**Exception handling:**

Exception is an abnormal condition. Exception handling is mechanism to handle errors (abnormal condition) such as ClassNotDoundException, IOException, SQLException…etc

Advantage is to maintain normal flow of the application

Types – Checked Exception, Unchecked Exception and Error

Checked exception – classes that inherit Throwable class except RuntimeException and error, for example IOException, SQLException, checked exception are checked at compile time

Unchecked exception – classes that inherit RuntimeException, for example ArithematicException, NullPointerException, ArrayIndexOutOfBoundException etc……these are checked at run time

Error – Error is not recoverable, for example OutOfMemoryError, VirtualMachine Error……etc

Try-catch block:

Try block: try block is used to enclose the code that might throw an exception. If an exception occurs at the particular statement in in try block, the rest of the code will not execute.

Java try block must be followed either catch or finally block

Java Catch block: catch block is used to handle the exception by declaring the type of exception within parameter.

The catch block must be used after try block. There can be multiple catch blocks for single try block.

Without exception handling example: In the below “rest of code” will not be executed since there was exception before that

**package** com.lokesh;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**int** a =50/0;

System.***out***.println("rest of code");

}

}

Using exception handling example: the “rest of code” gets executed

**package** com.lokesh;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try**{

**int** a =50/0;

}**catch**(Exception e) {

System.***out***.println(e);

}

System.***out***.println("rest of code");

}

}

Below is an example where rest of code doesn’t get executed even when we use exception handling because the pointer doesn’t reach there it goes to catch block after exception occurs

**package** com.lokesh;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try**{

**int** a =50/0;

System.***out***.println("rest of code");

}**catch**(Exception e) {

System.***out***.println(e);

}

}

}

Below is an example we are giving custom message

**package** com.lokesh;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try**{

**int** a =50/0;

}**catch**(Exception e) {

System.***out***.println("divide by zero");

}

}

}

Example to resolve exception in catch block

**package** com.lokesh;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**int** a=50;

**int** b=0;

**int** c;

**try**{

c =a/b;

}**catch**(Exception e) {

c =a/(b+2);

System.***out***.println(c);

}

}

}

Another example for unchecked exception:

**package** com.lokesh;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try** {

**int** a[]= {1,3,5,7,9};

System.***out***.println(a[10]);

}**catch**(ArrayIndexOutOfBoundsException e) {

System.***out***.println(e);

}

System.***out***.println("rest of code");

}

}

Another Example for checked Exception

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.PrintWriter;

**import** java.nio.file.FileAlreadyExistsException;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try** {

PrintWriter pw =**new** PrintWriter("abc.txt");

pw.print("hello");

}**catch**(FileNotFoundException e){

System.***out***.println(e);

}

System.***out***.println("rest of code");

}

}

Multi-catch block example:

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.PrintWriter;

**import** java.nio.file.FileAlreadyExistsException;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try** {

**int** a[]=**new** **int**[5];

a[5]=30/0;

}**catch**(ArithmeticException e){

System.***out***.println(e);

}**catch**(ArrayIndexOutOfBoundsException e){

System.***out***.println(e);

}**catch**(Exception e){

System.***out***.println(e);

}

System.***out***.println("rest of code");

}

}

Another example for multi-catch block:

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.PrintWriter;

**import** java.nio.file.FileAlreadyExistsException;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try** {

**int** a[]=**new** **int**[5];

a[5]=30/0;

}**catch**(ArithmeticException e){

System.***out***.println("arithematic exception occured");

}**catch**(ArrayIndexOutOfBoundsException e){

System.***out***.println("array exception occured");

}**catch**(Exception e){

System.***out***.println("sub exception didn't get catched came to main exception");

}

System.***out***.println("rest of code");

}

}

Nested try block: try block inside a try block. For example Inner try block can handle arrayindex exception while outer try block can handle arithematic exception

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.PrintWriter;

**import** java.nio.file.FileAlreadyExistsException;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try** {

//inner try 1

**try** {

**int** a=50/0;;

}**catch**(Exception e){

System.***out***.println("arithematic exception occured");

}

//inner try 2

**try** {

**int** a[] = {1,3,5,7,9};

System.***out***.println(a[10]);

}**catch**(Exception e) {

System.***out***.println("array exception has reached");

}

}**catch**(Exception e) {

System.***out***.println("hande at outer try catch");

}

System.***out***.println("normal flow again");

}

}

Java finally block: used to execute import code such as closing the connection …etc

Finally block is always executed whether the exception is handled or not

Finally block follows the try-catch block

Case 1: When exception doesn’t occur – below is an example where program didn’t throw any exception, finally block is executed after try block

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.PrintWriter;

**import** java.nio.file.FileAlreadyExistsException;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try** {

**int** a=30/2;

System.***out***.println(a);

}**catch**(Exception e) {

System.***out***.println(e);

}**finally** {

System.***out***.println("finally block is executed");

}

System.***out***.println("rest of code");

}

}

Case 2: When an exception occurred but not handled by catch block – below is an example where arithematic exception occurred since catch has Nullpointer exception it didn’t handle

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.PrintWriter;

**import** java.nio.file.FileAlreadyExistsException;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try** {

**int** a=30/0;

System.***out***.println(a);

}**catch**(NullPointerException e) {

System.***out***.println(e);

}**finally** {

System.***out***.println("finally block is executed");

}

System.***out***.println("rest of code");

}

}

Case 3: When an exception handled by catch block - – below is an example where arithematic exception occurred and it is handled by catch block

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.PrintWriter;

**import** java.nio.file.FileAlreadyExistsException;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

**try** {

**int** a=30/0;

System.***out***.println(a);

}**catch**(ArithmeticException e) {

System.***out***.println(e);

}**finally** {

System.***out***.println("finally block is executed");

}

System.***out***.println("rest of code");

}

}

Java throw keyword: used to throw an exception explicitly, we can throw checked and unchecked exception using throw

We can define our own set of conditions and throw an exception explicitly suing throw keyword

For example we can throw arithmetic exception if we divide number by another number instead of zero

Syntax (example):

throw new ArithmeticException (“sorry some arithmetic error”);

Throwing unchecked exception example: we have created a method named validate() that accepts an integer as a parameter. If the age is less than 18, we are throwing the ArithmeticException otherwise print a message welcome to vote.

**package** com.lokesh;

**public** **class** HelloWorld {

**public** **void** validate(**int** age) {

**if**(age>18) {

**throw** **new** ArithmeticException("not eligible tovote");

}**else** {

System.***out***.println("eligible to vote");

}

}

}

**package** com.lokesh;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

HelloWorld hw=**new** HelloWorld();

hw.validate(13);

System.***out***.println("rest of code");

}

}

Throwing checked exception example: if we throw checked exception it must be exception using either catch block or the method must declare it using throws declaration

**package** com.lokesh;

**import** java.io.BufferedReader;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.nio.Buffer;

**import** javax.xml.parsers.FactoryConfigurationError;

**public** **class** HelloWorld {

**public** **void** method1() **throws** FileNotFoundException {

FileReader fr=**new** FileReader("D:\\abc.txt");

BufferedReader br=**new** BufferedReader(fr);

**throw** **new** FileNotFoundException("I came here 1");

}

}

In main class below you need to handle from Try and catch since the expectation is it throws an error

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

HelloWorld hw=**new** HelloWorld();

**try**{

hw.method1();

}**catch**(FileNotFoundException e) {

System.***out***.println("i came to catch block");

}

System.***out***.println("rest of code");

}

}

Throwing user defined exception:

**package** com.lokesh;

**public** **class** HelloWorld **extends** Exception{

**public** HelloWorld(String str) {

**super**(str);

}

}

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) {

// **TODO** Auto-generated method stub

HelloWorld hw=**new** HelloWorld("hello");

**try**{

**throw** **new** HelloWorld("this is user defined exception");

}**catch**(HelloWorld e) {

System.***out***.println("caught user defined exception");

System.***out***.println(e.getMessage());

}

}

}

Java Throws keyword: used to declare exception. It gives information to programmers that there may occur an exception

Syntax -

Return\_type method\_name() throws exception\_class\_name{

}

Throws keyword are only for checked exception because we are telling that these may occur exception while unchecked are under our control to correct and error is beyond our control

Throws just tells that an error may occur but it doesn’t handle the exception

Java throws Example: Below is an example where an IO exception occurred and it has been handled by try-catch block

**package** com.lokesh;

**import** java.io.IOException;

**public** **class** HelloWorld **extends** Exception{

**public** **void** method1() **throws** IOException{

**throw** **new** IOException("device error"); //checked exception

}

**public** **void** method2() **throws** IOException{

**try** {

method1();

}**catch** (Exception e) {

System.***out***.println("exception handled");

}

}

}

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**public** **class** HelloWorld2 {

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

// **TODO** Auto-generated method stub

HelloWorld hw=**new** HelloWorld();

hw.method2();

System.***out***.println("normal flow");

}

}

Another Java throws Example (In throws there should be only checked exception I just gave unchecked exception just for example):

**package** com.lokesh;

**import** java.io.IOException;

**public** **class** HelloWorld **extends** Exception{

**public** **void** method1() **throws** ArithmeticException{

**int** a =50/0;

}

}

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) **throws** ArithmeticException{

// **TODO** Auto-generated method stub

HelloWorld hw=**new** HelloWorld();

**try**{

hw.method1();

}**catch**(Exception e) {

System.***out***.println("exception handled");

}

System.***out***.println("normal flow");

}

}

Another Java throws Example: Below is an example where no exception occurred and program continued normally

**package** com.lokesh;

**import** java.io.IOException;

**public** **class** HelloWorld **extends** Exception{

**public** **void** method1() **throws** IOException{

System.***out***.println("something inside method1");

}

}

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**public** **class** HelloWorld2 {

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

// **TODO** Auto-generated method stub

HelloWorld hw=**new** HelloWorld();

**try**{

hw.method1();

}**catch**(Exception e) {

System.***out***.println("exception handled");

}

System.***out***.println("normal flow");

}

}

Another Java throws Example: Below is an example where an exception occurred and there was no try-catch block

**package** com.lokesh;

**import** java.io.IOException;

**public** **class** HelloWorld **extends** Exception{

**public** **void** method1() **throws** ArithmeticException{

**throw** **new** IOException("device error"); //checked exception

}

}

**package** com.lokesh;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**public** **class** HelloWorld2 {

**public** **static** **void** main(String args[]) **throws** ArithmeticException{

// **TODO** Auto-generated method stub

HelloWorld hw=**new** HelloWorld();

hw.method1();

System.***out***.println("exception handled");

System.***out***.println("normal flow");

}

}

Difference between throw and throws is that the throw keyword will throw an exception explicitly while throws is used in method signature to declare exception that tells(programmers) exception might be thrown while execution of code of this method

Rules of using exception handling in method overriding

If the superclass method does not declare an exception, subclass overridden method cannot declare checked exception but it can declare unchecked exception

If the superclass method declare an exception, subclass overridden method cannot declare exception or no exception but it cannot declare parent exception