David Torres Aranha

Actividad 6.2: Introspección y Reflexión

Investigar sobre la característica de “Reflexión” que ofrece JAVA

Brevemente, ¿qué es y para qué sirve la característica de JAVA llamada Introspección? ¿y la característica de JAVA llamada Reflexión? ¿Qué paquete java ofrece esas características?

- Introspección: Permite examinar y "leer" la estructura de un objeto (atributos y métodos) en tiempo de ejecución.
- Reflexión: Permite no solo examinar, sino modificar el comportamiento, instanciar objetos e invocar métodos dinámicamente.
- Paquetes: java.beans (Introspección) y java.lang.reflect / java.lang.Class (Reflexión).

¿Cuáles son los principales métodos que se usan en Introspección?

Basados en la API JavaBeans:

- Introspector.getBeanInfo(): Obtiene la información de la clase.
- PropertyDescriptor.getReadMethod(): Obtiene el método getter.
- PropertyDescriptor.getWriteMethod(): Obtiene el método setter.

¿Cuáles son los principales métodos que se usan en Reflexión?

- Class.forName(): Carga una clase desde un texto (String).
- getDeclaredFields() / getDeclaredMethods(): Obtienen los atributos y métodos.
- Method.invoke(): Ejecuta un método.
- Field.set() / Field.get(): Modifican o leen el valor de un atributo.

Busca un ejemplo sencillo de uso.

Averiguar la superclase de String en tiempo de ejecución:

```
System.out.println(String.class.getSuperclass().getName());
// Salida: java.lang.Object
```

¿Qué harías si tuvieras el nombre de una clase como "literal" (con su paquete incluido) y necesitas crear un objeto de esa clase en tiempo de ejecución?

La cargaría con Class.forName("paquete.Clase") y crearía el objeto usando getDeclaredConstructor().newInstance().

¿Puedes invocar un método de un objeto sin conocerlo en tiempo de compilación? ¿Cómo?

Sí, mediante reflexión. Se busca el método por su nombre usando getDeclaredMethod("nombreMetodo") y se ejecuta llamando a .invoke(objeto, parametros).

¿Qué riesgos o desventajas crees que tiene usar reflexión en exceso?

- Peor rendimiento: Es más lento porque la JVM no puede optimizar estas llamadas.
- Inseguridad: Rompe la encapsulación al permitir acceder a variables y métodos private.
- Errores ocultos: Los errores tipográficos en los nombres (Strings) fallarán en tiempo de ejecución, no al compilar.

Dentro de acceso a datos (2º DAM), indícame 3 ejemplos de proyectos populares en los que se esté utilizando la introspección y la reflexión.

- Hibernate / JPA: Para mapear tablas de base de datos a objetos Java (ORM).
- Spring Boot: Para la Inyección de Dependencias (crear e inyectar beans automáticamente).
- Jackson / Gson: Para transformar objetos Java en archivos JSON y viceversa.