Exception Handling

➢ What are Exceptions?

- Exceptions are **Runtime Errors**.
- There are various types of errors
  - Syntax Error
  - Logical Error
  - Runtime Error.
  -

- **Syntax Error**
- Syntax and Logical errors are faced by Programmers, and runtime errors are faced by user.
- Spelling or grammatical mistakes are syntax errors, for example using uninitialised variable it and using undefined variable etc and missing a semicolon etc.
- Syntax errors can be removed with the help of **compiler.**

- **Logical Error**
- Logical error is a bug in program that it to operate incorrectly, for example missing parenthesis in the calculation.
- Logical errors are removed with the help of **debugger.**

- **Runtime Error.**
- Mishandling of a program causes Runtime error.
- Causes of runtime errors are bad input, unavailability of resources.
- Major problems with runtime errors is program will crash.
- Exception handling is process of responding to the runtime errors.

➢ How to Handle Exceptions

- **Try and Catch Exceptions**

Example program

```
public class MyClass

{
  public static void main(String[] args)
  {
    try
    {
      int[] myNumbers = {1, 2, 3};
      System.out.println(myNumbers[10]);
    }
    catch (Exception e)
    {
      System.out.println("Something went wrong.");
    }
  }
}
```

- The try statement allows you to define a block of code to be tested for errors while it is being executed.
- The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.
- The try and catch keywords come in pairs.
- A try block can have **Multiple catch blocks**
- Try and catch block can be nested.
- When a try catch block is present in another try block then it is called a **nested try catch block.**

➢ Class Exception

- Object is the mother class for all the java classes.
- Exception is the parent class for all the exceptions.
- **ClassNotFoundException** this exception is raised when the object is used but the class is not found.
- **IOException** accessing input output, mostly accessing files where files are not there or file is corrupted one of the famous exception is the FileNotFoundException.
- **InterruptedException** it is related to Multithreading if a thread stops abnormally it throws an Interrupted exception
- **NumberFormatException** when input is given as a number string form but not have a proper number, then it throws NumberFormatException.
- **RuntimeExceptions** under these there are ArithmeticExceptions, IndexOutOfBoundException, NullPointerException.

- Exception classes are categorised into two types **checked exception** and **unchecked exception.**
- Checked exception must be handle by try and catch, java compiler forces you to write try and catch.
- Unchecked exception are not mandatory to be handled Only Runtime Exceptions are the unchecked exceptions.

- **Methods of class exceptions used for Error Message**
    - string getMessage()
    - string toString()
- these method returns a string with a message written about exception, purpose of both the methods is same, but the usage of both the methods are different.

➢ Throw VS Throws

- **throw** keyword is used to throw an exception logically.
- Only one exception can be thrown at a time by using throw keyword.
- It is used within the method.
- It is followed by instance variable.
- **throws** is used for declaring that a method may throw exception
- **throws** is written in signature of method

Example program

```
public class GFG {

  public static void main(String[] args)
  {
   // Use of unchecked Exception
   try {
    // double x=3/0;
    throw new ArithmeticException();
   }
   catch (ArithmeticException e)
   {
     e.printStackTrace();
   }
  }
}
```

-

Example program

```
import java.io.IOException;


public class UseOfThrowAndThrows {

    public static void main(String[] args)
        throws IOException
    {
    }
}
```

➢ Try with Resources

- All the things that are outside the program are resource to a program.
- Heap is also a resource to a program.
- Whenever a program needs a resource it should acquire it and when do not need we should return it.
- To write an object in heap we write new.
- In java heap memory objects are deallocated automatically by garbage collector.
- Finally keyword is used in association with a try/catch block.
- Finally keyword is meant to execute whether an exception occurs or not.
- Resources are needed to be closed in finally block.
- The try-with-resources statement is a try statement that declares one or more resources.
- The try-with-resources statement ensures that each resource is closed at the end of the statement.

Example program

```
static String readFirstLineFromFile(String path) throws
IOException
{
    try (BufferedReader br =new BufferedReader(new
FileReader(path)))
   {
    return br.readLine();
    }
}
```