

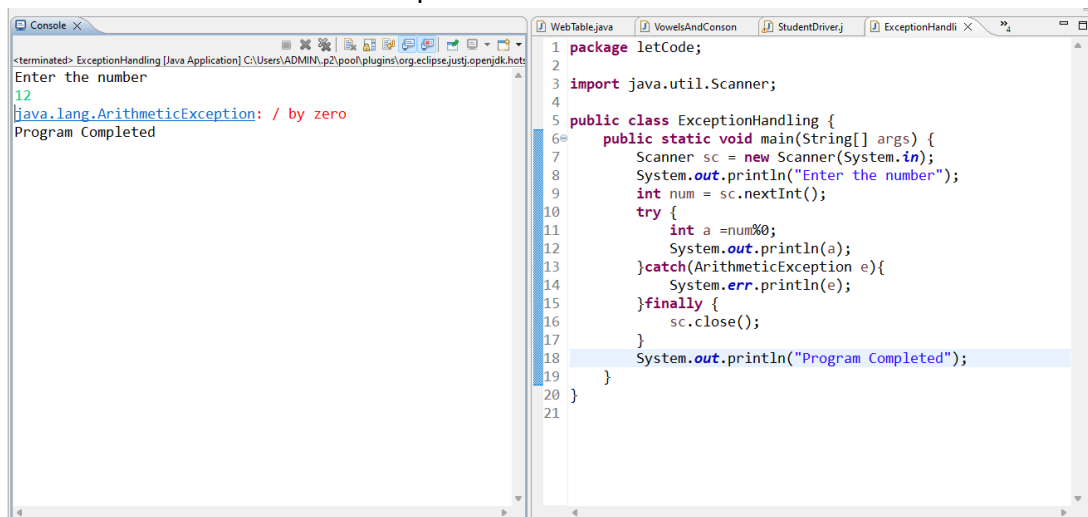
Exception Handling

1. What is an Exception?

➔ It is an unexpected break in the program's flow that can cause the program to crash.

2. How the exception is handled in java?

➔ In Java, exceptions are handled using a "try-catch" block, where the code that might throw an exception is placed within the "try" block, and the "catch" block contains the code to handle the exception.



The screenshot shows an IDE with two panels. The left panel is the Console, and the right panel is the Source Editor.

Console Output:

```
<terminated> ExceptionHandling [Java Application] C:\Users\ADMIN\p2\poo\plugins\org.eclipse.justj.openjdk.hot
Enter the number
12
java.lang.ArithmeticException: / by zero
Program Completed
```

Source Code (WebTable.java):

```
1 package letCode;
2
3 import java.util.Scanner;
4
5 public class ExceptionHandling {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         System.out.println("Enter the number");
9         int num = sc.nextInt();
10        try {
11            int a = num%0;
12            System.out.println(a);
13        } catch (ArithmeticException e) {
14            System.err.println(e);
15        } finally {
16            sc.close();
17        }
18        System.out.println("Program Completed");
19    }
20 }
21 }
```

3. Difference Between Error and Exception in Java?

➔ **Error:** It is a mistake done in a code by the programmer.

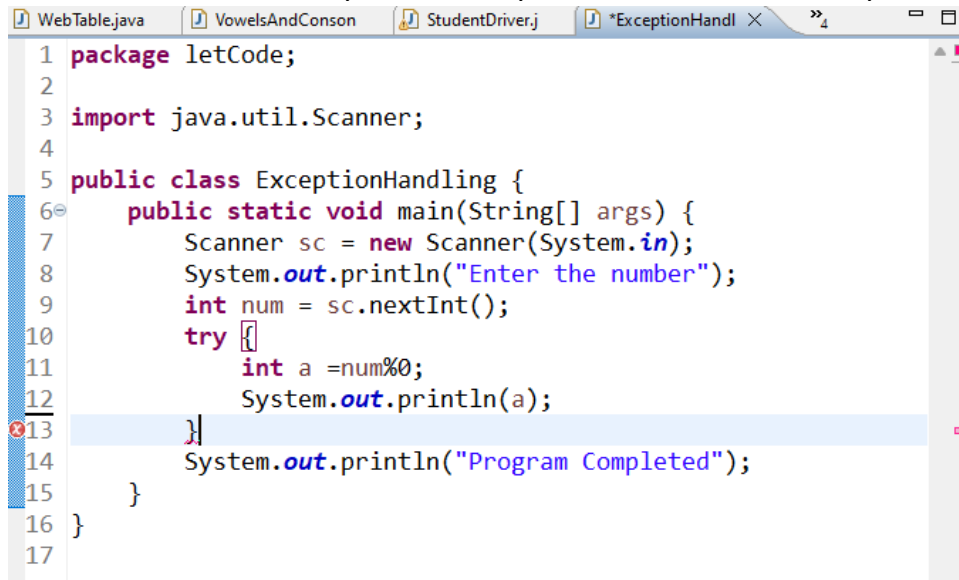
➔ **Exception:** It is an unexpected break in the program's flow, that can cause the program to crash.

4. Can we keep other statements in between try, catch and finally blocks?

➔ **No.** Between the try, catch, and finally blocks, no further statements should be written. They work together as a single entity.

5. Can we write try block without catch and finally block?

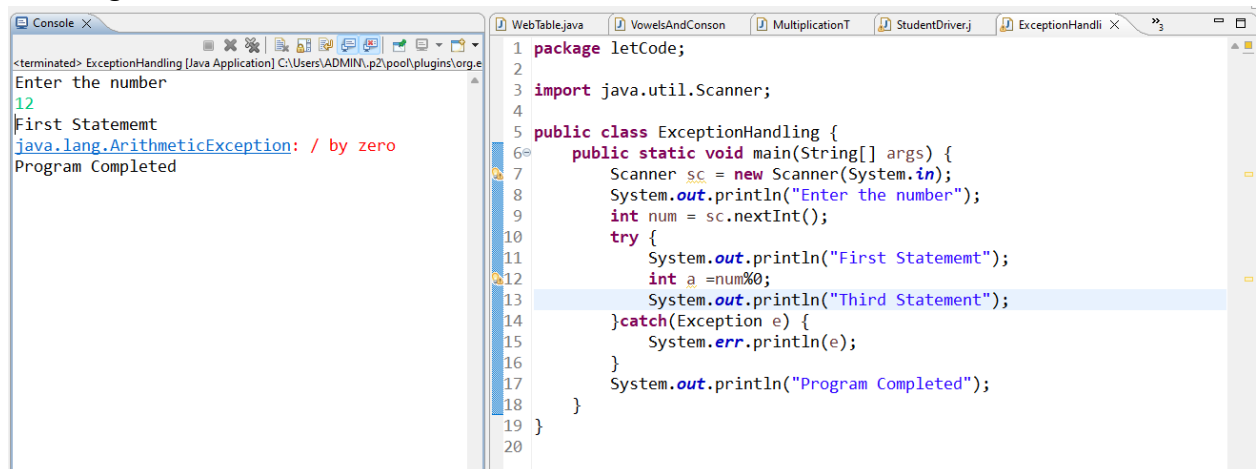
➔ No, it is not mandatory that each try block must be followed by a catch block in Java.



```
1 package letCode;
2
3 import java.util.Scanner;
4
5 public class ExceptionHandling {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         System.out.println("Enter the number");
9         int num = sc.nextInt();
10        try {
11            int a = num%10;
12            System.out.println(a);
13        }
14        System.out.println("Program Completed");
15    }
16 }
17
```

6. There are Three Statements in try block – Statement1, Statement2, Statement3. After that there is a catch block to catch the exceptions occurred in the try block. Assume that exception has occurred in statement2, Does the statement3 get executed or not?

➔ No. If the statement2 is having exception within try block then the statement3 will not get executed.



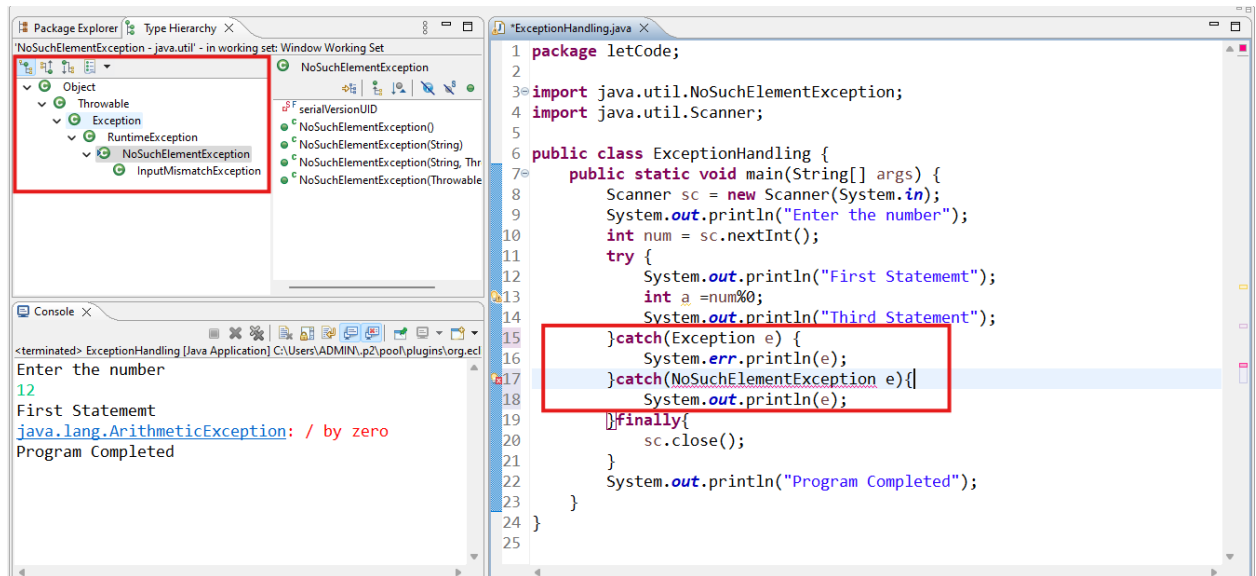
```
1 package letCode;
2
3 import java.util.Scanner;
4
5 public class ExceptionHandling {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         System.out.println("Enter the number");
9         int num = sc.nextInt();
10        try {
11            System.out.println("First Statement");
12            int a = num%10;
13            System.out.println("Third Statement");
14        } catch (Exception e) {
15            System.err.println(e);
16        }
17        System.out.println("Program Completed");
18    }
19 }
20
```

Console Output:

```
<terminated> ExceptionHandling [Java Application] C:\Users\ADMIN\p2\poo\plugins\org.e
Enter the number
12
First Statement
java.lang.ArithmeticException: / by zero
Program Completed
```

7. What is unreachable catch block error? Explain the hierarchy of exceptions in Java?

➔ A block of statements to which the control can never reach under any case can be called as unreachable blocks.



i.e., In a catch, if we mentioned **Exception** and, in another catch, we cannot able to mention the **type of Exception** because the Exception is **parent** of NoSuchElementException. Either we should use NoSuchElementException first, before Exception or we should not use NoSuchElementException once we declared the Exception in catch block.

8. What is Runtime Exception in Java and give example?

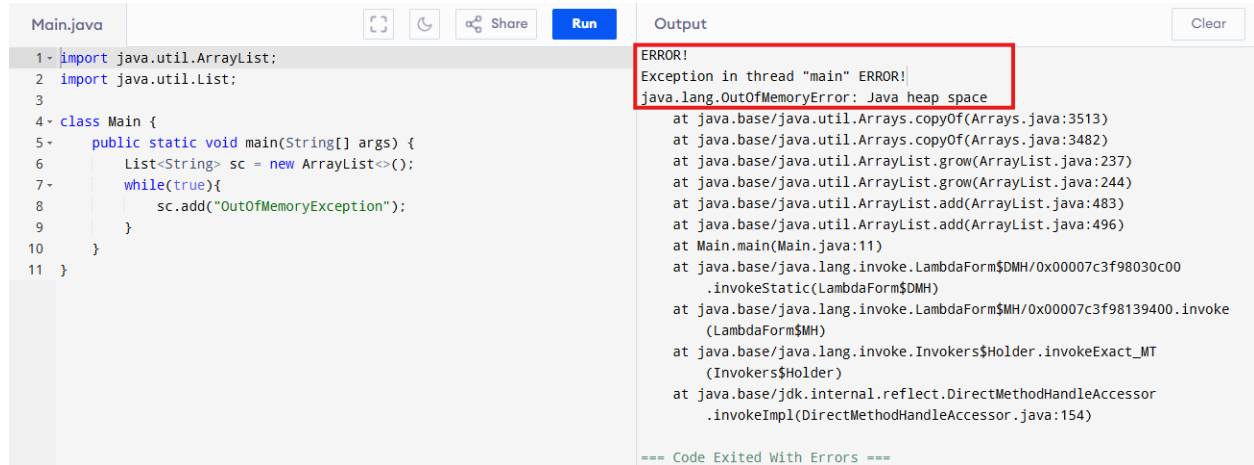
➔ In Java, a **RuntimeException** is a type of **unchecked exception** that occurs during the execution of a program.

Common Examples of RuntimeException:

- **NullPointerException**: Attempting to access or modify a null object.
- **ArrayIndexOutOfBoundsException**: Trying to access an index of an array that is out of bounds.
- **ArithmeticException**: Performing illegal arithmetic operations, like dividing by zero.

9. what is out of memory error in java?

- ➔ In Java, an `OutOfMemoryError` is a **serious runtime error** that occurs when the Java Virtual Machine (JVM) is unable to allocate enough memory to continue execution. This usually happens when the heap, stack, or other memory areas are exhausted.



```
1- import java.util.ArrayList;
2- import java.util.List;
3-
4- class Main {
5-     public static void main(String[] args) {
6-         List<String> sc = new ArrayList<>();
7-         while(true){
8-             sc.add("OutOfMemoryException");
9-         }
10-    }
11- }
```

Output

```
ERROR!
Exception in thread "main" ERROR!
java.lang.OutOfMemoryError: Java heap space
    at java.base/java.util.Arrays.copyOf(Arrays.java:3513)
    at java.base/java.util.Arrays.copyOf(Arrays.java:3482)
    at java.base/java.util.ArrayList.grow(ArrayList.java:237)
    at java.base/java.util.ArrayList.grow(ArrayList.java:244)
    at java.base/java.util.ArrayList.add(ArrayList.java:483)
    at java.base/java.util.ArrayList.add(ArrayList.java:496)
    at Main.main(Main.java:11)
    at java.base/java.lang.invoke.LambdaForm$DMH/0x00007c3f98030c00
        .invokeStatic(LambdaForm$DMH)
    at java.base/java.lang.invoke.LambdaForm$MH/0x00007c3f98139400.invoke
        (LambdaForm$MH)
    at java.base/java.lang.invoke.Invokers$Holder.invokeExact_MT
        (Invokers$Holder)
    at java.base/jdk.internal.reflect.DirectMethodHandleAccessor
        .invokeImpl(DirectMethodHandleAccessor.java:154)

=== Code Exited With Errors ===
```

10. What are Checked and Unchecked exception in Java?

- ➔ **Checked Exception:** The Exception that are checked at Compile Time. The java compiler forces the developer to handle these exceptions using **try-catch** or declare them in the method signature using **throws**.
- ➔ **Unchecked Exception:** Also called as Runtime Exceptions, this will **not-check** at compile time, These occur due to programming logic errors and **do not require explicit handling**.

11. Difference between `ClassNotFoundException` and `NoClassDefFoundError` in java?

- ➔ Both `ClassNotFoundException` and `NoClassDefFoundError` in Java are related to **missing classes**, but they occur in different scenarios.

ClassNotFoundException	NoClassDefFoundError
1. The requested class file is not available in the classpath.	➔ The class was compiled successfully but deleted or moved before runtime.
2. The class name is misspelled while loading it dynamically.	➔ A dependency issue in a JAR file prevents the class from loading. The class depends on another class that is missing.