Generic Class & Methods

Prepared by
Rubyeat Islam
Lec, CSE Dept, MIST
rubyeat88@gmail.com

Java-Generics

- Generics also provide compile-time type safety that allows programmers to catch invalid types at compile time.
- Using Java Generic concept, we might write a generic method for sorting an array of objects, then invoke the generic method with integer arrays, Double arrays, String arrays and so on, to sort the array elements.

Why Generic instead of Non-Generic?

- Find errors in compile time
- The concept of type casting is removed.
- It forces to enforce on using the concept of Algorithm.

Main Utilization of Generic

Java Generic methods and generic classes enable programmers to specify, with a single method declaration, a set of related methods or with a single class declaration , a set of related types respectively.

For example:

It would be nice if we could write a single sort method that could sort the elements in an integer array, a String array or an array of any type that supports ordering.

So remember:

The concept is same but the values are different.

Generic Method

```
package Generic;

public class GenericMethodTest {
   public static < E > void printArray(E [] inputArray){

      for(E element : inputArray){
         System.out.printf("%s ",element);
      }
      System.out.println();
   }

   public static void main(String[] args) {
      //Create two arrays of integer and character types

      Integer [] intArray = {1, 2, 3, 4, 5};
      Character [] charArray = {'A', 'E', 'I', 'O', 'U'};

      System.out.println("IntArray Contains : ");
      printArray(intArray);
      System.out.println("\nCharArray Contains : ");
      printArray(charArray);
   }
}
```

Type Parameter

- When you create a Method in a class that method has certain type of parameters.
- These parameters may be of any type (i.e. integer, string, double and so so) and hold the value.
- But if you don’t know which type of value will be passed as parameter then it is recommended to use **Type Parameter.**

Type Parameter is nothing but is declared in angular brackets < > before the return type.

Generic Class

```
package Generic;

public class GenericClass  < T1, T2 >{
   public void display (T1 var1, T2 var2){
      System.out.println("Name: " + var1 + " ID: " + var2);
   }

   public static void main(String[] args) {
      GenericClass<String, Integer> obj = new
GenericClass<String, Integer>();
      obj.display("Monica", 1234);
   }
}
```

Keep Practicing