**Practice Exercise**

Create a class MyClass and create three methods myMethod1(), Method2() and Method3().

Invoke Method2() from Method1() and Method3() from Method2().

Write a code that can throw an exception inside myMethod3() and compile:

File file=new File("filename.txt");

Scanner sc=new Scanner(file);

You will get compilation errors now as you are not handling a checked exception FileNotFoundException.

Declare throws over myMethod3() and myMethod2(). You will need to add throws FileNotFoundException on myMethod() as:

public void myMethod3() throws FileNotFoundException {

File file=new File("filename.txt");

Scanner sc=new Scanner(file);

}

Handle the exception in myMethod1() by enclosing the code that can throw exception using a try-catch block:

public void myMethod1(){

try{

myMethod2();

}

catch(FileNotFoundException e)

{

e.printStackTrace();

}

}

Now retry the excercise by throwing an IOException manually using the throw keyword from inside method3 as:

throw new IOException();

**Trace output Exercise**

**Example 1: Finally is always executed.**

What will the below method return if invoked?

public int myMethod(){

  try {

  return 1;

  }

  Catch (Exception e){

  return 2;

  }

  finally{

  return 3;

 }

}

**Result**

Though it might look like returning 1, it will return 3, as finally block is always executed after a try and/or catch blocks. This is a common interview question asked by many to test your understanding of try-catch-finally.

**Example 2: Child exceptions should come before parent exceptions**

Assumption: filename.txt does not exist.

Fact: FileNotFoundException is a child of IOException.

Program:

 try {

      File file = new File("filename.txt");

      Scanner sc = new Scanner(file);

      throw new IOException();

    }

    catch (FileNotFoundException e) {

      System.out.println("FileNotFoundException called!!!");

    }

    catch (IOException e) {

      System.out.println("IOException called!!!");

    }

**Result**

This will print:

FileNotFoundException called!!!

**Example 3: Parent exception can also catch child exception**

Assumption: filename.txt does not exist.

Fact: FileNotFoundException is a child of IOException.

Program:

 try {

      File file = new File("filename.txt");

      Scanner sc = new Scanner(file);

      throw new IOException();

    }

    catch (IOException e) {

      System.out.println("IOException called!!!");

    }

**Result**

This will print:

IOException called!!!