week 7 (IV)
24/11/2020

DOMS Page No. Date / /

1. write a program to demonstrate generics with multiple object parameters.

```
class GENERICSC<F, S>
{
  F object1;
  S object2;

  GENERICSC(F O1, S O2)
  {
    object1 = O1;
    object2 = O2;
  }
  void printName()
  {
   System.out.println("Type of object1 is "
+ object1.getclass().getName());
  System.out.println("Type of object2 is "
+ object2.getclass().getName());
  }
   F getob1()
  {
   return object1;
  }
   S getob2()
   {
    return object2;
   }
  }
   public class DGenerics
  { GENERICSC<Float, String> G1 = new
    GENERICSC<Float, String>(10f, "CGPA");
    G1.printname();
    float FL = G1.getob1();
```

```
System.out.println("The number given to
object 1 is" +FL);
String ST = h1.getOb2();
System.out.println("The name given to object
P3" +ST);
}
}
```

Week 8 (IV)
24/11/2020

① write a program that demonstrate handling of exceptions in inheritance tree. create a base class called "Father" and derived class called "Son" which extends base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge() when the input age<0, In Son class implement a constructor that takes both father and Son's age and throws an exception if Son's age is >= father's age.

```
Class Father
{
 Static void AcceptName (int InputAge)
  throws ArithmeticException
 {
  try
  {
   if (InputAge < 0)
   throw new ArithmeticException("wrong
    Age");
  }
```

```
catch (ArithmeticException e)
{
System.out.println("caught Exception "+e);
}
}
}
Class Son extends Father
{
Static void checkFage(int S-Age, int F-Age)
throws ArithmaticException
{
try {
if (S-Age >= F-Age)
throw new ArithmaticException("Son cannot be
elder than father");
System.out.println("the Son's age is "+S-
Age + "The fathers age is "+F-Age);
}
catch (Arithmatic exception e){
System.out.println("F caught"+e);
}
}
}
public class ExceptionHandling{
public static void main (String args[]){
Father.acceptName(-10);
Son.checkFage(30, 20);
}
}
=
```