Exceptions:

What is an exception?

An exception is a problem that arises during the execution of a program.

When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore these exceptions are to be handled.

An exception can occur for many different reasons, below given are some scenarios where exception occurs.

🡪A user has entered invalid data.

🡪A file that needs to be opened cannot be found.

🡪A network connection has been lost in the middle of communications or the JVM has run out of memory.

What happens when we don’t handle exceptions? What is the use of exception handling?

So let’s say we have a program,

Class Test{

Psv Main(){

P v m1(){

Connect to the database;

Read the data from the database; //sqlException may occur due to

Close dbConnection;

}

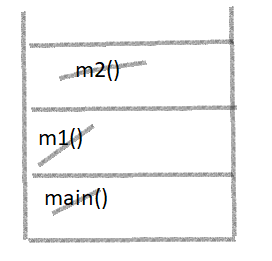
}

}

Talk about Default exception handling:

Call stack, what is stack trace?

There is always one thread known as the main thread called by JVM



**package** com.te.string;

**public** **class** test {

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

*m1*();

System.***out***.println(10/0);

}

**public** **static** **void** m1() {

*m2*();

System.***out***.println("hi");

}

**public** **static** **void** m2() {

System.***out***.println("hello");

}

}

TYPES OF EXCEPTIONS:

**Checked exceptions**:

A checked exception is an exception that is checked at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be ignored at the time of compilation, the Programmer should take care of handle these exceptions.

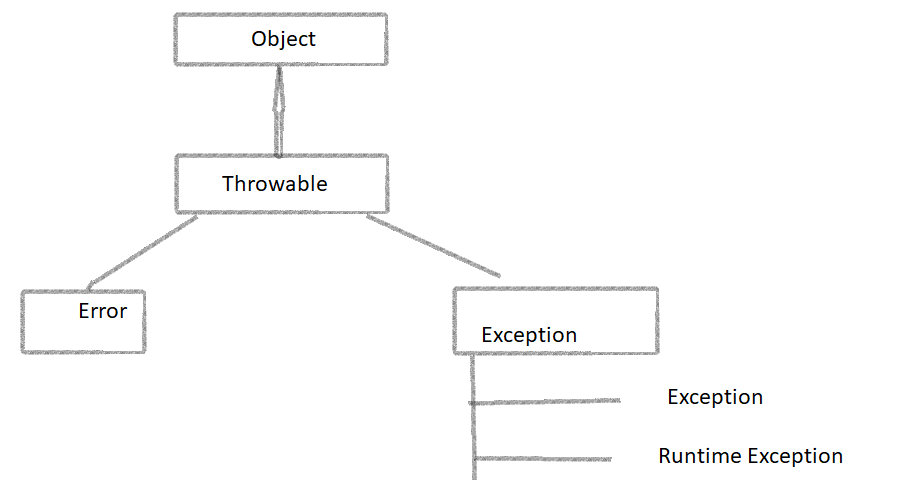
For example, if you use FileReader class in your program to read data from a file, if the file specified in its constructor doesn't exist, then an FileNotFoundException occurs, and compiler prompts the programmer to handle the exception.

**Unchecked exceptions:**

An Unchecked exception is an exception that occurs at the time of execution, these are also called as Runtime Exceptions, these include programming bugs, such as logic errors or improper use of an API. runtime exceptions are ignored at the time of compilation.

For example, if you have declared an array of size 5 in your program, and trying to call the 6th element of the array then an ArrayIndexOutOfBoundsExceptionexception occurs.

Talk about the Hierarchy of exception:



Exception🡪 ClassNotFound

🡪IOException

🡪SQLException

Runtime🡪Arithmetic

🡪ClassCast

🡪NullPointer

🡪ArrayIndexOutOfBounds

Fully checked and partially checked:

Fully checked is when

**Simple exception example**

**package** com.te.string;

**import** java.util.Scanner;

**public** **class** Excep {

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

Scanner scanner = **new** Scanner(System.***in***);

**int** result = 0;

{

**while**(**true**) {

System.***out***.println("Enter the numerator");

**int** a = scanner.nextInt();

System.***out***.println("Enter the denominator");

**int** b = scanner.nextInt();

**try** {

result = a / b;

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

**break**;

} **catch** (ArithmeticException e) {

System.***out***.println("Please try again. Something went wrong");

}

}

}

}

}

**Ways to handle exceptions:**

1. Try catch block

**Creating file exception: IOException.// Try catch**

**package** com.te.string;

**import** java.io.IOException;

**public** **class** File {

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

java.io.File file = **new** java.io.File("D:\\newFile.txt");

**try** {

file.createNewFile();

} **catch** (IOException e) {

System.***out***.println("Unable to create");

}

}

}

**Handling exception with throws: By using throws keyword we can propagate the exception occurring in the method to the calling method and now the calling method becomes responsible to handle the exception.  
This is also known as exception propagation too.**

* **So what happens if there’s is an exception in a method, jvm checks for catch block. If catch block is not available that method is terminated and the exception is propagated to it’s calling method.**
* **And it continues till it finds a proper catch block.**

**The**

**package** com.te.string;

**import** java.io.File;

**import** java.io.IOException;

**public** **class** test {

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

File file = **new** File("D:\\\\newfile.txt");

file.createNewFile();

}

}

TRY WITH MULTPLE CATCH BLOCKS:

We use this to handle one code but some particular exceptions with different codes.

Rules:

All sub class exceptions should be written prior to the superclass type.

From JDK 8:

We can also use “|” operator.

Try{

}

Catch(exception1 |exception2)

**Talk about the finally block, with the same database eg.:**

Class Test{

Psv Main(){

P v m1(){

Connect to the database;

Read the data from the database; //sqlException may occur due to

Close dbConnection;

}

Another ex for finally:

**package** com.te.string;

**import** java.io.IOException;

**public** **class** File {

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

java.io.File file = **new** java.io.File("D:\\newFile.txt");

**try** {

file.createNewFile();

}

Finally{

Syso(“finally is executing”);

}

}

}

**Custom checked exceptions:**

package com.tyss.custom\_exception.common;

public class Account {

double balance;

Account(double balance){

this.balance=balance;

}

public void withdraw(double amount) throws InsufficientBalanceException {

if(amount<=balance) {

System.out.println("Rs. "+ amount + " is successfully withdrawn");

System.out.println("Remaining balance= Rs."+ (balance-amount));

}

else {

throw new InsufficientBalanceException ("Insufficient Balance");

}

}

}

**Custom Unchecked Exceptions:**

package com.tyss.custom\_runtime\_exception.common;

public class Election {

public void vote(int age) {

if(age<18) {

throw new AgeValidatorException("Not eligible to cast vote");

}

else {

System.out.println("Your vote has been successfully casted");

}

}

}

package com.tyss.custom\_runtime\_exception.common;

public class AgeValidatorException extends RuntimeException {

String message;

public AgeValidatorException() {

super();

}

public AgeValidatorException(String message) {

super(message);

this.message=message;

}

@Override

public String getMessage() {

return this.message;

}

}

package com.tyss.custom\_runtime\_exception.common;

import java.util.Scanner;

public class Test1 {

public static void main(String[] args) {

Scanner scn = new Scanner(System.in);

System.out.println("Enter your age:");

int s = scn.nextInt();

Election el = new Election();

try {

el.vote(s);

} catch (AgeValidatorException e) {

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

}

scn.close();

}

}