

3) `java.util.InputMismatchException` :

\* While reading the data from the Scanner class, We should read appropriate value based on Scanner class method otherwise we will get `java.util.InputMismatchException`

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter Your Roll :");
int roll = sc.nextInt();
System.out.println("Your Roll Number is :" + roll);
```

Note : If we will read any value except integer then we will get Exception

4) `java.lang.NegativeArraySizeException` :

\* Array size must be positive integer only, If we try to pass negative size then we will get an exception `java.lang.NegativeArraySizeException`

```
int size = -12;
int []arr = new int[size];
```

5) `java.lang.NumberFormatException` :

\* If a reference variable is pointing to null and by using that reference variable, If we are accessing any non static field OR non static method then we will get `java.lang.NullPointerException`.

```
String str = "NIT";
int val = Integer.parseInt(str);
System.out.println("Value is :" + val);

String str = "Java";
Integer obj = Integer.valueOf(str);
System.out.println(obj);
```

6) `java.lang.NullPointerException` :

\* If a reference variable is pointing to null and by using that reference variable, If we are accessing any non static field OR non static method then we will get `java.lang.NullPointerException`

Case 1 :

```
String str = null;
System.out.println(str.length());
```

Case 2 :

```
String str = "null";
System.out.println(str.length());
```

Note : It is valid String so, We will not get NPE

Case 3 :

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter a String value");
String str = sc.nextLine();
System.out.println(str.length());
```

`java.lang.ArrayStoreException` :

\* In an array, If we try to insert Illegal data then It will generate an Exception i.e `java.lang.ArrayStoreException`

```
Object obj[] = new String[3];
obj[0] = "Scott";
obj[1] = "Smith";
obj[2] = 12;
```

\*\*\*Exception Hierarchy :

```
classDiagram
    class Throwable(C)
    class Error
    class RuntimeException
    class Exception
    class UncheckedExceptions {
        class LinkageError
        class VerifyError
        class NoClassDefFoundError
        class VirtualMachineError
        class StackOverflowError
        class OutOfMemoryError
        class ArithmeticException
        class NullPointerException
        class NumberFormatException
        class ArrayStoreException
        class ClassCastException
        class NegativeArraySizeException
        class InputMismatchException
        class NoSuchElementException
        class EmptyStackException
        class IndexOutOfBoundsException
        class ArrayIndexOutOfBoundsException
        class StringIndexOutOfBoundsException
    }
    class CheckedExceptions {
        class FileNotFoundException
        class ClassNotFoundException
        class IOException
        class EOFException
        class SQLException
        class InterruptedException
        class CloneNotSupportedException
    }
    Error <|-- RuntimeException
    RuntimeException <|-- UncheckedExceptions
    RuntimeException <|-- Exception
    Exception <|-- CheckedExceptions
```

\* A developer is not responsible for Error because We cannot recover Error, A developer is responsible for Exception.

WAP in java that shows `Exception` is the super class of all the Exceptions we have in java.

```
package com.ravi.exception_demo;

import java.io.FileNotFoundException;
import java.io.IOException;

public class ExceptionDemo
{
    public static void main(String[] args)
    {
        Exception e1 = new ArithmeticException();
        System.out.println(e1.toString());

        Exception e2 = new ArithmeticException("Divide By Zero");
        System.out.println(e2.toString());

        Exception e3 = new FileNotFoundException();
        System.out.println(e3);
    }
}
```

Exception format :

The java software people has provided the format of exception so whenever we print exception object by using `toString()` then the format is

Fully Qualified Name : errorMessage

Package Name + Class Name : errorMessage