WEEK 10 : 24/11/2020 Mallika Prasad 1BM19CS081

LP #7 : WAP to demonstrate generics with multiple object parameters

```
class TwoGen<T, V> {
T ob1 ; V ob2;
TwoGen (T o1 , V o2) {
  ob1 = o1; ob2 = o2;
 }
void showTypes () {
System.out.println ("Type of T is " + ob1.getClass().getName());
System.out.println ("Type of V is " + ob2.getClass().getName());
}
T getob1 () { return ob1; }
```

```
V getob2 { return ob2; }
}

class SimpgenMain {
public static void main (String args[]) {
TwoGen<Integer, String> tgObj = new TwoGen<Integer, String>
(88, "Generics");
tgObj.showTypes();
int v = tgObj.getob1();
System.out.println("value : " +v);
String str = tgObj.getob2();
System.out.println("value: " + str); }
}
```

LP#8: WAP that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge() when the input age > father's age.

```
import java.util.Scanner;
class WrongAge extends Exception {
public WrongAge extends Exception {
public WrongAge(String s) {
    super(s); }
  }
```

```
class Father {
 int fatherAge;

Father(int fAge, int sAge) throws WrongAge {
 if (fAge <= sAge) { throw new WrongAge ("[redacted] Son's age
 can't be greater than or equal to the father's age "); }
 else {
  this.fatherAge = fAge; }
    }
}

class Son extends Father {
 int sonAge;
 Son (int fAge, int sAge) throws WrongAge {
   super (fAge, sAge);
   this.sonAge = sAge;
  }

 void print () {
 System.out.println("Father's age: " + fatherAge);
 System.out.println("Son's age: " + sonage);
   }
}

public class Main {
 public static void main (String [] args) {
 int fAge, sAge;
```

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter father's age");
fAge = sc.nextInt();
System.out.println("Enter son's age");
sAge = sc.nextInt();
try {
Son son = new Son(fAge, sAge);
son.print(); }
catch (WrongAge err) {
System.out.println("Exception" + err); }
}
}
```