Exception Handling

✓ 1) Exceptn vs Error

✓ 2) Exception Hierarchy

✓ 3) Compile Time (Checked) vs RunTime (Unchecked)

✓ 4) Exceptn Handling :- What? Why? How?

✓ 5) Exceptn object & Default Error Handling by JVM

✓ 6) 5 keywords: try, catch, finally, throw, throws

✓ 7) valid orders of try, catch & finally

8) User Defined/Custom Exceptions

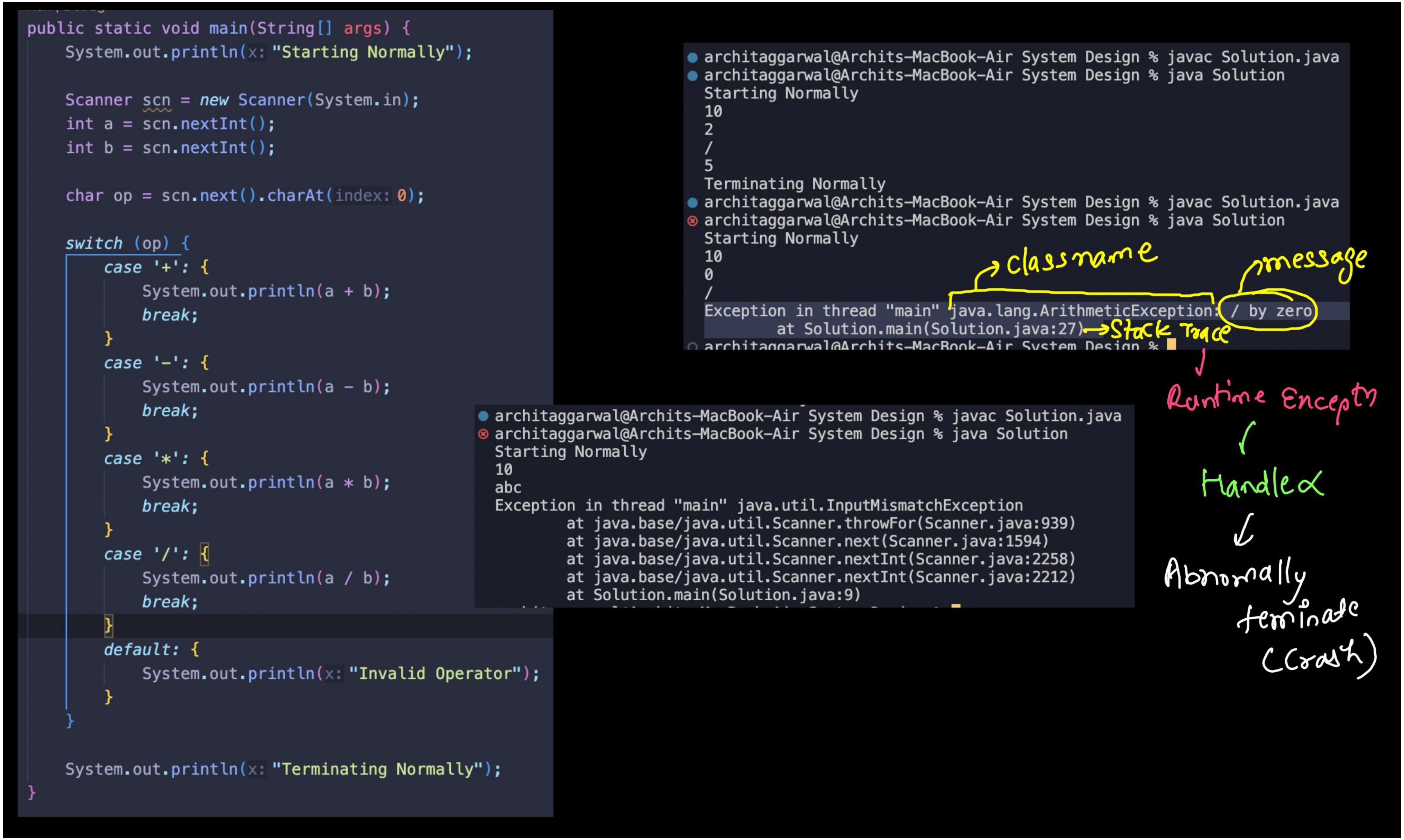
9) Differences: final, finally, finalize

10) Differences: throw vs throws

Exceptn :→ Any abnormal behavior in your code which occurs at run time & distrubes the normal flow by abnormal terminatn (crashing).

Exceptn Handling :→ Alternate sequence flow provided using 5 keywords to normally terminate the program is known as exceptn handling.

WHAT?



```java
public static void main(String[] args) {
    System.out.println(x: "Starting Normally");

    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();

    char op = scn.next().charAt(index: 0);

    switch (op) {
        case '+': {
            System.out.println(a + b);
            break;
        }
        case '-': {
            System.out.println(a - b);
            break;
        }
        case '*': {
            System.out.println(a * b);
            break;
        }
        case '/': {
            System.out.println(a / b);
            break;
        }
        default: {
            System.out.println(x: "Invalid Operator");
        }
    }

    System.out.println(x: "Terminating Normally");
}
```

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
2
/
5
Terminating Normally
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
0
/
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Solution.main(Solution.java:27)
architaggarwal@Archits-MacBook-Air System Design %
```

classname

message

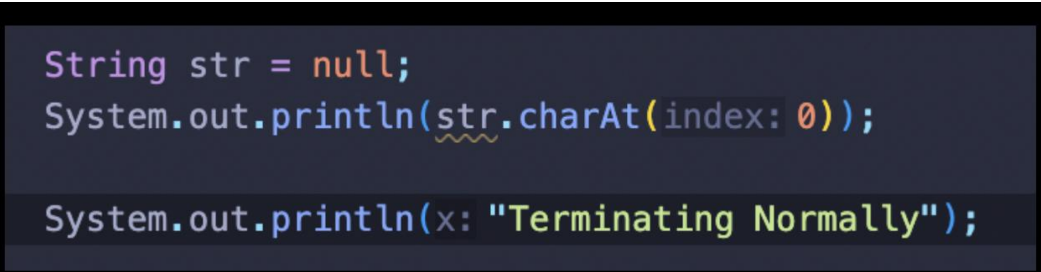
Stack Trace

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
abc
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
        at Solution.main(Solution.java:9)
```

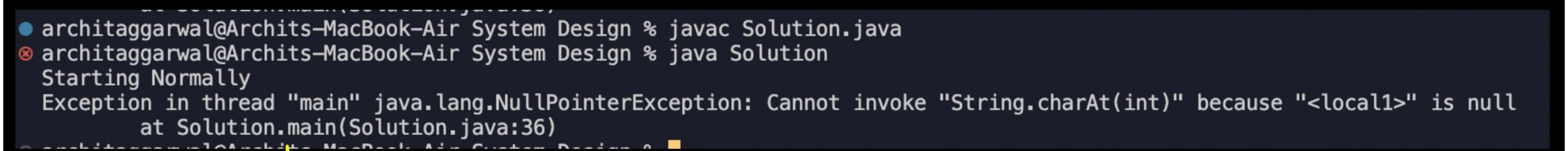
Runtime Exception

Handled

Abnormally terminate (Crash)



```java
String str = null;
System.out.println(str.charAt(index: 0));

System.out.println(x: "Terminating Normally");
```



```
        at Solution.main(Solution.java:36)
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.charAt(int)" because "<local1>" is null
        at Solution.main(Solution.java:36)
architaggarwal@Archits-MacBook-Air System Design %
```

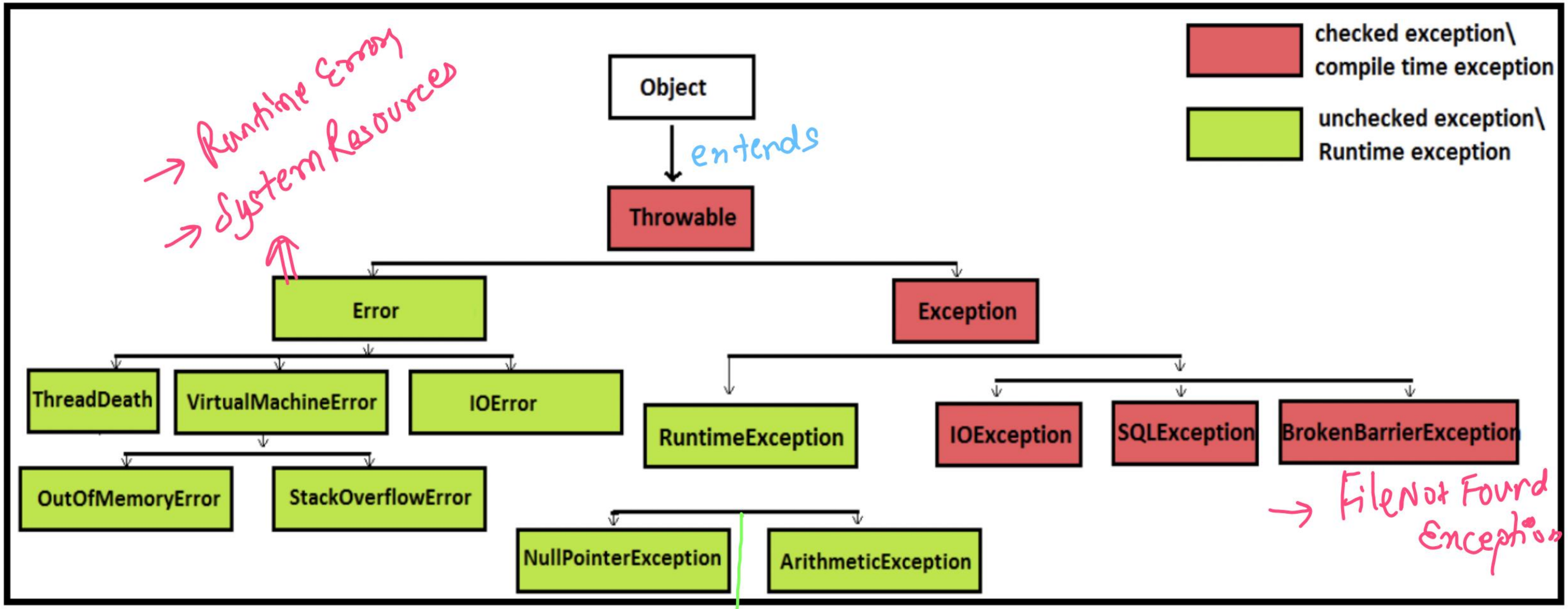
↓

unchecked (Runtime Excepn)

↓

Compiler was not able to check these Exceptions

Exception hierarchy >

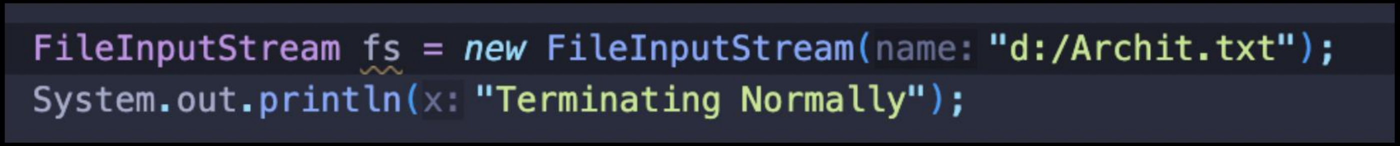


Index OutOf Bound Excepn,

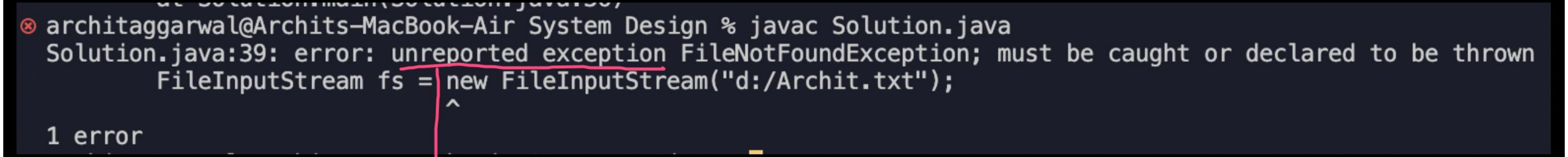
Number Format Excepn

Class Cast Exception

→ ClassNot Found Exception



```
FileInputStream fs = new FileInputStream(name: "d:/Archit.txt");
System.out.println(x: "Terminating Normally");
```



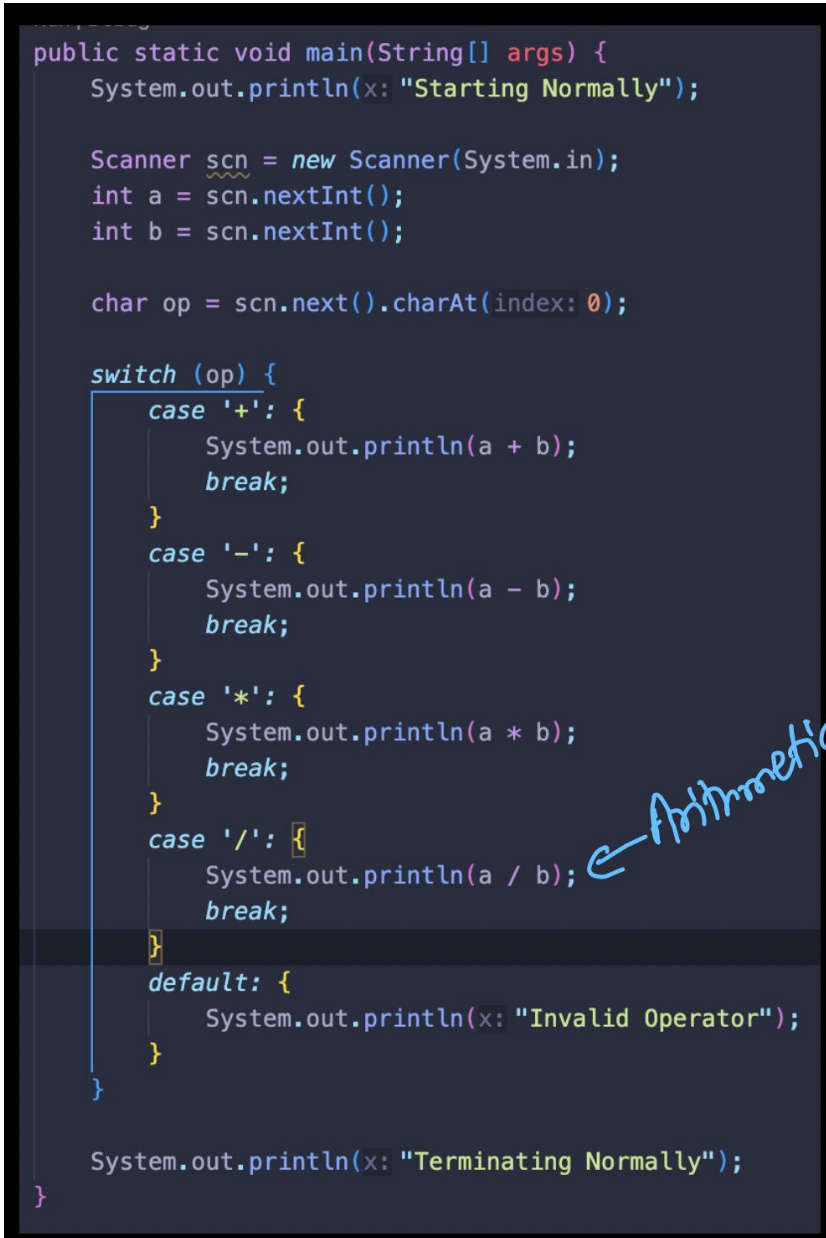
```
        at Solution.main(Solution.java:30)
⊗ architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
  Solution.java:39: error: unreported exception FileNotFoundException; must be caught or declared to be thrown
          FileInputStream fs = new FileInputStream("d:/Archit.txt");
                               ^
  1 error
```

Compiler is complaining that you have not handled the FileNotFound Exception

So i will not compile it!

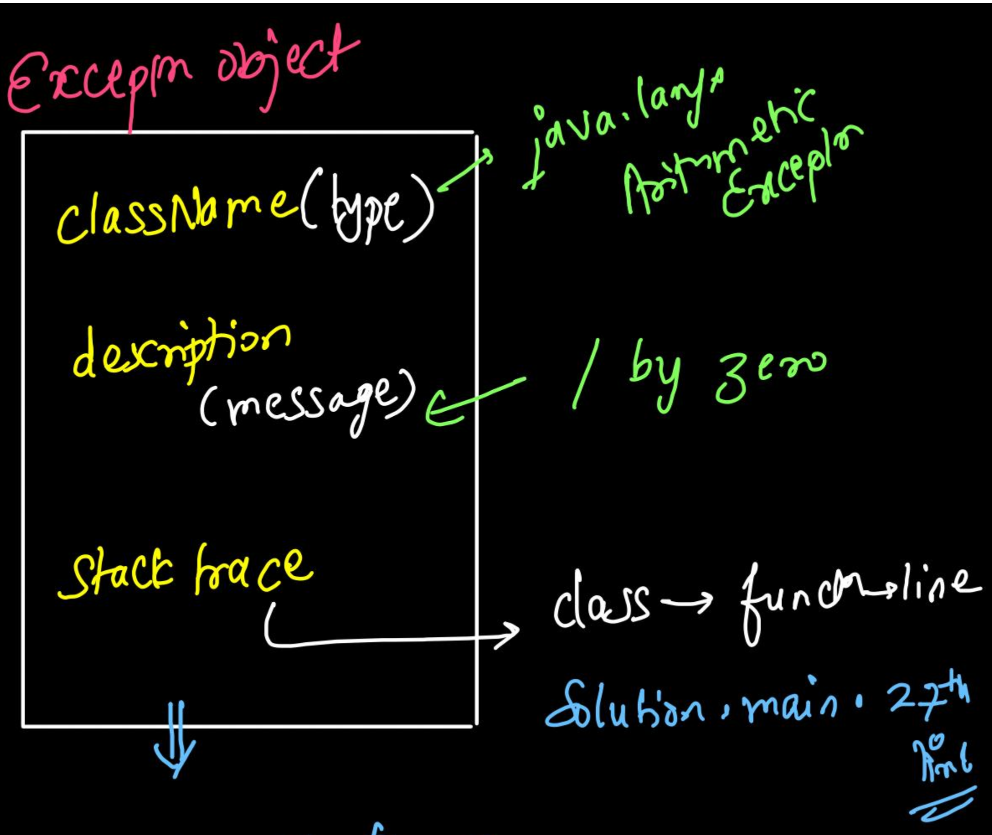
Checked / Compile Time Excepn

(Excepn will still happen at run time!)



```
public static void main(String[] args) {
    System.out.println(x: "Starting Normally");

    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();

    char op = scn.next().charAt(index: 0);

    switch (op) {
        case '+': {
            System.out.println(a + b);
            break;
        }
        case '-': {
            System.out.println(a - b);
            break;
        }
        case '*': {
            System.out.println(a * b);
            break;
        }
        case '/': {
            System.out.println(a / b);
            break;
        }
        default: {
            System.out.println(x: "Invalid Operator");
        }
    }

    System.out.println(x: "Terminating Normally");
}
```

← ArithmeticException



main will throw this object to JVM.

⇓

It will handle it via Default Exceptn Handling

5 keywords

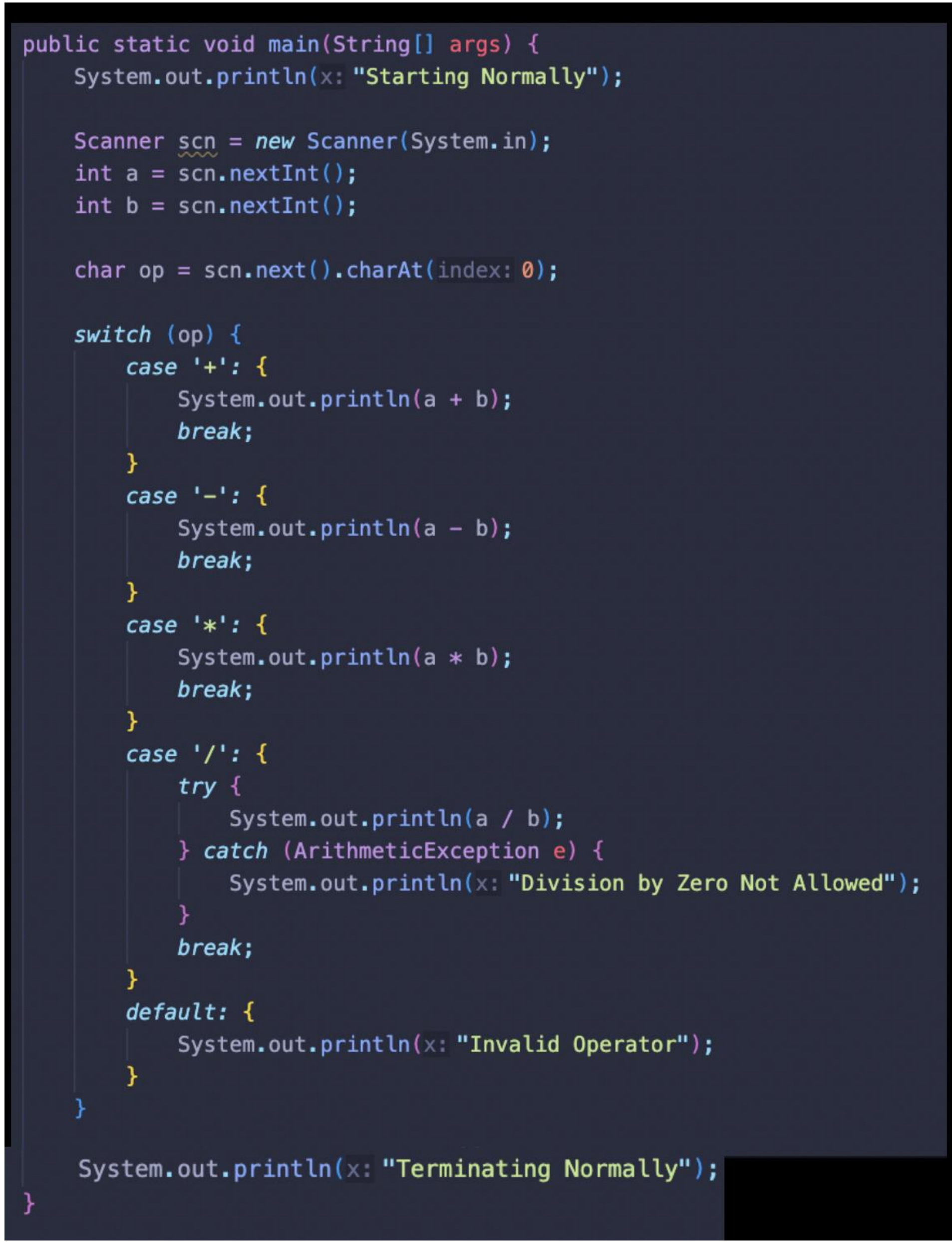
(1) try → risky code (chances of Excepn)

(2) catch → alternate flow (handlig the Exception)

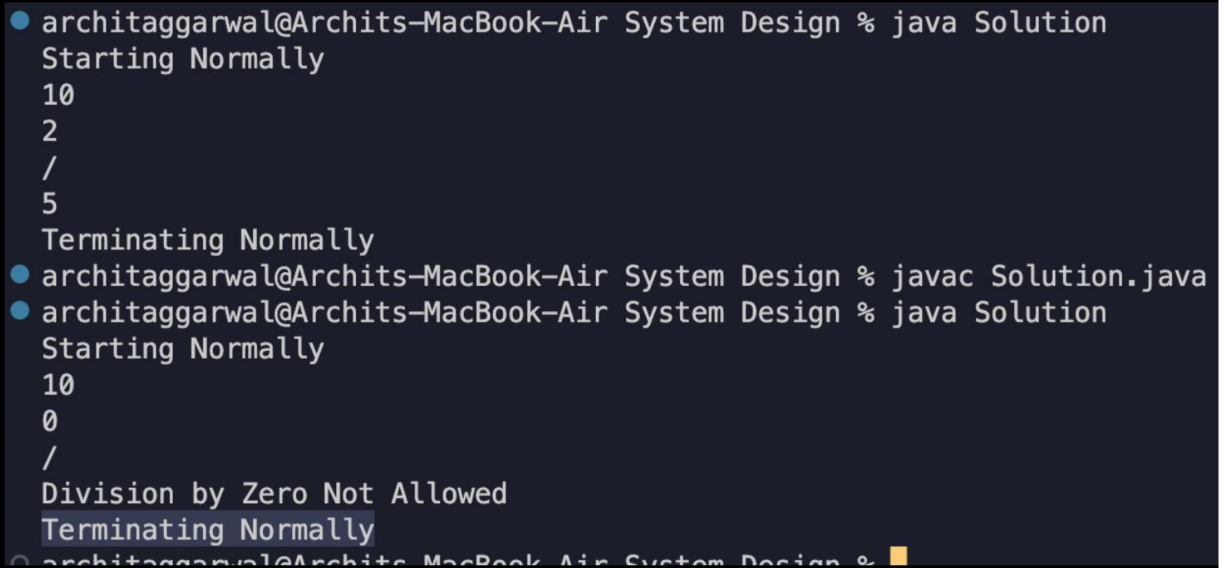
(3) finally → cleanup code (FileInputoutput, I/o stream close, memory release

(4) throw

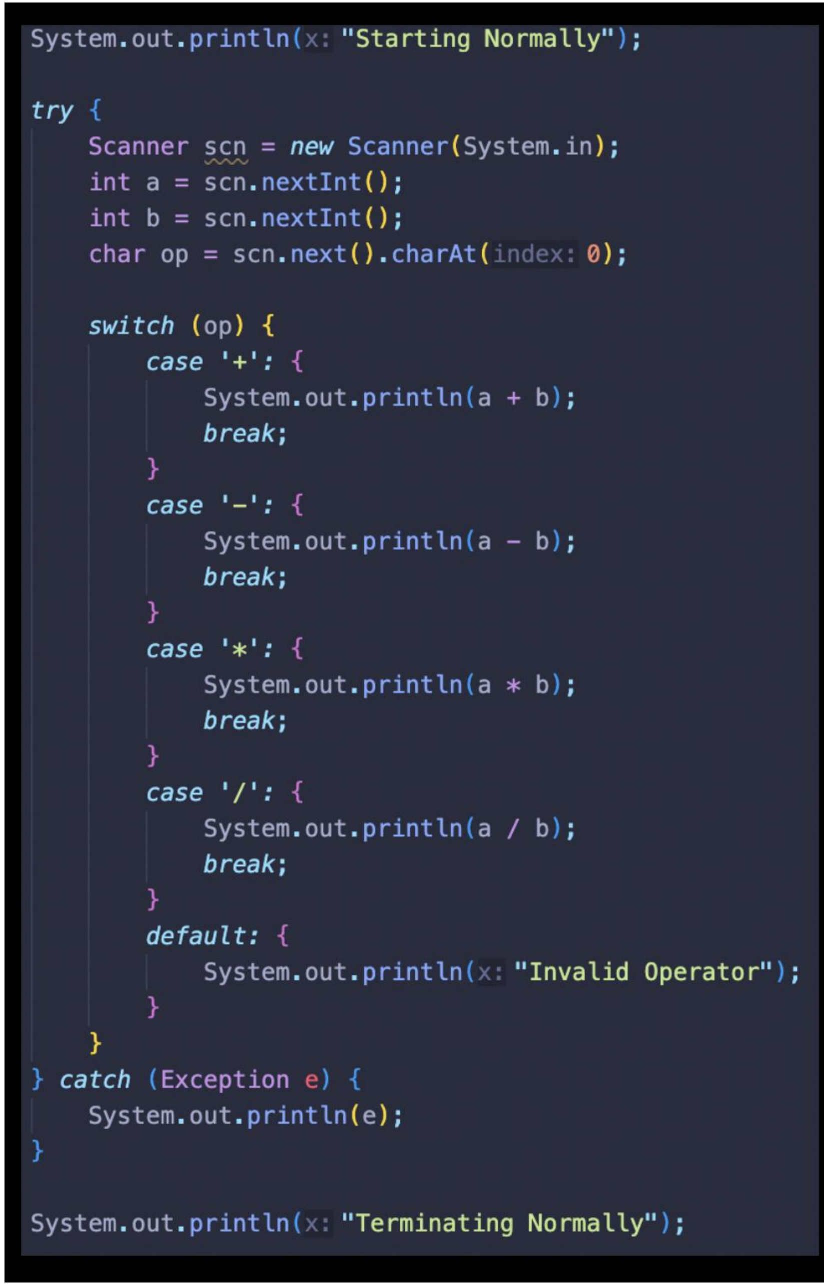
(5) throws



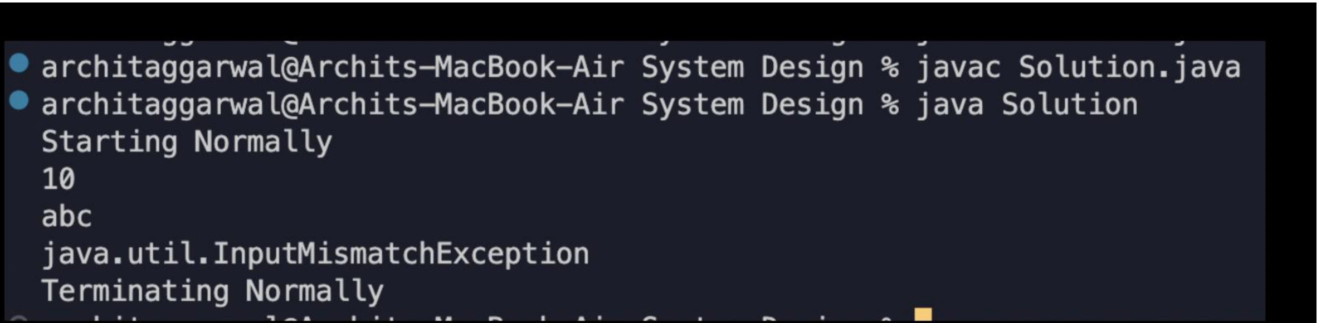
```java
public static void main(String[] args) {
    System.out.println(x: "Starting Normally");

    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();

    char op = scn.next().charAt(index: 0);

    switch (op) {
        case '+': {
            System.out.println(a + b);
            break;
        }
        case '-': {
            System.out.println(a - b);
            break;
        }
        case '*': {
            System.out.println(a * b);
            break;
        }
        case '/': {
            try {
                System.out.println(a / b);
            } catch (ArithmeticException e) {
                System.out.println(x: "Division by Zero Not Allowed");
            }
            break;
        }
        default: {
            System.out.println(x: "Invalid Operator");
        }
    }

    System.out.println(x: "Terminating Normally");
}
```



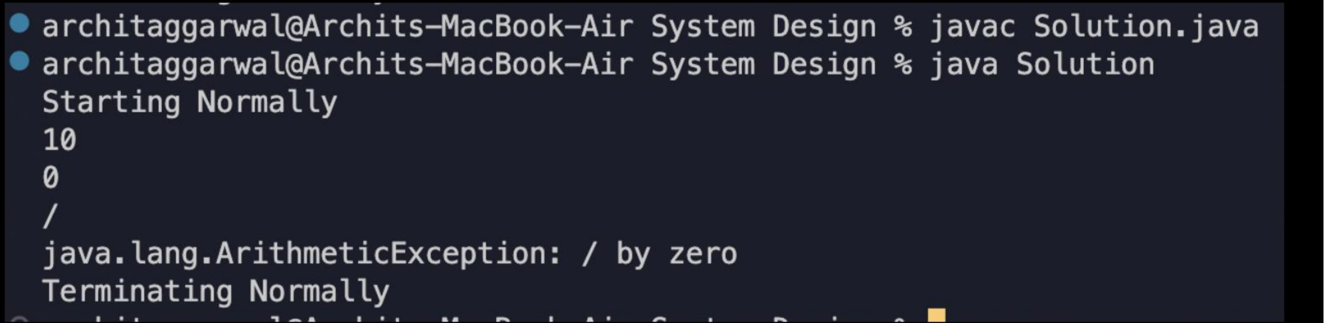
```
● architaggarwal@Archits-MacBook-Air System Design % java Solution
  Starting Normally
  10
  2
  /
  5
  Terminating Normally
● architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
● architaggarwal@Archits-MacBook-Air System Design % java Solution
  Starting Normally
  10
  0
  /
  Division by Zero Not Allowed
  Terminating Normally
○ architaggarwal@Archits-MacBook-Air System Design %
```



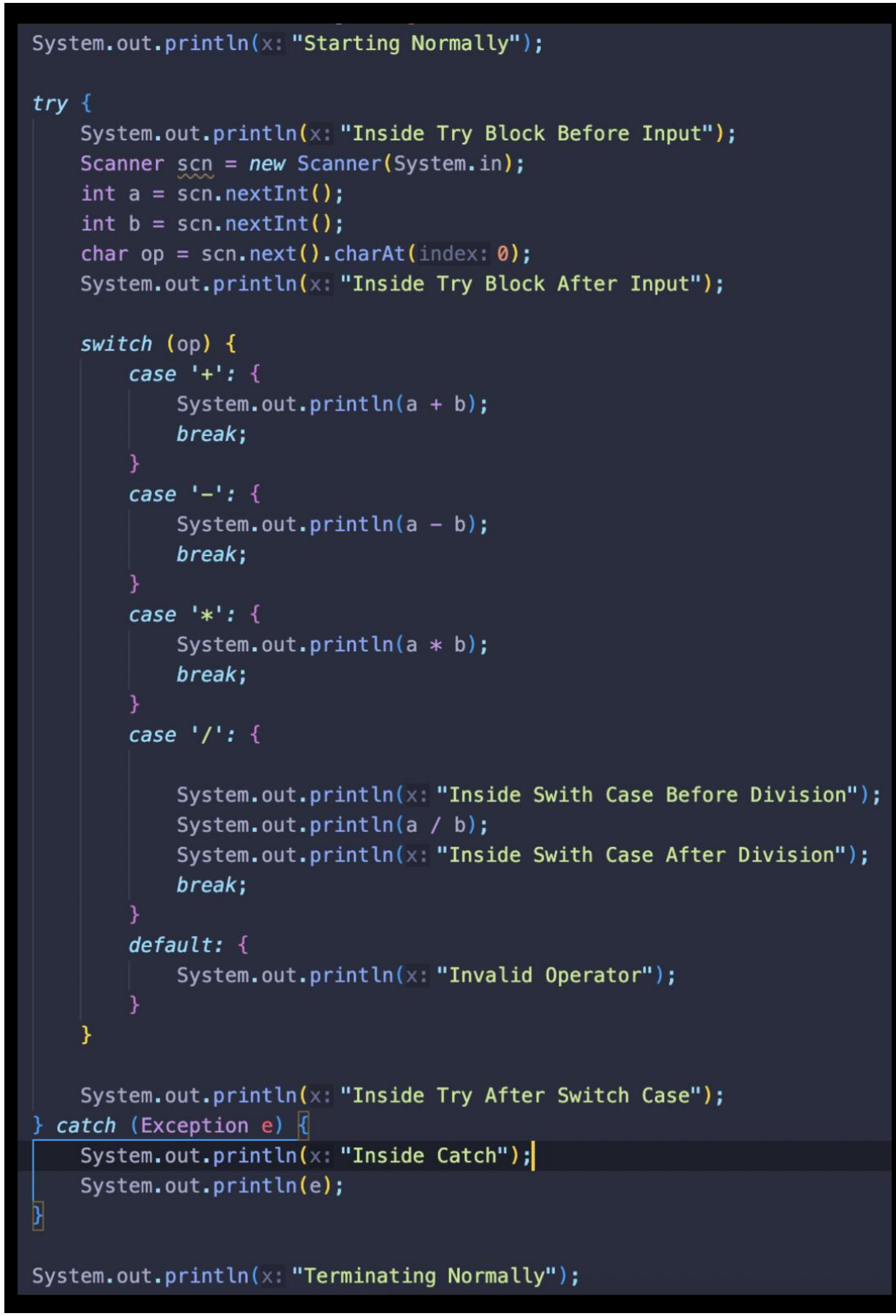
```java
System.out.println(x: "Starting Normally");

try {
    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();
    char op = scn.next().charAt(index: 0);

    switch (op) {
        case '+': {
            System.out.println(a + b);
            break;
        }
        case '-': {
            System.out.println(a - b);
            break;
        }
        case '*': {
            System.out.println(a * b);
            break;
        }
        case '/': {
            System.out.println(a / b);
            break;
        }
        default: {
            System.out.println(x: "Invalid Operator");
        }
    }
} catch (Exception e) {
    System.out.println(e);
}

System.out.println(x: "Terminating Normally");
```



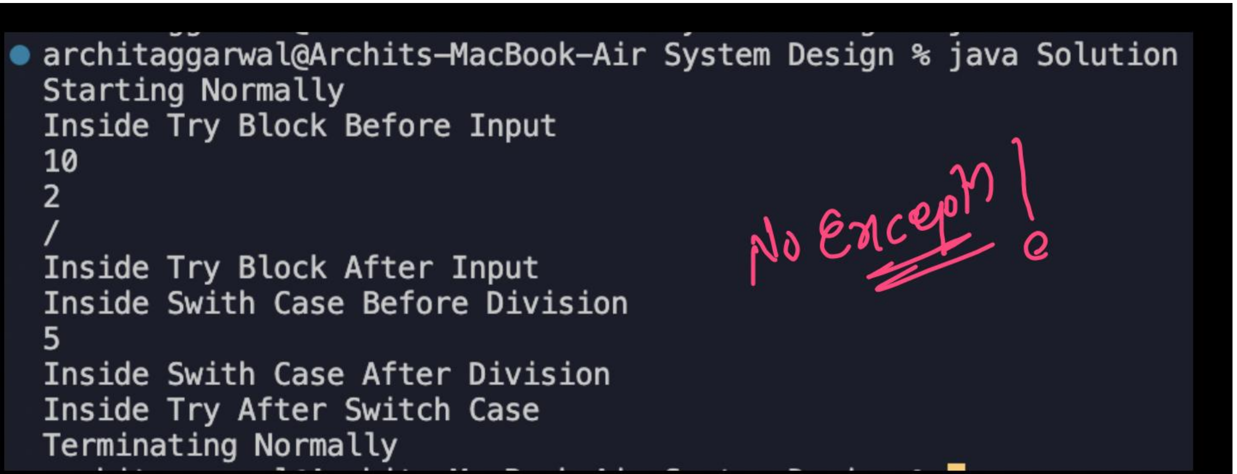
```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
abc
java.util.InputMismatchException
Terminating Normally
```



```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
0
/
java.lang.ArithmeticException: / by zero
Terminating Normally
```

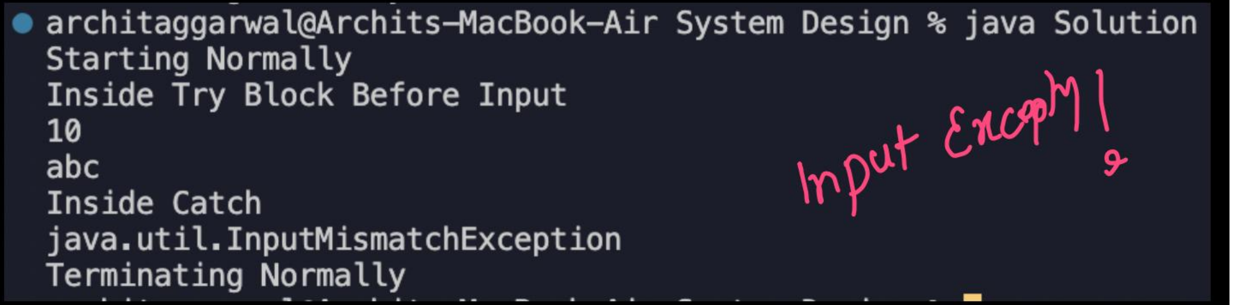


```java
System.out.println(x: "Starting Normally");

try {
    System.out.println(x: "Inside Try Block Before Input");
    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();
    char op = scn.next().charAt(index: 0);
    System.out.println(x: "Inside Try Block After Input");

    switch (op) {
        case '+': {
            System.out.println(a + b);
            break;
        }
        case '-': {
            System.out.println(a - b);
            break;
        }
        case '*': {
            System.out.println(a * b);
            break;
        }
        case '/': {

            System.out.println(x: "Inside Swith Case Before Division");
            System.out.println(a / b);
            System.out.println(x: "Inside Swith Case After Division");
            break;
        }
        default: {
            System.out.println(x: "Invalid Operator");
        }
    }

    System.out.println(x: "Inside Try After Switch Case");
} catch (Exception e) {
    System.out.println(x: "Inside Catch");
    System.out.println(e);
}

System.out.println(x: "Terminating Normally");
```



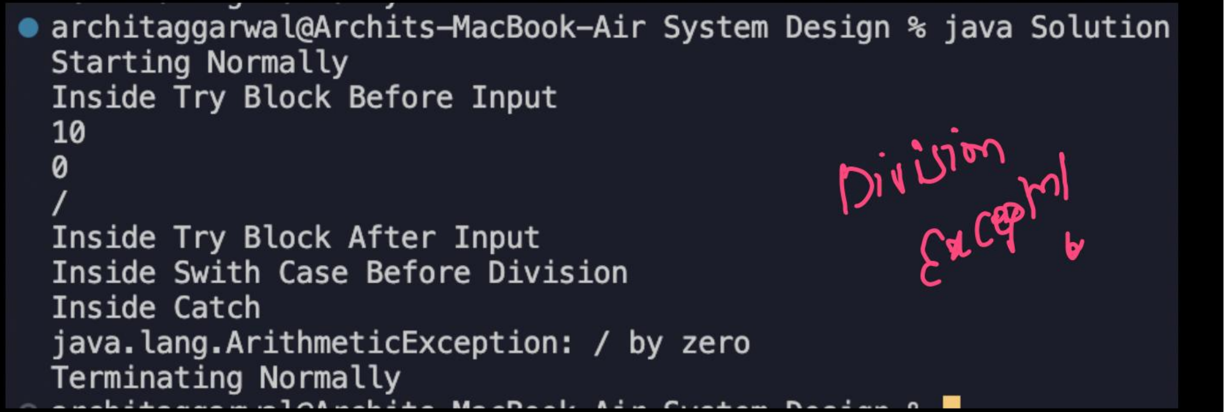
```
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
Inside Try Block Before Input
10
2
/
Inside Try Block After Input
Inside Swith Case Before Division
5
Inside Swith Case After Division
Inside Try After Switch Case
Terminating Normally
```

No Exception!



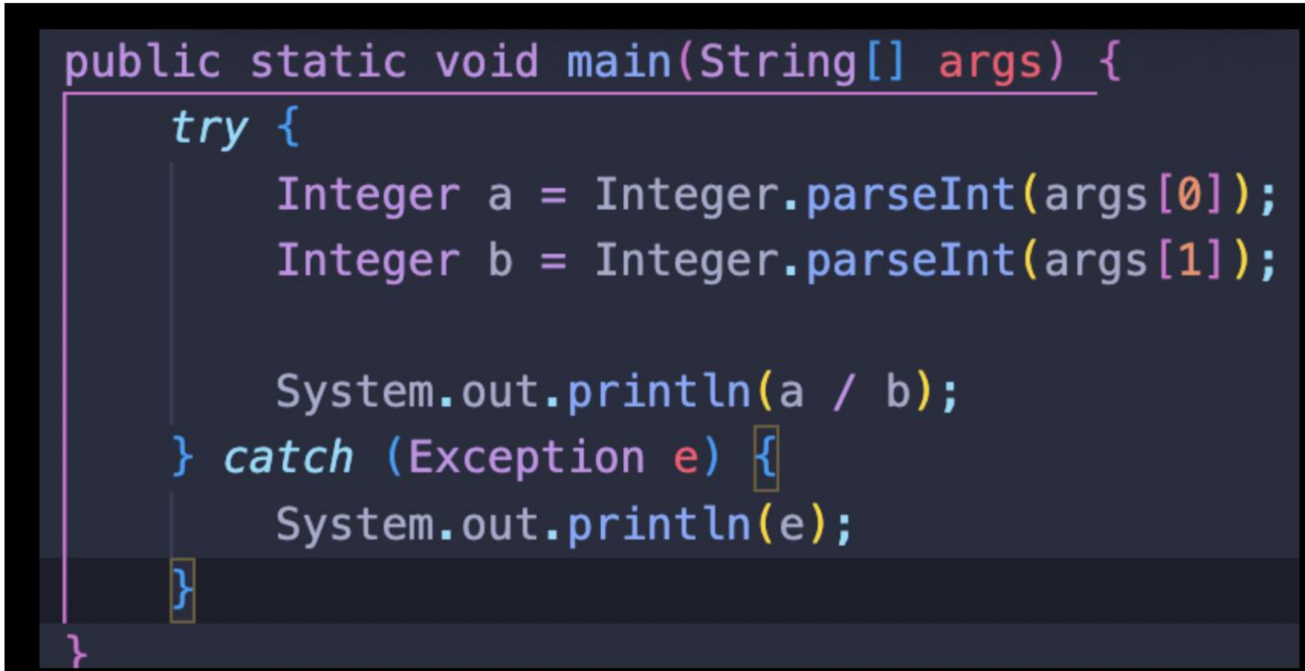
```
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
Inside Try Block Before Input
10
abc
Inside Catch
java.util.InputMismatchException
Terminating Normally
```

Input Exception!

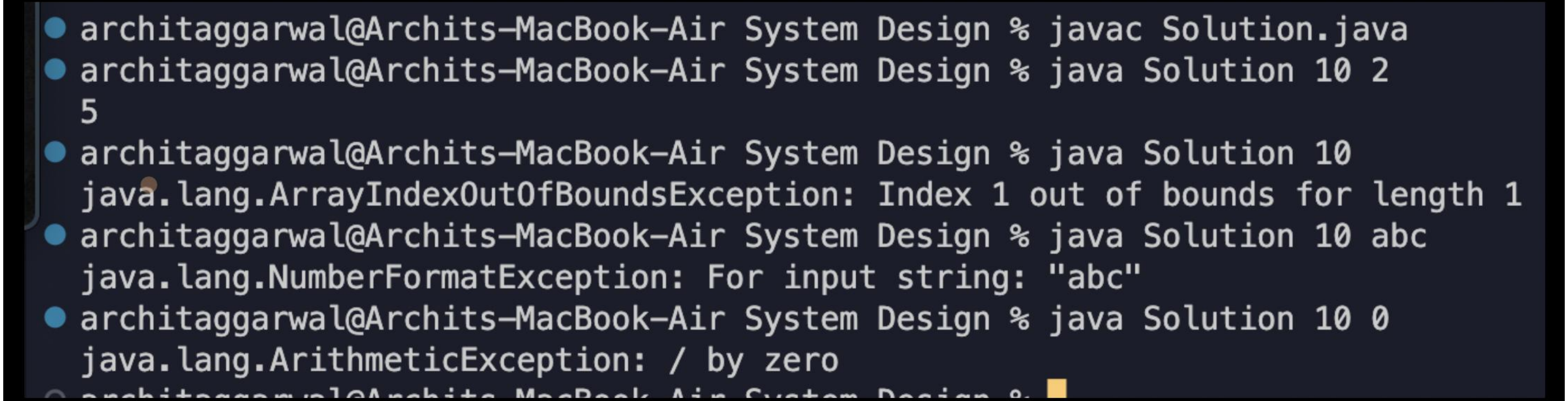


```
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
Inside Try Block Before Input
10
0
/
Inside Try Block After Input
Inside Swith Case Before Division
Inside Catch
java.lang.ArithmeticException: / by zero
Terminating Normally
```

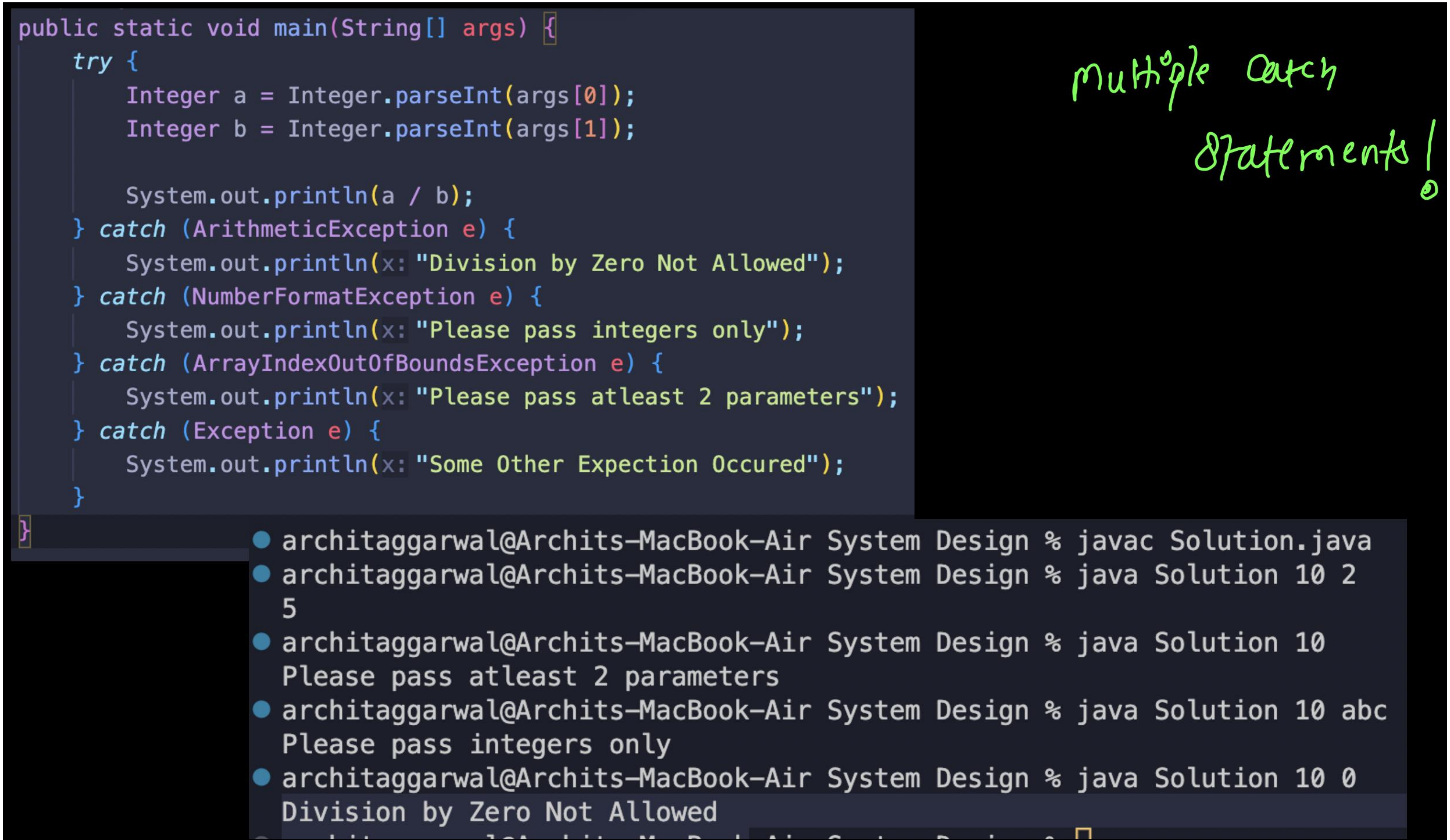
Division Exception!



```java
public static void main(String[] args) {
    try {
        Integer a = Integer.parseInt(args[0]);
        Integer b = Integer.parseInt(args[1]);

        System.out.println(a / b);
    } catch (Exception e) {
        System.out.println(e);
    }
}
```



```
● architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
● architaggarwal@Archits-MacBook-Air System Design % java Solution 10 2
  5
● architaggarwal@Archits-MacBook-Air System Design % java Solution 10
  java.lang.ArrayIndexOutOfBoundsException: Index 1 out of bounds for length 1
● architaggarwal@Archits-MacBook-Air System Design % java Solution 10 abc
  java.lang.NumberFormatException: For input string: "abc"
● architaggarwal@Archits-MacBook-Air System Design % java Solution 10 0
  java.lang.ArithmeticException: / by zero
○ architaggarwal@Archits-MacBook-Air System Design %
```



```java
public static void main(String[] args) {
    try {
        Integer a = Integer.parseInt(args[0]);
        Integer b = Integer.parseInt(args[1]);

        System.out.println(a / b);
    } catch (ArithmeticException e) {
        System.out.println(x: "Division by Zero Not Allowed");
    } catch (NumberFormatException e) {
        System.out.println(x: "Please pass integers only");
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println(x: "Please pass atleast 2 parameters");
    } catch (Exception e) {
        System.out.println(x: "Some Other Expection Occured");
    }
}
```

multiple catch statements!

```
● architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
● architaggarwal@Archits-MacBook-Air System Design % java Solution 10 2
  5
● architaggarwal@Archits-MacBook-Air System Design % java Solution 10
  Please pass atleast 2 parameters
● architaggarwal@Archits-MacBook-Air System Design % java Solution 10 abc
  Please pass integers only
● architaggarwal@Archits-MacBook-Air System Design % java Solution 10 0
  Division by Zero Not Allowed
```

Valid orders of try, catch

```
try{ }
```

✗ only try not possible

```
catch{ }
```

✗ only catch not possible

```
try{
    try{ }
    catch(e){ }
}
catch{ }
```

both are valid

```
try{ }
catch(e1){ }
catch(e2){ }
   ⋮
catch(ek){ }
```

✓ multiple catch

```
try{ }
catch{
        try{ }
        catch(e){ }
}
```

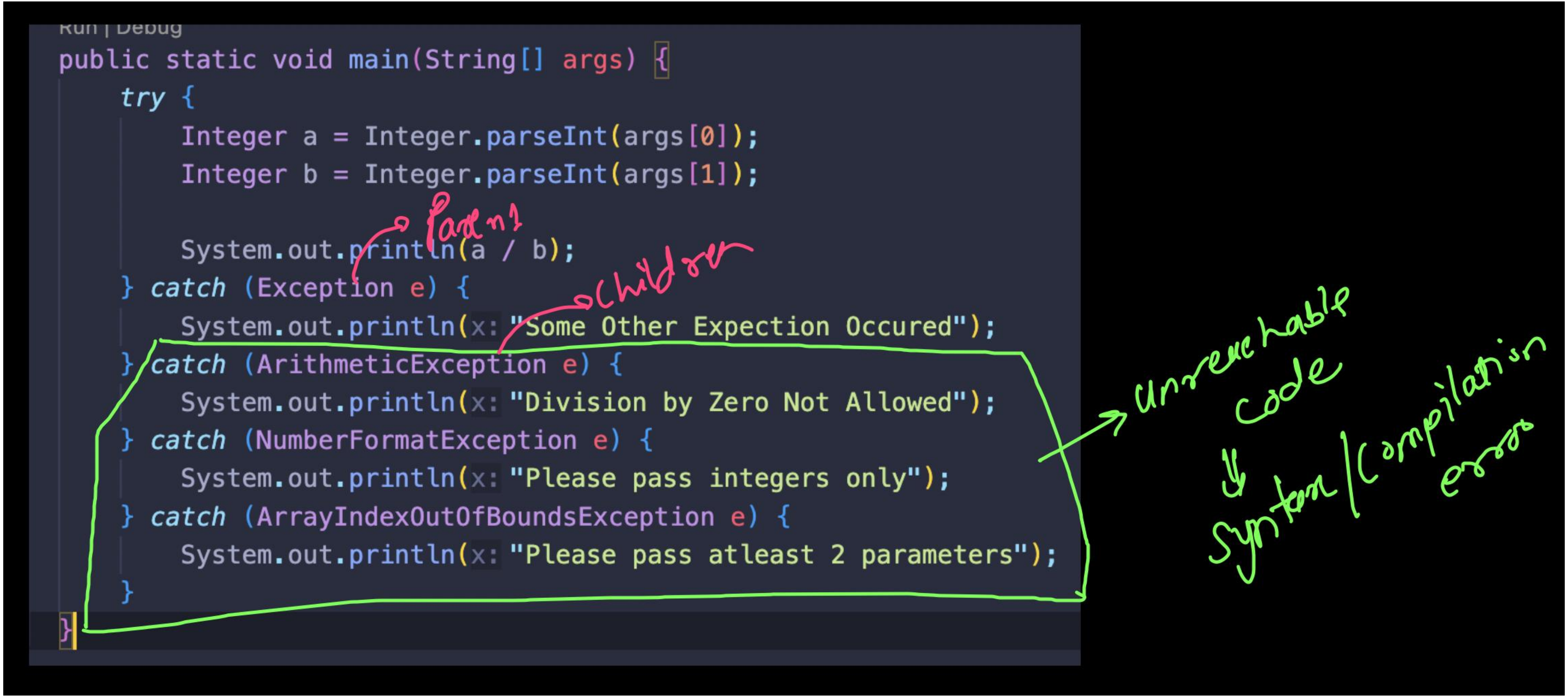
```
catch( ){ }
  try { }
```

✗ not possible

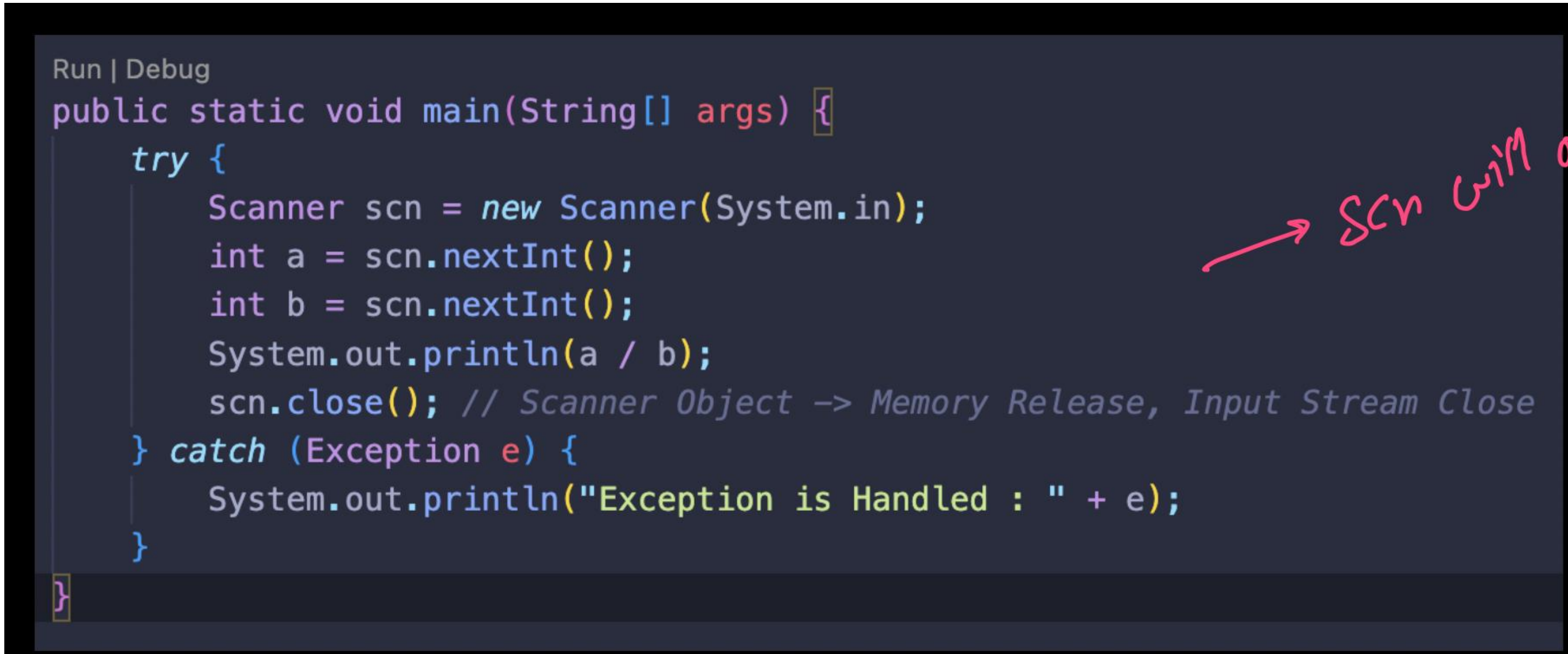
```
try{ --- }
Syso("In b/w try & catch")
catch( ){ }
```

✗ not possible

✓ Nested Try Loop

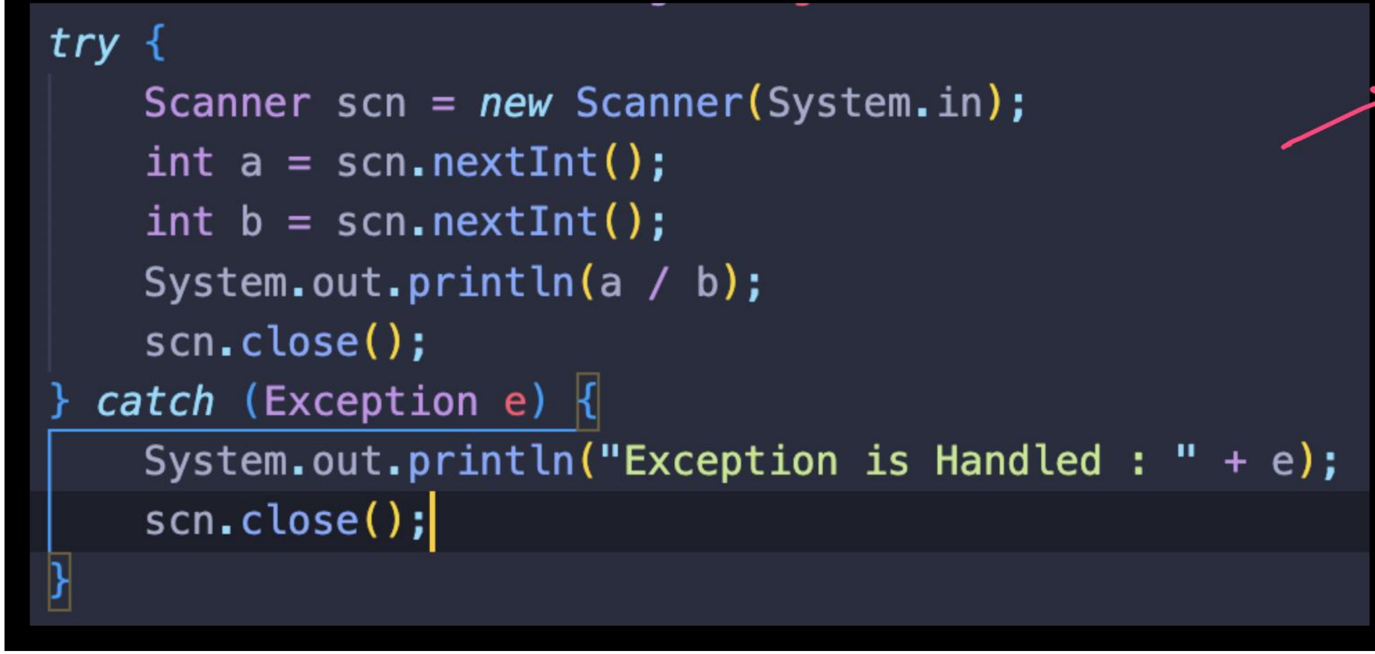


```
Run | Debug
public static void main(String[] args) {
    try {
        Integer a = Integer.parseInt(args[0]);
        Integer b = Integer.parseInt(args[1]);

        System.out.println(a / b);
    } catch (Exception e) {
        System.out.println(x: "Some Other Expection Occured");
    } catch (ArithmeticException e) {
        System.out.println(x: "Division by Zero Not Allowed");
    } catch (NumberFormatException e) {
        System.out.println(x: "Please pass integers only");
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println(x: "Please pass atleast 2 parameters");
    }
}
```



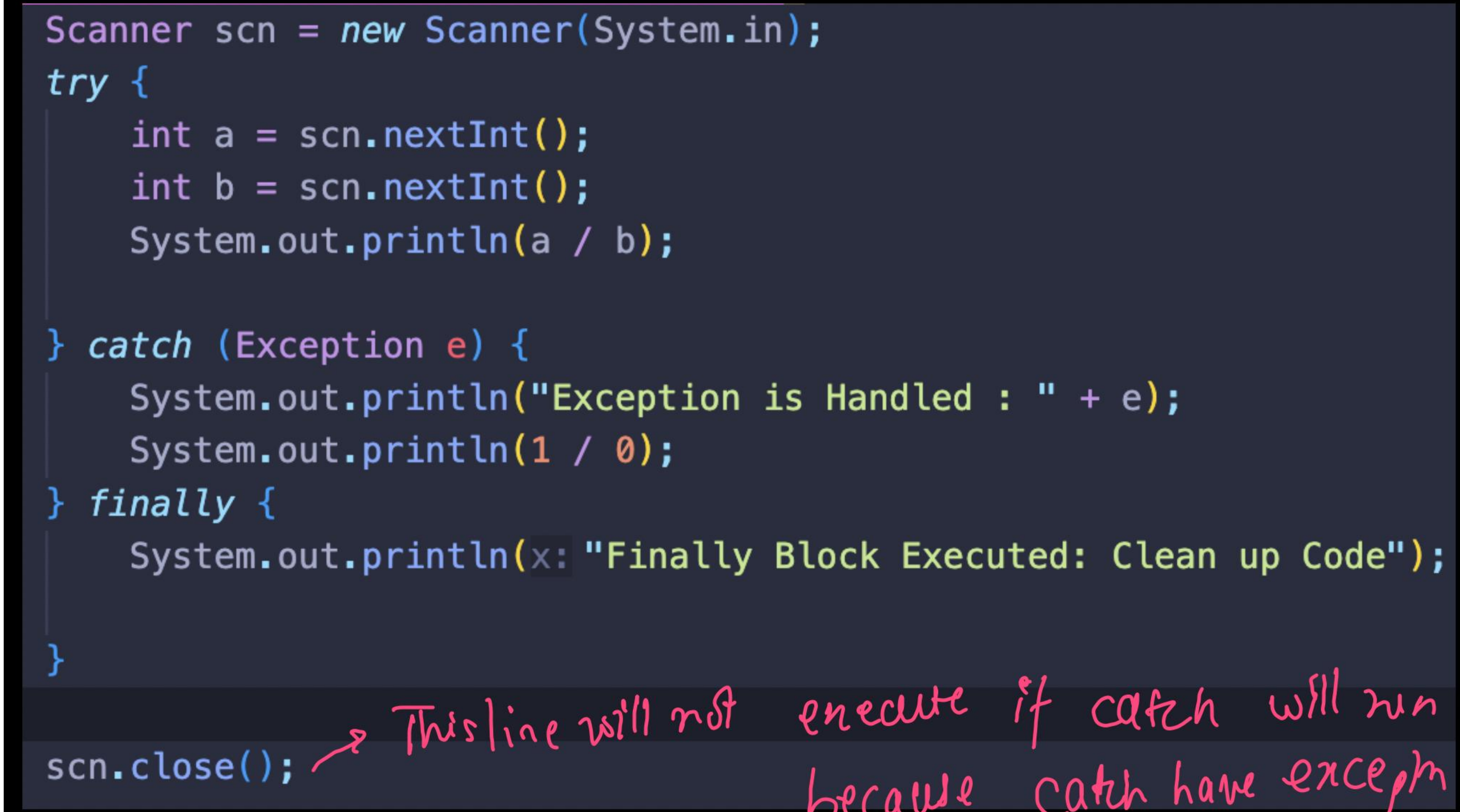
```java
public static void main(String[] args) {
    try {
        Scanner scn = new Scanner(System.in);
        int a = scn.nextInt();
        int b = scn.nextInt();
        System.out.println(a / b);
        scn.close(); // Scanner Object -> Memory Release, Input Stream Close
    } catch (Exception e) {
        System.out.println("Exception is Handled : " + e);
    }
}
```

→ Scn will only close if Exception is not occurred



```java
try {
    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();
    System.out.println(a / b);
    scn.close();
} catch (Exception e) {
    System.out.println("Exception is Handled : " + e);
    scn.close();
}
```

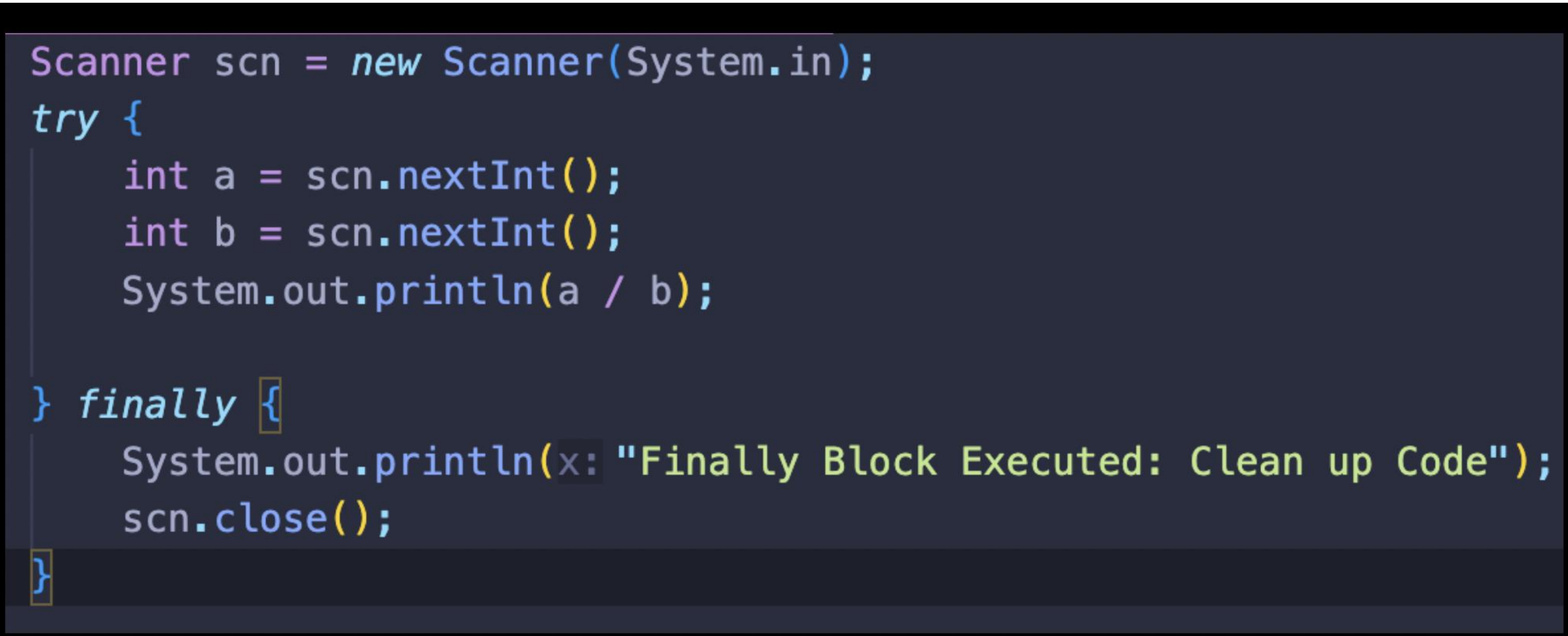
→ Code redundancy



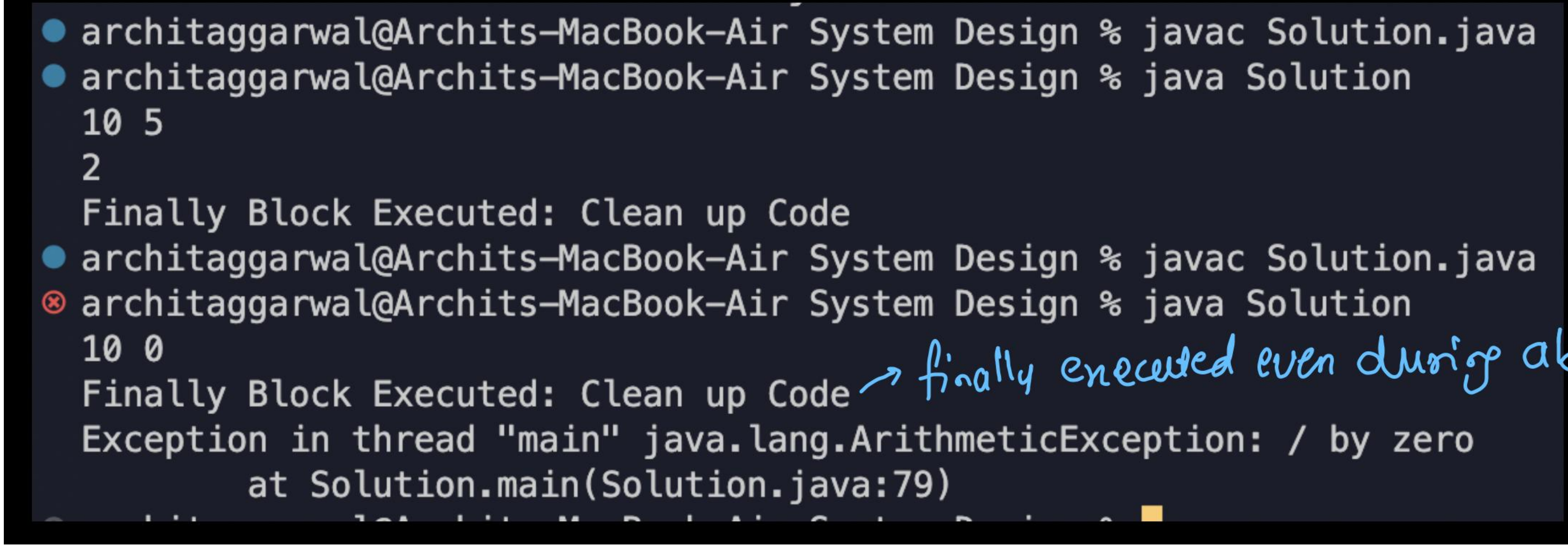
```java
Scanner scn = new Scanner(System.in);
try {
    int a = scn.nextInt();
    int b = scn.nextInt();
    System.out.println(a / b);

} catch (Exception e) {
    System.out.println("Exception is Handled : " + e);
    System.out.println(1 / 0);
} finally {
    System.out.println(x: "Finally Block Executed: Clean up Code");

}

scn.close();
```

This line will not execute if catch will run because catch have exceptn

↳ abnormal termination!

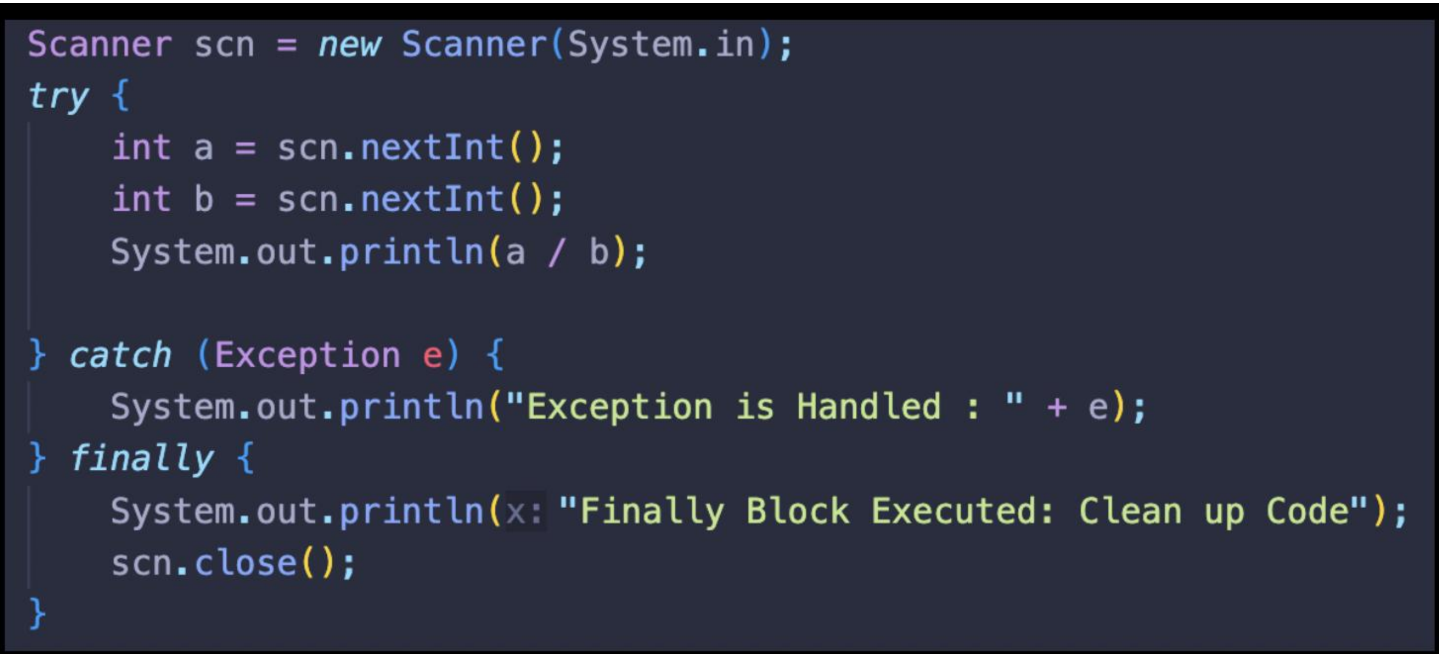


```java
Scanner scn = new Scanner(System.in);
try {
    int a = scn.nextInt();
    int b = scn.nextInt();
    System.out.println(a / b);

} finally {
    System.out.println(x: "Finally Block Executed: Clean up Code");
    scn.close();
}
```

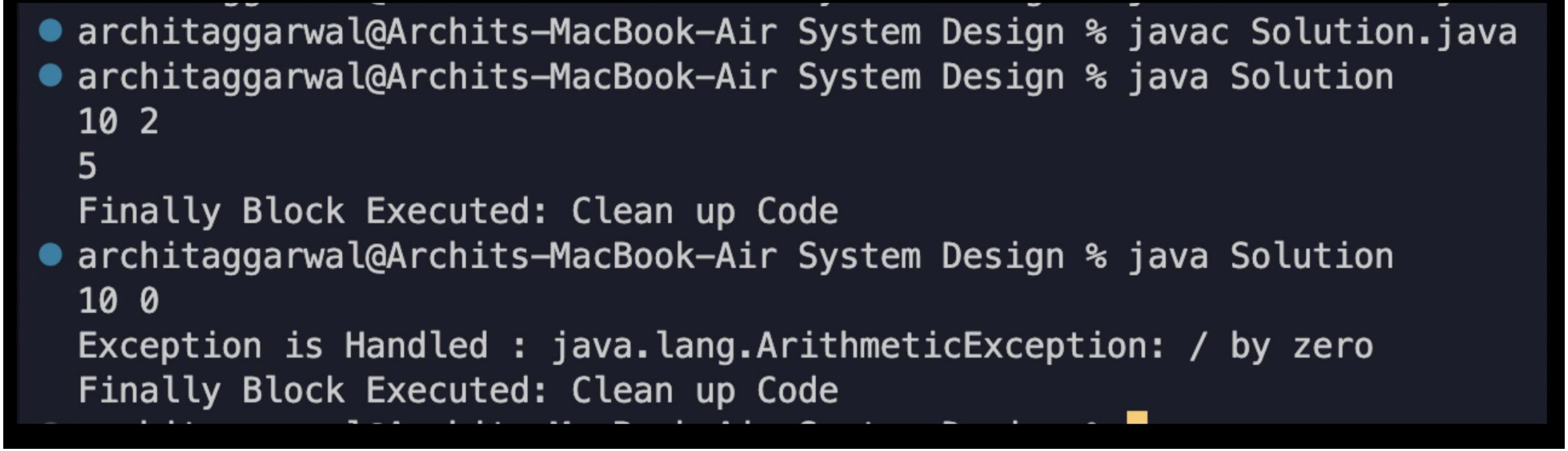


```
● architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
● architaggarwal@Archits-MacBook-Air System Design % java Solution
  10 5
  2
  Finally Block Executed: Clean up Code
● architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
⊗ architaggarwal@Archits-MacBook-Air System Design % java Solution
  10 0
  Finally Block Executed: Clean up Code
  Exception in thread "main" java.lang.ArithmeticException: / by zero
          at Solution.main(Solution.java:79)
```

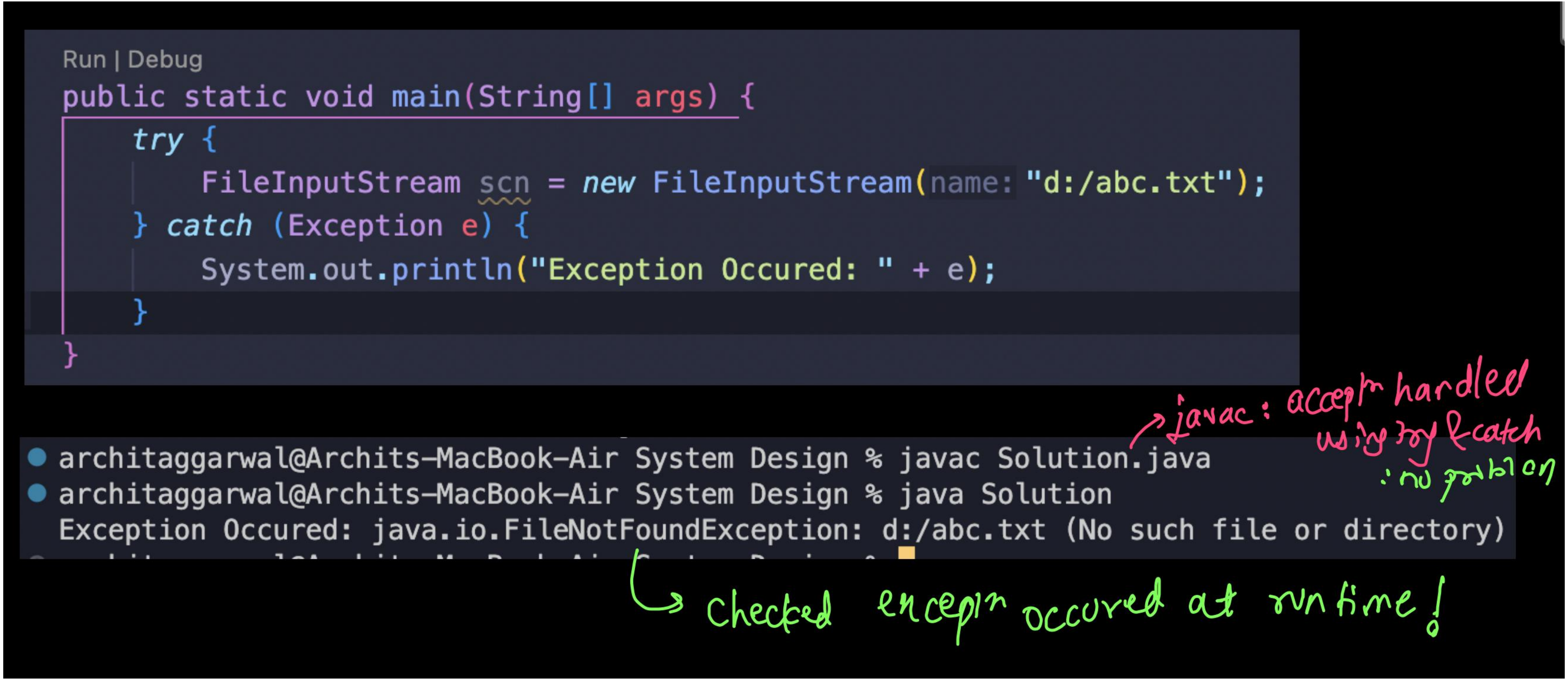
→ finally executed even during abnormal termination!



```java
Scanner scn = new Scanner(System.in);
try {
    int a = scn.nextInt();
    int b = scn.nextInt();
    System.out.println(a / b);

} catch (Exception e) {
    System.out.println("Exception is Handled : " + e);
} finally {
    System.out.println(x: "Finally Block Executed: Clean up Code");
    scn.close();
}
```



```
● architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
● architaggarwal@Archits-MacBook-Air System Design % java Solution
  10 2
  5
  Finally Block Executed: Clean up Code
● architaggarwal@Archits-MacBook-Air System Design % java Solution
  10 0
  Exception is Handled : java.lang.ArithmeticException: / by zero
  Finally Block Executed: Clean up Code
```



```
Run | Debug
public static void main(String[] args) {
    try {
        FileInputStream scn = new FileInputStream(name: "d:/abc.txt");
    } catch (Exception e) {
        System.out.println("Exception Occured: " + e);
    }
}
```

```
● architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
● architaggarwal@Archits-MacBook-Air System Design % java Solution
  Exception Occured: java.io.FileNotFoundException: d:/abc.txt (No such file or directory)
○ architaggarwal@Archits-MacBook-Air System Design %
```

javac : accep^n handled using try & catch : no problem

Checked excep^n occured at runtime!