Ex:No:12

GENERIC PROGRAMMING

Date: 13/11/2023

Aim:

Write a Java program using genetic function with the help of built-in function for the given problems.

1.Write a program to perform the following operations on an array using generic classes

- **Add an element in the beginning/middle/end**
- **Delete an element from a given position**

Algorithm:

```
Step1: program to perform generic function using array.
Step2: to add an element in beginning /middle /end.
Step3: to delete an element from given position.
Step4: display the operation performed in generic function.
```

Program:

```
 import java.util.Arrays;
 class ArrayOperations<T> {
   private T[] array;
   private int size;
  public ArrayOperations(int capacity) {
      this.array = (T[]) new Object[capacity];
      this.size = 0;
   }
 public void addElementAtBeginning(T element) {
      ensureCapacity();
      System.arraycopy(array, 0, array, 1, size);
      array[0] = element;
      size++;
   }
public void addElementAtMiddle(T element, int position) {
      ensureCapacity();
      System.arraycopy(array, position, array, position + 1, size - position);
      array[position] = element;
      size++;
   }
   public void addElementAtEnd(T element) {
      ensureCapacity();
      array[size++] = element;
   }
 public void deleteElement(int position) {
      if (position < 0 || position >= size) {
        throw new IllegalArgumentException("Invalid position");
      }
System.arraycopy(array, position + 1, array, position, size - position - 1);
      size--;
   }

   private void ensureCapacity() {
        if (size == array.length) {
            array = Arrays.copyOf(array, 2 * size);
        }
    }
public void printArray() {
        System.out.println(Arrays.toString(array));
    }
}

public class GenericArrayOperationsDemo {
    public static void main(String[] args) {
        ArrayOperations<Integer> arrayOps = new ArrayOperations<>(5);
        arrayOps.addElementAtEnd(10);
        arrayOps.addElementAtEnd(20);
        arrayOps.addElementAtEnd(30);
        System.out.println("Original Array:");
        arrayOps.printArray();
       arrayOps.addElementAtBeginning(5);
       arrayOps.addElementAtMiddle(15, 2);
      System.out.println("\nArray after adding elements:");
        arrayOps.printArray();
 arrayOps.deleteElement(1);
        System.out.println("\nArray after deleting an element:");
        arrayOps.printArray();
    }
```

```
}
```

Output:

```
Array after adding elements:

[5, 10, 15, 20, 30]

Array after deleting an element:

[5, 15, 20, 30, 30]
```

2.Write a java program to find the maximum value from the given type of elements using a generic function.

Algorithm:

Step1: program to find a maximum value from the element.

Step2:use generic function to find the maximum element in it.

Step3: find the maximum value from the given element.

Program:

```
public class GenericMinFinder {

public static <T extends Comparable<T>> T findMin(T[] array) {

 if (array == null || array.length == 0) {

            throw new IllegalArgumentException("Array is empty or null");

        }

 T min = array[0];

for (T element : array) {

            if (element.compareTo(min) < 0) {

                min = element;

            }

        }
return min;
    }
  public static void main(String[] args) {
        Integer[] intArray = {5, 3, 8, 2, 7, 1};
        Integer minInt = findMin(intArray);
        System.out.println("Minimum Integer: " + minInt);
        Double[] doubleArray = {2.5, 1.2, 3.8, 0.7, 4.2};
        Double minDouble = findMin(doubleArray);
        System.out.println("Minimum Double: " + minDouble);
        String[] stringArray = {"apple", "banana", "orange", "kiwi", "grape"};
        String minString = findMin(stringArray);
        System.out.println("Minimum String: " + minString);
    }
}
```

Output:

```
Minimum Integer: 1
Minimum Double: 0.7
Minimum String: apple
```

Result:

Thus, java program using generic function is executed and output is verified.

Code/output (15)	
Quiz (5)	
Record (5)	
Total (25)	
Initial	