Exception Handling

Basically the programmers sufferers from two kinds of errors

1.sysntactic errors:

These errors are identified at compile time by the compiler so we can call them as compile time errors…Programmer no need to worry about these errors bcs compiler will identify and intimate to the programmer as error message…….

Ex: missing of ;

missing of }

2.Logical errors:

These errors are not identified by the compiler ….. Ex:

Programmer may use “-“ instead of “+” when he is performing addition operation

And system is unable to execute some kind of complicated instructions …. These are called Exceptions…..

Ex…..

A=10;

b=0;

c=a/b

the above statement cauese an exception

**We can handle the Exceptions using 3 blocks**

**1.try Block:whatever the statement causes the exception in the program are written in try block**

**try**

**{**

**Statements;**

**}**

**2.Catch Block:**when Exception risese in the try block the control will jumps to concern catch block

Catch(Exception e)

{

Statement;

}

**3.Finally Block:**

Whatever the statements we want to execute irrespective of Exception are written in finally block

Finally

{

Statements;

}

Handling Single Excepetion

class Exp1

{

public static void main(String args[])

{

int a,b,c;

a=20;

b=0;

System.out.println("File Opend");

try

{

c=a/b;

System.out.println("Division is :"+c);

}

catch(ArithmeticException e)

{

System.out.println("b val not lessthan 1:");

}

finally

{

System.out.println("File Close");

}

}

}

Handling Multiple Exceptions

class Exp1

{

public static void main(String args[])

{

int[] arr=new int[5];

int a,b,c;

a=20;

b=5;

//Reading elements into array

for(int i=0;i<=4;i++)

{

arr[i]=i+10;

}

System.out.println("File Opend");

try

{

c=a/b;

System.out.println("Division is :"+c);

//printing

for(int i=0;i<=5;i++)

{

System.out.println( arr[i]);

}

}

catch(ArithmeticException e)

{

System.out.println("b val not lessthan 1:");

}

catch(ArrayIndexOutOfBoundsException e)

{

System.out.println("i val Must less than 5");

}

finally

{

System.out.println("File Close");

}

}

}

Throws clause

Even ifthe programmer is not handling runtime exceptions the java compiler will not give any error related to runtime exception.But the rule is the programer shouled handled checked exceptions(The exceptions that are checked at compilation time).if he is not willing to handle he should throw them out using throws clause…Oterwise there will be an error by compiler…there is an IOException raised by readLine() in BufferedReader class…This is checked exception hence the compiler checkes it at compilation time…if it is not handled .the compiler expects at least to thtow it out…..

import java.io.\*;

class Demo

{

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

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter ur Name");

String Name=br.readLine();

System.out.println(" Your Name is : "+Name);

}

}

Throw Clause

There is also a throw statement available in java to throw an exception explicitly and catch it ….

Throw new NulPointerException(“Exception Data”);

In the Above Example NullPointerException class Object is created and “Exception Data” stored into its object …then it thrown using throw statement..we can catch using catch block……

class Sample

{

static void Demo()

{

try

{

System.out.println("We are inside Dmeo");

throw new NullPointerException("Exception Data");

}

catch(NullPointerException e)

{

System.out.println(e);

}

}

}

class Demo

{

public static void main(String args[])

{

Sample.Demo();

}

}