Day 28

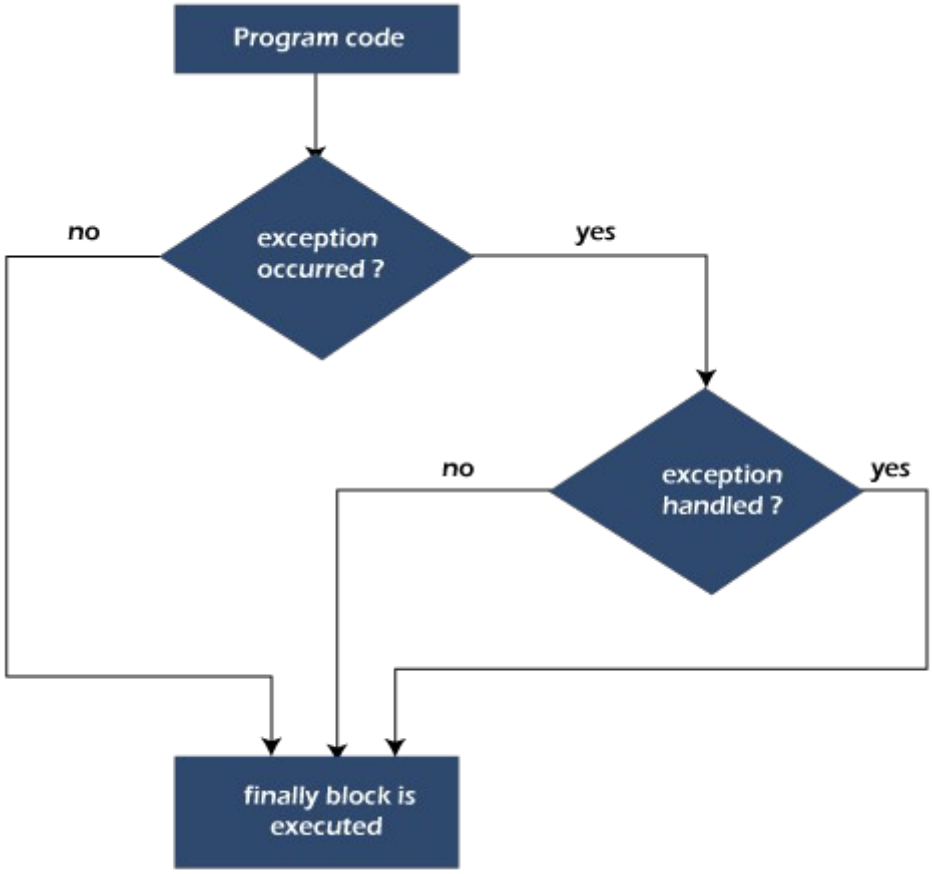
Java finally block

Java finally block is a block used to execute important code such as closing the connection, etc.

Java finally block is always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not.

The finally block follows the try-catch block.

Flowchart of finally block



Why use Java finally block?

finally block in Java can be used to put "cleanup" code such as closing a file, closing connection, etc.

The important statements to be printed can be placed in the finally block.

Usage of Java finally

Let's see the different cases where Java finally block can be used.

x
Case 1: When an exception does not occur

Let's see the below example where the Java program does not throw any exception, and the finally block is executed after the try block.

TestFinallyBlock.java

```
class TestFinallyBlock {
  public static void main(String args[]){
  try{
//below code do not throw any exception
   int data=25/5;
   System.out.println(data);
  }
//catch won't be executed
  catch(NullPointerException e){
System.out.println(e);
}
//executed regardless of exception occurred or not
 finally {
System.out.println("finally block is always executed");
}

System.out.println("rest of phe code...");
  }
}
```

Output:
Java finally block
Case 2: When an exception occurr but not handled by the catch block

Let's see the the fillowing example. Here, the code throws an exception however the catch block cannot handle it. Despite this, the finally block is executed after the try block and then the program terminates abnormally.

TestFinallyBlock1.java

```
public class TestFinallyBlock1{
      public static void main(String args[]){

      try {

        System.out.println("Inside the try block");

        //below code throws divide by zero exception
       int data=25/0;
       System.out.println(data);
```

```
        }
        //cannot handle Arithmetic type exception
        //can only accept Null Pointer type exception
        catch(NullPointerException e){
          System.out.println(e);
        }

        //executes regardless of exception occured or not
        finally {
          System.out.println("finally block is always executed");
        }

        System.out.println("rest of the code...");
        }
      }
```

Output:
Java finally block
Case 3: When an exception occurs and is handled by the catch block

Example:

Let's see the following example where the Java code throws an exception and the catch block handles the exception. Later the finally block is executed after the try-catch block. Further, the rest of the code is also executed normally.

TestFinallyBlock2.java

```
public class TestFinallyBlock2{
    public static void main(String args[]){

    try {

      System.out.println("Inside try block");

      //below code throws divide by zero exception
     int data=25/0;
     System.out.println(data);
    }

    //handles the Arithmetic Exception / Divide by zero exception
    catch(ArithmeticException e){
      System.out.println("Exception handled");
      System.out.println(e);
    }

    //executes regardless of exception occured or not
    finally {
      System.out.println("finally block is always executed");
    }
```

```
    System.out.println("rest of the code...");
    }
}
```

Output:
Java finally block
Rule: For each try block there can be zero or more catch blocks, but only one finally block.
Note: The finally block will not be executed if the program exits (either by calling System.exit() or by causing a fatal error that causes the process to abort).