1. Concepto, Utilidad y Paquete Principal

- **Introspección:** Es la capacidad de examinar las propiedades, eventos y métodos expuestos por una clase JavaBean en tiempo de ejecución. Sirve para que frameworks y herramientas descubran automáticamente la estructura de los objetos (getters, setters y propiedades) sin necesidad de instanciarlos ni modificar sus campos directamente.

- **Reflexión:** Es la capacidad que tiene un programa en tiempo de ejecución para inspeccionar, modificar su propio comportamiento, instanciar clases dinámicamente, acceder a campos privados e invocar métodos sin conocer los tipos concretos en tiempo de compilación.

- **Paquete de JAVA:** La reflexión se ofrece principalmente a través de java.lang.reflect (junto con la clase **java.lang.Class**), mientras que la introspección orientada a JavaBeans se encuentra en java.beans.

2. Principales Métodos en Introspección

- **Introspector.getBeanInfo(Class<?> beanClass):** Analiza la clase dada y devuelve un objeto BeanInfo con la metadata del JavaBean.
- **BeanInfo.getPropertyDescriptors():** Devuelve un array de PropertyDescriptor con información de todas las propiedades expuestas (asociadas a métodos getter y setter).
- **BeanInfo.getMethodDescriptors():** Devuelve un array de MethodDescriptor con los métodos públicos del Bean.
- **PropertyDescriptor.getReadMethod() y getWriteMethod():** Obtienen respectivamente el método de lectura (getter) y de escritura (setter) asociado a una propiedad.

3. Principales Métodos en Reflexión

- **Class.forName(String className):** Carga dinámicamente la clase dada por su nombre cualificado (*Fully Qualified Domain Name*).
- **Class.getDeclaredConstructor(Class<?>... parameterTypes):** Obtiene un constructor específico de la clase.
- **Constructor.newInstance(Object... initargs):** Crea una nueva instancia de la clase invocando el constructor obtenido.
- **Class.getDeclaredFields() / getDeclaredMethods():** Obtienen todos los atributos o métodos declarados en la clase (incluidos los privados).
- **Method.invoke(Object obj, Object... args):** Invoca dinámicamente un método sobre una instancia especificada.
- **Field.setAccessible(boolean flag):** Desactiva el control de acceso en tiempo de ejecución de Java para permitir el acceso a miembros private o protected.
- **Field.set(Object obj, Object value) / Field.get(Object obj):** Asigna o lee directamente el valor de un atributo en un objeto.

```
import java.lang.reflect.Method;

public class EjemploReflexion {
    public static void main(String[] args) {
        try {
            Class<?> clazz = Class.forName("java.lang.String");

            Method method = clazz.getMethod("toUpperCase");
            Object resultado = method.invoke("hola mundo");

            System.out.println("Resultado: " + resultado); // Imprime HOLA MUNDO
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

5. Creación de un Objeto a partir de un Literal con Nombre de Clase

Para instanciar una clase disponiendo únicamente de un `String` con su nombre completo (paquete + clase), se utiliza `Class.forName()` en combinación con `getDeclaredConstructor().newInstance()`

6. Invocación de Métodos en Tiempo de Ejecución sin Conocimiento Previo

Sí, es posible invoking un método dinámicamente usando reflexión mediante los métodos `getMethod()` o `getDeclaredMethod()` e invocándolo con `.invoke()`

7. Riesgos y Desventajas de Usar Reflexión en Exceso

- **Sobrecarga de Rendimiento (*Performance Overhead*):** La inspección dinámica impide ciertas optimizaciones en tiempo de compilación y del compilador JIT, resultando en una ejecución más lenta.
- **Quiebre de Encapsulación y Seguridad:** Al utilizar setAccessible(true) se rompen las reglas de visibilidad del lenguaje (private/protected), exponiendo datos sensibles a modificaciones no deseadas.
- **Pérdida de Verificación en Tiempo de Compilación (*Type Safety*):** Los errores de tipo o nombres de métodos incorrectos no son detectados al compilar, provocando excepciones en tiempo de ejecución (ClassNotFoundException, NoSuchMethodException, etc.).
- **Dificultad de Mantenimiento y Refactorización:** Los IDEs no pueden actualizar automáticamente los literales en texto cuando se renombran clases o métodos.

8. Tres Proyectos/Frameworks Populares en Acceso a Datos

1. **Hibernate / Jakarta Persistence (JPA):** Utiliza introspección y reflexión para mapear entidades POJO/Beans con tablas relacionales de la base de datos, instanciando objetos e inyectando datos directamente en campos privados.
2. **Spring Data / Spring Framework:** Emplea reflexión para la Inyección de Dependencias (@Autowired), resolución dinámica de anotaciones (@Entity, @Table) y generación de repositorios DAO proxies en tiempo de ejecución.
3. **Jackson / Gson:** Librerías de mapeo e integración de datos que inspeccionan JavaBeans mediante introspección para serializar objetos a formatos XML/JSON y deserializarlos dinámicamente.