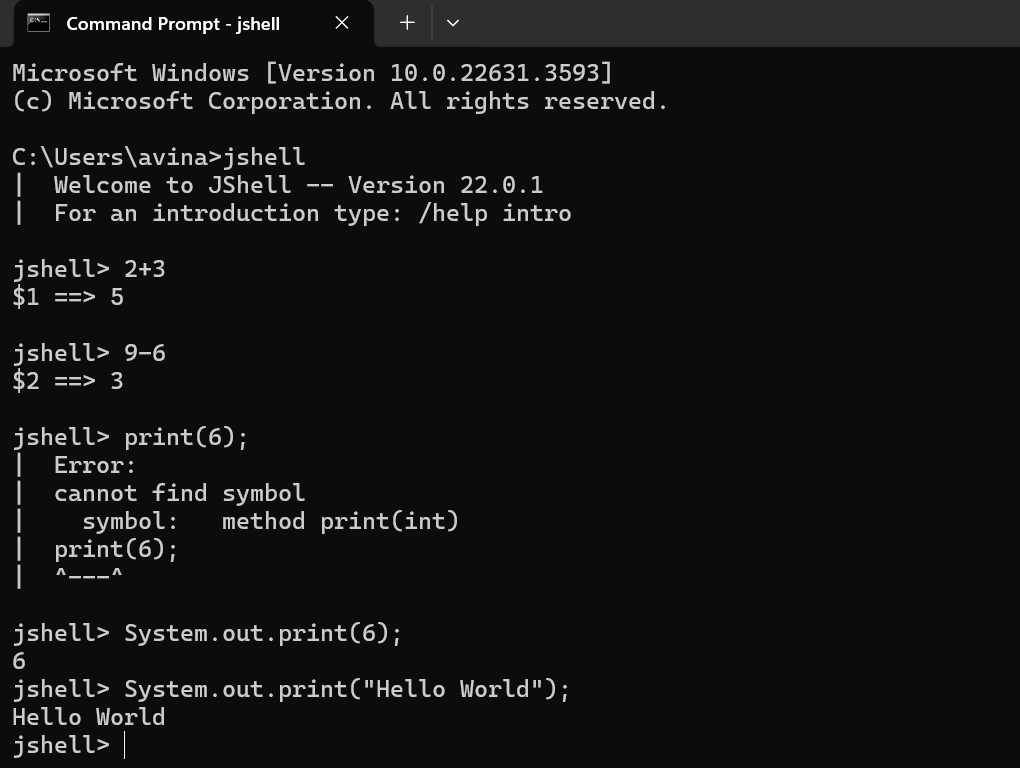
Package -> All the related file of java or we can say dependency.

IDE -> Type code, Run Code, compile code and Run Code.

The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), and other tools needed in Java development.

javac –version -> java Compiler

Jshell introduced java9 in which we can write one line code and get output.



Java is Platform Independent means you can your application on any machine but JVM is platform dependent.

Your machine should have JVM

JVM will execute the code, but JVM will not accept the code directly because it only understands byte code.

Compiler is responsible to convert machine code into byte code and that byte code goes on JVM and it will run.

The first file should have main method so JVM will understand from where I have to execute the code even if you have a thousand files.

The extension of byte file is .class and it is automatically when you compile your java code. Notice…

JVM is a part of JRE.

Java Is WORA.

Using Terminal to run java file -> javac filename.java this is done for compilation

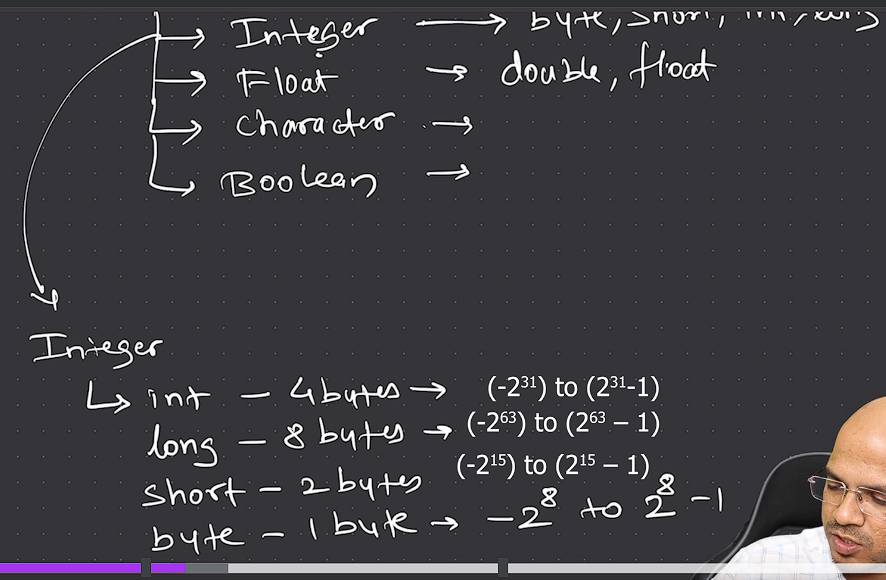
And then to run code we have to type java filename.

**1. JDK** (Java Development Kit) is a Kit that provides the environment to **develop and execute(run)** the Java program. JDK is a kit(or package) that includes two things

* Development Tools(to provide an environment to develop your java programs)
* JRE (to execute your java program).

**2. JRE** (Java Runtime Environment) is an installation package that provides an environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.

**3.**[**JVM** (**Java Virtual Machine)**](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/)is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an [**i*nterpreter***](https://www.geeksforgeeks.org/compiler-vs-interpreter-2/)**.**

 byte is -27 to 27-1 (Wrong in Photo) therefore range is 256 i.e (128+128=256)

By default Java support double(if you want to go with higher precision) data type. Explicitly you have to define float like this -> float a =5.6f;

Char here 2bytes(It has bigger range) unlike in C++ char is 1 byte

Java UNICODE and C++ ASCII code.

In java Booleans have true and false value unlike in C++ we can represent true with 1 and false with 0 but in Java we don’t.

When You do explicitly it is casting or Explicit Conversion and when it is happening automatically is called conversion (Implicit Conversion).

There is also one type of conversion which is Type promotion which is basically int = byte\*byte;

Default value of Boolean data type is false.

High order bit represent sign of integer.

When You try to fetch the value from Increment and decrement it behaves differently.

Post Increment -> fetch the value then increment

Pre Increment -> Increment and then fetch the value.

In Relational Operator the output we receive Boolean value.

&& and || -> double and and double pie is called short circuit.(In or operator if first one is true ,you are not checking the second one so u r saving your time that’s why its called short circuit.)

**Switch Case:** Based on that some particular value we will execute a particular case.

There is certain use cases where you might want to repeat the same things multiple times then loop comes into picture.

In do-while -> if you want to execute the code blocks even if the condition is false.

If You know how many iteration you want to go for then go for loop.

**OOPS:-**

A class is a blueprint of object.(Designing will going to happen here)

Java is **statically** **typed** means at initialisation you have define what type of data it is and also what type of data it is going to return.

Every object known something(methods) and every object does something(behaviour).

JVM is responsible for creating object.

\*\*\*

Calculator calc // just creating reference not creating object. (calc is reference variable)

= new Calculator(); // here we are creating the object of class.

**JDK,JVM and JRE:-**

What ultimately run onto your machine is byte code.

To make java platform independent we went for a virtual layer on top of it. The idea is to have JVM on every machine it will run your java code.

**JRE**-> suppose in java file your code uses some inbuilt classes and libraries maybe we need some extra files that you are going to use in your java file -> JRE comes into pictures

To run your code on JVM you need some extra file and that’s JRE will provide you.

JRE is environment where you run your code.

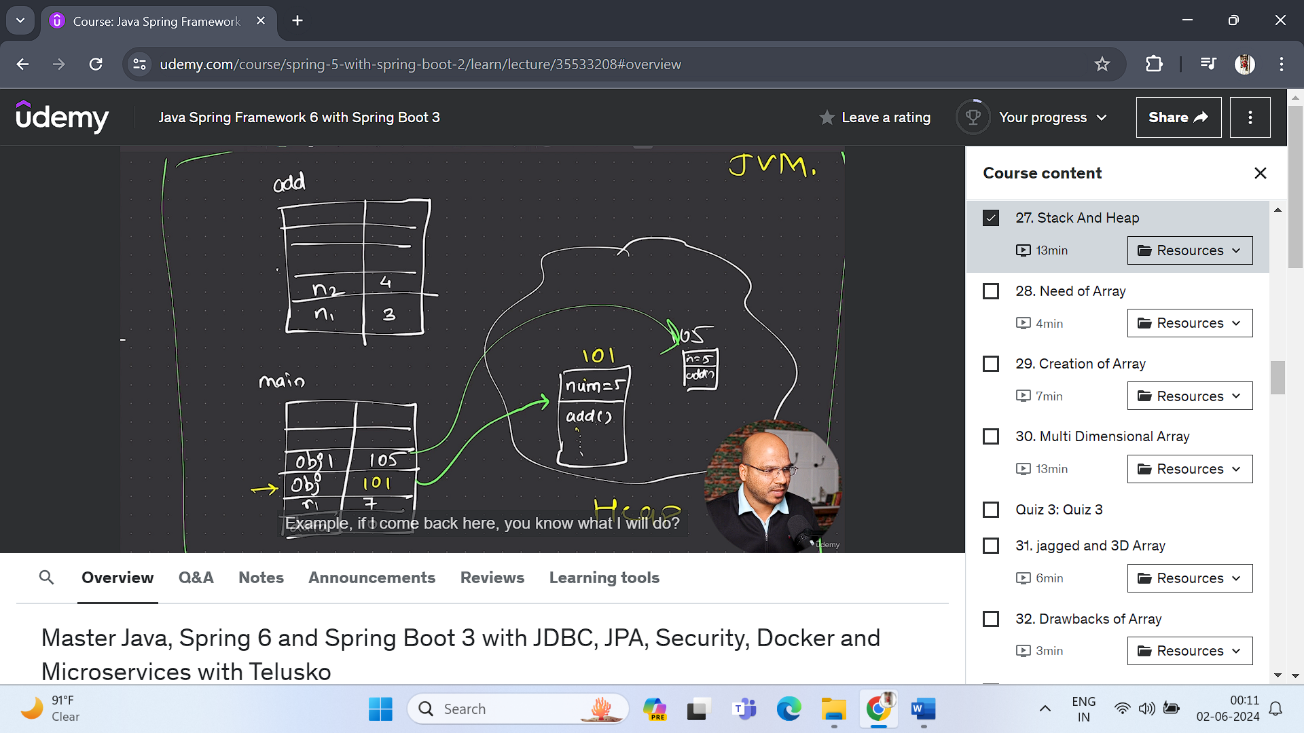
We create any software with different component like in programming we create class.

After return statement is executed it will stop execution.

**Method Overloading ->** Multiple function can have the same name but type or number of parameters are different.(The return type doesn’t matter)

**Stack and Heap: (**Refer Telusko Udemy**)**

In stack we store data in different sequence and in heap you have a open space.

Every method will have its own stack. Stack will have a key(name of variable) and value.

Local variables are part of stack not heap memory.

Instance variables are the part of heap memory

Object will be created inside heap memory.

In heap memory it will have add() method but it only have definition of it, the actual area which add will consume will be of stack.

**Types of variables**: Static variable, instance and local variables

**Instance variable** : A variable which is declared inside a class but outside of all methods.

**Static Variable :** Static variable are shared by different object and if you want refer static variable you can use class name.

In non-static method you can use static variable there is no harm.

OR for static variable we can say we are making something a class member not an object member.

And Also you are saving memory as well because not every object will have their own value for it.

If we want common value for all object we can got for Static variable.

* The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc.
* The static variable gets memory only once in the class area at the time of class loading.
* It makes your program **memory efficient** (i.e., it saves memory).

We can call static method with the help of class name.(No creation of object)

We can use static variable inside static method but you cannot use non-static variable inside static method.

Instance variable/non static variable will be different for different object.

Non-static method (creation of object is required)

Static method (No creation of object is required)

**Needs of Array :** If you want to store multiple value in one particular variable we can use array.

int num[]={1,2,3};

int num[] = new int[4]; // if we don’t know the value initially. By default all the values will be zero. (Dynamic Way)

Top of Form

Which subsystem of the JVM is responsible for loading class files -> Class LoaderBottom of Form

What does arr[arr.length-1].length represents in a multidimensional array? -> Length of the last row in the array.

In Multi-Dimension(Array of Array) we can specified number of rows you want.

A jagged[array](https://en.wikipedia.org/wiki/Jagged_array) is an array of arrays such that member arrays can be of different sizes, i.e., we can create a 2-D array but with a variable number of columns in each row. These types of arrays are also known as Jagged arrays.

**Drawback of Array:**

In java array is object(the moment u said new it is an object it is created inside heap memory)

Array: It will occupy memory space continuously.

**Fixed Size** (You cannot increase size after allocating size, you can do one thing that you will new array with that size and copy those elements into a new array but it will consume lot of time and also computational will happening)

So, **for searching, for inserting value we have to traverse it does consume a lot of time.**

What if **we want array of different type**

These are drawback of Array and then come **Collections.**

**Exception are runtime errors**, the error which comes at runtime. At compile time there is no error.

Student students[] = new Student[3] // here we not creating three indistinct object, here we are creating an array which can hold student references.(It will not create object itself)

You have manually create object and assign to an array.

**Refer-> Student.java and ArrayOfObj.java**

**For-each LOOP: (Refer-> foreach.java)**

It will only array type of data or collections

In For-each loop or enhanced loop there is no counter variable, condition, No exception and length of array to check, we are giving, we just simply printing the value for array.

It will just iterate between the nums and iterate till last elements.

for(int n: nums)

System.out.println(n);

**String:-** If we want to store multiple characters. In java String is a class and also, we have some methods.

String num = “Avi”;

String num = new String(“Avi”);

Strings are **immutable** that are unchanged or once you created object you cannot changed it.

What if we want mutable string then we can go for **String Buffer and String Builder.**

Example: String s1 = ”Avi”;

String s2 = ”Avi”;

Here in above example here we are creating two references(s1 and s2 in stack) and only one object (**String Constant Pool**) -> We have a special area in heap memory

**String literal is actually a constant** and this is how we are trying to **save** **memory**.

A Java String Pool is a place in heap memory where all the strings defined in the program are stored. A separate place in a stack is there where the variable storing the string is stored. Whenever we create a new string object, JVM checks for the presence of the object in the String pool, If String is available in the pool, the same object reference is shared with the variable, else a new object is created.

Refer (Mutable and Immutable String).

The capacity of **String** **Buffer** changes it will store your name and also it will give some extra space,(if it consume continuous memory location inside your heap memory) what if there is no extra space available next, in that case it has to relocate. So, reduce relocation of string it gives you a buffer memory.

**String Buffer is thread safe and String Builder is not thread safe.**

The Static variable is shared by all object.

**Static Method :- (Refer Static method)**

You can static method directly with the help of class name.

You can use static variable inside static method but you cannot used a **directly** non-static variables inside a static methods because non-static variable or instance variable is different for different object.

**Indirectly** we can use by passing object references.

**Static Block:-** In Static block we basically initialise static variable because in constructor if we initialise static variable it will initialise static variable every time when object is created.

The good thing is static block will be called only **once** irrespective how many objects you create.

**Static blocks is used to initialise the static variables.**

Class is loaded only once. Then object is created. There is Two Steps..

In JVM there is special area which is called **class** **loader** and this will have all the class which are loaded and it will happen only once.

Every time you load a class it will call **static** **block**.

That’s why static block is executed first before creating of object.

**If we don’t create object, it will not load the class as well. But what if we want to load the class?**

**For that we have special class inside java which is called Class Class.**

**Inside class there is method forName is method which does one thing, which loads your class.(Not create object)**

**Class.forName(className: “Mobile”); //Mobile is a class name.**

**Encapsulation:-** Keeping it tight that no one from outside world can use it.

We are making our variable private and then the only way to access them is with the help of methods(Getter and Setter). So basically we are binding data/variables(private) with the methods. The only way to access that data is with methods. So no one can outside world can use your variable directly.

So somewhere we encapsulating our data and methods and that is Encapsulation.

**Getters and Setters:-** There is no compulsion that there is should be both methods getter and setter.

**TIPS-** Right click source action and generate getter and setter methods in VS code.

**This keyword :-** this is a keyword which represent current object. The current object means the object which is calling the method.

**Constructor:-** What if I want to assign value when object is created, I don’t want default value.

Constructor looks like a method itself.

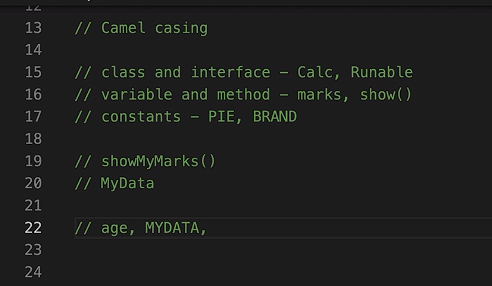
There’s compulsion constructor name should be same as class name.

**Constructor is special method, it has same name as the class name, the constructor never return anything.**

**Every time you create an object it will call constructor.**

**Two Types of constructor : Default/Normal constructor and Parameterised Constructor.**

**How can I achieve constructor overloading?**

****

**Naming CONVENTION**

**Anonymous object :-** Object which have no name for anonymous object there is no references. -> new A() // creation of object. Or Anonymous object

A obj = new A() // obj is not object, obj is reference variable. Or we called reference object.

**Inheritance :**

Basically you can use the features of super class.

For Inheritance we need class file not java file. To inherit superclass we use a keyword called **extends.**

**Multilevel Inheritance : Whenever we have multi-level inheritance -> Class A -> Class B -> Class C.**

**Multiple Inheritance does not support in Java.**

**Whenever you get object of a class, it will call the constructor of subclass and superclass both.**

**NOTE: Every constructor in Java has a method which is there and that method is super(). This method is there even you don’t mention. So by default, every constructor the first statement is super().**

**Super() method simply means call the constructor of a super class. (CALL THE CONSTRUCTOR OF SUPER CLASS)**

**Every class in JAVA extends(Inherit) the object class. Object is a class in Java.**

**this() method will execute the constructor of a same class.**

**Refer -> This and super method video.**

**Method Overriding : Where we can override the methods of parents class.(name and type/number of parameter should be same)**

**Package:** A set of classes and interfaces grouped together are known as Packages.

It is always a good idea to put your classes in a packages.(Just to give organized structure) Package is like a folder.

Whatever class you use in java is belongs to package.

If you want all file in package just use (\*)

**import java.lang\*; // \* simply means all the files not all the folders.**

A library is a **collection of pre-written code that you can use to perform specific tasks.** Libraries are often used to reduce the amount of code a programmer needs to write by providing reusable functions or classes that can be called upon as needed.

A framework is a set of pre-written code that provides a structure for developing software applications. A library, on the other hand, is a collection of pre-written code that can be used to perform specific tasks or to optimize task.

**Access Modifier: To Control the visibility of class members (The security level of each individual class and class member)**

If want something to be accessed outside the package always make it public. Public means it can accessed from anywhere.

Private variable can be used in the same class irrespective of the package.

By default there is default access modifier when you not mention anything. Default simply means it can be accessed in the same package.

If you make it protected you can’t use outside the package but you can access outside the package but it should be subclass (**INHERITANCE**) -> (Work in the same package).

If you don't use any modifier, it is treated as **default** by default. The default modifier is accessible only within package. It cannot be accessed from outside the package.

**Polymorphism:**

Theirs is something which is behaves differently in different situation based on situation.

Your behaviour or your execution will change based on some situation.

Compile time polymorphism(Early Binding) -> if behaviour is defined at compile time.

And Runtime time Polymorphism(Late Binding)-> if behaviour is defined at runtime

**Dynamic Method Dispatch :**  Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time.

So, it’s not about what is a type of the object or reference, it will only refer **object**

**NOTE :** Irrespective what type of object you have or what type of variable you create , it depend what object you have. Refer -> Dynamic Method Dispatch.

**Final Keyword :**

Final keyword is used with variable, class and method.

If you make your variable constant you can not change the variable later. In java we use final to achieve this(In c++ we const).

Final variable simply means making constant.

Final class -> No one should able to extends my class (Inheritance) -> You are basically stopping the Inheritance.

Final method -> To stop method overriding. (No one can override it)

**Object class equals to string hashcode:**

Every time you try to print the **object** by **default** it will call the **toString**() method

Hashcode : it tries to create a single string of all data which you have, So lets say if you have object which has five variables, it will apply hash algorithm and will generate a string number string which will be a fixed size.

When you say two object are equal, they should have same value and they should have same hashcode. (Go with Source Action With IDE) (Refer This Video)

**Upcasting and Downcasting:**

**Wrapper class : For every primitive type we are going to have a class for it (Integer class extends the object class)**

**It goods if we want to work with Collection framework which only support object. (There are certain requirement where you have to use classes)**

**Boxing -> Taking primitive value and storing that in object.**

**Unboxing -> Object type to primitive type. (When you take out primitive value from an object)**

parseInt is a method which will take the string and it will convert into Integer.

This particular syntax has been deprecated. Deprecated simply means it was there, Don’t use it now because in future release they might remove it or we have a better alternative for that.

**Abstract keyword:** You can use abstract keyword here, it means I’m giving you the abstract data of a car should have drive features but I don’t know how to implement it.

If you want to declare a method.

Abstract method only belongs to abstract class.

You can’t create object of abstract class. (You can create a reference of object class.)

A **concrete** class is **a class that has an implementation for all of its methods**.

**Inner Class :** A class inside a class.

**Anonymous Inner class :** Anonymous means something which doesn’t have name(Class).

(Inner class)

Refer : VIDEO

**Interface:** An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.

There are mainly three reasons to use interface. They are given below.

* It is used to achieve abstraction.
* By interface, we can support the functionality of multiple inheritance.
* It can be used to achieve loose coupling.

When we have class which only have abstract method then another alternative is Interface.

Every Interface method is Public Abstract by default.

Interface tell you what are the method you need but I will not implement that.

Whenever you want to implement the interface, you say **implements** keywords.

All the variables inside the interface are by default **final and static.**

* If you want to print object you have to use toString() method. Every time you try to print object it will call toString(). toSting() will print the data in a particular format.
* We can use == operators for reference comparison (**address comparison**) and .equals() method for **content comparison**. In simple words, == checks if both objects point to the same memory location whereas .equals() evaluates to the comparison of values in the objects.

| **Class** | **Interface** |
| --- | --- |
| In class, you can instantiate variables and create an object. | In an interface, you can’t instantiate variables and create an object. |
| A class can contain concrete (with implementation) methods | The interface cannot contain concrete (with implementation) methods. |
| The access specifiers used with classes are private, protected, and public. | In Interface only one specifier is used- Public. |

**Java Abstract Method**

The abstract Method is used for creating blueprints for classes or interfaces. Here methods are defined but these methods don’t provide the implementation. Abstract Methods can only be implemented using subclasses or classes that implement the interfaces.

**Can a class implement multiple inheritance? -> one class can implements multiple interface.**

**That’s not the case of abstract class, in abstract class we can extend only one class**

**Interface also support inheritance.**

**Need of Interface:**

To achieve lossely coupled.

**Enum:** An enum is a special "class" that represents a group of **constants** (unchangeable variables, like final variables).

An enum cannot be used to create object and it cannot extends other classes (but it can implement interface).

Enum in java extends Enum class.

**Annotation :** Sometimes you want to interact with compiler by saying something, i.e **it’s just that we want to supply some extra information to the compiler or to the runtime. We also sometimes refer as meta data.**

There are some annotation which works on variables, methods and also at class level.(Deprecited)

You can also set the retention level whether it is compiler level or runtime level.

@override -> Method Overriding.

**Types of interface : Normal Interface, Functional Interface(SAM) and Marker Interface.**

**Normal Interface –** Whenever you have a interface with two or more methods is Normal Interface.

**SAM(Single Abstract Method) –** if you have only **one** method.

**Marker Interface –** A marker interface is interface which has **no** methods. **Blank** Interface -> is to update something to the compiler, if we want to talk to compiler by saying “please allow this”.

**Serialization** : You can take the object and you can store the value of object in your hard drive.

By default, every object is not allowed to do that, you can give the permission of marker interface by creating class.

Serialization in Java is **a mechanism of writing the state of an object into a byte-stream**

**Functional Interface(SAM):**

@FuntionalInterface -> Annotation

**Verbose :** Java is no more verbose (in java 8) means is specifying everything, making a detailed code (Its good most the of time to read the code, but sometimes you need to shorten the code to reduce the number of line(To reduce bug)).

For this lambda expression is introduced and we can use lambda expression only with functional interface.

Verbose is an option that can be used on the Java command to enable debug output for class loading.

In computing, Verbose mode is an option available in many computer operating systems and programming languages that provides additional details as to what the computer is doing and what drivers and software it is loading during startup

**Lambda Expression -> Reduces the code**

**Lambda** Expression is only work with **functional** interface.

**Exceptions: Runtime error we called them Exceptions. Runtime Error will stop your application.**

We can define **checks** to avoid exceptions.

Error – Compiler time, Run time, and Logical Error

In try blocks we are handling the exception if there is exception or not, if there is exceptions, the compiler would throw an error (object will throw) and catch block will accept that error (Exception e).

And it will not stop your program executions.

Put only **critical** statement in try blocks that may throw an error.

**Try is basically try to execute the statement, if everything goes good, it will execute the remaining statements, and if somethings goes wrong if there’s Exception, it will execute the catch block.**

**Exceptions extends Throwable class.**

**Make sure when you are handling exceptions your Exception class should be at bottom. (Parent class at the end is a compulsion)**

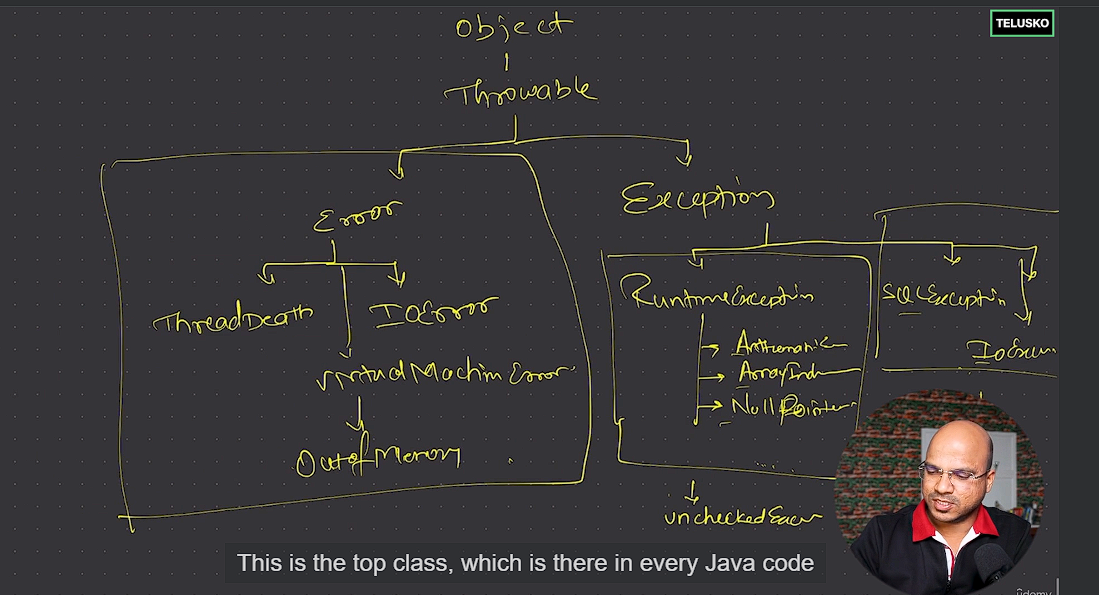
Arithmetic Exception, ArrayIndexOutOfBound Exception and NullPointerException are different type of exception which extend Exception class (Parent class)

Exception is a superclass and all exceptions above are subclasses of its.

Every **class** in java **extends** **object** class.

**Checked** **Exceptions** -> You have to handle it exceptions compulsory.

**Unchecked Exceptions ->** You have a choice whether you have to handle or not.



**Exception throw Keyword:** Throw keyword is used to throw(object of that exceptions) the exception and catch block will be catching it.(if we want to call the catch block then you can go for throw keyword (In case of you don’t have any specific exceptions-> Custom Exceptions))

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](https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/). The throw keyword is mainly used to throw custom exceptions.

**Custom Exception:**

**Ducking Exception using throws:**

throws is a keyword in Java that is used in the signature of a method to indicate that this method might throw one of the listed type exceptions. The caller to these methods has to handle the exception using a try-catch block.

**User Input Using BufferReader and Scanner:**

println method belongs to **PrintStream** Class and **out** is the **object** of **printStream** and this object is created as **static** variable inside the system class.

BufferedReader is a class which basically work with I/O and belongs to a package java.io.

**Try with resources:**

Normally we put critical statement in try blocks and exception is throw in a catch block.

Try is basically used to handle the exceptions

**Finally** (You can write statement in finally block) has the features irrespective if u got the exception or not, it will execute the finally block.

**Finally** block will executed **irrespective** of **exceptions**.

**Finally** make sense when we want to **close** the **resources** (Database/Network Connectivity and BufferedReader and any file to save data leakage and to save resources)

**Try** with **Resources**(Resources closed automatically)

**Threads:**

Ram acts like a temporary memory for your processing and CPU is something which executes somethings.

**Dividing your task into small things.(The smallest unit)**

**Threads allows a program to operate more efficiently by doing multiple things at the same time.**

**Threads can be used to perform complicated tasks in the background without interrupting the main program.**

A thread in Java is the direction or path that is taken while a program is being executed.

A thread in Java is like a separate **mini**-**program** running inside your main program.