# CST-239 Activity 5 Guide

## Contents



## 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.

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 }
```

   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.

© 2020. Grand Canyon University. All Rights Reserved.

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

```
public class MyArray
{
    public <E> void printArray(E[] inputArray)
    {
        // Iterate over the array and print each element
        for(E element : inputArray)
        {
            System.out.printf("%s ", element);
        }
    }

    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' };

        // Print each array out
        MyArray ma = new MyArray();
        System.out.println("Array integerArray contains:");
        ma.printArray(intArray);   // pass an Integer array
        System.out.println("\nArray doubleArray contains:");
        ma.printArray(doubleArray);   // pass a Double array
        System.out.println("\nArray characterArray contains:");
        ma.printArray(charArray);   // pass a Character array
    }
}
```

```
public class MyNumbersArray
{
    public <E extends Number> void printArray(E[] inputArray)
    {
        // Iterate over the array and print each element
        for(E element : inputArray)
        {
            System.out.printf("%s ", element);
        }
    }

    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 };
        Float[] floatArray = { 0.0f, 1.0f, 2.5f, 3.5f };

        // Print each array out
        MyNumbersArray ma = new MyNumbersArray();
        System.out.println("Array integerArray contains:");
        ma.printArray(intArray);   // pass an Integer array
        System.out.println("\nArray doubleArray contains:");
        ma.printArray(doubleArray);   // pass a Double array
        System.out.println("\nArray floatArray contains:");
        ma.printArray(floatArray);   // pass a Float array
    }
}
```

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.

© 2020. Grand Canyon University. All Rights Reserved.

GRAND CANYON UNIVERSITY

# 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.

PlayList.java

```
1
2
3  import java.util.ArrayList;
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 }
```

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.

PlayList.java | *PlayMap.java

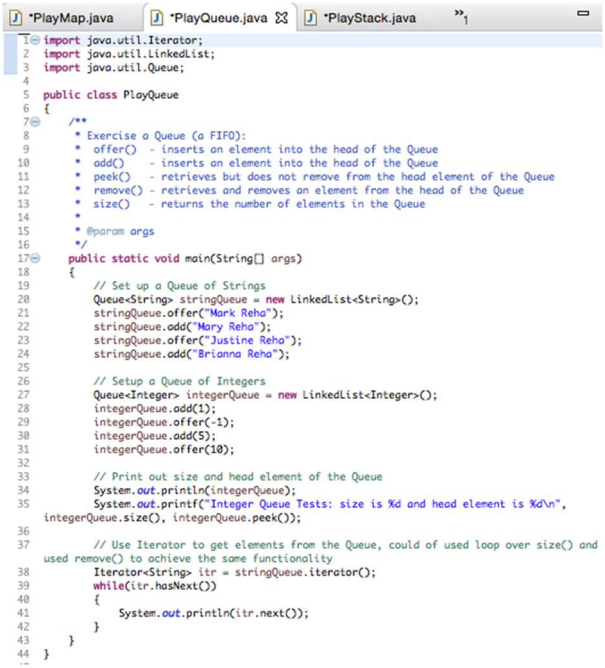
```
1  import java.util.HashMap;
2  import java.util.Map;
3
4  public class PlayMap
5  {
6      public static void main(String[] args)
7      {
8          // Create a Map of Integers keyed by Integers
9          Map<Integer, Integer> integerMap = new HashMap<Integer, Integer>();
10         integerMap.put(1,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

© 2020. Grand Canyon University. All Rights Reserved.

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.

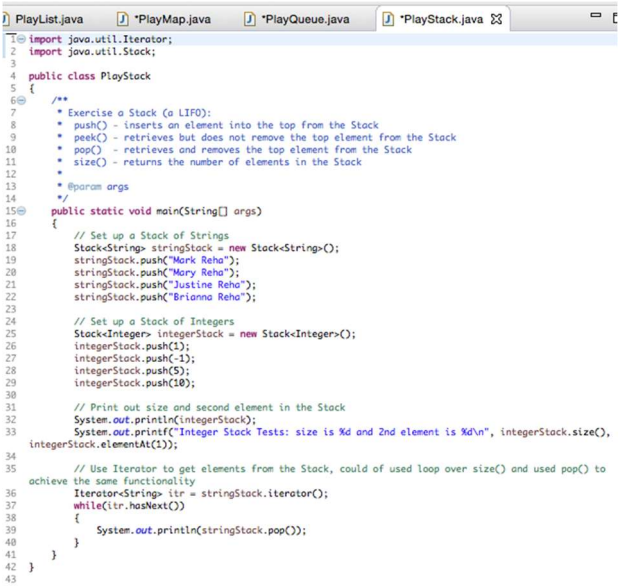
4. Using a Queue:

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.




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.




© 2020. Grand Canyon University. All Rights Reserved.

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.

© 2020. Grand Canyon University. All Rights Reserved.

GRAND CANYON UNIVERSITY™

**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).

© 2020. Grand Canyon University. All Rights Reserved.