GENERIC PROGRAMMING

TUTORIAL

Q1

```
class Box<T> {
    private T t;

    public void add(T t) {
        this.t = t;
    }

    public T getT() {
        return t;
    }
}
```

Given this class, write a function to create an Integer type and String type object of Box. Initialize the objects with suitable values using the add method, and print them using the getT method of Box class.

Solution:

```
public static void main(String[] args) {
    Box<Integer> integerBox = new Box<Integer>();
    Box<String> stringBox = new Box<String>();

    integerBox.add(new Integer(10));
    stringBox.add(new String("Hello World"));

    System.out.printf("Integer Value :%d\n", integerBox.getT());
    System.out.printf("String Value :%s\n", stringBox.getT());
}
```

Note that 10, and “Hello World” are only example input values.

Q2 Pair of Generics

```
class Box<T, S> {
    private T t;
    private S s;

    public void add(T t, S s) {
        this.t = t;
        this.s = s;
    }

    public T getFirst() {
        return t;
    }

    public S getSecond() {
        return s;
    }
}
```

The Box class above has been modified to accommodate a pair of generic data types. Now write a method to create an object where T is an Integer, and S is List<Integer>. Now in this method, create a List<Integer> of size 10, where each element is a random number between 1 and 8. Take the product of all elements in the list, and store it in a variable. Now initialize the box object with the product as the integer argument, and the list being passed as the second argument. Print these using the getFirst(), and getSecond() methods.

Solution:

```
public static void main(String[] args) {
    Random rand = new Random();
   Box<Integer, List<Integer>> box
      = new Box<Integer, List<Integer>>();

   List<Integer> messages = new ArrayList<Integer>();

   for(int i=1;i<=10;i++){
    messages.add(1+rand.nextInt(bound: 8));
   }
   int prod = 1;
   for(int i=0;i<10;i++){
    prod*= messages.get(i);
   }
   box.add(Integer.valueOf(prod),messages);
   System.out.printf(format: "Integer Value :%d\n", box.getFirst());
   System.out.printf(format: "List Values :%s\n", box.getSecond());


}
```

Note: Output will be different because of random values.

Q3

```java
import java.util.*;

public class Classroom {

   public static double sum(List<? extends Number> numberlist) {
      double sum = 0.0;
      for (Number n : numberlist){
        sum+= n.doubleValue();
      }
      return sum;
   }

   Run | Debug
   public static void main(String args[]) {
      List<Integer> integerList = Arrays.asList(...a: 1, 2, 3);
      System.out.println("sum = " + sum(integerList));

      List<Double> doubleList = Arrays.asList(...a: 1.2, 2.3, 3.5);
      System.out.println("sum = " + sum(doubleList));
   }
}
```

a) Run this code, and identify where a generic type has been used.
b) Name some inbuilt classes that the generic type will accept as arguments.
c) Why has the primitive data type “double” been used for the sum variable, and not int, float, etc.? Can you use a generic operator in place of double?

Solution:

a) The ? represents the generic type. It is called the wildcard generic operator.
b) Some inbuilt classes that are accepted are Double, Integer, Float, BigInteger, Byte, etc.
c) The data type double is sort of all-encompassing when it comes to numeric data types (it can store all types of information). For example, you can store an integer value in double, but not vice-versa. No, a generic operator cannot be used in place of double. When implementing a functionality for a data structure(like taking sum), we have to specify a concrete data type. The generic types work well in blueprint definitions like classes (as seen in the Box class above), as they allow a variety of objects to be created,(an object of a Class can be instantiated using String, as well Integer, or other classes).

Note: The wildcard generic operator allows you to restrict the types of Objects you want to create. In the above function, it ensures that a list of numbers is passed (integers or decimals). In this case, you CANNOT use an alphabet like T/S to restrict object types. Try replacing the ? in the above function, and you will see an error.

Some Things to be kept in mind while using Generic Types:

1) It does NOT allow for primitive data types. For example, if you want to create an integer object, use Integer to specify it, using int will give an error.

2) Generic types CANNOT be declared static. For example, private static T t; is NOT allowed. This is because a static variable is common across objects. So if you want T to represent both String and Integer, the ambiguity for variable t will not be resolved(whether it is Integer or String).

3) You CANNOT use the instanceof operator with such objects. If you declare a class with a generic variable T, and create an object like Box<Integer> integerBox = new Box<Integer>();, then an integerBox instanceof Box<Integer> will give error.

4) Generic arrays are NOT allowed. This means, Box<Integer> [] arrayOfLists = new Box<Integer>[2]; does NOT work.