Write a program to demonstrate generics with multiple Object parameter

```
Class Test <X, Y, Z, W) {
  X projects;
  Y name;
  Z exp;
  W languages;

 Test (X projects, Y name, Z exp,
          W languages) {

  this.projects = projects;
  this.name = name;
  this.exp = exp;
  this.languages = languages;
 }

 public X getprojects () {
   return this.projects;
 }

 public Y getName() {
  return this.name;
 }

 public Z getExp () {
   return this.exp;
 }

  public ~~W~~ getLang() {
     return this.languages;
  }
```

```
public void print () {
    System.out.println ("Number of
project: " + getproject () + "\n Developer Name:
" + getName() +" \n Industry experience:" +
getExp() + " \n Known programming languages
: " + getLang());
  }
}

class generics {
  public static void main (String[] args) {
    Test <Integer, String, Integer, Integer>
    t= new Test <Integer, String, Integer,
      Integer > (10, "Vineeth", 11, 12);
    t.print ();
  }
}
```

Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" & derived class called "Son" which extends the base class.

```
import java.lang.*;
import java.util.Scanner;

class WrongAge extends Exception
{
    public WrongAge (String msg) {
        Super (msg);
    }
}

class Father
{
    private int age;

    Father (int age) {
        this.age = age;
    }

    public int getAge () {
        return this.age;
    }
}

class Son extends Father
{
    private int sAge;

    Son (int fAge, int sAge) {
        Super (fAge);
        this.sAge = sAge;
    }
    public int getSAge() {
        return this.sAge;
    }

    public void disp() throws WrongAge {
     if (Super.getAge() <= getSAge() ||
     Super.getAge() < 0 || getSAge() < 0)
        throw new wrongAge("please enter
            a vaid age value.");
       else
        System.out.println("Father's age: " +
         Super.getAge() + "\n son's age: " +
            this.getSAge());
      }
    }

  class ExceptionalHanding {
     public static void main(String args[]) {
       int dad= 1, son=1;
       Scanner s= new Scanner(System.in);

      try {
        dad = s.nextInt();
        son = s.nextInt();
        son S1= new Son(dad, son);

        S1.disp();
      }
      catch (WrongAge e) {
         e.printStackTrace();
      }
    }
}
```