SANYA VIRMANI 4EA

BCA
4th Java

Ds. Neha Verma
V.SST

(41)

EXCEPTION HANDLING.

WHY REQUIRE EXCEPTION HANDLING :-

(i) An error may produce an incorrect output or may terminate the execution of the program abruptly.

(2) It is therefore important to detect & manage properly all the possible error conditions in the program.

TYPES OF ERRORS :-

We have 2 categories for

(A) COMPILE TIME ERRORS.

~~Exa~~ (B) RUN TIME ERROR.

COMPILE TIME ERROR :-

1) All Syntax errors will be detected & display by JAVA COMPILER & therefore these errors are known as "COMPILE TIME ERRORS".

2) Eg.

```
(error object)     error
ERROR1.Java : 7 :  ;        // file name
System.out.println ("JAVA")  // where is error
^
1 Error.
```

3) Common Errors :-

(A) Missing Semicolon. ;

(B) use of Undeclared Variables.

× (C) use of = in place of == (runtime error)

(d) Misspelling of identifiers & Keywords.

<u>RUNTIME ERRORS</u> :- (i) A program may compile Successfully creating the <u>.class</u> file but not run properly. Such Programs may produce wrong results due to wrong logic.

<u>COMMON RUN-TIME ERRORS</u> :-

(a) Dividing an Integer by zero. ∞(not defined)

(b) Accessing an element that is out of bounds of an array.

(c) Converting invalid String to a number.

<u>EXAMPLE OF RUN TIME ERROR</u> :- (Dr. Neha Vimal VCIT, VIPS)

called with class (static method)

```
class ERROR2
{
    Public Static Void main ( String args [ ])
    {   int a=10,b=5, C=5;
        int x =a / ( b-c); ← division by zero.
        System. out. println (" X:" + X);
        int y =  a/(b+c);
        System. out. println(" Y" + y);
    }
}
```

abnormal termination

This program is <u>Syntactically correct</u> & does not cause any problem while executing. However, while Executing, it displays following Message & <u>stops without executing further statements</u>.

exception object → java. lang .ArithmeticException : / by zero } printable error

at Error2. main(ERROR2. Java : 10) (printed by toString() [DEII])

folder (package) ↙ ; Error → called-in function ; main → ; ERROR2. Java → file ; 10 → line no.

EXCEP

error

objec

(2)

EXCEPTION:- (1) when the java interpreter encounters an error such as dividing by zero. It creates an exception object & throws it. [Informs us that error has occured

(2) If we want the program to continue with the execut-ion of remaining code, then we should try to catch the exception object thrown by error condition & then display an appropirate message for taking corre-tive actions. This task is known as "EXCEPTION HANDLING".

(Dr. Neha Verma) VSIT, VIPS)

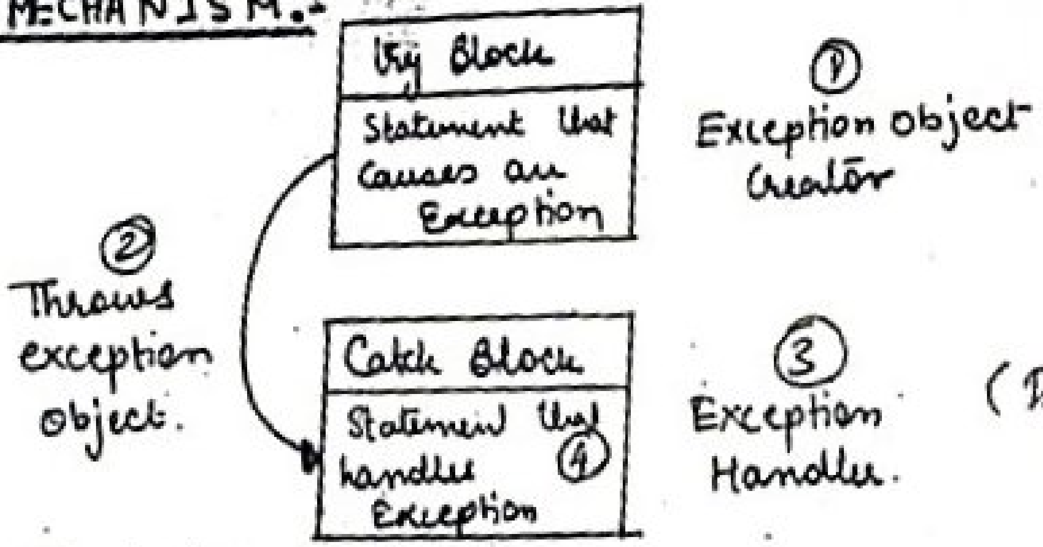
(3) So, we have following tasks for error handling:

(i) Find the Problem. [Hit] // maybe
(2) Inform that Error has occured. [Throw]
(3) Receive the error information. [Catch]
(4) Take Corrective Actions. [Handle].

COMMON JAVA EXCEPTION:-

EXCEPTION TYPE	CAUSE OF EXCEPTI
(1) Arithmetic Exception	Caused by Errors Such as/by
(2) FileNotFoundException	Caused by an attempt to access a non-existent file.
(3) NumberFormatException	Caused when a conversion b/w Strings & number fails.
(4) IO Exception	Input/output Exception
(5) SQL Exception	

MECHANISM:-


Try Block
Statement that causes an Exception
① Exception object creator
② Throws exception object.
Catch Block
Statement that handles Exception ④
③ Exception Handler.

(Dr Neha Verma
VSIT, VIPS

Basic Syntax:-

```
try
{
   statements;      // Generates An Exception
}
catch( Exception-Type e)
{
   statements;      // Processes the Exception
}
```

EXAMPLE:-

```
class Error3
{
 Public static void main (String args[])
 {
   int a=10, b=5, c=5, x, y;
   try
  {
     x= a/(b-c);      // 10/0 (wrong logic)
  }
normal
termination  catch (ArithmeticException e)    //exception (chec
  {
   System.out.println (" DIVISION BY ZERO");
  }
  y= a/(b+c);        // 10/10
  System.out.println (" Y = "+y);
 }
}
```

OUTPUT:

```
DIVISION BY ZERO
Y =1
```

MULTIPLE CATCH STATEMENTS :-

(Dr. Neha Verma)
USIT, VIPS

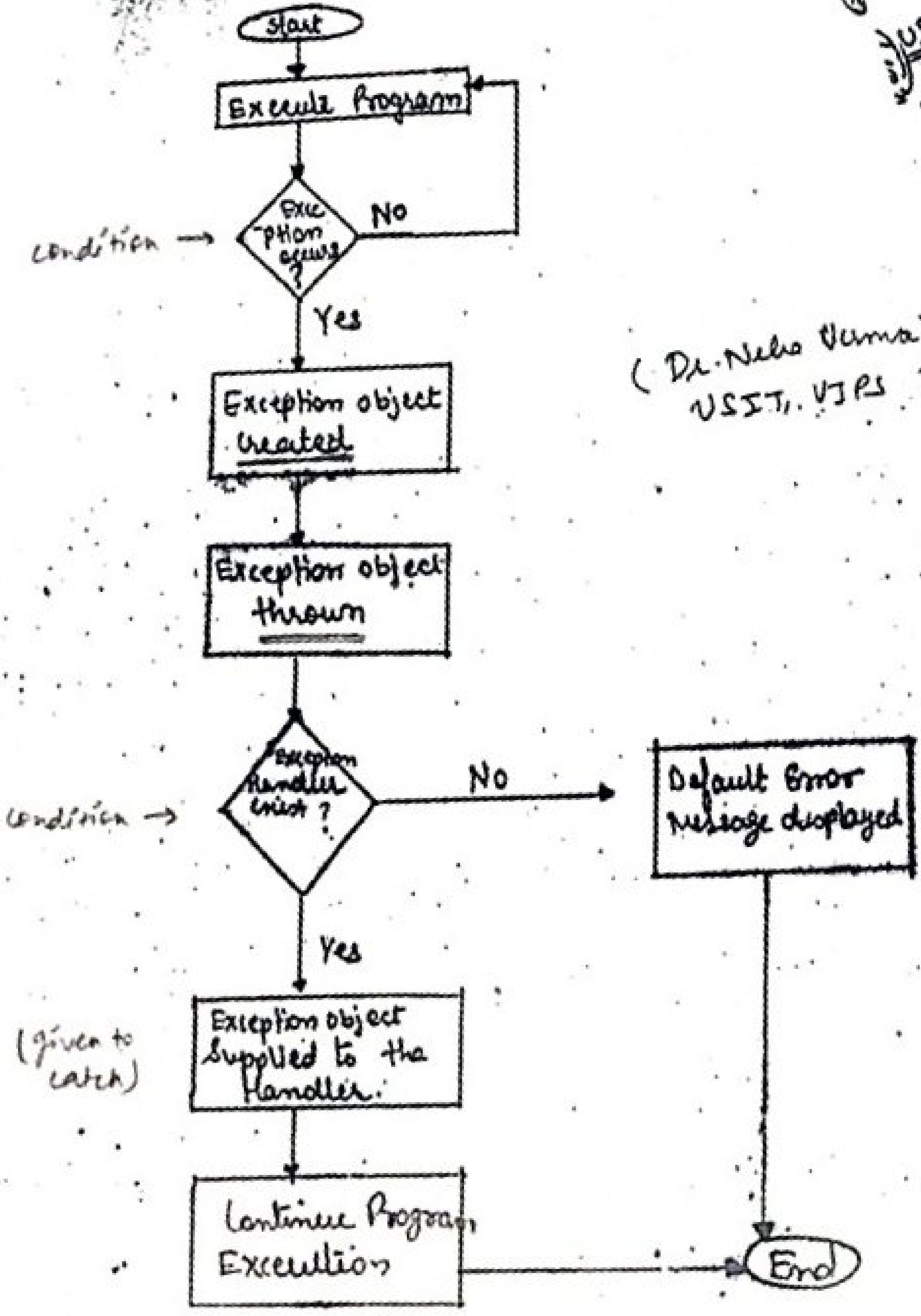
```
class Error4
{
    Public static void main( String args [])
    {   int a[]= {5,10};
        int b=5;
        try
        {
           int x =. a[2]/b- a[1];
        }
        Catch (ArithmeticException e)
        {
          System.out.println( "DIVISION BY ZERO");
        }
        Catch ( ArrayIndexOutOfBoundsException e)
        {
          System.out.println( "ARRAY INDEX ERROR");
        }
        Catch( ArrayStoreException e)
        {
          System.out.println( "WRONG ~~DATA~~ DATA TYPE");
        }
        int y= a[1]/a[0];
        System.out.println(" Y "+y);
    }
}
```

runtime error (exception)

// printing this even after catch

OUTPUT:

```
Array INDEX ERROR.
Y=2.
```



(Dr. Neha Verma)
USIT, VIPS

Full Exception Handling Mechanism.

6   Q final, finally, finalize difference
constant [illegible]

(i) USING FINALLY (1) finally statement that can be used to handle an exception that is not caught by any of the previous catch statements. (47)

(2) It may be added immediately after try block or after the last catch block.

(3) when a finally block is defined that is guaranteed to execute regardless of whether or not an exception is thrown

Example:

```
try
{
    openFile();
    WriteFile();
}
catch(- - -)
{
    // Processes Exception
}
finally
{
    closeFile();
}
```

// will be executed in se case (checks everything in code & points to the end)

(Dr. Neha Verma
VSIT, VIPS)

Note: It is NOT MANDATORY to have a finally block.

Q. throw/throws
Q. user defined exception.