

**1. Distinguish between String and String Buffer class.**

**Answer :-**

**2. Demonstrate the methods used in StringBuffer class for inserting ,deleting and appending operations.**

**Answer :-**

(A)**Insert() method :-** The insert() method inserts the given string with this string at the given position. This method used for inserting operation

import java.io.\* ;

**class A {**

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

**StringBuffer sb = new StringBuffer("Hello "); sb.insert(1, "Java");**

**// Now original string is changed System.out.println(sb);**

**} }**

(B)**Delete() method :-** The delete() method of StringBuffer class deletes the string from the specified beginIndex and endIndex-1.

**import java.io.\* ;**

**class A{**

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

**StringBuffer sb=new StringBuffer("Hello"); sb.delete(1,3);**

**System.out.println(sb); }**

**}**

**© append() method :-** The append() method concatenates the given argument with this string

**import java.io.\* ;**

**class A {**

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

**{**

**StringBuffer sb = new StringBuffer("Hello ");**

**sb.insert(1, "Java");**

**// Now original string is changed**

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

**}**

**}**

3. Demonstrate the different methods used in String class with an example.

**4. Demonstrate the working multiple catch statements with example.**

**Answer :-** A try block can be followed by multiple catch blocks. The syntax for multiple catch blocks looks like the following

**Syntax :-**try {

} catch (Exception e) {

// TODO: handle exception }

catch (Exception e) {

// TODO: handle exception }

**Example :-**

public class mmmm {

public static void main(String[] args) {

try{

int a[]=new int[5]; a[5]=30/0;

}

catch(ArithmeticException e) {

System.out.println("Arithmetic Exception occurs"); }

catch(ArrayIndexOutOfBoundsException e) {

System.out.println("ArrayIndexOutOfBounds Exception occurs");

} catch(Exception e)

{

System.out.println("Parent Exception occurs"); }

System.out.println("rest of the code"); }

}

**5. Define Exception Explain the different types of exception. Demonstrate the working of try and catch statement.**

**Answer :-** An exception (or exceptional event) is a problem that arises during the execution of a program

**Types of Exceptions**

There are three categories of Exceptions. They are 1. Checked Exception

2. Unchecked Exception 3. Error

**1.Checked exceptions** − A checked exception is an exception that is checked (notified) by the compiler at compilation-time, these are also called as compile time exceptions.

 These exceptions cannot simply be ignored, the programmer should take care of (handle) these exceptions.

 E.g. IOException, SQLException etc.

**2.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.

 E.g. ArithmeticException, NullPointerException

**3.Errors** − These are not exceptions at all, but problems that arise beyond the control of the user or the programmer.

 They are also ignored at the time of compilation. Error is irrecoverable.  E.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

Working of try and catch Block :- Syntax

try {

} catch (Exception e) {

// TODO: handle exception }

The code which is prone to exceptions is placed in the try block. When an exception occurs, that exception occurred is handled by catch block associated with it.

Every try block should be immediately followed either by a catch block or finally block.

A catch statement involves declaring the type of exception you are trying to catch. If an exception occurs in protected code, the catch block (or blocks) that follows the try is checked. If the type of exception that occurred is listed in a catch block, the exception is passed to the catch block.

Example :-

public class mmmm{

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

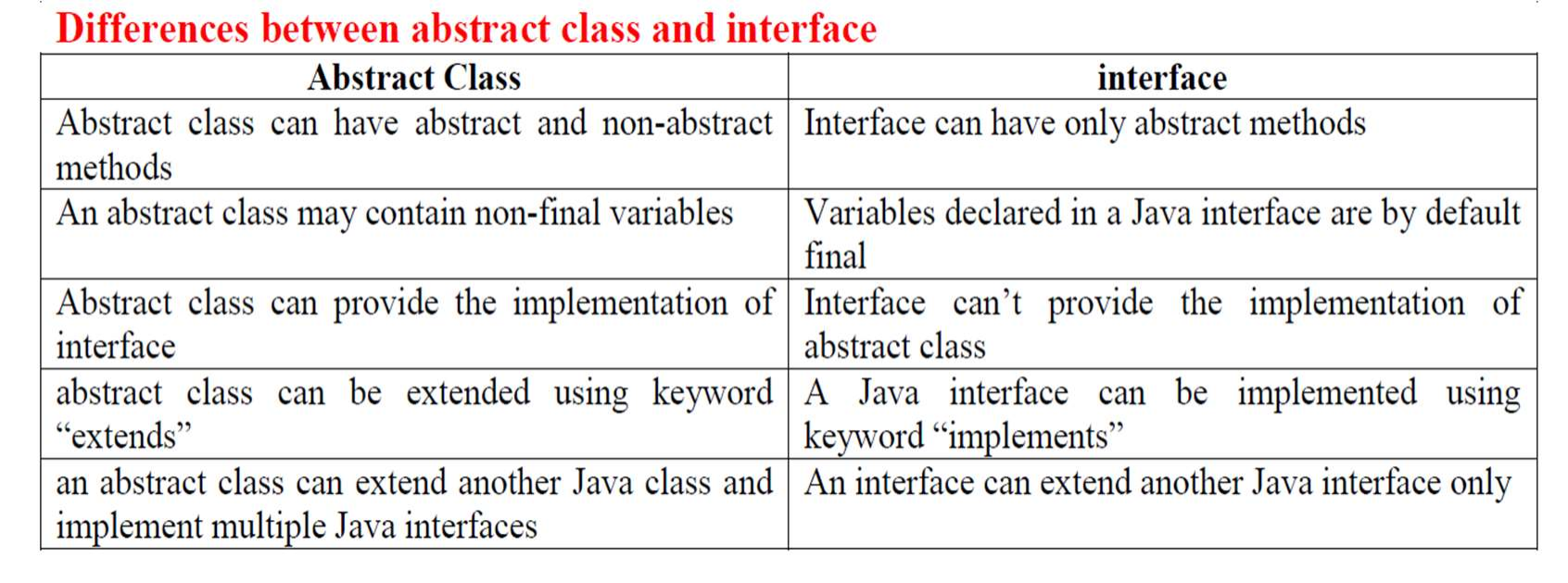
//code that may raise exception int data=100/0; }catch(ArithmeticException e) {

System.out.println(e); }

//rest code of the program System.out.println("rest of the code..."); }

}

**6. What is multiple inheritance ?Multiple inheritance is not supported through class in java, but it is possible by an interface, why? Explain with an example.**



**Answer :-** When the child class extends from more than one superclass, it is known as multiple inheritance. However, Java does not support multiple inheritance.

Multiple inheritance is not supported in the case of class because of ambiguity. However, it is supported in case of an interface because there is no ambiguity. It is because its implementation is provided by the implementation class.

interface Printable{ void print();

}

interface Showable{ void print();

}

class TestInterface3 implements Printable, Showable{ public void print(){System.out.println("Hello");} public static void main(String args[]){ TestInterface3 obj = new TestInterface3(); obj.print();

} }

Output :- Hello

As we can see in the above example, Printable and Showable interface have same methods but its implementation is provided by class TestTnterface3, so there is no ambiguity.

**7. Distinguish between the abstract class and interface.**

**Answer :-**

8. **What is thread and multi-threading in Java? Discuss the life cycle of the thread in java with neat diagram**.

Answer :-

Thread: thread is an individual working component. It is a separate path of execution. Threads are independent. If there occurs exception in one thread, it doesn't affect other threads. It uses a shared memory area.

**Multithreading in Java** is a process of executing multiple threads simultaneously.

****

**BORN STATE(new state):** Whenever a new thread is created, it will be in born state(new state)

**Active:** When a thread invokes the start() method, it moves from the new state(born state) to the active state. The active state contains two states within it: one is **runnable**, and the other is **running**. from runnable state u can explicitly kill the thread by thread.stop().

**Blocked or Waiting:** Whenever a thread is inactive for a span of time (not permanently)or whenever we make thread to sleep then, either the thread is in the blocked state or is in the waiting state, whenever sleep time completes the thread again goes to runnable state if u stop the thread explicitly by stop() then thread goes to dead state

**Terminated(Dead state):** A terminated or dead thread means the thread is no more in the system. In other words, the thread is dead, and there is no way one can respawn the dead thread.

9. What is Multi-threading? What are the ways to create thread and multiple threads in java.

Multithreading in Java is a process of executing multiple threads simultaneously.

A thread is an individual working element .

Multiprocessing and multithreading, both are used to achieve multitasking. However, we use multithreading than multiprocessing because threads use a shared

memory area. They don't allocate separate memory area so saves memory, and

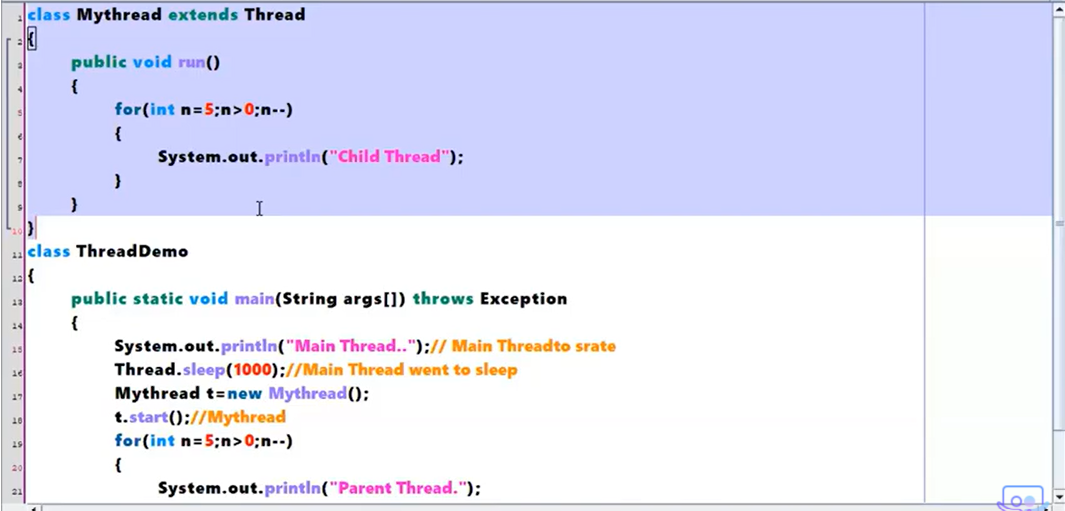
context-switching between the threads takes less time than process. Java Multithreading is mostly used in games, animation, etc.

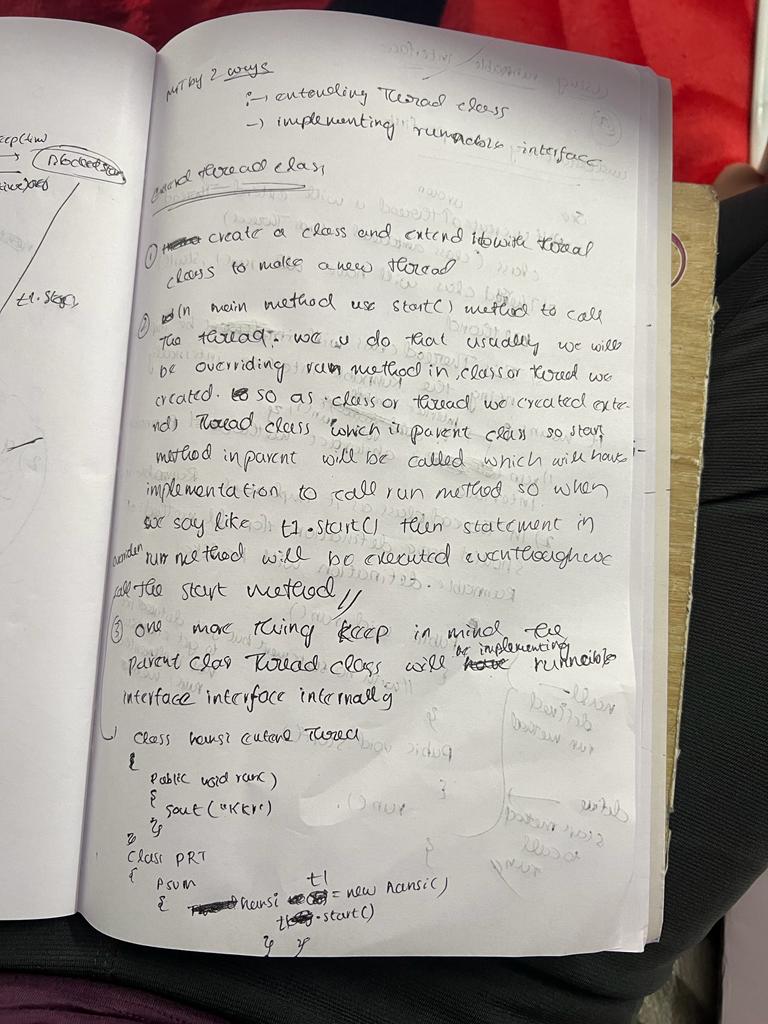
**way to create a thread:**

**1)by extending Thread**

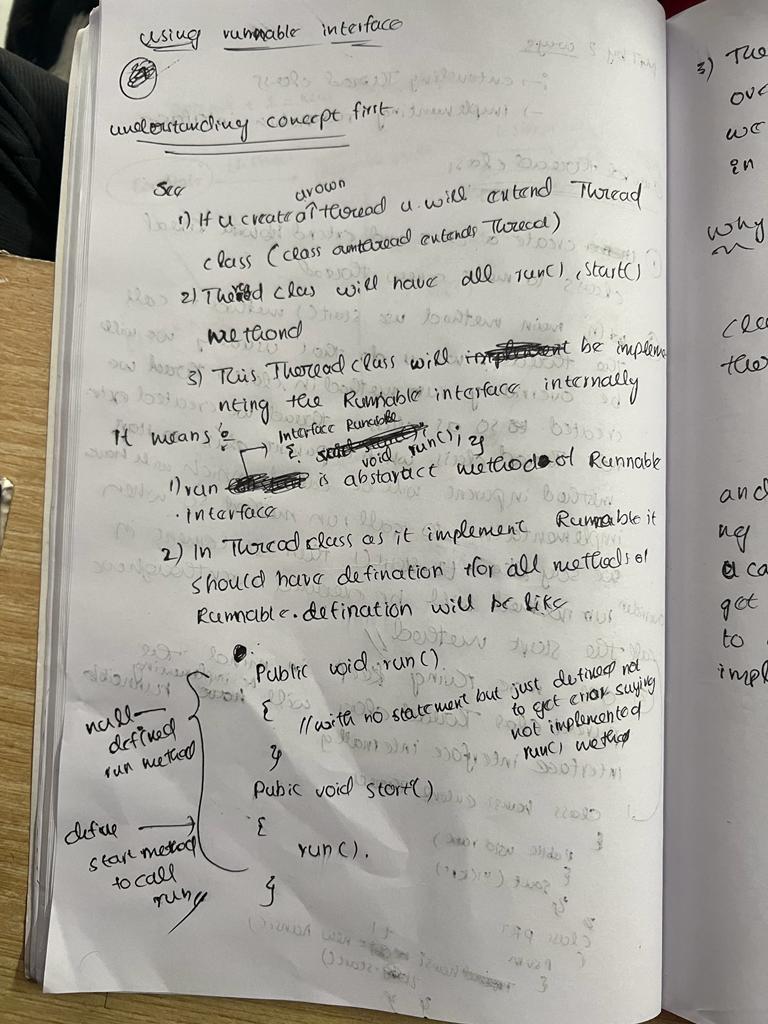
**2)by implementing Runnable**

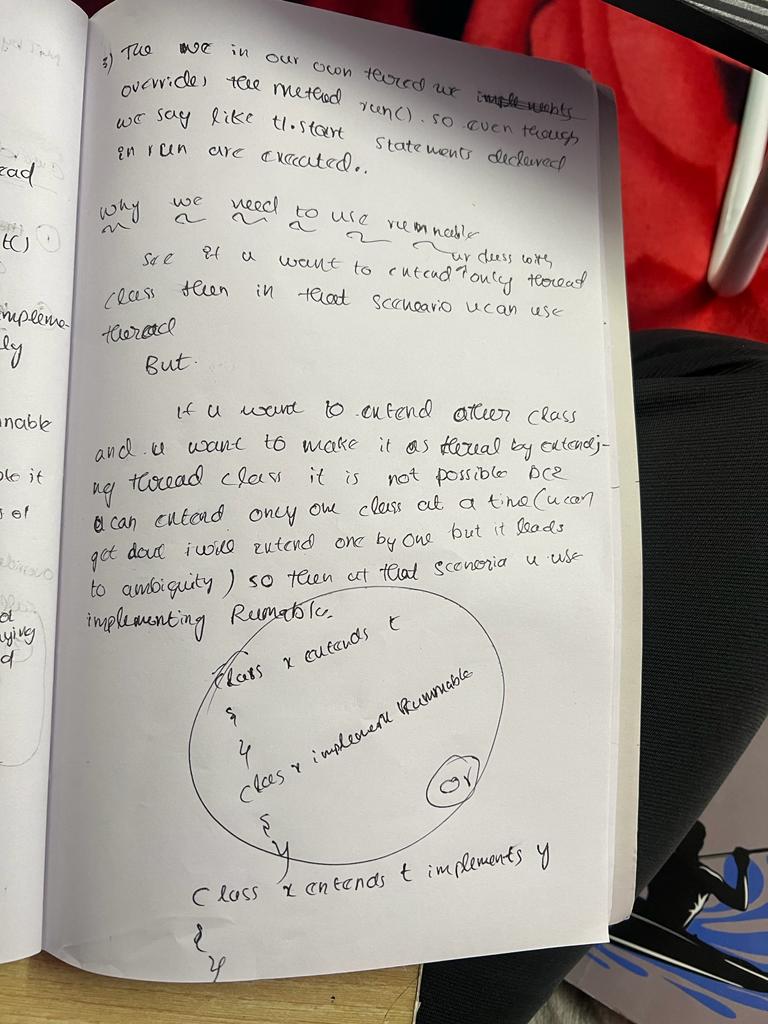
**1)by extending Thread**

****



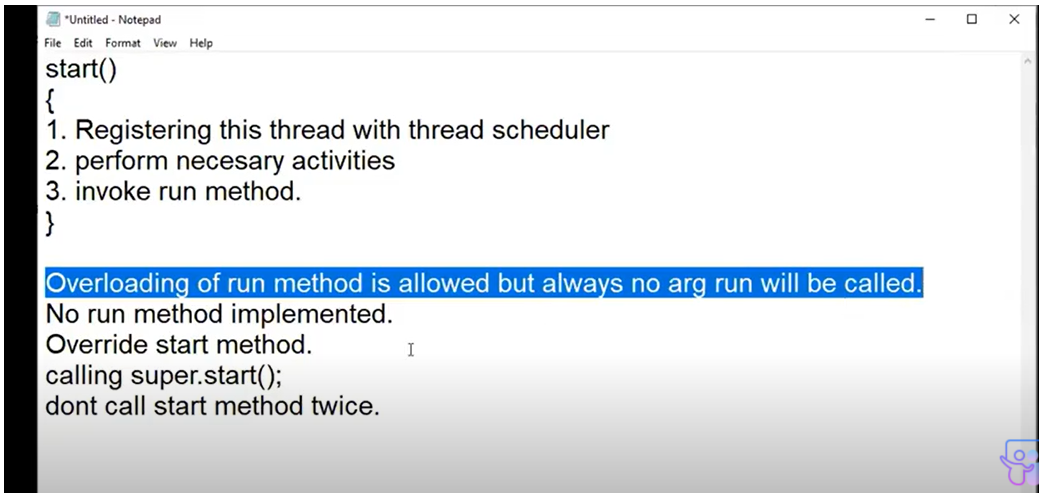
**2)by implementing Runnable**

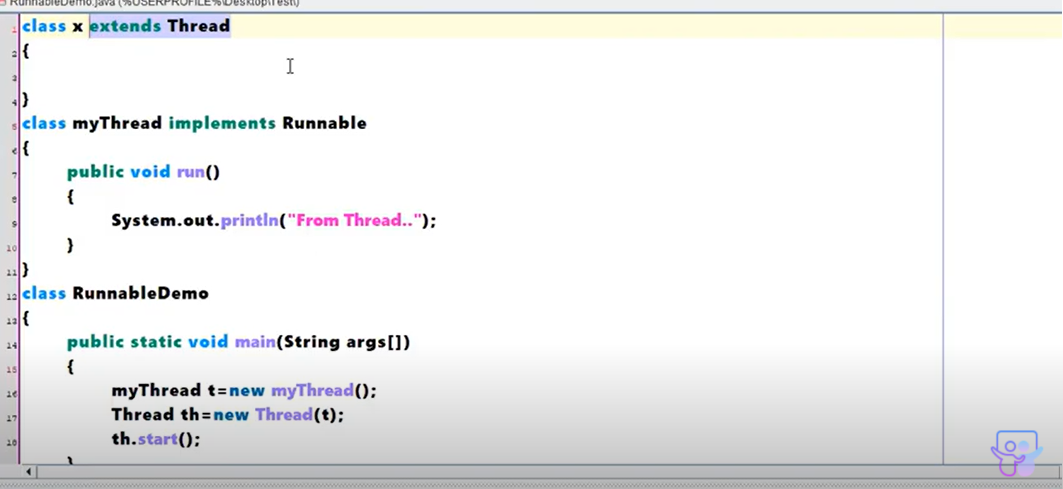




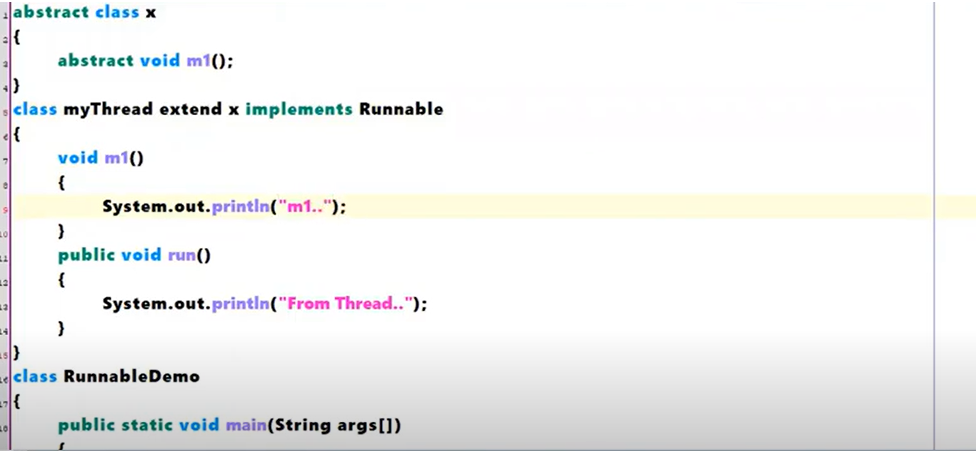
**CREATING MULTIPLE THREADS IN JAVA**

**CREATE MULTIPLE OBJ OF SAME THREAD LEADS TO CREATION OF MULTIPLE THREADS**

****

****

****

****

**10.** **Write short notes on:1.finally block 2.throws 3.throw**

**Answer :- Finally block :-**

**Usually finally block is declared after try and catch .u can use it directly after try also DIRECTLY without catch.working scenario of finally is like the statements in the block will be excecuted even if exception arises and also even if exception not arises.**

The finally block follows a try block or a catch block. A finally block of code always executes, irrespective of occurrence of an Exception.

 Using a finally block allows you to run any clean up-type statements that you want to execute, no matter what happens in the protected code.

try {

// Protected code

} catch (ExceptionType1 e1) { // Catch block

} catch (ExceptionType2 e2) { // Catch block

} catch (ExceptionType3 e3) { // Catch block

}finally {

// The finally block always executes. }

**2. Throws :-**

**See when comes to throws throws is generally used when we know that an exception arises in a method and u don’t need to do something with respect to that exception then we uses the throws**

If a method is capable of causing an exception that it does not handle, it must specify this behavior so that callers of the method can guard themselves against that exception. We do this by including a **throws** clause in the method’s declaration. A **throws** clause lists the types of exceptions that a method might throw.

type method-name(parameter-list) throws exception-list {

// body of method }

**3. Throw :-**

**See when it comes to throw we use it we wanted to handle an exception with a set of instruction explicitly.**

The throw keyword in Java is used to explicitly throw an exception from a method or any block of code. We can throw either checked or unchecked exception. The throw keyword is mainly used to throw custom exceptions.

**11.** **Discuss the differences between checked and unchecked exceptions.**

**Answer :-**

|  |  |  |
| --- | --- | --- |
| Checked Exception | | Unchecked Exception |
| Checked exceptions occur at compile time. | Unchecked exceptions occur at runtime. | |
| The compiler checks a checked exception. | | The compiler does not check these types of exceptions. |
| These types of exceptions shoulf be handled at the time of compilation. | | These types of exceptions cannot be a catch or handled at the time of compilation, because during run time the exception is being caught. |
| They are the sub-class of the exception class. | | They are runtime exceptions and hence are not a part of the Exception class. |
| Here, the JVM needs the exception to catch and handle. | | Here, the JVM does not require the exception to catch and handle. |
| Examples of Checked exceptions:   * File Not Found Exception * I.O Exception * Remote Exceptions * Interrupted Exception * No Such Method Exception * Class Not Found Exception | | Examples of Unchecked Exceptions:   * Arithmetic Exception * Null Pointer Exception * Array Index Out of Bounds Exception * Class cascade exception * Illegal argument exception * Security Exception |

12.Create two threads in which one displays “Dept of CSE” for every 50 seconds and the other displays “ Jain University” for every 100 seconds continuously.

13.Explain the visibility of each access specifier with respect to package.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| public | The fields with public access specifier are accessible for all classes inside and outside of a package | | | | [Try it »](https://www.w3schools.com/java/tryjava_multi.asp?filename=demo_mod_public2&multi=demo_mod_public2_multi) |
| private | The fields with private are only accessible within the declared class | | | | [Try it »](https://www.w3schools.com/java/tryjava.asp?filename=demo_access_mod) |
| *default* | The fields with default only accessible by classes in the same package. This is used when you don't specify a modifier. | | | | [Try it »](https://www.w3schools.com/java/tryjava.asp?filename=demo_mod_default2) |
| protected | The fields with protected are accessible in the same package and **subclasses**  **Of other packages also** . | | | |  |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only** | **outside**  **package** |
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

**14.** **Explain the differences between throw and throws keywords in java.**

**Answer :-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. no.** | **Basis of Differences** | **throw** | **throws** |
| 1. | Definition | Java throw keyword is used throw an exception explicitly in the code, inside the function or the block of code. | Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code. |
| 2. | Syntax | The throw keyword is followed by an instance of Exception to be thrown.  Or the throw keyword is followed by catch | The throws keyword is followed by class names of Exceptions to be thrown. |
| 3. | Declaration | throw is used within the method. | throws is used with the method signature. |
| 4. | Internal implementation | We are allowed to throw only one exception at a time i.e. we cannot throw multiple exceptions. | We can declare multiple exceptions using throws keyword that can be thrown by the method. For example, main() throws IOException, SQLException. |
| 5 | When we use | When we explicitly wanted to something w.r.t exception we uses throw | We wanted not to do anything w.r.t exception we use throws and we we are in a scenario of it might throw an exception and not wanted to do anything explicitlys |