



CST-239 Activity 5 Guide

Contents

Part 1: Java Generics.....	1
Part 2: Java Collections Framework	3

Part 1: Java Generics

Overview

Goal and Directions:

In this activity, you will develop classes that use Java generic classes, methods, and bounded generics. Complete the following tasks for this activity:

Execution

1. Create a new Java Project named *topic5-1*.
2. Generic Class Type:

- a. Create a new class named *Storage* class in the *app* package with a *main()*.
- b. Define the *Storage* class with a Generic Class Type *T*.
- c. Create a private class member variable named *s* of type *T*.
- d. Create a non-default constructor that takes a single method argument of type *T*. Save the method argument in the private class member variable.
- e. Create a public method named *getData()* that returns private class member variable (of type *T*).
- f. In *main()*, instantiate an instance of *Storage* class to supports a *String* and call its *getData()* method and prints its return value to the console.
- g. In *main()*, instantiate an instance of *Storage* class to supports an *Integer* and call its *getData()* method and prints its return value to the console.
- h. Run the application.
- i. Take a screenshot of the console of the output.
- j. Generate the JavaDoc for all classes.

```
Storage.java
1
2
3 public class Storage<T>
4 {
5     private T s = null;
6
7     public Storage(T s)
8     {
9         this.s = s;
10    }
11
12    public T getData()
13    {
14        return this.s;
15    }
16
17    public static void main(String[] args)
18    {
19        Storage<String> storage1 = new Storage<String>("Mark Reha");
20        System.out.println("This is the data " + storage1.getData());
21        Storage<Integer> storage2 = new Storage<Integer>(0);
22        System.out.println("This is the data " + storage2.getData());
23    }
24 }
```



3. Generic Method Type and Bounded Generic:

- a. Create a new class named *MyArray* class in the *app* package with a *main()*.
- b. Create a public method named *printArray()* that takes a single method argument of a generic array (of type E). In its implementation loop over the array and print each value of the array to the console.
- c. In *main()* create 3 arrays of type Integer, Double, and Character.
- d. In *main()*, instantiate an instance of *MyArray* class and print each array by calling its *printArray()* method.
- e. Run the application.
- f. Take a screenshot of the console.
- g. Copy the *MyArray* class to a new class named *MyNumberArray*.
- h. Update the *printArray()* method to restrict the Method Generic Type to a Number type.
 - i. Fix the array types declared in *main()* to resolve any compiler errors.
 - j. Run the application.
 - k. Take a screenshot of the console output.
 - l. Generate the JavaDoc for all classes.

```
Storage.java  MyArray.java  x
1 public class MyArray
2 {
3     public <E> void printArray(E[] inputArray)
4     {
5         // Iterate over the array and print each element
6         for(E element : inputArray)
7         {
8             System.out.printf("%s ", element);
9         }
10    }
11
12    public static void main(String[] args)
13    {
14        // Create arrays of Integer, Double and Character
15        Integer[] intArray = { 1, 2, 3, 4, 5 };
16        Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };
17        Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };
18
19        // Print each array out
20        MyArray ma = new MyArray();
21        System.out.println("Array integerArray contains:");
22        ma.printArray(intArray); // pass an Integer array
23        System.out.println("\nArray doubleArray contains:");
24        ma.printArray(doubleArray); // pass a Double array
25        System.out.println("\nArray characterArray contains:");
26        ma.printArray(charArray); // pass a Character array
27    }
28 }
29
30
31
32
33 public class MyNumbersArray
34 {
35     public <E extends Number> void printArray(E[] inputArray)
36     {
37         // Iterate over the array and print each element
38         for(E element : inputArray)
39         {
40             System.out.printf("%s ", element);
41         }
42     }
43
44     public static void main(String[] args)
45     {
46        // Create arrays of Integer, Double and Character
47        Integer[] intArray = { 1, 2, 3, 4, 5 };
48        Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };
49        Float[] floatArray = { 0.0f, 1.0f, 2.5f, 3.5f };
50
51        // Print each array out
52        MyNumbersArray ma = new MyNumbersArray();
53        System.out.println("Array integerArray contains:");
54        ma.printArray(intArray); // pass an Integer array
55        System.out.println("\nArray doubleArray contains:");
56        ma.printArray(doubleArray); // pass a Double array
57        System.out.println("\nArray floatArray contains:");
58        ma.printArray(floatArray); // pass a Float array
59    }
60 }
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Deliverables:

The following need to be submitted as this part of the activity:

- a. All screenshots of application in operation.
- b. ZIP file of the code in the project folder. Include the JavaDoc generated for the project.



Part 2: Java Collections Framework

Overview

Goal and Directions:

In this activity, you will develop classes that use the ArrayList, HashMap, Queue, and Stack from the Java Collections Framework. Complete the following tasks for this activity:

Execution

1. Create a new Java Project named *topic5-2*.
2. Using an ArrayList:
 - a. Create a new class named *Playlist* class in the *app* package with a *main()*.
 - b. Create an ArrayList of Integers. Add 5 numbers to the List.
 - c. Create an ArrayList of Strings. Add 5 strings to the List.
 - d. Print the first element of each ArrayList to the console.
 - e. Print the Integer List using a for loop to the console.
 - f. Print the String List using a while loop to the console.
 - g. Run the application.
 - h. Take a screenshot of the console of the output.
 - i. Generate the JavaDoc for all classes.
3. Using a HashMap:
 - a. Create a new class named *PlayMap* class in the *app* package with a *main()*.
 - b. Create an HashMap of Integers. Add 5 numbers to the Map.
 - c. Create an HashMap of Strings. Add 5 strings to the Map.
 - d. Print the size and if empty for each HashMap to the console.
 - e. Print the String Map using a for loop to the console.

```
1 |
2 |
3 import java.util.ArrayList;
4 |
5 |
6 |
7 public class Playlist
8 {
9     public static void main(String[] args)
10    {
11        // Create a List of Integers and add elements using add()
12        List<Integer> integerList = new ArrayList<Integer>();
13        integerList.add(Integer.valueOf(10));
14        integerList.add(Integer.valueOf(11));
15
16        // Create a List of Strings and add elements using add()
17        List<String> stringList = new ArrayList<String>();
18        stringList.add("Hello World");
19        stringList.add("Hi World");
20
21        // Get first element from the List using get()
22        System.out.printf("Integer Value :%d\n", integerList.get(0));
23        System.out.printf("String Value :%s\n", stringList.get(0));
24
25        // Loop over the List using a For Loop
26        for (Integer data : integerList)
27        {
28            System.out.printf("Integer Value :%d\n", data);
29        }
30
31        // Loop over the List using an Iterator
32        Iterator<String> stringIterator = stringList.iterator();
33        while (stringIterator.hasNext())
34        {
35            System.out.printf("String Value :%s\n", stringIterator.next());
36        }
37    }
38 }
```

```
1 |
2 import java.util.HashMap;
3 import java.util.Map;
4 |
5 |
6 |
7 public class PlayMap
8 {
9     public static void main(String[] args)
10    {
11        // Create a Map of Integers keyed by Integers
12        Map<Integer, Integer> integerMap = new HashMap<Integer, Integer>();
13        integerMap.put(1, 1);
14        integerMap.put(2, 2);
15
16        // Create a Map of Strings keyed by Integers
17        Map<Integer, String> stringMap = new HashMap<Integer, String>();
18        stringMap.put(1, "One");
19        stringMap.put(2, "Two");
20
21        // Create a Map of Strings keyed by Strings
22        Map<String, String> nameMap = new HashMap<String, String>();
23        nameMap.put("FirstName", "Mark");
24        nameMap.put("LastName", "Rehe");
25
26        // Print out size and if a Map is empty
27        System.out.printf("Name Map Tests: size is %d and is empty %b\n", nameMap.size(), nameMap.isEmpty());
28
29        // Use a For Loop to loop over the Keys to retrieve each Map Value
30        for (Map.Entry<String, String> m : nameMap.entrySet())
31        {
32            System.out.printf("Key: %s Value: %s\n", m.getKey(), m.getValue());
33        }
34
35        integerMap.clear();
36        stringMap.remove();
37        stringMap.clear();
38        nameMap.clear();
39    }
40 }
```



- f. Remove all elements for each of the Maps.
- g. Run the application.
- h. Take a screenshot of the console of the output.
- i. Generate the JavaDoc for all classes.

4. Using a Queue:

- a. Create a new class named *PlayQueue* class in the *app* package with a *main()*.
- b. Create a Queue of Integers. Add 5 numbers to the Queue.
- c. Create a Queue of Strings. Add 5 strings to the Queue.
- d. Print the size and if head element for each Queue to the console.
- e. Print the Integer Queue using *toString()* to the console.
- f. Print the String Map using a while loop to the console.
- g. Run the application.
- h. Take a screenshot of the console of the output.
- i. Generate the JavaDoc for all classes.

```
1 import java.util.Iterator;
2 import java.util.LinkedList;
3 import java.util.Queue;
4
5 public class PlayQueue
6 {
7     /**
8      * Exercise a Queue (a FIFO):
9      * offer() - Inserts an element into the head of the Queue
10     * add() - Inserts an element into the head of the Queue
11     * peek() - retrieves but does not remove from the head element of the Queue
12     * remove() - retrieves and removes an element from the head of the Queue
13     * size() - returns the number of elements in the Queue
14     */
15     @param args
16
17     public static void main(String[] args)
18     {
19         // Set up a Queue of Strings
20         Queue<String> stringQueue = new LinkedList<String>();
21         stringQueue.offer("Mark Reha");
22         stringQueue.add("Mary Reha");
23         stringQueue.offer("Justine Reha");
24         stringQueue.add("Brianna Reha");
25
26         // Setup a Queue of Integers
27         Queue<Integer> integerQueue = new LinkedList<Integer>();
28         integerQueue.add(1);
29         integerQueue.offer(-1);
30         integerQueue.add(5);
31         integerQueue.offer(10);
32
33         // Print out size and head element of the Queue
34         System.out.println(integerQueue);
35         System.out.printf("Integer Queue Tests: size is %d and head element is %d\n",
36             integerQueue.size(), integerQueue.peek());
37
38         // Use Iterator to get elements from the Queue, could of used loop over size() and
39         // used remove() to achieve the same functionality
40         Iterator<String> itr = stringQueue.iterator();
41         while(itr.hasNext())
42         {
43             System.out.println(itr.next());
44         }
45     }
46 }
```

5. Using a Stack:

- a. Create a new class named *PlayStack* class in the *app* package with a *main()*.
- b. Create a Stack of Integers. Push 5 numbers to the Stack.
- c. Create a Stack of Strings. Push 5 strings to the Stack.
- d. Print the size and if 2nd element for each Stack to the console.
- e. Print the Integer Stack using *toString()* to the console.
- f. Print the String Stack using a while loop to the console.
- g. Run the application.
- h. Take a screenshot of the console of the output.
- i. Generate the JavaDoc for all classes.

```
1 import java.util.Iterator;
2 import java.util.Stack;
3
4 public class PlayStack
5 {
6     /**
7      * Exercise a Stack (a LIFO):
8      * push() - Inserts an element into the top from the Stack
9      * peek() - retrieves but does not remove the top element from the Stack
10     * pop() - retrieves and removes the top element from the Stack
11     * size() - returns the number of elements in the Stack
12     */
13     @param args
14
15     public static void main(String[] args)
16     {
17         // Set up a Stack of Strings
18         Stack<String> stringStack = new Stack<String>();
19         stringStack.push("Mark Reha");
20         stringStack.push("Mary Reha");
21         stringStack.push("Justine Reha");
22         stringStack.push("Brianna Reha");
23
24         // Set up a Stack of Integers
25         Stack<Integer> integerStack = new Stack<Integer>();
26         integerStack.push(1);
27         integerStack.push(-1);
28         integerStack.push(5);
29         integerStack.push(10);
30
31         // Print out size and second element in the Stack
32         System.out.println(integerStack);
33         System.out.printf("Integer Stack Tests: size is %d and 2nd element is %d\n", integerStack.size(),
34             integerStack.elementAt(1));
35
36         // Use Iterator to get elements from the Stack, could of used loop over size() and used pop() to
37         // achieve the same functionality
38         Iterator<String> itr = stringStack.iterator();
39         while(itr.hasNext())
40         {
41             System.out.println(stringStack.pop());
42         }
43     }
44 }
```



GRAND CANYON
UNIVERSITY™

6. Tutorials and Quiz

- Go the Java Collections Tutorial at <https://www.javatpoint.com/collections-in-java>.
- Review all the tutorials.
- Complete Collection Quiz-1. Take a screenshot of your completed Quiz.

Deliverables:

The following need to be submitted as this part of the activity:

- a. All screenshots of application in operation.
- b. ZIP file of the code in the project folder. Include the JavaDoc generated for the project.



Research Questions

1. Research Questions: Online students will address these in the Discussion Forum and traditional on ground students will address them in this assignment.
 - a. Create a Java project that uses an ArrayList and a LinkedList. Show the most appropriate choice for inserting and deleting elements at the beginning of the list. Summarize your answers and explanation for how your code examples work in 300 words.
 - b. Create a Java project that shows the benefits of using generic types besides the examples from the activity. Summarize your answers and explanation for how your code examples work in 300 words.

Final Activity Submission

1. In a Microsoft Word document, complete the following for the Activity Report:
 - a. Cover sheet with the name of this assignment, date, and your name.
 - b. Section with a title that contains all the diagrams, screenshots, and theory of operation write-ups.
 - c. Zip file with all code and generated JavaDoc documentation files.
 - d. Section with a title that contains the answers to the Research Questions (traditional ground students only).
2. Submit the Activity Report and zip file of the code and documentation to the Learning Management System (LMS).