19/09/18

UNIT-3

Exception Handling

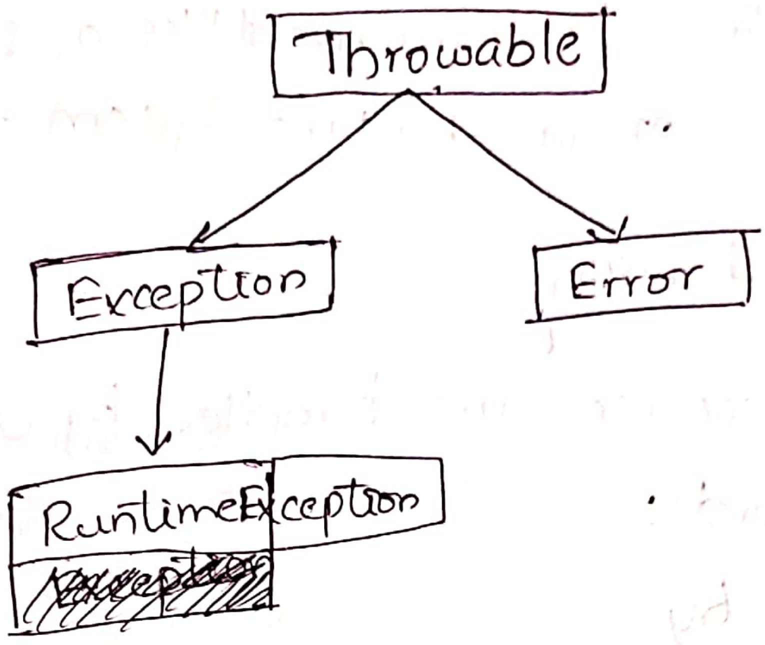
Write a c-program to read 2 integer values from consoul (or) Runtime.

```
import java.util.scanner;
class MathsDemo
{
    Public static void main (String ar[ ])
    {
        Scanner sc=new Scanner(System.in)
        System.out.println("Enter a value");
        int a=sc.nextInt( );
        System.out.println("Enter b value");
        int b=sc.nextInt( );
        int c=a/b;
        System.out.println("Division a/b is"+c);
    }
}
```

Exception :- The run time error in a Program is called exception.

exception is an event that interrupts the flow of execution (or) program.

Exception Types:



(1.) Throwable:- it is a base class for all the Exception type Bases.

(2.) The Execption Types are clasified in to 2 ways.

1. Exception
2. Error.

1. Exception is used to rectify (or) handle the errors in user program ~~like I/O Exception.~~ at runtime

3. Runtime Exception: It is a derived class for Exception class and handles the errors in runtime

(4) Exceptions ~~class~~ are automatically defined by the programs.

5. Error class is used to ~~def~~ handle the errors in a environment like a stack overflow or java runtime systems.

Exception handling :-

Exceptions are handled by using (5) keywords:

1. try
2. catch
3. throw
4. throws
5. finally.

(1) try :-

tryblock

it contains block of statements (or) program statements that you want to monitor for an Exception.

if an Exception exists. sensitive catch block.

(2) Catch Block :

~~it takes~~

catch Block catches the Exception and handle in a ~~manner~~ rational manner

(3.) throw: To throw an Exception mannually by using throw keyword.

(4.) Throws :- It is used to throws clause an enception from any method. ~~and any enception~~. by using throws clause.

(5) Finally contains Block of statemements(er) code. it is absolutely must be enecuted.

Syntax's:

```
try
{
  //block of programs
}
catch
{ // enception handler for Exception
}
//...
finally
{
   block of code to be enecuted after
  try block ends.
}
```

Example:

```
import java.util.*;
class MathDemo
{
 public static voidmain(String a[ ])
 {
  Scanner   sc=new   Scanner(System.in)
  try
  {
   System.out.println("Enter a value");
    int  a=sc.nextInt();
    System.out.println ("Enter b value");
    int  b=sc.nextInt( );
     int c=a/b;
      System.out.println("~~Do not give
                    dividend as zero"~~);
            (Division a/b  is": +c);
    }
  Catch (ArthematicException ae)
  {
   System.out.println ("Do not give dividen
                as zero");
  }
catch (InputMissmatchException ie)
{
    System.out.println("please enter ~~your~~ integer
                         value");
}
finally
{
    System.out.println("Finally block is executed");
}
}
}
```

20/9/19

Ex:2:

Write a java program to demonstrate the Exception handling:

Source code:

```
import java.util.*;
class ExceptionDemo
{
 public static void main(String ar[])
 {
   try{
      String lan[ ] = {"c", "c++", "java", "c#"};
      for(int i=0; i<10; i++)
```

```
    {
      System.out.println (lan [i]);
    }
  }

  catch (Index Out Of Bound Exception ie)
    {
      System.out.println ("There is no index
                         in array");
    }
  }
}
```

P. 90% User define exception (or) create your own exception:

→ A class is extends with Exception class then the derived class is called User defined Exception class. It having 2 constructors. They are:

st Exception( )

Exception (String msg)

There are many methods but, Mainly Two Methods

String toString ( )

String to getMessage( )

User defined Exception is must be thrown by using throw keyword.

Example program: (Lab Ex. 1.)

```
class EvenNumberException extends Exception.
{
    String message;
    EvenNumberException(String msg)
    {
        message = msg;
    }
    public String toString()
    {
        return message;
    }
}
class MyException
{
    static void even(int a) throws EvenNumberException
    {
        if (a%2 = =0)
            System.out.println("Number is :"+a);
        else
            throw new EvenNumber Exception
                        ("enter even number ....");

    }
```

```
public static voidmain (String str[ ])
{
    try
    {
        even(10);
        even (20);
        even(5);
    }
    Catch (EvenNumberException ene)
    {
     System .out.println ("Exception caught :"+e
    }
  }
}
```

Scanned by CamScanner

Write a c-program to demonstrate the exceptional handling.

```
import java.util.*;
class Exception
{
 public static void main(String args[])
 {
  try
  {
   String lan[]={"c", "c++", "java", "c#"};
   for(int i=0; i<10; i++)
   {
    system.out.println(lan[i]);
   }
  }
  catch (Index out of Bound Exception i.e)
  {
  System.out.println("There is no index in array");
  }
 }
}
```

Throw :- Raise an exception manually throw

throwableinstance

Example

```
ArithmeticException   ae=new   ArithematicException
                                      ("msg")
        throw(ae);
```

throwable instance must be a object type of throwable (or) sub classess of throwable

primitive types like int, char cannot be used as enception.

we can create throwable instance in 2 ways:-

one is by using new operator

second is using a parameter in catch block.

when throw is find that immediately stop the flow of enecuition and any subseaquent statements are not enecuted

The try block inspects the if it has any catch statement that match with enception type

* if it is found the control is transfer to the catch statement.

* if it is not found then the nearest try block is inspected and so on...

* if no catch block is found then it will handle by default exception handler and prints the stack trace.

→Write a java program to demonstrate throw or rethrow

```
class ThrowDemo
{
 public static void display( );
  {
   try {
     System.out.println ("Welcome");
     throw new Nullpointer Exception("Null ref
                                       exception");
   }
   catch (Null pointer Exception ie)
    {
     System.out.println(ie);
     throw new Arthematic Exception ("Invalid
                          operation");
                   // rethrow exception.

   }
 }
public static void main(String args[ ])
 {
   try {
    Display ( );
  }
```

```
    catch (Arthematic Exception ie) {
     System.out.println(ie);
   }
  }
 }
```

Throws:-

```
type method name (parameter-list) throws exception
                                          = list
{
   //body of method
}
```

* Throw block raise an enception from method signature.

* The throw clause takes the multiple enceptions with comma (,)

* throws clause doesn't handle type of Error, Runtime Exception or any of its sub classes.

→

```
class ThrowsDemo
{
 static void Display ( ) throws illegalAcess Exception
  {
   System.out.println ("Inside display.");
   Throw new illegalAccess Exception("Illegal
                                        Exception");
 }
public static void main(string args[])
{
 try
 {
  Display( );
 }
 catch (IllegalAccessException ae)
 {
  System.out.println(ae);
 }
}
}
```

```
→ import java.util.*;
class ExceptionDemo
{
 p.s.v.m (string args[])
 {
  try
  {
   string lan[] ={"C","C++","java"};
   for(int i=0; i<10; i++)
   {
    System.out.println (lan[i]);
   }
  }
  catch (IndexoutofBoundException ie)
  {
   System.out.println ("There is no index inarray");
  }
```

2nd Edition, ... (Early Objects), ...

```
finally
{
 system.out.println ("final block is executed");
}
}
}
```

Nested try:- A try block is placed in another try block is called nested try.

Each time the try statements enters the content of exception.

Each content of exception is placed in stack. If the nested try block doesn't have any catch handler for particular exception then the next try statement catch handler are inspected for match.

If there is no match found then default exception handler will handle.

Program:

```
import java.util.*;
Class MathsDemo
{
  p    s    v  m (string args[])
 {
```

After the

CLO1: Understand funda

execution, methods, etc.

```
Scanner sc=new Scanner(system.in);
try
{
 System.out.println ("enter a value");
   int a=sc.nextInt( );
  System.out.println ("enter b value");
    int b =sc.next int( );
     int c=a/b;
  System.out.println ("Division of a/b "+c);

try
{
String lan[ ] ={"c", "c++", "java"};
 for (int i=0; i<10; i++)
 {
  System.out.println(lan[i]);
 }
}
catch(indexoutofBoundsException ie)
{ System.out.println("there is no index in array");
}
}
catch(ArithmeticException ae)
{
 System.out.println("Do not give dividend as zero");
}
```

```
Catch (InputMismatchExceptoon ae)
 {
  system.out.println("please enter only integer");
 }
finally
 {
  system.out.println("final block is executed");
  }
 }
}
```

Scanned by CamScanner

Exception Types:-

1. Build in enception
2. User defined enpception

1. Build in enception

Exception	meaning
Arthematic Exception	Arithematic error, such as dividend as zero
Array Inden Out of Bounds Exception	Array inden is out of bounds.
Illegal Argument Exception	Illegal argument used to invoke a method
Illegal state Exception	Environment or application is in incorrect state
Inden Out of Bounds Exception	Some type of inden is out of bounds.
Null pointer Exception	Invalid use of null reference.

User defined Exception:

userdefined Exceptions are used to create our own exception. A class is extends with Exception class then the defined class is called user defined Exception class.

It having two constructors

(i) Exception()

(ii) Exception(string msg)

methods are string to string()

string getMessage()

User defined exceptions must be thrown by using "throw" keyword.

Example:

```
class EvenNumberException extends Exception
{
 String message;
 EvenNumberException (string msg)
  {
   message = msg;
  }
public String to string()
  {
   return message;
}  }
```

```
class exception
{  static void even (int a) throws EvenNumberException
   {  if (a%2 ==0)
         system.out.println("Number is:"+a);
      else
         throw new EvenNumber Exception ("enter new
                                          number...");
   }
   public static void main (string a[])
   {
         try
         {
            even (10);
            even (20);
            even (5);
         }
         catch (Even Number Exception ene)
         {
          System.out.println ("exception Caught :"+ene);
       }
   }
}
```

~~Child~~ chained Exception:-

-An enception describes the cause of another enception is called chained Exception.

For Ex:

-A method Throws arthemathic ~~express~~ enception because of divide by zero. However the actual cause of the problem was I/O error occur. Finally, Here I/O enception ~~causes (or) dimatches~~ describes the cause of arthemathic enceptions

it is having two constructors and two methods

constructors:-

(i) Throwable (Throwable causeExc)

(ii) Throwable (String msg, Throwable causeExc)

Methods:

Throwable initCause (Throwable causeExc)

Throwable getcause ( )

Example:

```
// Demonstrate enception chaining
 class chainExcDemo
 {
   Static void d~~emoproc~~splayc( )
   {
     NullPointerException e=new NullpointerExcept
                                     ("top layer");
                            // Create an enception
     e.initcause(new Arthematic Expression ("cause"));
                            //add a cause

     throw e;

 }
 public static void main (String args[ ])
 {
   try
   {
     Display ( );
   }
   catch(NullpointerException e)
   {
     System.out.println ("caught:"+e);
                    //display top level enception
    System.out.println (" original cause:"+e.getcau
                                         ( ));
//display class Exception.

      }

    }

  }
```