombre del enum seguido del nombre de la constante.
    - Ejemplo: Season.SUMMER
  - Métodos de utilidad:
    - values(): devuelve un array con todas las constantes del enum.
    - valueOf(String name): devuelve la constante del enum con el nombre especificado.
    - name(): devuelve el nombre de la constante del enum.
    - ordinal(): devuelve el índice (posición) de la constante del enum.
  - En sentencias switch:
    - Se pueden usar en sentencias switch.
    - Los valores de los casos son los nombres de las constantes del enum.
    - Ejemplo:

```
switch(summer) { case WINTER:
System.out.print("Get out the
sled!"); break; case SUMMER:
System.out.print("Time for the
pool!"); break; default:
System.out.print("Is it summer
yet?"); }
```
- 4. Declaración de Enums
  - Sintaxis básica:
    - Se utiliza la palabra clave 'enum'.
    - Se especifica un nombre para el enum.
    - Se enumeran los valores del enum entre llaves, separados por comas.
    - Ejemplo:

```
public enum Season { WINTER,
SPRING, SUMMER, FALL; }
```
  - Enums complejos:
    - Pueden tener constructores, campos y métodos.
    - Los constructores son siempre privados.
    - Los valores del enum se pasan como argumentos al constructor.
    - Ejemplo:

```
public enum Season
{ WINTER("Low"),
SPRING("Medium"),
SUMMER("High"),
FALL("Medium"); private final
String expectedVisitors; private
Season(String expectedVisitors)
{ this.expectedVisitors =
expectedVisitors; } public void
printExpectedVisitors()
{ System.out.println(expectedVi
sitors); } }
```
- Herencia
  - Enums implicitly extend the java.lang.Enum class. They cannot inherit from any other class
  - Enums, however, can implement interfaces.
  - An enum is implicitly final, which means you cannot extend it. For the same reason, it cannot be sealed.