Control flow of Exception Handling concept

Control flow in try catch finally

```
try{
 statement-1
 statement-2
 statement-3

}catch(Exception e){
 statement-4

}finally{
 statement-5
}
 statement-6
```

Case1: If there is no exception.

Case2: If an exception is raised at statement2 and the corresponding catch block is matched.

Case3: If an exception is raised at statement2 and corresponding catch block is not matched

Case4 : If an exception is raised at statement4

Case5 : If an exception is raised at statement5

Control flow in Nested try-catch-finally

```
try{
 stmt-1
 stmt-2
 stmt-3

 try{
     stmt-4;
     stmt-5;
     stmt-6;
 }catch(X e){
  stmt-7;
 }finally{
     stmt-8;
 }
 stmt-9;
}catch(Y e){
 stmt-10;
}finally{
 stmt-11;
}
    stmt-12;
```

Case1 : If there is no exception

Case2 : If an exception is raised at statement2 and the corresponding catch block is matched.

Case3 : If an exception is raised at statement2 and the corresponding catch block is not matched.

Case4 : If an exception is raised at statement5 and corresponding inner catch block is matched

Case5 : If an exception is raised at statement5 and the inner catch has not matched but the outer catch block is matched.

Case6 : If an exception is raised at statement5 and both inner catch and outer catch block is not matched.

Case7 : If an exception is raised at statement7 and corresponding catch block is matched

Case8 : If an exception is raised at statement7 and corresponding catch block is not matched

Case9 : If an exception is raised at statement8 and corresponding catch block is matched.

Case10 : If an exception is raised at statement8 and the corresponding catch block is not matched.

Case11 : If an exception is raised at statement 9 and the corresponding catch block is matched.

Case12 : If an exception is raised at statement 9 and the corresponding catch block is not matched.

Case13 : If an exception is raised at statement 10

Case14 : If an exception is raised at statement 11 or 12

- If we are not entering into a try block then finally the block won't be executed.
- If we are entering into a try block without executing the final block we can't come out.
- We can write try inside try, nested try-catch is possible.
- Specific exceptions can be handled using inner try catch and generalised exceptions can be handled using outer try catch.

Note:

```
public class TestApp{
 public static void main(String... args){
  try{
   System.out.println(10/0);
  }catch(ArithmeticException ae){
   System.out.println(10/0);
  }finally{
   String s=null;
   System.out.println(s.length());
  }
 }
}
```

Default exception handler handles the most recent exception and it can handle only one exception.
RE: java.lang.NullPointerException

Various possible cases of Exception

1.
```
try{
}catch(X e){     //valid
}
```

2.
```
try{
}catch(X e){
                //valid
}catch(Y e){
}
```

3.
```
try{
}catch( X e){
//invalid
}catch( X e){
}
```

4.
```
try{
 }finally{      //valid

}
```

5.
```
try{
}catch(X e){
      //valid
}finally{
}
```

6. `try{}  //invalid`

7. `catch(){}    //invalid`

8. `finally{}   //invalid`

9.
```
try{}
System.out.println("Hello");   //invalid
catch(){}
```

10.
```
try{}
 catch(X e){}
 System.out.println("hello");  //invalid
 catch(Y e){}
```

11.
```
try{}
catch(X e){}
System.out.println("hello");       //invalid
finally{}
```

12.
```
try{}
finally{}
catch(X e){}     // invalid
```

13.
```
try{}
catch(X e){}
try{}
finally{}
```

14.
```
try{}
catch(X e){}
finally{}
finally{}    //invalid
```

15.
```
try{}
catch(X e){
   try{}
     catch(Y e1){}                 //valid
}
```

16.
```
try{}
catch(X e){}
finally{
  try{}
    catch(Y e1){}                  //valid
    finally{}
}
```

17.
```
try{
 try{}                             //invalid
}
```

18.
```
try
 System.out.println("hello");      //invalid
catch(X e){}
```

19.
```
try{}
catch( X e1)
  System.out.println("hello");     //invalid
```

20.
```
try{}
catch( NullPointerException e1){}  //invalid
finally
 System.out.println("Hello");
```