University Institute of Information Technology
Faculty of Computing

Course Code:	CS-432	**Subject:**	MPL
Lab Manuals		**Class/Lab Instructor:**	

-------------------- LAB 03 --------------------

Learning Objective:

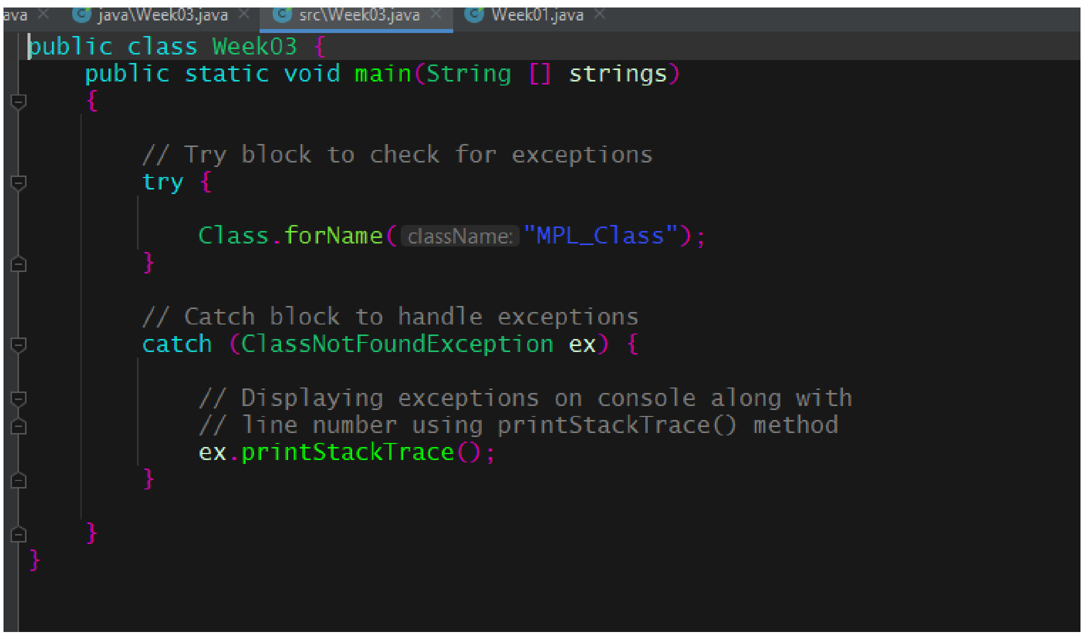
1. Exception Handling
2. Types of Exception
3. Scenarios where exception may occur

Checked Exception:

These are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using the throws keyword.

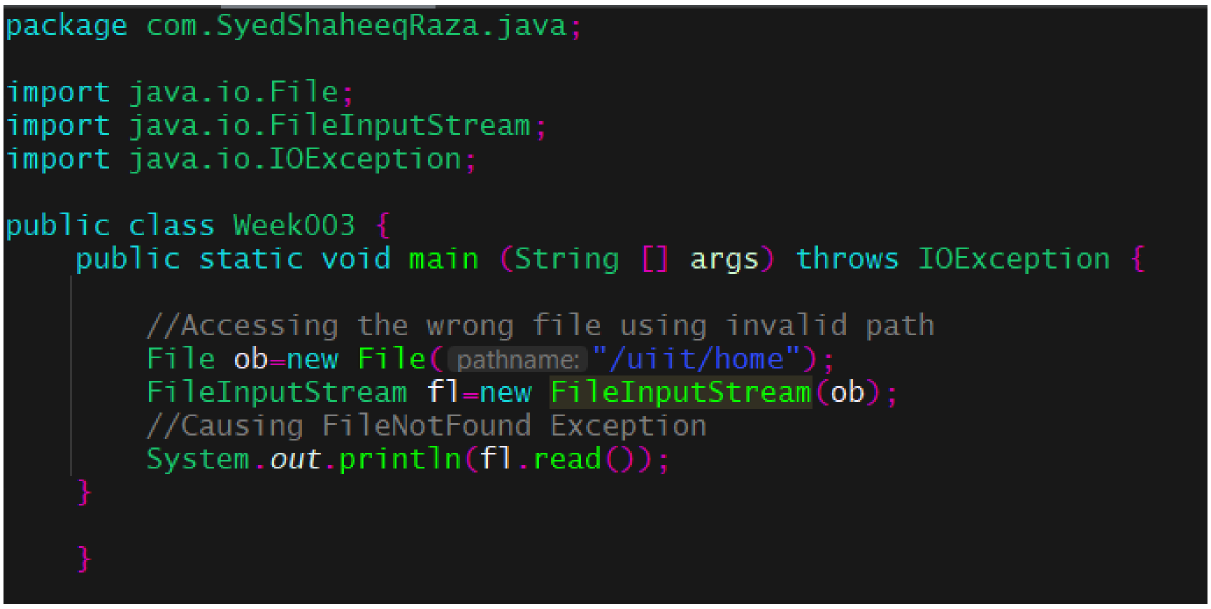
1. ClassNotFoundException
2. InterruptedException
3. IOException
4. SQLException
5. IllegalAccessException
6. FileNotFoundException

Example # 1 ClassNotFoundException



```
public class Week03 {
    public static void main(String [] strings)
    {

        // Try block to check for exceptions
        try {

            Class.forName( className: "MPL_Class");
        }

        // Catch block to handle exceptions
        catch (ClassNotFoundException ex) {

            // Displaying exceptions on console along with
            // line number using printStackTrace() method
            ex.printStackTrace();
        }

    }
}
```

Example # 2 IOException


```
package com.SyedShaheeqRaza.java;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

public class Week003 {
    public static void main (String [] args) throws IOException {

        //Accessing the wrong file using invalid path
        File ob=new File( pathname: "/uiit/home");
        FileInputStream fl=new FileInputStream(ob);
        //Causing FileNotFound Exception
        System.out.println(fl.read());
    }

    }
```

Unchecked Exception:

An unchecked exception (also known as a runtime exception) in Java is something that has gone wrong with the program and is unrecoverable. Just because this is not a compile time exception, meaning you do not need to handle it, that does not mean you don't need to be concerned about it.

- ArithmeticException.
- NullPointerException.
- ArrayIndexOutOfBoundsException.
- NumberFormatException.
- InputMismatchException.
- IllegalStateException.
- Missing Resource Exception.
- No Such Element Exception.

Scenarios where exception may occur:

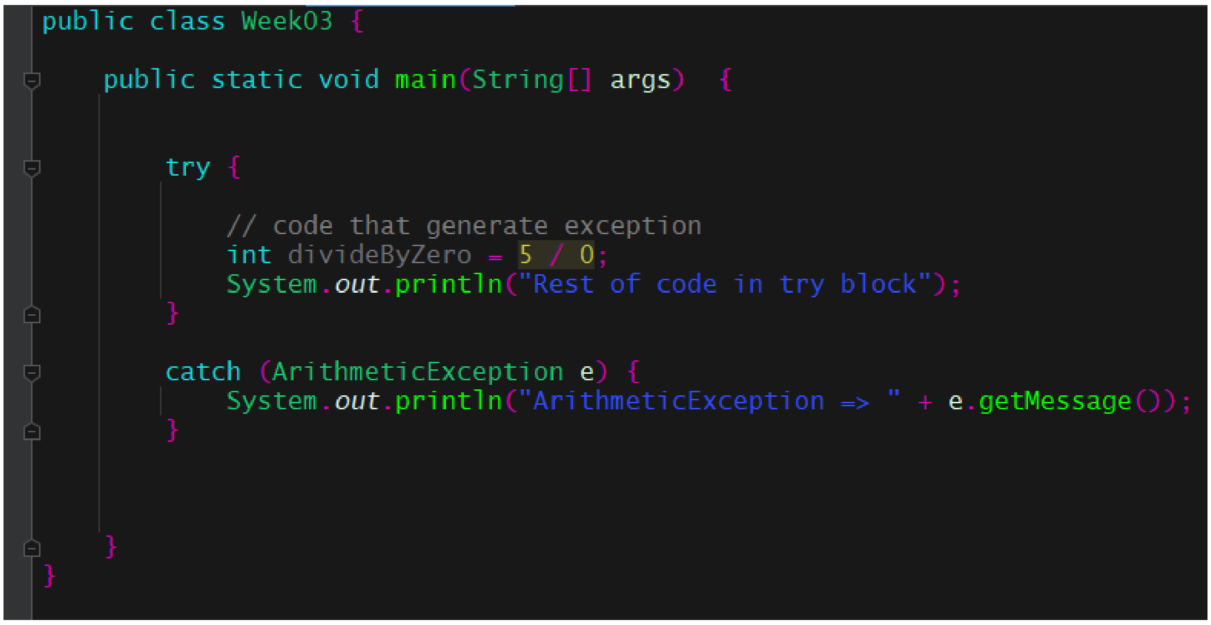
In a program, exceptions can occur due to invalid user actions, insufficient disk space, or loss of the network connection with the server. Exceptions can also result from programming errors or incorrect use of an API. Unlike humans in the real world, a program must know exactly how to handle these situations.

Java try...catch block:

The try-catch block is used to handle exceptions in Java. Here's the syntax of try...catch

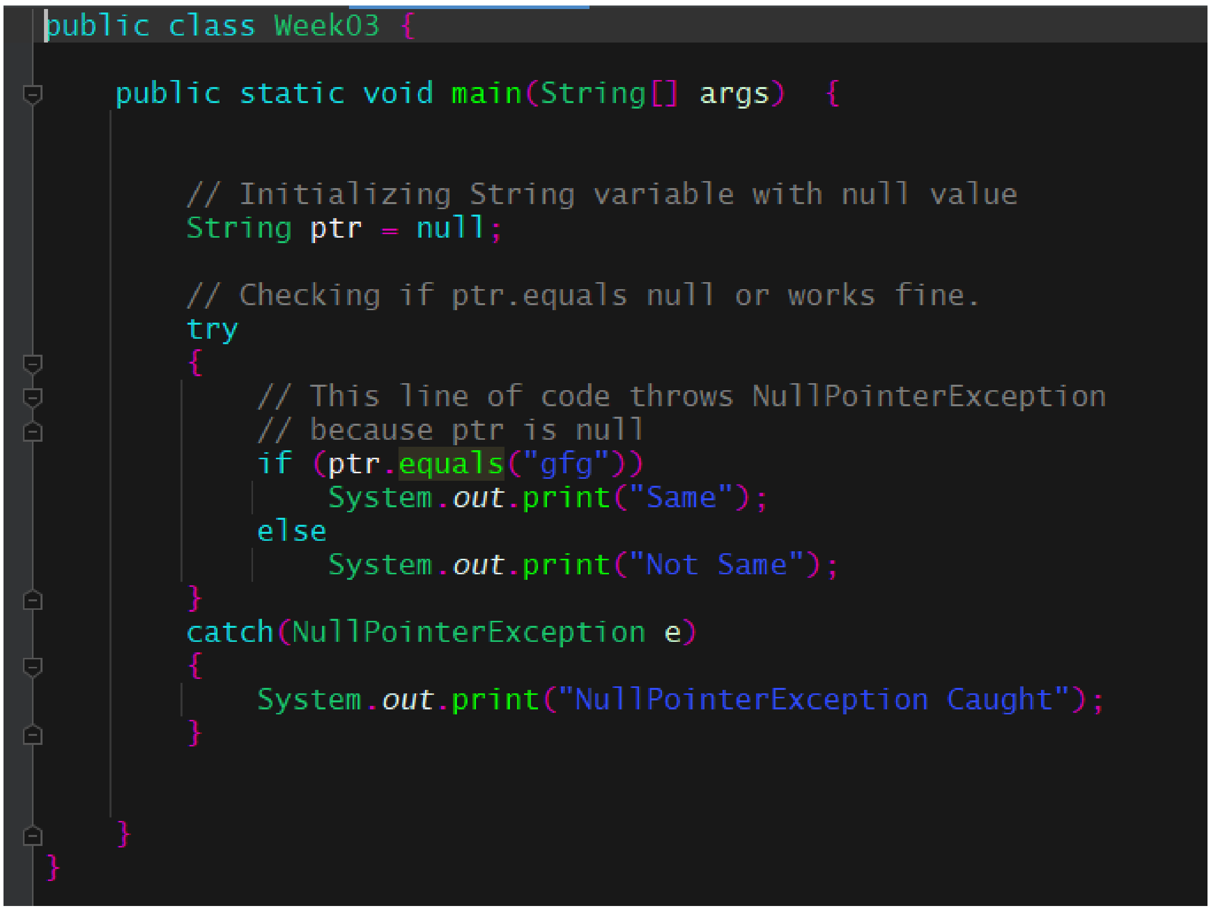
```
try {
  // code
}
Catch (Exception e) {
  // code
}
```

Example # 3 Exception handling using try...catch



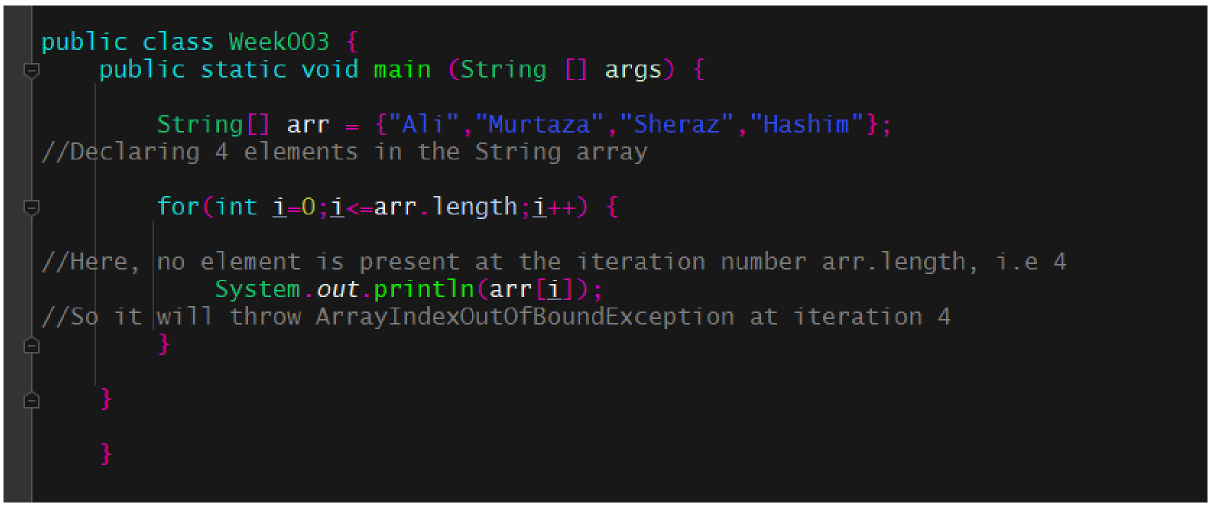
```
public class Week03 {

    public static void main(String[] args)  {


        try {

            // code that generate exception
            int divideByZero = 5 / 0;
            System.out.println("Rest of code in try block");
        }

        catch (ArithmeticException e) {
            System.out.println("ArithmeticException => " + e.getMessage());
        }



    }
}
```

Example #4 NullPointerException.



```
public class Week03 {

    public static void main(String[] args)  {


        // Initializing String variable with null value
        String ptr = null;

        // Checking if ptr.equals null or works fine.
        try
        {
            // This line of code throws NullPointerException
            // because ptr is null
            if (ptr.equals("gfg"))
                System.out.print("Same");
            else
                System.out.print("Not Same");
        }
        catch(NullPointerException e)
        {
            System.out.print("NullPointerException Caught");
        }


    }
}
```

Example #5 ArrayIndexOutOfBoundsException

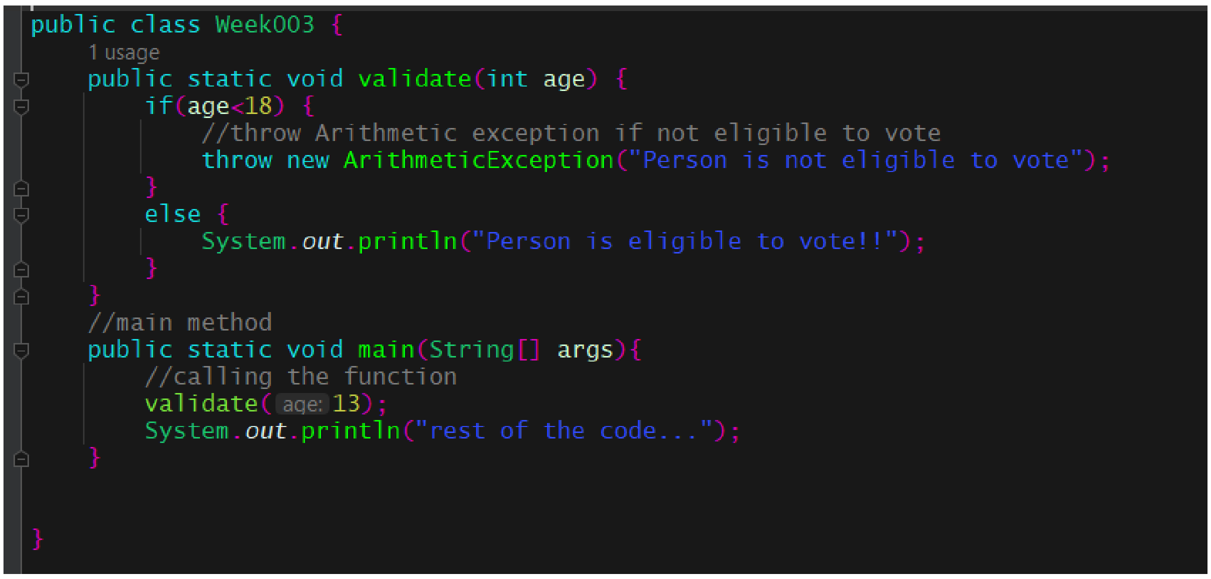


```
public class Week003 {
    public static void main (String [] args) {

        String[] arr = {"Ali","Murtaza","Sheraz","Hashim"};
//Declaring 4 elements in the String array

        for(int i=0;i<=arr.length;i++) {

//Here, no element is present at the iteration number arr.length, i.e 4
            System.out.println(arr[i]);
//So it will throw ArrayIndexOutOfBoundException at iteration 4
        }

    }

    }
```

Java throw keyword:

The throw keyword in Java is used for explicitly throwing a single exception. This can be from within a method or any block of code. Both checked and unchecked exceptions can be thrown using the throw keyword.

Example #6 Java throw keyword

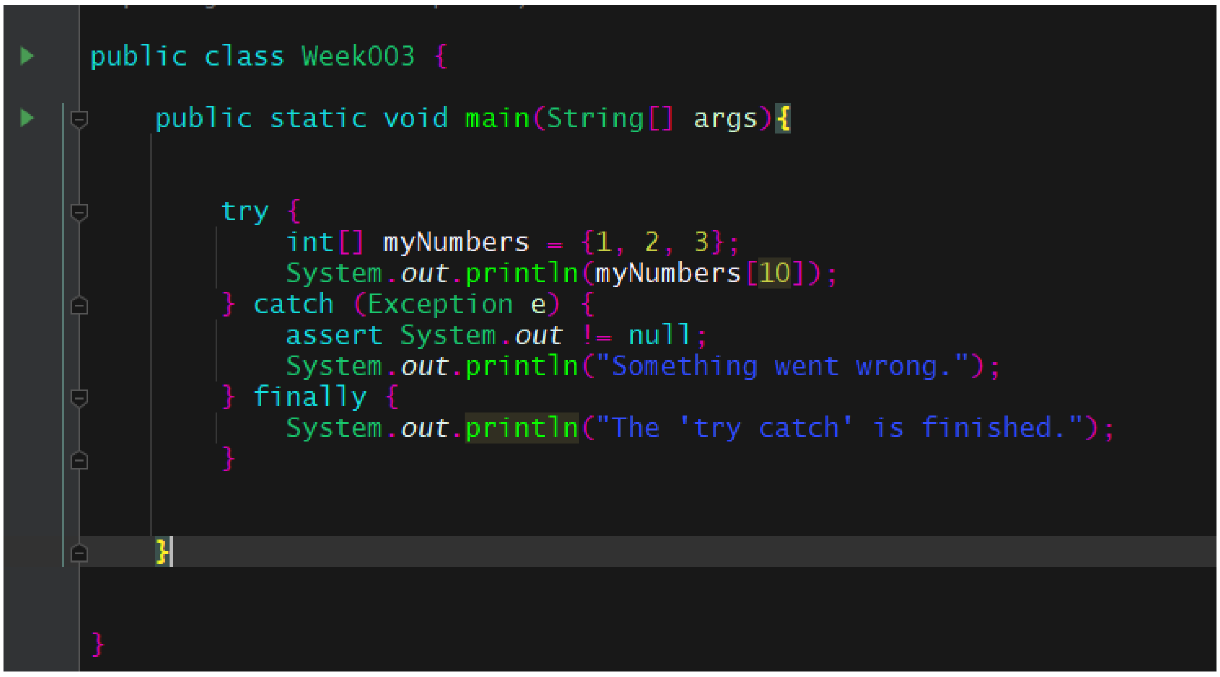


```java
public class Week003 {
    1 usage
    public static void validate(int age) {
        if(age<18) {
            //throw Arithmetic exception if not eligible to vote
            throw new ArithmeticException("Person is not eligible to vote");
        }
        else {
            System.out.println("Person is eligible to vote!!");
        }
    }
    //main method
    public static void main(String[] args){
        //calling the function
        validate( age: 13);
        System.out.println("rest of the code...");
    }


}
```

Java finally Keyword:

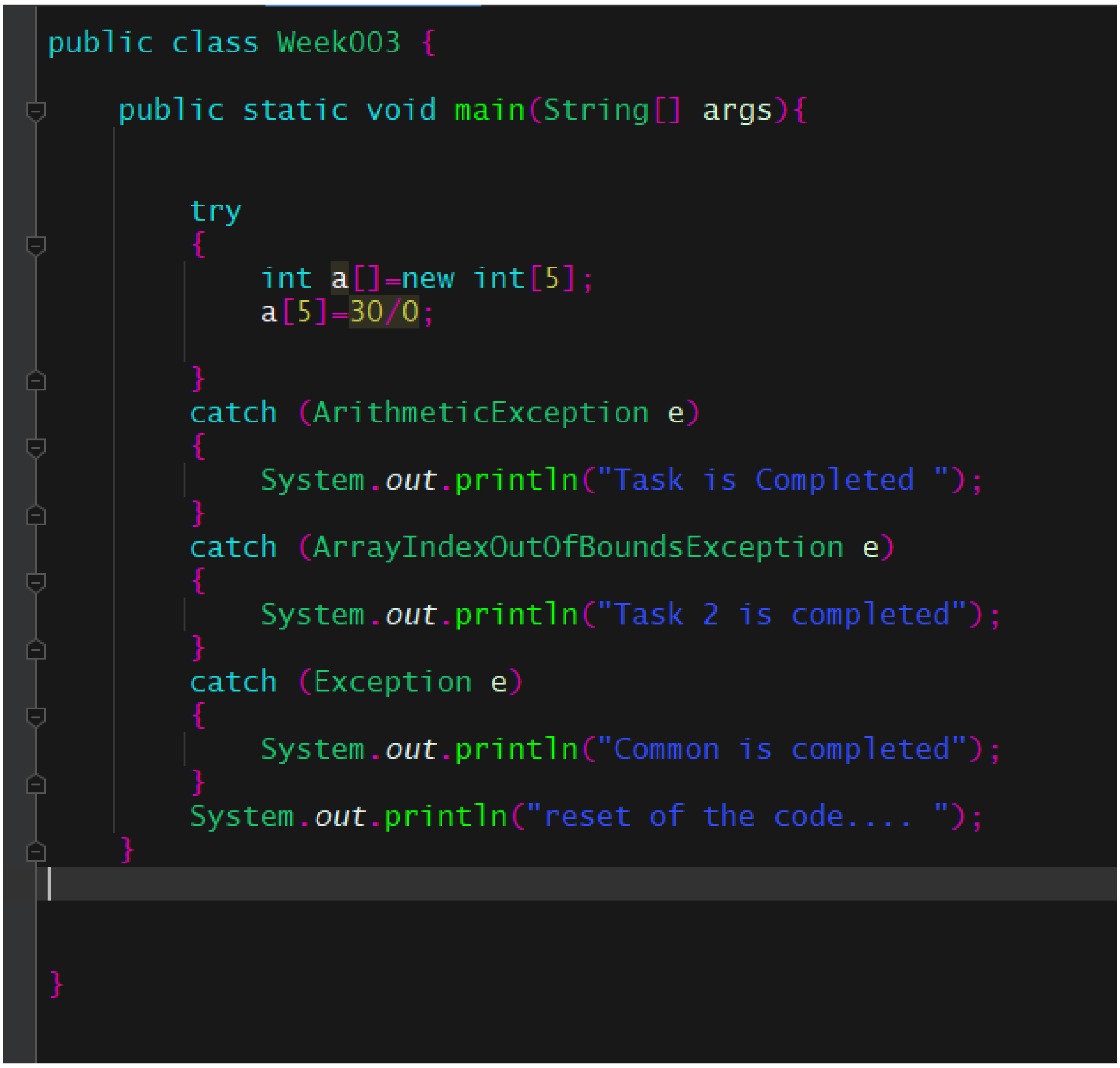
The finally keyword is used to execute code (used with exceptions - try.. catch statements) no matter if there is an exception or not.

Example #7 Java finally Keyword



```java
public class Week003 {

    public static void main(String[] args){


        try {
            int[] myNumbers = {1, 2, 3};
            System.out.println(myNumbers[10]);
        } catch (Exception e) {
            assert System.out != null;
            System.out.println("Something went wrong.");
        } finally {
            System.out.println("The 'try catch' is finished.");
        }


    }


}
```

Example #8 Catch Multiple Exceptions



```
public class Week003 {

    public static void main(String[] args){


        try
        {
            int a[]=new int[5];
            a[5]=30/0;

        }
        catch (ArithmeticException e)
        {
            System.out.println("Task is Completed ");
        }
        catch (ArrayIndexOutOfBoundsException e)
        {
            System.out.println("Task 2 is completed");
        }
        catch (Exception e)
        {
            System.out.println("Common is completed");
        }
        System.out.println("reset of the code.... ");
    }



}
```

Lab Task(s)

1. Write a Java program to find the null pointer exception with example.
2. Write a Java program to find the arithmetic exception with example.
3. Write a Java program to handle the multiple exception with example.
4. Write another example of try and catch function.
5. Writ a java program to use the keywords is throw and throws in same example.