

Program 7:

Write a java program that demonstrates exceptional handling of inheritance tree. class "father" & derived class "son".

```
-> class wrongAge extends Exception {  
    public wrongAge() {  
        super("age cannot be negative");  
    }  
}  
  
class input extends Exception {  
    public input() {  
        super("wrong input");  
    }  
}  
  
class Father {  
    public int age;  
    Father(int age) throws wrongAge {  
        if (age > 0) {  
            throw new wrongAge();  
        }  
        this.age = age;  
    }  
}  
  
class Son extends Father {  
    int s-age;  
    Son(int f-age, int s-age) throws wrongAge, input {  
        super(f-age);  
        if (f-age < 0 && s-age < 0) {  
            throw new wrongAge();  
        }  
        else if (f-age <= s-age) {  
            throw new input();  
        }  
        this.s-age = s-age;  
    }  
}  
  
public class J {  
    public static void main(String[] args) {  
        try {
```



```

try {
    Father f = new Father(40);
    System.out.println("Father's age: " + f.age);
    Son s = new Son(40, 50);
    System.out.println("Son's age: " + s.sage);
} catch (WrongAge e) {
    System.out.println(e.toString());
}
catch (Input ae) {
    System.out.println(ae.toString());
}
}
}
}

```

Algorithm:

Step 1: Start

Step 2: Initialize variable f-age, s-age

Step 3: Create user defined exception
class WrongAge extends Exception {
 public WrongAge() {
 super("Age cannot be negative")
 }
}

Step 4: Create another user defined exception
class Input extends Exception {
 public Input() {
 super("Wrong input")
 }
}

Step 5: Create class Father {
 public int age;
 Father(int age) throws WrongAge {
 if (age > 0) {
 throw new exception WrongAge();
 }
 this.age = age;
 }
}

Step 6: Create class Student extends Father {
 int s-age
 Son(int f-age, int s-age) throws WrongAge, Input {
 super(f-age);
 }
}


```

    if (t-age < 0 & s-age < 0) {
        System.out.println new wrongAge();
    }
    else if (t-age <= s-age) {
        System.out.println (throw new input());
    }
    this.s-age = s-age;
}

```

write a program which create 2 thread where one is executed every 10 second & another for 2 second.

```

class One extends Thread {
    public void run() {
        int i=0;
        while (i < 2) {
            i++;
            try {
                System.out.println ("BMS College of Engineering");
                Thread.sleep(10000);
            } catch (Exception e) {
                System.out.println (e.toString());
            }
        }
    }
}

```

```

class Two extends Thread {
    public void run() {
        int i=0;
        while (i < 2) {
            i++;
            try {
                System.out.println ("CSE");
                Thread.sleep(2000);
            } catch (Exception e) {
                System.out.println (e.toString());
            }
        }
    }
}

```


public class I {

public static void main(String[] args) {

One t1 = new One();

Two t2 = new Two();

t1.start();

t2.start();

}

}

algorithm:

step 1: Start

Step 2: Initialize variable i = 0

Step 3: Construct a class One & Two which extends Thread.

Step 4: Under class One create method run(), under which, while (i < 5) {

i++
try {

System.out.println("BMS College of Engineering");

Keep Thread in sleep(1000)

} catch (Exception e) {

System.out.println(e.toString());

}

}

Step 5: Under class Two create method run() under which, while (i < 5) {

i++;

try {

System.out.print/print("GE")

} catch (Exception e) {

System.out.println(e.toString());

}

}

Step 6: Create a main class

Call class One t1 = new One();

Call class Two t2 = new Two();

t1.start();

t2.start();

Step 7: stop.

Output:

7) Continuation of ~~the~~ per algorithm of program 7

Step 7: call main class J

try calling class, Father f = new Father(40);

~~son s = new Son(16, 40);~~
age: 40, age)

call class, son s = new Son(16, 40);

System.out.println("son's age: " + s.age)

} catch (Exception e) {

~~System.out.println(e.toString());~~

} catch (Exception e) {

catch (WrongAge w) {

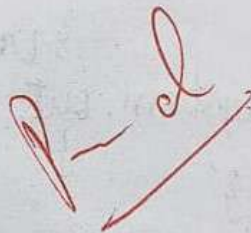
System.out.println(w.toString());

} catch (Input i) {

System.out.println(i.toString());

}

Step 8: stop.



Output:

- (i) ~~Father's~~ input: wrong input
- (ii) wrongAge: Age cannot be negative
- (iii) Father's Age: 40
son's age: 16

8) Output:

BMS College of Engineering

CSE

CSE

~~BMS College of Engineering~~

16.02.24

```
D:\java\oops>javac J.java
```

```
D:\java\oops>java J
```

```
Program 7
```

```
Name: Aditya Dinesh Netrakar
```

```
USN: 1BM22CS017
```

```
input: Wrong input
```

```
D:\java\oops>|
```

```
D:\java\oops>javac I.java
```

```
D:\java\oops>java I
```

```
Program 8
```

```
Name: Aditya Dinesh Netrakar
```

```
USN: 1BM22CS017
```

```
BMS College of Engineering
```

```
CSE
```

```
CSE
```

```
BMS College of Engineering
```

```
D:\java\oops>|
```