SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

19CS204 OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Topic – Generic Programming

Generic Programming

Java Generics is a set of **related methods or a set of similar types**.

Generics allow types **Integer, String, or even user-defined types to be passed as a parameter to classes, methods, or interfaces**.

Generics are mostly used by classes like **HashSet or HashMap.**

Advantages of using generics

Generics ensure **compile-time safety** which allows the programmer to catch the invalid types while compiling the code.

Java Generics helps the programmer to **reuse** the code for whatever type he/she wishes. For instance, a programmer writes a generic method for sorting an array of objects. Generics allow the programmer to use the same method for Integer arrays, Double arrays, and even String arrays.

Another advantage of using generics is that **Individual typecasting isn't required.** The programmer defines the initial type and then lets the code do its job.

Type Parameters

The type parameters naming conventions are important to learn generics thoroughly. The common type parameters are as follows:

1. T - Type
2. E - Element
3. K - Key
4. N - Number
5. V - Value

Types of Java Generics

Generic method

Generic Java method takes a **parameter and returns some value after performing a task**.

It is exactly like a normal function, however, a generic method has **type** parameters which are cited by actual type.

This allows the generic method to be used in a more general way.

The compiler takes care of the type of safety which enables programmers to code easily since they do not have to perform long, individual type castings.

Generic Class

Generic classes

A generic class is implemented **exactly like a non-generic class**.

The only difference is that it contains a **type** parameter section.

There can be more than **one type of parameter, separated by a comma**.

The classes, which accept one or more parameters, are known as parametrized classes or parameterized types.

Generic method

```java
public class TestGenerics4{

  public static < E > void printArray(E[] elements) {
      for ( E element : elements){
          System.out.println(element );
      }
      System.out.println();
  }
  public static void main( String args[] ) {
      Integer[] intArray = { 10, 20, 30, 40, 50 };
      Character[] charArray = { 'J', 'A', 'V', 'A', 'T','P','O','I','N','T' };

      System.out.println( "Printing Integer Array" );
      printArray( intArray  );

      System.out.println( "Printing Character Array" );
      printArray( charArray );
  }
}
```

Generic class

```
class container<T> {
  private T obj1;

  public void add(T obj1) {
    this.obj1 = obj1;
  }

  public T get() {
    return obj1;
  }

  public static void main(String[] args) {
    container<Integer> integerContainer= new container<Integer>();
    container<String> stringContainer = new container<String>();

    integerContainer.add(new Integer(7));
    stringContainer.add(new String("You are awesome"));

    System.out.printf("Integer Value :%d\n\n", integerContainer.get());
    System.out.printf("String Value :%s\n", stringContainer.get());
  }
}
```

Generic Methods

- It is possible to declare a generic method that uses one or more type parameters of its own.
- Furthermore, it is possible to create a generic method that is enclosed within a non-generic class.
- It allows static as well as non-static methods.
- Here, the scope of arguments is limited to the method where it is declared

Generic Methods

```
class GenMethDemo {
// Determine if an object is in an array.
static <T extends Comparable<T>,
V extends T> boolean isIn(T x, V[] y) {
for(int i=0; i < y.length; i++)
if(x.equals(y[i])) return true;
return false;
}
```

```
public static void main(String args[]) {
// Use isIn() on Integers.
Integer nums[] = { 1, 2, 3, 4, 5 };
if(isIn(2, nums))
System.out.println("2 is in nums");
if(!isIn(7, nums))
System.out.println("7 is not in nums");
System.out.println();
// Use isIn() on Strings.
String strs[] = { "one", "two", "three",
"four", "five" };
if(isIn("two", strs))
System.out.println("two is in strs");
if(!isIn("seven", strs))
System.out.println("seven is not in strs");
// Oops! Won't compile! Types must be compatible.
// if(isIn("two", nums))
// System.out.println("two is in strs");
}
}
```

THANK YOU