Generics in Java

Generics: Introduction, Generic Example, Generic Class, Generic Method Practice problems

Generics Introduction

Java Generics are a way to create **classes, interfaces, and methods** that work with **any data type**, while still keeping **type safety**.

Why use Generics?

- To write **flexible** and **reusable** code.
- To avoid **typecasting**.
- To catch **type errors at compile-time** instead of runtime.

Generic Example

- Think of a **tiffin box** — it can carry **roti**, **rice**, or **fruits**.
- The box shape is the same, only the **type of content** changes.
- Generics are like the tiffin box — **one structure**, many **data types**.

Generic Class

✓ A generic class can handle multiple types of data.
✓ We use angle brackets <> to define the type.

Syntax:

```
class ClassName<Type> {

    T variable; // T can be any type: Integer, String, Float, etc.

}
```

```
class StudentRecord<T> {
    String name;
    T marks;

    void displayRecord() {
        System.out.println("Name: " + name);
        System.out.println("Marks: " + marks);
    }
}
class Main {
    public static void main(String[] args) {
        StudentRecord<Integer> s1 = new StudentRecord<>();
        s1.name = "Amit";
        s1.marks = 85;
        s1.displayRecord();
        StudentRecord<Float> s2 = new StudentRecord<>();
        s2.name = "Neha";
        s2.marks = 91.5f;
        s2.displayRecord();
    }
}
```

Generic Method

A generic method is a method that works with any data type, declared using angle brackets <> before the return type.

✓ Generic methods allow us to write one method that works with any data type.

✓ They are declared with <T> before the return type.

✓ T can be Integer, String, Float, or any Object.

✓ They improve reusability and remove the need for multiple overloaded methods.

Syntax:

```
public <T> void methodName(T parameter) {
    // method body
}
```

Example

```
class StudentUtil {
    public static <T> void compareMarks(T m1, T m2) {
        System.out.println("Mark 1: " + m1 + ", Mark 2: " + m2);
        if (m1.equals(m2)) {
            System.out.println("Both marks are equal");
        } else {
            System.out.println("Marks are different");
        }
    }
}
class Main {
    public static void main(String[] args) {
        StudentUtil.compareMarks(85, 85);         // Integer
        StudentUtil.compareMarks(91.5f, 89.5f);   // Float
        StudentUtil.compareMarks("A+", "A");      // Grade (String)
    }
}
```

Test Question 1.

Q1. What is the correct syntax for declaring a generic method?

A) public <T> void show(T item)

B) B) public void <T> show(T item)

C) C) public void show<T>(T item)

D) D) public T void show(T item)

Test Question 2.

Q2. What is the purpose of <T> before the return type in a method declaration?

A) To indicate that the method is static

B) B) To define a generic type parameter for that method

C) C) To enforce encapsulation

D) D) To import java.util.* automatically

Practice Questions

Q1. Print Elements from Any Type of Array

Q2. Check if Two Values Are Equal

Q3. Swap Two Elements in an Array

Q4. Count Occurrences of an Element