JEPPIAAR INSTITUTE OF TECHNOLOGY

“Self-Belief | Self Discipline | Self Respect”

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

LECTURE NOTES

CS8392 – Object Oriented Programming

(Regulation 2017)

Year/Semester: II/03 CSE

2021 – 2022

Prepared by

Ms. R. Revathi

Assistant Professor/CSE

UNIT-III :-

Exception Handling and I/o :-

VR 2m. **Exception :- - Definition:**

An Exception is an abnormal (or) Unusual Condition which may occur at run time..

An Exception may occur due to the following reasons:

✓ Invalid data as input
✓ File Cannot be found/opened
✓ Network Connection may be disturbed.

Based on the above exceptions, the exceptions are classified into three categories.

① Checked Exception:- It is an exception that occurs at Compile time. These exceptions cannot be ignored at the time of Compilation. So, the Programmer should handle these exceptions.

② Unchecked Exception:- An Unchecked exception is an exception that Occurs at run time. These includes Program bugs, Such as logic errors.

Exception Hierarchy:-

$$\begin{tikzcd}
 & \text{java. lang.} \arrow[d] & & \\
 & \text{Object} \arrow[d] & & \\
 & \text{Throwable} \arrow[dl] \arrow[drr] & & \\
\text{Errors} & & & \text{Exceptions} \arrow[dl] \arrow[d] \\
 & & \text{Run time exceptions} & \text{Other exceptions}
\end{tikzcd}$$

- The java. lang. Exception class is the base class for all exception classes. All Exceptions and errors types are sub class of class "Throwable", which is base class of hierarchy. ① One branch is headed by Exception. This class is used for exceptional conditions that user Programs should catch NullpointerException is an example of such a Exception.

② Another branch is headed by Errors that are used by the Java run time System (JVM) to indicate errors. StackOverflowError is an example of such an error.

✓ The Exception class has two main sub classes.

✓ IoException class
✓ RuntimeException class.

VQ: 13m ⓧⓧ

Throwing and Catching Exceptions (or) Exception handling mechanism:-

Consider an example, that divides the 'no' in Java, as follows:

```
Class Example1
 {
  Public static void main (String
                          args[])
   {
      int a=42, d=0, c;
      c=a/d;
      System.out.Println("The division
                          is"+c);
   }
 }
```

In the above example, when the Value of a (ie) 42 when it is divided by the Value of d (ie) 0 it throws the run time exception, which we did not able to Catch the exception. (Uncaught Exception)

In Order to catch and handle the run time exception, we are in need of "Exception handling" mechanism. It uses 5 Keywords in Java".

✓ The 5 Keywords are

① try
② throw
③ Catch
④ throws
⑤ finally.

① try:-

Use:-

✓ A try block is Placed around the Code, that might monitor for exception.

✓ If any exception Occurs inside the try block, then it stops the execution, and throw the exception to the Catch block.

Syntax:-

```
try {
    // monitor for exception.
    // Protected Code.
    }
```

② throw:-

Use:-

✓ throw block is used to throw the exception to Catch block, explicitly

✓ throw block will be automatically thrown the error by JVM. [No need to use throw inside the try block]

Syntax:-

```
try {
    throw // optional.


}
```

③ Catch:-

Use:-

✓ It is used to Catch the Exception which was thrown from try block.

✓ It will Catch and handle the Exception.

Syntax:-

```
try {
    // moniter the Exception

} Catch (ExceptionType exceptionobject)
{
    // handle the exception

}
```

④ throws:-

Use:

throws Keyword, will throw the Exception only for a specific "method".

✓ The throws keyword appears at the end of the method's signature.

<u>Syntax:-</u>

```
Accessspecifier returntype Methodname (arguments)
                               throws Exception
{

}
```

⑤ <u>finally:-</u>

<u>Use:-</u>

A finally is a <u>block of code</u> that <u>always executes</u>, <u>irrespective</u> of occurrence of an <u>Exception</u>.

<u>Syntax:-</u>

```
try {

} Catch (Exceptiontype exob)
{

}
finally {
   // block of code that always get
                      // executed.
}
```

Program:- To illustrate Exception handling mechanism:- (eg) of try, throw and Catch

```
Class Example2
{
 Public static Void main(String args[])
 {    int d, a;
   try {
      d = 0;
     a = 42/d;
   System.out.Println(a);
   } Catch (ArithmeticException e)
   {
   S.o.p (" Division by zero" + e);
   }
 }
}
```

Output:-

```
Division by Zero
```

Multiple Catch statements:-

Def: If a try block has more than One catch block, then it is called multiple Catch statements.

Syntax:-

```
try {

}
```

```
Catch ( ExceptionType1 exob1)
{

}
Catch ( ExceptionType2 exob2)
{

}
.
.
.
Catch ( Exception Type n exobn)
{

}
```

Program:-

```
Class Example3
{
 Public static void main (String args[]
 {
  try {
 int a = args. length;
  int b = 42/a;
  int c[] = {1};
   c[42] = 99;
 } catch ( ArithmeticException e)
  { S.o.P ("Divide by 0" + e);
  }
Catch ( ArrayIndexOutOfBoundsException e)
  {
   S.O.P(" Array index Out of bound" + e);
   }
 }
}
```

<u>Output:-</u>

```
① C:/> java Example3
     a = 0
     Divide by 0 : java.lang. ArithmeticException:
                                         / by zero

② C:/> java Example3
     a = 1
     Array index Out of bound : ArrayIndexOut
                  OfBoundsException : 42 .
```

<u>Nested try Statements:-</u>

Definition:- If a try within a try statement, then it is called nested try statements.

Program:-

```
      Class nestedtry
      {
      Public static Void main (String args[])
       {
          try {
              int a= args. length ;
              int b= 42/a ;
              S.O.P(a);
```

```
try {
    if (a==1)
        a = a/(a-a);

    if (a==2)
    {  int c[] = {1};
            c[42] = 99;

    }
 } catch (ArrayIndexOutOfBoundsException e)
    {
      S.O.P("Array index out of bounds"+e);

    }
 } catch ( ArithmeticException e)
    {
      S.O.P("Divide by 0"+e);

    }
 }
}
```

Output:

```
    C:\> java  nestedtry
    Divide by 0 : java.lang. ArithmeticException:
                                      : / by zero

    c:\> java  nestedtry
      a = 1
    Divide by 0: java. lang. ArithmeticException:
                                      :/ by zero
    c:\> java  nestedtry.
         a = 2
```

Array index out of Bounds
java.lang.ArrayIndexOutOfBoundsException: 12

✓ throws :- Use: When a method wants to throw an exception then throws keyword is used.

Syntax:

```
method name ( Parameter_list ) throws exception
{


}
```

Program:

```
Class Exception3
{
  Static Void fun(int a, int b) throws
                      ArithmeticException

  {
     int c;
     try {
        c = a/b;

   }
   Catch (ArithmeticException e)
   {
      S.O.P ("Caught" + e);
   }
}
 Class Exception4
 {
   Public static Void main (String args[]
   {
```

```
int a = 5;
fun (a, 0);
 }
}.
```

✓ Finally:-

Use and Syntax: The finally block will be always gets executed and Provides the Assurance of execution of important Code that must be executed after the try block.

Syntax: finally {   }

Program:-

```
      class finallydemo
       { Public static Void main (string args[])
         {  int a=10, b=1;
        try {
             b = a/0;
           }
        Catch (ArithmeticException e)
         {  S.O.P (e);
         }
       finally {
         if (b!=-1) {
         S.O.P( " finally block executes without
                         Occurrence of exception");
         } else {
         S.O.P ("With exception");
}} } }
```

UR.
4m:

Java Built in Exceptions:- ~~(or) User defined Exception~~

Built in Exceptions (or) Predefined Exceptions which are available in Java libraries.

Exception	Description.
(i) ArithmeticException	It is thrown, When an Exceptional Condition has Occured in arithmetic operation.
(ii) IoException	When an illegal i/p / o/p operation is Performed then this Exception is raised.
(iii) ArrayIndexOutOfBoundsException	When array index gets out of bound, this Exception will be caused
(iv) NumberFormatException	When we try to Convert an invalid string to number.
(v) NullPointerException	Caused, When an attempt to access an object with a null reference is made.

UQ 13m

User defined Exception (or) Creating Own Exceptions:

✓ We Can Create Our Own Exceptions using the Keyword `throw'

Syntax:

throw new Usedefined Exception (Parameter

eg

throw new MyownException (" your age is less ");

Program:

```
class MyownException extends Exception
{
    MyownException ( String msg)
    {
        Super(msg);
    }
}
Class myexception
{
  Public static Void main (String a[])
  {
    int age;
    age = 15;
  try {
    if(age < 21)
  throw new MyownException ("your
                  age is less");
}
Catch (MyOwnException e)
  {
    S.o.p(" This is Exception" + e);

  }
finally
  {
    S.o.p ("finally block");
  }
 }
}
```

<u>Output:-</u>

```
This is Exception
your age is less
finally block.
```