

LAB # 7

Generics in Java

OBJECTIVE:

Implementing generic classes and methods for ensuring compile time type safety of data.

Generics:

Generics is a type of specified data type in the related context of class, method or object. In order to restrict any class, method or object with specified type then generics should be introduced in the code. Generics means parameterized types. The idea is to allow type (Integer, String etc., and user-defined types) to be a parameter to methods, classes, and interfaces. Using Generics, it is possible to create classes that work with different data types. An entity such as class, interface, or method that operates on a parameterized type is called generic entity.

Importance of generics:

Object is the superclass of all other classes and Object reference can refer to any type object. These features lack type safety. Generics adds that type safety feature.

Generics Parameters name:

- E – Element (used extensively by the Java Collections Framework, for example ArrayList, Set etc.)
- K – Key (Used in Map)
- N – Number
- T – Type
- V – Value (Used in Map)
- S,U,V etc. – 2nd, 3rd, 4th types

Types of generics:

1. Generic Class

we use <> to specify parameter types in generic class creation. To create objects of generic class, we use following syntax.

```
class Test<T>
{
    // An object of type T is declared
    T obj;
    Test(T obj)
    {
        this.obj = obj;
    } // constructor
    public T getObject()
    {
        return this.obj;
    }
}

// Driver class to test above

class Main
{
    public static void main (String[] args)
    {
        // instance of Integer type
        Test <Integer> iObj = new Test<Integer>(84);
        System.out.println(iObj.getObject());

        // instance of String type
        Test <String> sObj = new Test<String>("Hiba Fatima");
        System.out.println(sObj.getObject());
    }
}
```

We can also pass multiple Type parameters in Generic classes:


```
class Test<T, U>
{
    T obj1; // An object of type T
    U obj2; // An object of type U

    // constructor
    Test(T obj1, U obj2)
    {
        this.obj1 = obj1;
        this.obj2 = obj2;
    }

    // To print objects of T and U
    public void print()
    {
        System.out.println(obj1);
        System.out.println(obj2);
    }
}

// Driver class to test above
class Main
{
    public static void main (String[] args)
    {
        Test <String, Integer> obj =
            new Test<String, Integer>("Hiba Fatima", 84);
    }
}
```

```
        obj.print();
    }
}
```



Hiba Fatima
84

2. Generic Functions:

We can also write generic functions that can be called with different types of arguments based on the type of arguments passed to generic method, the compiler handles each method.

```
class Main
{
    // A Generic method example
    static <T> void genericDisplay (T element)
    {
        System.out.println("Value is" +
                           " = " + element);
    }

    // Driver method
    public static void main(String[] args)
    {
        // Calling generic method with Integer argument
        genericDisplay(11);
    }
}
```

```
// Calling generic method with String argument
genericDisplay("Software Construction and Development");

// Calling generic method with double argument
genericDisplay(1.0);
}
}
```

```
Value is = 11
Value is = Software Construction and Development
Value is = 1.0
```

Example of a program without generic method and classes:

```
public class Main
{
    public static void main(String[] args) {
        // create arrays of Integer, Double and Character
        Integer[] integerArray = {1, 2, 3, 4, 5, 6};
        Double[] doubleArray = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7};
        Character[] characterArray = {'H', 'E', 'L', 'L', 'O'};

        System.out.printf("Array integerArray contains:%n");
        printArray(integerArray);
        System.out.printf("%nArray doubleArray contains:%n");
        printArray(doubleArray);
    }
}
```

```
System.out.print("Array characterArray contains:\n");
printArray(characterArray);
}
public static void printArray(Integer[] inputArray)
{
    for (int element : inputArray)
        System.out.printf("%s ", element);
}
public static void printArray(Double[] inputArray)
{
    for (double element : inputArray)
        System.out.printf("%s ", element);
    System.out.println();
}
public static void printArray(Character[] inputArray)
{
    for (char element : inputArray)
        System.out.printf("%s ", element);
    System.out.println();
}
}
```

```
Array integerArray contains:
1 2 3 4 5 6
Array doubleArray contains:
1.1 2.2 3.3 4.4 5.5 6.6 7.7
Array characterArray contains:
H E L L O
```

In the above code, three overloaded methods are implemented for getting integer, double or character type input. This long described pattern can be transformed into a single method by simply adding generic data type as below:

```
public static void printArray(E[] inputArray)
{
    // display array elements
    for (E element : inputArray)
        System.out.printf("%s ", element);
    System.out.println();
}
```

Solution with Generics:

```
public class Main {
    // generic method printArray
    public static < E > void printArray( E[] inputArray ) {
        // Display array elements
        for(E element : inputArray) {
            System.out.printf("%s ", element);
        }
        System.out.println();
    }
}
```

```
}

    public static void main(String args[]) {
        // Create arrays of Integer, Double and Character
        Integer[] intArray = { 1, 2, 3, 4, 5 };
        Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };
        Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };

        System.out.println("Array integerArray contains:");
        printArray(intArray);    // pass an Integer array

        System.out.println("\nArray doubleArray contains:");
        printArray(doubleArray); // pass a Double array

        System.out.println("\nArray characterArray contains:");
        printArray(charArray);   // pass a Character array
    }
}
```

Output:

```
terminated - Main Java Application C:\Users\delnape\AppData\Local\Temp\plugins\org.eclipse.jdt.ui.openjdk102sp0.jar
Array integerArray contains:
1 2 3 4 5

Array doubleArray contains:
1.1 2.2 3.3 4.4

Array characterArray contains:
H E L L O
```


Lab Task:

Write a program that takes integer array, double array and character array. Make a generic function that prints these array in reverse order.