EXCEPTION HANDLING

CSE215 | DAFFODIL INTERNATIONAL UNIVERSITY

OBJECT ORIENTED PROGRAMMING LAB

SESSION - 08

08

**What is exception?**

* Exception is a situation faced by a program while running or executing which disrupts and terminates the program abnormally.

For example:

int x=0,y=0;   
 int z=x/y; // here 0/0 which is an arithmetic exception. Which detected at runtime

2.

String s=null;

System.out.println(s.length()); //NullPointerException

3.

int a[]=new int[5];

a[10]=50; //ArrayIndexOutOfBoundsException

**Exception Handing:**

-Exception Handling is a mechanism to handle runtime errors.

**How to handle exception?**

- We use **try-catch-finally** block to handle exception.

* Java try block is used to enclose the code that might throw an exception or may cause error.
* Java catch block is used to handle the Exception. It must be used after the try block only. Multiple catch block can be used with a single try.

try {

// Protected code where error may cause

}

Catch (ExceptionName e) {

// Catch block   
 }

finally

{

// block of code to be executed after try block ends

}

**EXAMPLE 8.1:**

int x=0,y=0,z;

try

{

System.out.println("z = "+(x/y));

}

catch(ExceptionName e)

{

e.printStackTrace();

}

finally

{

System.out.println("Done!!");

}

**Built in exception**

**ilt in exception?useme e1)ck to handle exception.while running Static keyword\*\*,**

|  |  |
| --- | --- |
| **Exception** | **Meaning** |
| ArithmeticException | Arithmetic error, such as divide-by-zero. |
| ArrayIndexOutOfBoundsException | Array index is out-of-bounds. |
| ArrayStoreException | Assignment to an array element of an incompatible type. |
| ClassCastException | Invalid cast. |
| EnumConstantNotPresentException | An attempt is made to use an undefined enumeration value. |
| IllegalArgumentException | Illegal argument used to invoke a method. |
| IllegalMonitorStateException | Illegal monitor operation, such as waiting on an unlocked thread. |
| IllegalStateException | Environment or application is in incorrect state. |
| IllegalThreadStateException | Requested operation not compatible with current thread state. |
| IndexOutOfBoundsException | Some type of index is out-of-bounds. |
| NegativeArraySizeException | Array created with a negative size. |
| NullPointerException | Invalid use of a null reference. |
| NumberFormatException | Invalid conversion of a string to a numeric format. |
| SecurityException | Attempt to violate security. |
| StringIndexOutOfBounds | Attempt to index outside the bounds of a string. |
| TypeNotPresentException | Type not found. |
| UnsupportedOperationException | An unsupported operation was encountered. |

**Create user defined Exception:**

* Create a subclass of “Exception”
* Call super() with the constructor
* You just need to extend the predefined **Exception** class to create your own Exception. The following **InsufficientFundsException** class is a user-defined exception that extends the Exception class. An exception class is like any other class, containing useful fields and methods.

**Example 8.2:**

import java.io.\*;

public class InsufficientBalanceException extends Exception {

private double amount;

public InsufficientBalanceException(String msg) {

super(msg);

}

}

-**How to use user defined Exception?**

double balance=0;

try{

if(balance<=0)

throw new InsufficientBalanceException(“Insufficient Balance”);

}

catch(Exception e) {

System.out.println(e);

}

**Avoid exception**

* Use the keyword **“throws”** (after method declaration)

**Example 8.3:**

class test{

public static void main(String[] x) **throws exception** {

int x=0,y=0;

System.out.println(x/y);

}

}

**Here main method handles any exception.**

class test{

public static void main(String[] x) **throws exception** {

int x=0,y=0;

System.out.println(x/y);

}

}