TEMA 5

CARACTERÍSTICAS AVANZADAS DE JAVA PARA PROGRAMACIÓN ORIENTADA A OBJETOS

TEMA 5. CARACTERÍSTICAS AVANZADAS

Lambdas en Java

- Java permite pasar comportamientos como parámetros.
  - Una función puede requerir en sus parámetros una clase que cumpla una interface.
  - Podemos crear una clase en el software que cumpla la interface y pasarla como parámetro.
- Antes de Java 8, se usaban clases anónimas.
  - Se puede crear una clase en dinámico, sin crear la clase como un fichero en el sistema, que después de la ejecución desaparece del sistema.
- Desde Java 8: expresiones lambda, más concisas y legibles.

TEMA 5. CARACTERÍSTICAS AVANZADAS

Clases Anónimas en Java

- Se usan desde Java 1.1.
  - Nos permiten crear una clase para un uso local sin necesidad de crear un fichero nuevo sobre el diseñó y su mantenimiento.
  - La vida de sea clase se queda en el ámbito de la función y solo pude usar los elemento públicos de la interface e internamente sus métodos privados. No hay mecanismo de acceder a sus métodos públicos propios si se defienen.
  - Muy útiles cuando se desea modificar comportamientos en dinámico (Botones o cálculos internos)

```
import java.util.*;

public class EjemploAnonima {
    public static void main(String[] args) {
        List<String> lista = Arrays.asList("A", "C", "B");

        // Orden con clase anónima
        Collections.sort(lista,
        new Comparator<String>() {
                public int compare(String s1, String s2) {
                    return s1.compareTo(s2);
                }
            }
        );

        System.out.println(lista);
    }
}
```

TEMA 5. CARACTERÍSTICAS AVANZADAS

Lambda en Java

- Se usan desde Java 8.
  - Simplifican el código y lo hacen mas directo.
  - Ayudan al manejo de clases de un solo método que modifican comportamiento.
  - Se basan en la programación funcional.
  - Permiten la implementación de acciones (parametros) -> { cuerpo }
  - Integracion con streams, colecciones, switch …..

```
() -> System.out.println("Hola mundo");     // sin parámetros
x -> x * x;                                 // un parámetro
(a, b) -> a + b;                            // dos parámetros
(String s) -> { return s.length(); }        // con tipo y cuerpo
```

```
List<String> lista = Arrays.asList("A", "B", "C");
lista.forEach(s -> System.out.println(s));
```

Lambda en Java

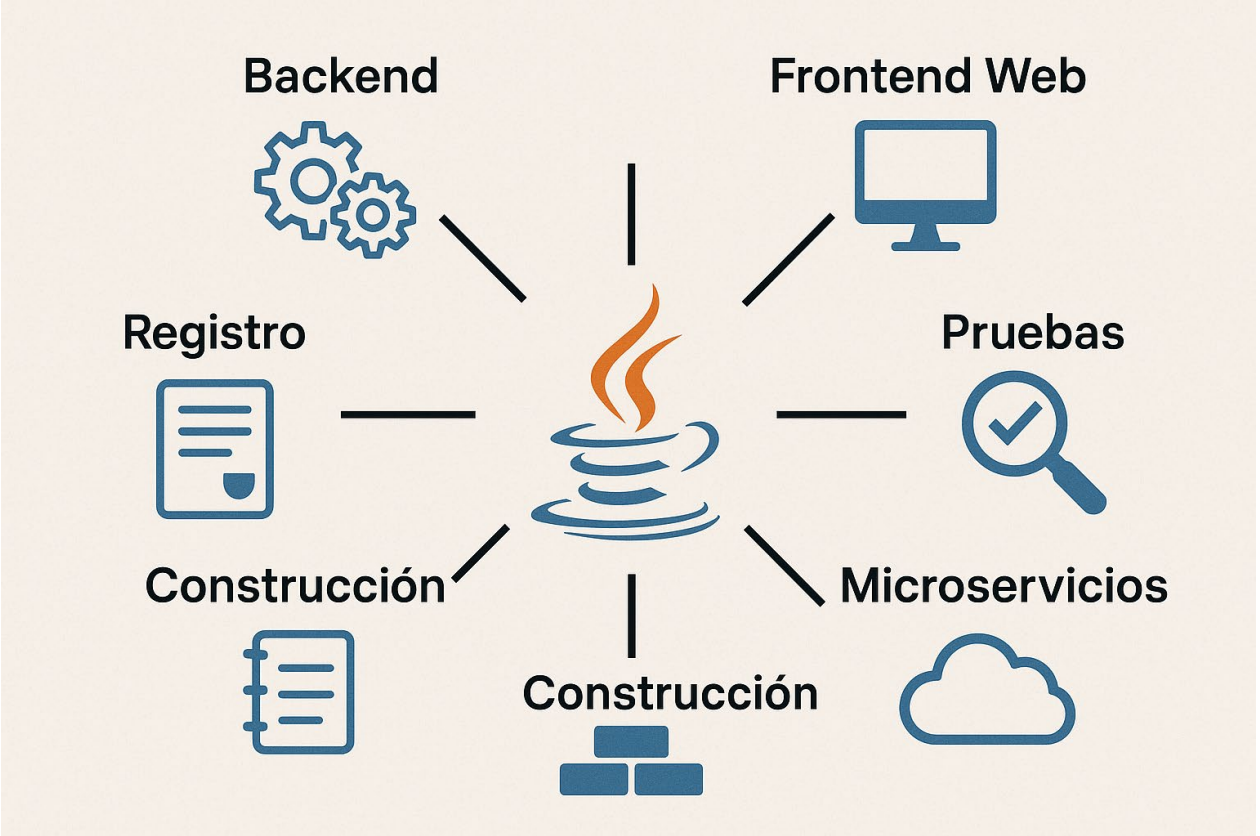
```
import java.util.*;

public class EjemploAnonima {
    public static void main(String[] args) {
        List<String> lista = Arrays.asList("A", "C", "B");

        // Orden con clase anónima
        Collections.sort(lista,
         new Comparator<String>() {
                    public int compare(String s1, String s2) {
                        return s1.compareTo(s2);
                    }
                }
        );

        System.out.println(lista);
    }
}
```

```
import java.util.*;

public class EjemploLambda {
    public static void main(String[] args) {
        List<String> lista = Arrays.asList("A", "C", "B");

        // Orden con lambda
        Collections.sort(lista, (s1, s2) -> s1.compareTo(s2));

        System.out.println(lista);
    }
}
```

TEMA 5. CARACTERÍSTICAS AVANZADAS

Frameworks

- Simplifican el desarrollo
- Ayudan a validar el trabajo.
- Mejoran la interconexión con agentes externos (BBDD, Cloud, interface Grafico…..)
- Es código *robusto* y validado.
- Reduce la complejidad de códigos complejos, pero repetitivos.



TEMA 5. CARACTERÍSTICAS AVANZADAS

Frameworks (lista)

- Spring Framework
  - Desarrollo de aplicaciones empresariales.
  - Inyección de dependencias, configuración, seguridad y acceso a datos.
- **Spring Boot**
  - Variante de Spring para crear aplicaciones listas para producción con mínima configuración.
- **Hibernate**
  - Mapeo objeto-relacional (ORM).
  - Simplifica el acceso y persistencia de datos en bases de datos relacionales.
- Jakarta EE (antes Java EE)
  - Conjunto de especificaciones para aplicaciones empresariales (servlets, EJB, JPA, JAX-RS…).
  - Base de servidores de aplicaciones como GlassFish, Payara, WildFly.

TEMA 5. CARACTERÍSTICAS AVANZADAS

Frameworks (lista 2)

- JSF (Jakarta Faces)
  - Framework para construir interfaces web basadas en componentes.
- Struts
  - Framework MVC para aplicaciones web, muy usado en los 2000.
- Play Framework
  - Desarrollo web reactivo y escalable con soporte para Java y Scala.
- GWT
  - Creación de interfaces de usuario web en Java, sin necesidad de JavaScript.
- **JUnit**
  - Framework de pruebas unitarias.

TEMA 5. CARACTERÍSTICAS AVANZADAS

Frameworks (lista 2)

- TestNG
  - Framework de pruebas más flexible que JUnit (tests unitarios, de integración, dependencias entre tests).
- **Apache Maven**
  - Gestión de proyectos y dependencias.
- Gradle
  - Alternativa a Maven, con un DSL más flexible para la construcción de proyectos.
- Log4j / SLF4J
  - Frameworks para logging en aplicaciones Java.
- Micronaut / Quarkus (modernos)
    - Frameworks ligeros para microservicios y aplicaciones cloud-native.

Validacion del Sofware

- El proceso de validación del software es una etapa tan importante como el desarrollo o el diseño.
- Un error en ejecución tiene un coste 10 veces superior a detectarlo en fase de pruebas
- Existen múltiples software/librerías/tecnicas de validación para las diferentes partes del software (interface de usuario, base de datos, modelo, conexiones externas …)

TEMA 5. CARACTERÍSTICAS AVANZADAS

JUnit

- Framework de pruebas unitarias.
- Valida el comportamiento mediante el proceso entrada controlada, salida esperada.
- Maneja excepciones.
- Base de contrato de las funciones. Invariantes, precondiciones y postcondiciones.
- No puede validar interface de usuario (aunque hay técnicas de simulación que pueden forzar la validación)

JUnit3

- Versión 3 del framework JUnit.
- Las clases deben heredar de TestCase
- Los métodos de validación deben comenzar su nombre con ***test*** para que se ejecuten como prueba.

```
import junit.framework.TestCase;

public class CalculadoraTest extends TestCase {

    public void testSuma() {
        assertEquals(5, 2 + 3);
    }

    public void testResta() {
        assertEquals(1, 3 - 2);
    }
}
```

TEMA 5. CARACTERÍSTICAS AVANZADAS

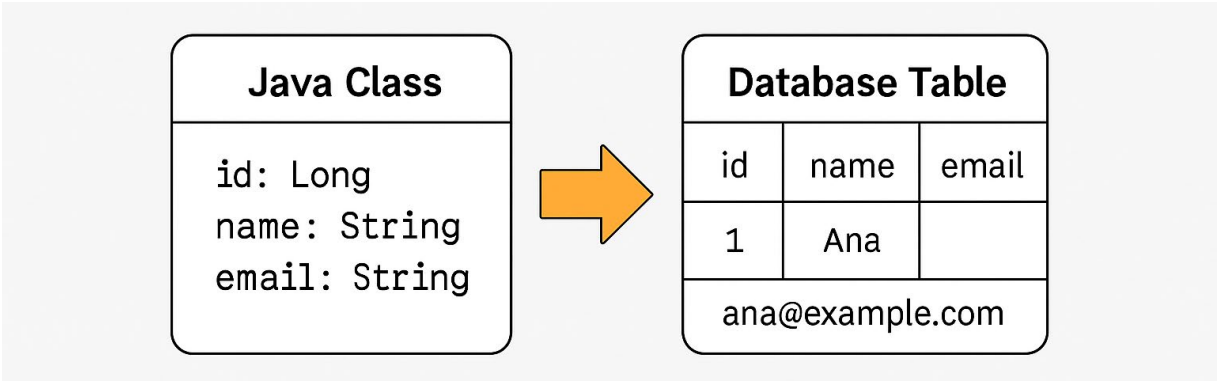
JUnit4

- Versión 4 del framework JUnit.
- Las clases **no necesitan** heredar de TestCase
- Los métodos de validación deben tener la etiqueta ***@Test*** para que se ejecuten como prueba.
- Mas flexible y limpio

```
import static org.junit.Assert.*;
import org.junit.Test;

public class CalculadoraTest {

    @Test
    public void testSuma() {
        assertEquals(5, 2 + 3);
    }

    @Test
    public void testResta() {
        assertEquals(1, 3 - 2);
    }
}
```

Hibernate

- Hibernate es un framework de mapeo objeto-relacional (ORM) para Java.
- ***Permite trabajar con bases de datos relacionales utilizando objetos Java en lugar de escribir directamente sentencias SQL.***



Hibernate

Ventajas

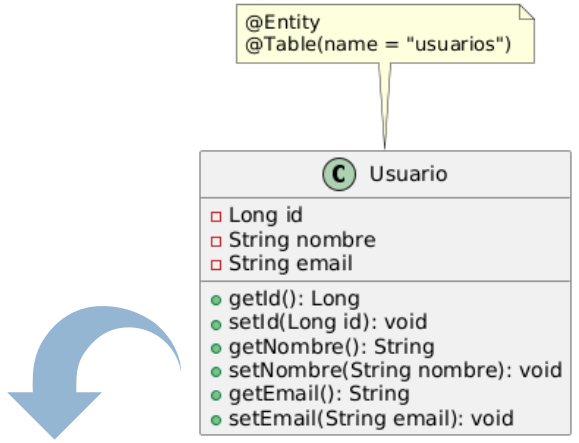
- **Abstracción de SQL** = evita escribir la mayoría de consultas SQL.
- **Productividad** = simplifica el acceso a datos y reduce código repetitivo (CRUD).
- **Portabilidad** = funciona con múltiples motores de bases de datos (MySQL, PostgreSQL, Oracle…).
- **Caché** = mejora el rendimiento gestionando datos en memoria.
- **Integración** = se integra fácilmente con frameworks como Spring.

Limitaciones

- **Curva de aprendizaje** = la configuración inicial puede ser compleja.
- **Sobreabstracción** = para consultas muy específicas puede ser menos eficiente que SQL nativo.
- **Rendimiento** = en operaciones masivas, puede ser más lento si no se ajusta bien.
- **Debugging** = la capa intermedia dificulta la depuración de errores SQL.
- **Remodelado** = Obliga a remodelar el software por incompatibilidad de persisntencia de ciertas estructuras (interfaces, frameworks…..)

Hibernate (ejemplo)

```
import jakarta.persistence.*;

@Entity
@Table(name = "usuarios")
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nombre;
    private String email;

    // ….. Getters y Setters
}
```




```
-- Crear tabla
CREATE TABLE usuarios (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(255),
    email VARCHAR(255)
);

-- Insertar datos
INSERT INTO usuarios (nombre, email) VALUES ('Ana', 'ana@example.com');
INSERT INTO usuarios (nombre, email) VALUES ('Juan', 'juan@mail.com');
INSERT INTO usuarios (nombre, email) VALUES ('María', 'maria@mail.com');
```

Hibernate (ejemplo)

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class Main {
    public static void main(String[] args) {
        SessionFactory factory = new Configuration()
                .configure("hibernate.cfg.xml")
                .addAnnotatedClass(Usuario.class)
                .buildSessionFactory();

        Session session = factory.getCurrentSession();
```

```
try {
            session.beginTransaction();

            Usuario u = new Usuario();
            u.setNombre("Ana");
            u.setEmail("ana@example.com");

            session.save(u); // INSERT automático

            session.getTransaction().commit();
        } finally {
            factory.close();
        }
    }
}
```

Spring

Framework para Java que facilita el desarrollo de aplicaciones empresariales

- Contiene múltiples módulos integrables multiproposito: Spring MVC, Spring Data, Spring Security, etc
- Facilitar la integración con bases de datos, test, servicios rest, gestión de la seguridad y acceso a la aplicación.
- Inyección de dependencias y Inversión de control integrado y sencillo de implementar
- Solucion basada en anotaciones que aplican soluciones de diseño optimas de manera transparente al usuario.

Spring Boot

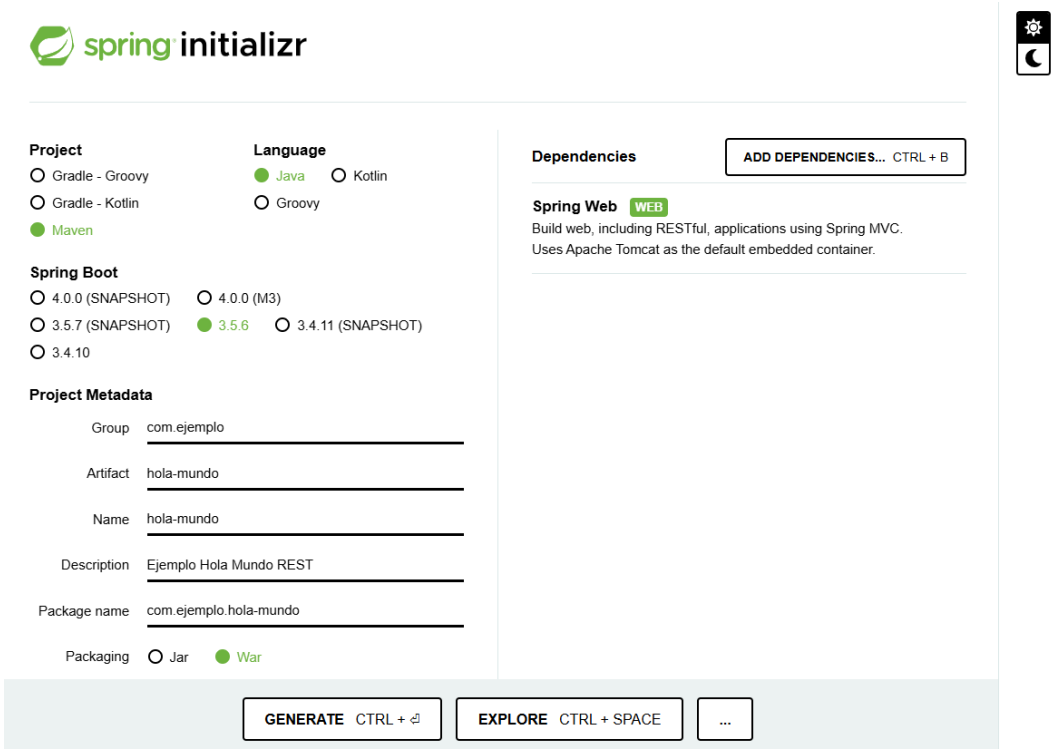
Extensión de Spring que simplifica la configuración

- Arranque rápido de proyectos con **dependencias preconfiguradas (starters)**
- **Servidor embebido** (Tomcat/Jetty).
- Configuración mínima y simplificada frente a Spring clásico con archivos properties e yml.
- Solucion de alta velocidad para la implementación de aplicaciones de APIs REST y enfocadas a microservicios.

Spring Boot (ejemplo Hola Mundo REST)

Configuracion:

- Descargar proyecto base de https://start.spring.io
  - *(Ó Spring Initializr integrado en IntelliJ Ultimate)*
  - ***Packagin : jar o war***
  - ***Dependecies : Spring Web***



Spring Boot (ejemplo Hola Mundo REST)

```
package com.ejemplo.hola_mundo;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplicatio
n;

@SpringBootApplication
public class HolaMundoApplication {
    public static void main(String[] args) {
        SpringApplication.run(HolaMundoApplication.class, args);
    }
}
```

```
package com.ejemplo.hola_mundo;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HolaController {

    @GetMapping("/hola")
    public String decirHola() {
        return "¡Hola desde Spring Boot!";
    }
}
```

```
http://localhost:8080/hola
```

Spring Boot (ejemplo Hola Mundo REST + API)

```
package com.ejemplo.hola_mundo;

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api")
public class SaludoController {

    @GetMapping("/saludo")
    public String saludo(@RequestParam String nombre) {
        return "Hola, " + nombre + "!";
    }
}
```

```
http://localhost:8080/api/saludo?nombre=Joaquin
```