Finally Block

A “finally” is a keyword used to create a block of code that follows a try or catch block.

A finally block contains all the crucial codes such as closing connections, stream, etc that is always executed whether an exception occurs within a try block or not.

When finally block is attached with a *try-catch block*, it is always executed whether the catch block has handled the exception thrown by try block or not.

Case 1: where there is no exception in the try block.

```
public class finallyBlockExample1
{
public static void main(String[] args)
{
  int a = 20, b = 30;
  try
  {
    int sum = a + b;
    System.out.println("Sum: " +sum);
  }
  catch(Exception e)
  {
    System.out.println(e);
  }

 finally
 {
    System.out.println("finally block must be executed");
 }
 System.out.println("Hello Java");
 }
}
```

Case 2: where an exception will occur inside try block and it will be handled by catch block.

```
public class finallyBlockExample2
{
public static void main(String[] args)
{
 int a = 20, b = 0;
 try
 {
    System.out.println("Value of a: " +a);
    System.out.println("Value of b: " +b);
    int div = a/b;
    System.out.println("Division: " +div);
 }
 catch(ArithmeticException ae)
 {
    System.out.println(ae); // prints corresponding exception.
 }

 finally
 {
   System.out.println("Denominator cannot be zero");
 }
 System.out.println("Hello Java");
 }
}
```

Case 3: where an exception will be raised but it is not handled by the corresponding catch block.

```
public class finallyBlockExample3
{
public static void main(String[] args)
{
  int a = 20, b = 0;
```

```
  try
  {
     System.out.println("Value of a: " +a);
     System.out.println("Value of b: " +b);
     int div = a/b;
     System.out.println("Division: " +div);
  }
  catch(NullPointerException npe)
  {
    System.out.println(npe); // prints corresponding exception.
  }

  finally
  {
    System.out.println("Denominator cannot be zero");
  }
  System.out.println("Hello Java");
 }
}
```

Case 4: where an exception occurs in catch block only. In this case, what will be the control flow?

```
public class finallyBlockExample4
{public static void main(String[] args)
{ try
  { System.out.println("111");
    System.out.println("222");
  }
  catch(Exception ae)
  {System.out.println(10/0);
  }
  finally
  { System.out.println("444");
  }
  System.out.println("555");
```

```
 }
}
```

Case 5: Suppose exceptions have raised in both try and catch block. In this case, what will be the control flow?

```
public class finallyBlockExample5
{
public static void main(String[] args) {
try
{
  System.out.println("111");
  System.out.println(20/0);
  System.out.println("222");
 }
catch(Exception ae)
{
  System.out.println(10/0);
}
finally
{
  System.out.println("444");
 }
System.out.println("555");
 }
}
```

Case 6: Suppose an exception occurs in finally block, then what will be control flow?

```
public class finallyBlockExample6 {
public static void main(String[] args) {
try
{
  System.out.println("111");
  System.out.println(20/0);
```

```
  System.out.println("222");
 }
catch(ArithmeticException ae)
{
  System.out.println("333");
 }
finally
{
  System.out.println(10/0); // Exception occurred in finally block.
 }
System.out.println("555");
 }
}
```

Case 7: Suppose exceptions occur in try block as well as finally block but the thrown exception object has not matched with the corresponding catch block. In this case, what will be the output of the program?

```
public class finallyBlockExample7 {
public static void main(String[] args) {
try
{
  System.out.println("111");
  System.out.println(20/0);
  System.out.println("222");
 }
catch(NullPointerException npe)
{
  System.out.println("333");
}
finally
{
  System.out.println(10/0); // Exception occurred in finally block.
 }
System.out.println("555");
```

```
 }
}
```

Return Statement in Try Catch Finally Block

Now, two famous questions arise in the topic “try catch finally block” that

1. Can we define return statement in try block or catch block or finally block in Java?
2. If we return a value in try block or catch block or finally block, what will happen?

Case 1: Return statement in try block but do not have return statement at the end of method

```
package finallyProgram;
public class TryReturnTest1
{
 int m1() // Compile time error.
 {
  try {
     System.out.println("I am in try block");
     return 50;
  }
  catch(Exception e)
  {
    System.out.println("I am in catch block");
  }
 // Here, no return statement at the end of method.
 }
 public static void main(String[] args)
 {
  TryReturnTest1 ft = new TryReturnTest1();
   System.out.println(ft.m1());
```

```
  }
}
```

Case 2: Return statement in try block and end of method.

```
package finallyProgram;
public class TryReturnTest2
{
 int m1()
 {
   try {
      System.out.println("I am in try block");
      return 50;
   }
   catch(Exception e)
   {
      System.out.println("I am in catch block");
    }
  return 20; // return statement at the end of a method.
}
public static void main(String[] args)
{
  TryReturnTest2 ft = new TryReturnTest2();
  System.out.println(ft.m1());
  }
}
```

Case 3: Return statement in try block and end of method but statement after return.

```
package finallyProgram;
public class TryReturnTest3
{
 int m1()
 {
   try {
```

```
      System.out.println("I am in try block");
      return 50;
    }
    catch(Exception e)
    {
      System.out.println("I am in catch block");
    }
  return 20;
  System.out.println("Statement after return"); // Unreachable code.
}
 public static void main(String[] args)
 {
  TryReturnTest3 ft = new TryReturnTest3();
   System.out.println(ft.m1());
  }
}
```

Case 4: Return statement in try block and at end of method but exception occurred in try block.

```
package finallyProgram;
public class TryReturnTest4
{
 int m1()
 {
   try {
      System.out.println("I am in try block");
      int x = 10/0;
      return 50;
    }
    catch(ArithmeticException ae)
    {
      System.out.println("I am in catch block");
    }
  return 20;
}
```

```
public static void main(String[] args)
 {
  TryReturnTest4 ft = new TryReturnTest4();
   System.out.println(ft.m1());
  }
}
```

Case 5: Return statement in try-catch block.

```
package finallyProgram;
public class TryCatchReturn1
{
 int m1()
 {
    try {
       System.out.println("I am in try block");
       return 50;
    }
    catch(Exception e)
    {
       System.out.println("I am in catch block");
       return 30;
    }
 }
 public static void main(String[] args)
 {
  TryCatchReturn1 obj = new TryCatchReturn1();
   System.out.println(obj.m1());
  }
}
```

Case 6: Return statement in try-catch block and a statement at end of method.

```
package finallyProgram;
public class TryCatchReturn2
{
```

```
 int m1()
 {
  try {
    System.out.println("I am in try block");
    return 50; // return statement inside try block.
  }
  catch(Exception e)
  {
    System.out.println("I am in catch block");
    return 30; // return statement inside the catch block.
  }
 System.out.println("Method at end"); // Unreachable code. So, compile time
error will occur.
}
public static void main(String[] args)
{
  TryCatchReturn2 obj = new TryCatchReturn2();
  System.out.println(obj.m1());
 }
}
```

Case 7: Return statement in catch block but no exception in try block

```
package finallyProgram;
public class CatchReturn1
{
 int m1()
 {
  try {
     System.out.println("I am in try block");
  }
  catch(Exception e)
  {
     System.out.println("I am in catch block");
     return 30; // return statement inside the catch block.
  }
```

```
  return 100; // return statement at the end of method
}
public static void main(String[] args)
{
 CatchReturn1 obj = new CatchReturn1();
   System.out.println(obj.m1());
  }
}
```

Case 8: Return statement in catch block but exception occurred in try block.

```
package finallyProgram;
public class CatchReturn2 {
int m1()
{
  try {
     System.out.println("I am in try block");
     int x = 20/0;
     System.out.println("Result: " +x);
  }
  catch(ArithmeticException ae)
  {
     System.out.println("I am in catch block");
     return 30;
  }
 return 100;
}
public static void main(String[] args)
{
  CatchReturn2 obj = new CatchReturn2();
  System.out.println(obj.m1());
 }
}
```

Case 9: Return statement in try block and finally block

```
package finallyProgram;
public class FinallyReturn1
{
 int m1()
 {
   try {
      System.out.println("I am in try block");
      return 30;
   }
  finally {
      System.out.println("I am in finally block");
      return 50;
  }
}
public static void main(String[] args)
{
 FinallyReturn1 obj = new FinallyReturn1();
   System.out.println(obj.m1());
  }
}
```

Case 10: Return statement in catch and finally blocks

```
package finallyProgram;
public class FinallyReturn2
{
@SuppressWarnings("finally")
int m1()
{
  try {
     System.out.println("I am in try block");
     int x = 10/0;
     System.out.println("Result: " +x);
  }
  catch(ArithmeticException ae)
```

```
  {
    System.out.println("I am in catch block");
    return 40;
  }
 finally {
     System.out.println("I am in finally block");
     return 50;
  }
}
public static void main(String[] args)
{
   FinallyReturn2 obj = new FinallyReturn2();
   System.out.println(obj.m1());
 }
}
```

Case 11: Return statement in catch and finally blocks but a statement after finally block

```
package finallyProgram;
public class FinallyReturn3
{
@SuppressWarnings("finally")
int m1()
{
  int a = 20, b = 0;
  try {
     System.out.println("I am in try block");
     int c = a/b;
     System.out.println("Result: " +c);
  }
  catch(ArithmeticException ae)
  {
    System.out.println("I am in catch block");
    return 40;
  }
```

```
  finally {
     System.out.println("I am in finally block");
     return 50;
  }
 System.out.println("Statement after finally block");
}
public static void main(String[] args)
{
   FinallyReturn3 obj = new FinallyReturn3();
   System.out.println(obj.m1());
 }
}
```