Assignments on Java Generics

1. Write a Java Program to demonstrate a Generic Class.
Write a Java Program to demonstrate Generic Methods.

```
class Box<T>
{
	T item ;

	void setItem(T item)
	{
		this.item=item;
	}
	T getItem()
	{
		return item;
	}
}
class BoxDemo
{
	public static void main(String[] args)
	{
		Box<Integer>b1 = new Box<Integer>();
		Box<String>b2 = new Box<String>();

		b1.setItem(10);
		b2.setItem("hii");

		System.out.println("Values: "+b1.getItem());
		System.out.println("Values: "+b2.getItem());
	}
}
```

2 Write a Java Program to demonstrate Wildcards in Java Generics.

Program that demonstrates Upper Bounded Wildcards

```
import java.util.Arrays;
import java.util.List;
class UpperBoundedWildcardDemo
{
	public static void main(String[] args)
	{
		//Upper Bounded Integer List
		List<Integer> list1= Arrays.asList(4,5,6,7);

		//printing the sum of elements in list1
		System.out.println("Total sum is:"+sum(list1));

		//Upper Bounded Double list
		List<Double> list2=Arrays.asList(4.1,5.1,6.1);

		//printing the sum of elements in list2
		System.out.print("Total sum is:"+sum(list2));
	}

private static double sum(List<? extends Number> list)
	{
		double sum=0.0;
		for (Number i: list)
		{
			sum+=i.doubleValue();
		}

		return sum;
	}
}
```

Program that demonstrates Lower Bounded Wildcards

```
import java.util.Arrays;
import java.util.List;

class LowerBoundedWildcardDemo
{
	public static void main(String[] args)
	{
```

```
		//Lower Bounded Integer List
		List<Integer> list1 = Arrays.asList(1,2,3,4);

		//Integer list object is being passed
		print(list1);

		//Lower Bounded Number list
		List<Number> list2 = Arrays.asList(1,2,3,4);

		//Integer list object is being passed
		print(list2);
	}

	public static void print(List<? super Integer> list)
	{
		System.out.println(list);
	}
}
```

Program that demonstrates UnBounded Wildcards

```
import java.util.Arrays;
import java.util.List;

class UnboundedWildcardDemo {

public static void main(String[] args)
   {

	// Integer List
	 List<Integer> list1 = Arrays.asList(1, 2, 3);

	 // Double list
	 List<String> list2 = Arrays.asList("s","d","f");

	printlist(list1);
	printlist(list2);
   }

   private static void printlist(List<?> list)
```

```
    {

        System.out.println(list);
    }
}
```

2. **Assignments on List Interface**

1. Write a Java program to create List containing list of items of type String and use for-each loop to print the items of the list.

```
import java.util.*;

public class ArrayListIteration
{
        public static void main(String args[])
        {
                List<String> al= new ArrayList<String>();

                al.add("Ann");
                al.add("Bill");
                al.add("Cathy");

                // Using the Get method and the for loop

                for (int i = 0; i < al.size(); i++)
                {
                        System.out.print(al.get(i) + " ");
                }

                System.out.println();

                // Using the for each loop
                for (String str : al)
                System.out.print(str + " ");
        }
}
```

2. Write a Java program to create List containing list of items and use ListIterator interface to print items present in the list. Also print the list in reverse/ backword

direction.

```
import java.util.*;

public class ListIterators
{
	public static void main(String[] args)
	{

		List<String> names = new LinkedList<>();

		 names.add("First");
		 names.add("Middle");
		 names.add("Last");

		// Getting ListIterator
		ListIterator<String> listIterator= names.listIterator();

		 // Traversing elements
		 System.out.println("Forward Direction Iteration:");

		while (listIterator.hasNext())
		{
			System.out.println(listIterator.next());
		}

		// Traversing elements, the iterator is at the end
		// at this point

		System.out.println("Backward Direction Iteration:");

		while (listIterator.hasPrevious())
		{
			System.out.println(listIterator.previous());
		}
	}
}
```

3. **Assignments on Set Interface**

1. Write a Java program to create a Set containing list of items of type String and print
the items in the list using Iterator interface. Also print the list in reverse/ backword direction.

```
import java.util.*;

public class SetTest
{
	public static void main(String[] args)
	{
		// Declaring object of type String
		Set<String> hashSet = new HashSet<String>();

		// Adding elements to the Set
		// using add() method
		hashSet.add("Car");
		hashSet.add("Bike");
		hashSet.add("Train");
		hashSet.add("Truck");
		hashSet.add("Helicopter");

		//using iterator interface
		Iterator<String>  hashIterator= hashSet.iterator();
		while (hashIterator.hasNext())
		{
			// Returns the next element.
			System.out.println(hashIterator.next());
		}

		// Printing elements of HashSet object
		System.out.println("\n"+hashSet);

		//convert the hashset into an array List
		ArrayList<String> a1 = new ArrayList<String>(hashSet);

		//using Listiterator to traverse through the arrayList
		ListIterator<String> Listiterator = a1.listIterator();

		//Reverse the ArrayList
```

```
                System.out.println("\nItems in reverse order: ");
                Collections.reverse(a1);

                while(Listiterator.hasNext())
                {
                        System.out.println(Listiterator.next());
                }
        }
}
```

2. Write a Java program using Set interface containing list of items and perform the following operations:
a. Add items in the set.
b. Insert items of one set in to other set.
c. Remove items from the set
d. Search the specified item in the set

```
import java.util.*;
public class SetApp
{
        public static void main(String args[])
        {
                HashSet <String> one = new HashSet <String>();
                HashSet <String> two = new HashSet <String>();

                //1. Add item in set
                one.add("English");
                one.add("Maths");
                two.add("Science");
                two.add("History");
                two.add("Geography");

                System.out.println("Set one"+ one);
                System.out.println("Set two: "+ two);

                //2. Add items from one set to another set
                one.addAll(two);
                System.out.println("Updated Set one: "+one);

                //3. Remove item from set
```

```
            one.remove("English");
            System.out.println("Set after removing "+ "English: " + one);

            //4. Search particular item from set
            System.out.println("Set contains History: "+one.contains("History"));
}
}
```

4. **Assignments on Map Interface**

Write a Java program using Map interface containing list of items having keys and associated values and perform the following operations:

a. Add items in the map.
b. Remove items from the map
c. Search specific key from the map
d. Get value of the specified key
e. Insert map elements of one map in to other map.
f. Print all keys and values of the map.

```
import java.util.*;

public class HashMapApp
{
    public static void main(String[] args)
    {
    HashMap<Integer, String> hashmap = new HashMap<Integer, String>();

    //1.  Mapping string values to int keys
    hashmap.put(10, "Angel");
    hashmap.put(30, "Liza");
    hashmap.put(20, "Steve");

    //2.  Displaying the HashMap=>
    for (Map.Entry<Integer, String> ViewMap : hashmap.entrySet())
    {
        System.out.println(ViewMap.getKey() + " " + ViewMap.getValue());
    }

    //3.  Removing the existing key mapping
    String RemovedValue = (String)hashmap.remove(20);
```

```
        //4. Verifying the returned value
        System.out.println("\nRemoved value is: "+ RemovedValue);

        //5. Displaying the new map
        System.out.println("New map is: "+ hashmap);

        //6. Search for a specific key
System.out.println("\nHashmap contains key 10: "+hashmap.containsKey(10));
System.out.println("\nHashmap contains key 60: "+hashmap.containsKey(60));

        //7. Get value of specified key
        System.out.println("\nThe Value for key 30 is: " + hashmap.get(30));

        //8. Insert map elements of one map in to other map.
        HashMap<Integer, String> secondmap = new HashMap<Integer, String>();
        hashmap.put(40, "Betty");

        //9. Print all keys and values of the map
        secondmap.putAll(hashmap);
        System.out.println("\nSecond map is: "+ secondmap);
}
}
```

5. **Assignments on Lambda Expression**

1. Write a Java program using Lambda Expression to print ”Hello World”.

```
//Functional Interface
interface MyFunctionalInterface
{
          //A method with single parameter
         public void say(String str);
}

public class SingleParamLambda
{
        public static void main(String args[])
         {
                // lambda expression
                MyFunctionalInterface msg = (str) ->
                {
```

```
                        System.out.println(str);
                    };
                msg.say("Hello World");
        }
}
```

2. Write a Java program using Lambda Expression with single parameters.

```
//Functional Interface
interface MyFunctionalInterface
{
          //A method with single parameter
          public void say(String str);
}

public class SingleParamLambda
{
        public static void main(String args[])
         {
                  // lambda expression
                  MyFunctionalInterface msg = (str) ->
                  {
                                System.out.println(str);
                   };
                msg.say("Hello World");
        }
}
```

3. Write a Java program using Lambda Expression with multiple parameters to add two numbers.

```
interface FunctionalInt
{
        int operation(int a,int b);
}
class LambdaDemo
{       public static void main(String[] args)
        {
                FunctionalInt addImpl=(x,y)->{return x+y;};
                System.out.println("ADDITION: "+addImpl.operation(4,5));
```

```
        }
}
```

4. Write a Java program using Lambda Expression to calculate the following:

a. Convert Fahrenheit to Celcius

```
interface FahrenToCels
{
        float InCelcius(float Fahrenheit );
}

public class FahrenheitToCelsius
{
       public static void main(String[] args)
        {
        FahrenToCels Convert= (Fahrenheit)-> ((Fahrenheit-32)*5)/9  ;
        System.out.println("Temperature in Celsius is: "+Convert.InCelcius(54));
        }
}
```

b. Convert Kilometers to Miles.

```
interface KmToMiles
{
        double InMiles(double Kilometers);
}

public class KilometerToMiles {

        public static void main(String[] args)
        {

                KmToMiles Convert=(Kilometers)-> Kilometers / 1.6;
               System.out.println( "Kilometer In Miles is: a" +Convert.InMiles(10));
        }
}
```

5. Write a Java program using Lambda Expression with or without return keyword.

```
interface Addable
{
   int add(int a,int b);
}

public class  ReturnValue
 {
      public static void main(String[] args)
      {

      // Lambda expression without return keyword.
      Addable ad1=(a,b)->(a+b);
      System.out.println(ad1.add(10,20));

      // Lambda expression with return keyword.
      Addable ad2=(int a,int b)->
      {
                  return (a+b);
      };

      System.out.println(ad2.add(100,200));
   }
}
```

6. Write a Java program using Lambda Expression to concatenate two strings.

```
//Functional Interface
interface MyFunctionalInterface
{
   //A method with single parameter
   public void say(String str1, String str2);
}
public class ConcateString
{
         public static void main(String args[])
          {
                  // lambda expression
```

```
                MyFunctionalInterface msg = (str1, str2) ->
                 {
                                System.out.println(str1 +" "+ str2);
                  };
                 msg.say("Hello", "Java");
        }
}
```