In **Java** ermöglicht die **Reflection API**, dass Programme zur **Laufzeit** Informationen über Klassen, Methoden, Konstruktoren, Felder und sogar die Eigenschaften eines Objekts abrufen und verwenden können. Hier sind einige Beispiele, die zeigen, wie Reflection in Java verwendet wird:

1. Abrufen von Klasseninformationen mit Reflection

Mit **Reflection** kannst du Informationen über eine Klasse abrufen, wie z.B. ihren Namen, ihre Methoden und Felder.

Beispiel: Klasseninformationen abrufen

Code kopieren

```java
class Person {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}

public class ReflectionExample {
    public static void main(String[] args) {
        // Klassenobjekt von Person abrufen
        Class<?> personClass = Person.class;

        // Klassennamen abrufen
        System.out.println("Class Name: " + personClass.getName());

        // Methoden abrufen
        System.out.println("\nMethods:");
        for (Method method : personClass.getDeclaredMethods()) {
            System.out.println(method.getName());
        }

        // Felder abrufen
        System.out.println("\nFields:");
        for (Field field : personClass.getDeclaredFields()) {
            System.out.println(field.getName());
        }
    }
}
```

Ausgabe:

```
Class Name: Person

Methods:
getName
getAge

Fields:
name
age
```

2. Methoden aufrufen mit Reflection

Mit **Reflection** kannst du Methoden eines Objekts dynamisch aufrufen, ohne die Methoden explizit zu kennen.

Beispiel: Methode zur Laufzeit aufrufen

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

public class ReflectionInvokeMethod {
    public static void main(String[] args) throws Exception {
        // Klassenobjekt von Person abrufen
        Class<?> personClass = Person.class;

        // Neues Person-Objekt instanziieren
        Person person = new Person("Alice", 30);

        // Methode getName() zur Laufzeit aufrufen
        Method getNameMethod = personClass.getMethod("getName");
        String name = (String) getNameMethod.invoke(person);
        System.out.println("Name: " + name);

        // Methode getAge() zur Laufzeit aufrufen
        Method getAgeMethod = personClass.getMethod("getAge");
        int age = (int) getAgeMethod.invoke(person);
        System.out.println("Age: " + age);
    }
}
```

Ausgabe:

```
Name: Alice
Age: 30
```

3. Felder ändern mit Reflection

Mit **Reflection** kannst du auch auf **private Felder** zugreifen und deren Werte ändern.

Beispiel: Ändern eines privaten Felds

```java
import java.lang.reflect.Field;

public class ReflectionModifyField {
    public static void main(String[] args) throws Exception {
        // Klassenobjekt von Person abrufen
        Class<?> personClass = Person.class;

        // Neues Person-Objekt instanziieren
        Person person = new Person("Alice", 30);

        // Privates Feld "name" abrufen
        Field nameField = personClass.getDeclaredField("name");

        // Privates Feld zugänglich machen
        nameField.setAccessible(true);

        // Wert des Felds ändern
        nameField.set(person, "Bob");

        // Methode getName() aufrufen, um den geänderten Namen zu erhalter
        Method getNameMethod = personClass.getMethod("getName");
        String updatedName = (String) getNameMethod.invoke(person);
        System.out.println("Updated Name: " + updatedName);
    }
}
```

Ausgabe:

```
Updated Name: Bob
```

4. Konstruktoren zur Laufzeit aufrufen

Du kannst auch **Konstruktoren** mit Reflection zur Laufzeit aufrufen, um ein neues Objekt zu instanziieren.

Beispiel: Konstruktor zur Laufzeit aufrufen

```java
import java.lang.reflect.Constructor;

public class ReflectionInvokeConstructor {
    public static void main(String[] args) throws Exception {
        // Klassenobjekt von Person abrufen
        Class<?> personClass = Person.class;

        // Konstruktor mit Parametern (String, int) abrufen
        Constructor<?> constructor = personClass.getConstructor(String.class, int.class);

        // Neues Objekt zur Laufzeit erstellen
        Person person = (Person) constructor.newInstance("Charlie", 25);

        // Name und Alter abrufen
        System.out.println("Name: " + person.getName());
        System.out.println("Age: " + person.getAge());
    }
}
```

Ausgabe:

```
Name: Charlie
Age: 25
```

Zusammenfassung:

- **Reflections** in Java ist eine Funktion, die es dem Programm erlaubt, **zur Laufzeit Informationen** über Klassen, Methoden und Objekte zu erhalten.
- Du kannst damit **Methoden aufrufen**, **Felder ändern** und **Objekte dynamisch instanziieren**, selbst wenn du die genauen Details zur Compilezeit nicht kennst.
- Dies ist besonders nützlich in Frameworks, wie z.B. in **JavaBeans**, **Dependency Injection** oder **Unit Testing**, aber sollte **vorsichtig** verwendet werden, da es zu **Sicherheits- und Leistungsproblemen** führen kann.