Java Generics:

Java generics allow you to create classes, interfaces, and methods that operate on types as parameters, rather than on specific concrete types. This helps you write more flexible and reusable code while ensuring type safety. Generics were introduced in Java 5.

Here are two examples to illustrate the concept of Java generics:

Example 1: Generic Class

In this example, we'll create a generic `Box` class that can hold any type of object. We'll use a type parameter `<T>` to make it generic.

```java
public class Box<T> {
    private T value;

    public Box(T value) {
        this.value = value;
    }

    public T getValue() {
        return value;
    }

    public void setValue(T value) {
        this.value = value;
    }

    public static void main(String[] args) {
        Box<Integer> integerBox = new Box<>(10);
        Box<String> stringBox = new Box<>("Hello, Generics!");

        System.out.println("Integer Value: " + integerBox.getValue());
        System.out.println("String Value: " + stringBox.getValue());
    }
}
```

In this example, the `Box` class is generic, and you can create instances of it with different types (e.g., `Integer`, `String`). The type parameter `<T>` allows you to specify the type of data the `Box` will hold when creating an instance.

Example 2: Generic Method

In this example, we'll create a generic method that can find the maximum value in an array of any comparable objects.

```java
public class GenericMethodExample {
    public static <T extends Comparable<T>> T findMax(T[] array) {
        if (array == null || array.length == 0) {
            return null;
        }

        T max = array[0];
        for (T element : array) {
            if (element.compareTo(max) > 0) {
                max = element;
            }
        }
        return max;
    }

    public static void main(String[] args) {
        Integer[] intArray = { 3, 8, 1, 6, 7, 2 };
        Double[] doubleArray = { 2.5, 5.0, 1.3, 9.8 };
        String[] stringArray = { "apple", "banana", "cherry", "date" };

        System.out.println("Max Integer: " + findMax(intArray));
        System.out.println("Max Double: " + findMax(doubleArray));
        System.out.println("Max String: " + findMax(stringArray));
    }
}
```

In this example, the `findMax` method is a generic method that can find the maximum value in an array of any type that implements the `Comparable` interface. The type parameter `<T extends Comparable<T>>` ensures that the elements in the array can be compared.

These examples demonstrate how Java generics allow you to write reusable and type-safe code by parameterizing classes and methods with type information.