Object Oriented Programming with Java (Subject Code: BCS-403)

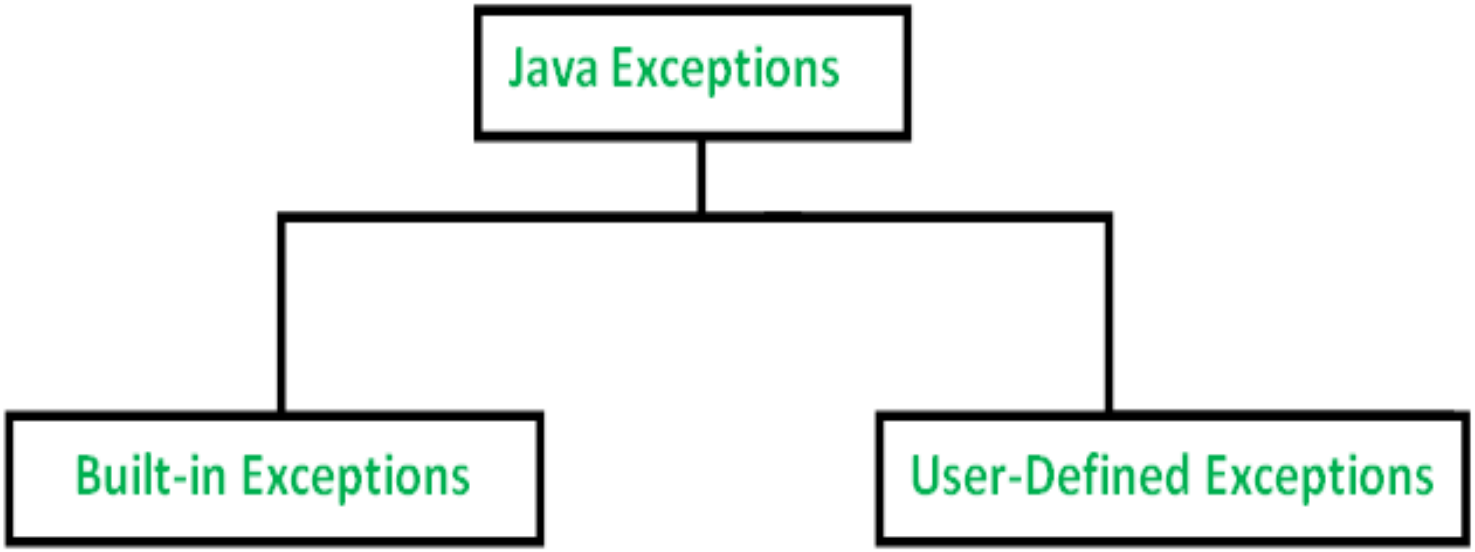
Unit 2

Lecture 16

Lecture 16

- In-built and User Defined Exceptions

Java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.



Built-in Exceptions

Built-in exceptions are the exceptions that are available in Java libraries. These exceptions are suitable to explain certain error situations.

- **ArithmeticException:** It is thrown when an exceptional condition has occurred in an arithmetic operation.
- **ArrayIndexOutOfBoundsException:** 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.
- **ClassNotFoundException:** This Exception is raised when we try to access a class whose definition is not found
- **FileNotFoundException:** This Exception is raised when a file is not accessible or does not open.

IOException: It is thrown when an input-output operation failed or interrupted

- **InterruptedException:** It is thrown when a thread is waiting, sleeping, or doing some processing, and it is interrupted.
- **NoSuchFieldException:** It is thrown when a class does not contain the field (or variable) specified
- **NoSuchMethodException:** It is thrown when accessing a method that is not found.
- **NullPointerException:** This exception is raised when referring to the members of a null object. Null represents nothing

NumberFormatException: This exception is raised when a method could not convert a string into a numeric format.

- **RuntimeException:** This represents an exception that occurs during runtime.
- **StringIndexOutOfBoundsException:** It is thrown by String class methods to indicate that an index is either negative or greater than the size of the string
- **IllegalArgumentException :** This exception will throw the error or error statement when the method receives an argument which is not accurately fit to the given relation or condition. It comes under the unchecked exception.

```
// Java program to demonstrate ArithmeticException
class ArithmeticException_Demo
{
    public static void main(String args[])
    {
        try {
            int a = 30, b = 0;
            int c = a/b;  // cannot divide by zero
            System.out.println ("Result = " + c);
        }
        catch(ArithmeticException e) {
            System.out.println ("Can't divide a number by 0");
        }
    }
}
```

//Java program to demonstrate NullPointerException

```
class NullPointer_Demo
{
    public static void main(String args[])
    {
        try {
            String a = null; //null value
            System.out.println(a.charAt(0));
        } catch(NullPointerException e) {
            System.out.println("NullPointerException..");
        }
    }
}
```

// Java program to demonstrate NumberFormatException

```
class  NumberFormat_Demo
{
    public static void main(String args[])
    {
        try {
            // "akki" is not a number
            int num = Integer.parseInt ("akki") ;

            System.out.println(num);
        } catch(NumberFormatException e) {
            System.out.println("Number format exception");
        }
    }
}
```

User-Defined Exceptions

- the built-in exceptions in Java are not able to describe a certain situation.
- In such cases, the user can also create exceptions which are called ‘user-defined Exceptions’.
- The user should create an exception class as a subclass of the Exception class.
- Since all the exceptions are subclasses of the Exception class, the user should also make his class a subclass of it.

Create user-defined Exception

The following steps are followed for the creation of a user-defined Exception.

class MyException extends Exception

We can write a default constructor in his own exception class.

MyException(){}

We can also create a parameterized constructor with a string as a parameter.

We can use this to store exception details. We can call the superclass(Exception) constructor from this and send the string there.

```
MyException(String str)
{
   super(str);
}
```

To raise an exception of a user-defined type, we need to create an object to his exception class and throw it using the throw clause, as:

```
MyException me = new MyException("Exception details");
throw me;
```