Exception:

Exception is run-time error. Normally whenever Exception occurs, the flow of execution just stops. Program will stop executing further even if it has some more statements. And you will get bunch of red lines. Which is not a good sign of Programming. We must handle the Exceptions. The Process of handling an Exception is called Exception Handling.

Types Of Exception:

Java has two types of Exception:

1) In Built Exception:

- Arithmetic Exception: It is thrown when an exceptional condition has occurred in an arithmetic operation.
- NullPointerException: This exception is raised when referring to the members of a null object. Null represents nothing
- ArrayIndexOutOfBoundException: It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.
- IOException: It is thrown when an input-output operation failed or interrupted.
- NoSuchFieldException: It is thrown when a class does not contain the field (or variable) specified.
- NoSuchMethodException: when accessing a method which is not found.

2) User-Defined Exception:

- LessAmountException
- FileNotFoundException

We can create Exception based on our needs.

Another two types are Checked Exception and UnChecked Exception:
Difference: Checked are noticed by compiler at compile time, and Unchecked aren't noticed by Compiler. They must be handled at runtime

Example Of Getting An Exception:

```
public class Unchecked_Demo {
   public static void main(String args[]) {
      int num[] = {1, 2, 3, 4};
      System.out.println(num[5]);
   }
}
```

```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 5

atExceptions.Unchecked_Demo.main(Unchecked_Demo.java:8)
```

Look at the example, We tried accessing the value from the array, which doesn't exist. And then got an ArrayIndexOutOfBound exception.

Example Of Handling An Exception:

Basic Structure:

```
 try {
// Protected code
} catch (ExceptionType1 e1) {
   // Catch block
}
```

Whenever you get suspicious, like “Oh, this portion of code will create problem. Always put them into **try** block. If there is an Exception the program control will be transferred to **catch** block. And your handling code should be there, in **catch** block.

Let’s talk about **finally** block as well,
finally block follows catch block. Finally block does not care if Exception has occurred or not. It will always execute.

```
import java.io.*;

public class ExcepTest {


   public static void main(String args[]) {
      try {
         int a[] = new int[2];
         System.out.println("Access element three :" + a[3]);
      } catch (ArrayIndexOutOfBoundsException e) {
```

```
            System.out.println("Exception thrown  :" + e);
        }
     finally {
            System.out.println("Finally Block Executes”);
         }
    }
}
```

User Defined Exception:

Before diving into User Defined Exception, let’s first talk about hierarchy of Exception:

There is one **Exception** class in java, which you have to extend while creating an User Defined Exception.

Now that **Exception** class is child of **Throwable** class. Throwable class has 2 children.

1) Errors

2) Exception

Now this **Throwable** class is Child of **Object** class, and **Object** class is part of **java.lang** package.

Figure:

java.lang → Object → Throwable → Error & Exception

Let's understand this topic with an Example:

```
public class InSufficient extends Exception {
   public InSufficient() {
      //Exception Got Initialized.
   }
    public String toString() {
      //This method will return following string to the catch
       block.
     return "Sorry, you don't have enough money";
   }
}
```

We created one class **InSufficient** which extends the **Exception** class.

```
public class DemoException {

   public static void main(String [] args) {

    int bankAmt = 1000;
    int withdrawAmt = 10000;
      try {
         if(withdrawAmt > bankAmt){
               throw new InSufficient();
```

```
            }
        } catch (InSufficient e) {
            System.out.println(e);
        }
    }
}
```

the **throw** keyword will create a new instance of **Insufficient** exception, and then It'll throw that **Insufficient exception** in catch block.

ToString() method in Exception will return some statements, that will be received in a **Catch** block. In e variable.

Now that little buddy 'e' has many methods within it.

1) getMessage() : Will fetch the error message.

2) getCause() : Will show you the cause of Error.

3) toString(): Same as getMessage() but will get the class name also.

This is everything you need to understand about Exception.

```
public class Main
{
        public static void main(String[] args) {
        int bankAmt = 1000;
    int withdrawAmt = 10000;
      try {
        if(withdrawAmt > bankAmt){
                        throw new InSufficient();
                }
      } catch (InSufficient e) {
        System.out.println(e);
      }
        }
}

class InSufficient extends Exception {
    InSufficient() {
      //Exception Got Initialized.
   }
    public String toString() {
      //Exception Got Initialized.
        return "Do Not Have Enough Money";
   }
}
```