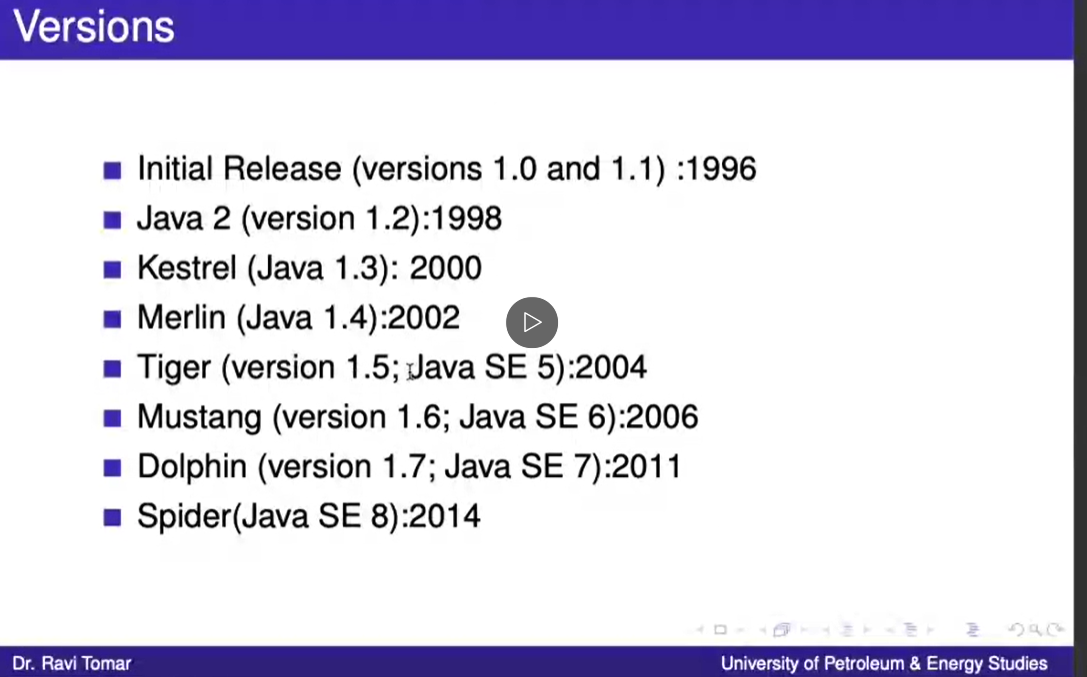
**ASSIGNMENT 2**

Based upon your understanding of the last 11 Lectures held, you are required to pen down your understanding Lecture wise.

**Lecture 1 Date: 21st January 2021**

**Versions of JAVA**



**Features of JAVA**

1. **PORTABILITY**

* Your applications are portable across multiple platforms. Write your applications once, and you never need to port them—they will run without modification on multiple operating systems and hardware architectures

**EXAMPLE**

**If we write a code in c language and compile it on linux it will create an filename.out file which can be executed on linux OS , While if we complie the same code on Windows it will create an filename.exe file**

**So we can not execute the .out file on windows or .exe on linux**

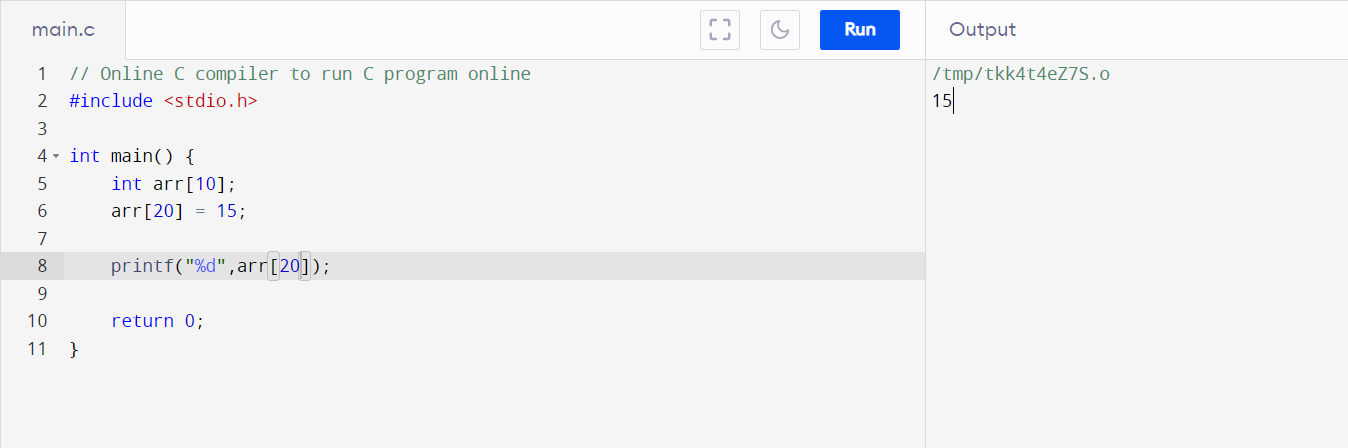
**This states that c is not a poprtable language.**

**In case of java .class file is created which can be executed on any operating system which has java installed in it via JVM (java virtual machine) which makes java a portable language.**

1. **ROBUST**

Your applications are robust because the Java run-time system manages memory for you.

EXAMPLE

In c language 

We will get the out put but its is not ethically correct .

Suppose if the base address is 100 so we are allowed to access only 100 + 10\*4 = 140 memory but here we are accessing 100 + 20\*4 =180 so the program executed successfully but ethically its not good as there can be some important process running at that part of the memory and you end up crashing the OS.

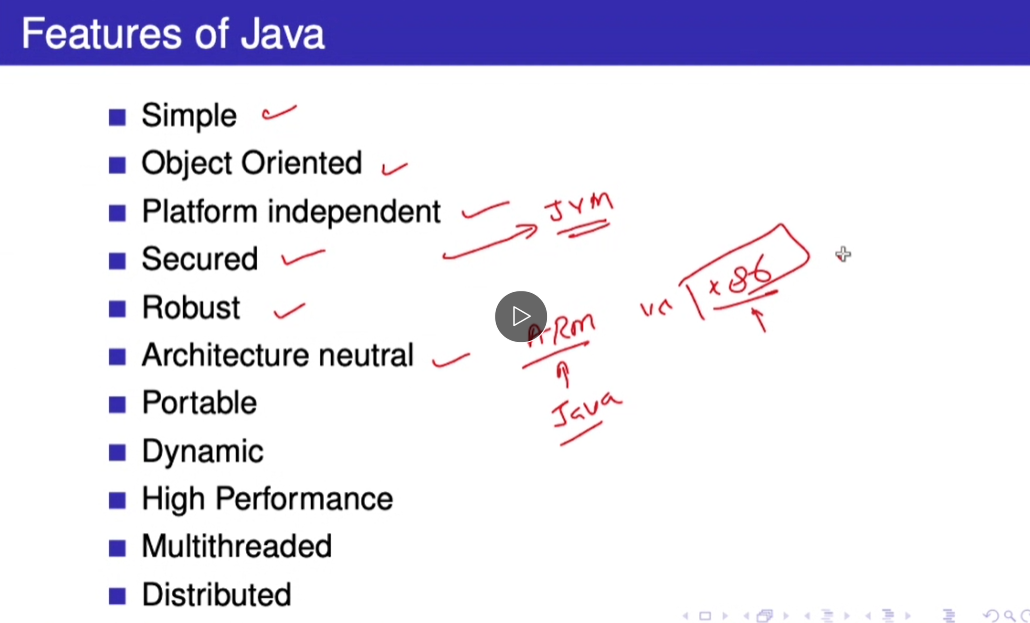
While in java JVM is assigned with a block of memory and no variable can access anything outside the block thus making it a robust language.

#Java is simple language as compared to C and C++

-----------------------------------------------------------------------------------------------

**Lecture 2 Date: 28st January 2021**

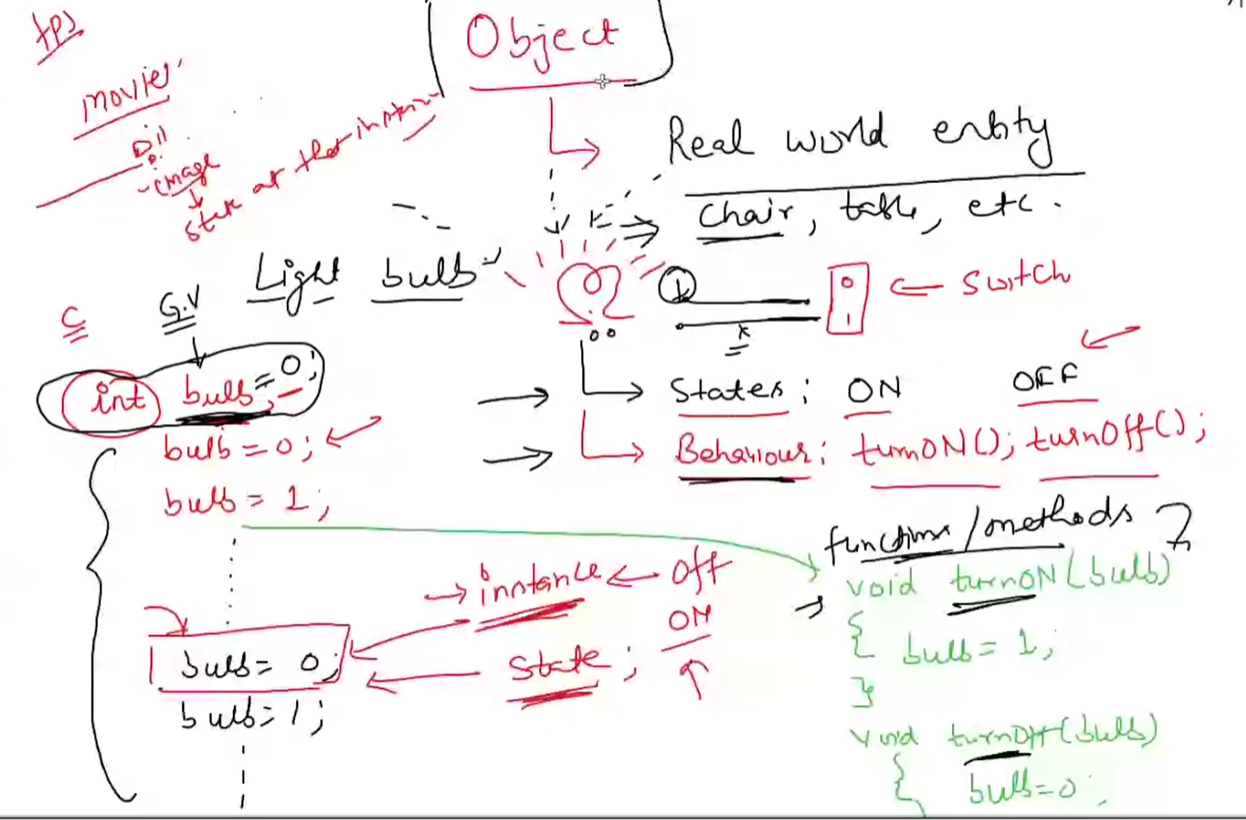
**FEATURES OF JAVA**



* **Object Oriented**  
  In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
* **Platform Independent**  
  Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
* **Simple**  
  Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
* **Secure**  
  With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
* **Architecture-neutral**  
  Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
* **Portable**  
  Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
* **Robust**  
  Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.
* **Multithreaded**  
  With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
* **Interpreted**  
  Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
* **High Performance**  
  With the use of Just-In-Time compilers, Java enables high performance.
* **Distributed**  
  Java is designed for the distributed environment of the internet.
* **Dynamic**  
  Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

**Source : Assignment 1**

**OBJECTS**



From the above example on white board of light bulb we understand that any real life entity can be called an object

Light bulb has 2 states ON and Off

And two behaviours TURN\_ON ,TURN\_OFF

So a bulb can be on or off at any instance and we have methods or functions to change its states .

# An object that is created using a class is said to be an instance of that class

We can say that anything that has some states and behaviours can be called as an object

---------------------------------------------------------------------------------------------------------

**Lecture 3 Date: 1st February 2021**

**Object inside object**

**Ex. Car -> engine**

**Fan -> capacitor,motor,blades,wires**

**MODULARITY**

Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality.

INFORMATION HIDING

It is a technique t hide the details of internal implementation hidden

From outside world.

CODE RE USE

Reusability in OOP achieves through the features of C++ where it possible to extend or reuse the properties of parent class or super class or base class in a subclass and in addition to that, adding extra more features or data members in the subclass or child class or derived class.

PLUGGABILITY

If a particular object turns out to be problematic, you can simply remove it from your application and plug in a different object as its replacement.

**Lecture 4 Date: 3rd February 2021**

**CLASSES**

a class is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods).

Syntax

Declaring a class

class ClassName

{

----- variables------

------functions------

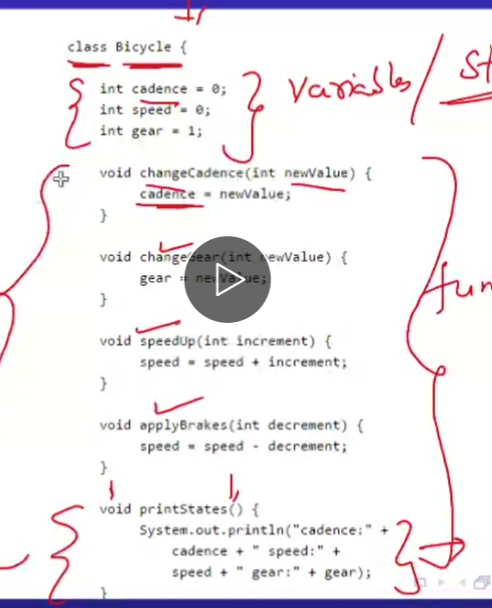
------dashboard-----

}

Given below is an example of class name bicycle to understand the above blueprint

We define class then we define the variables or states , further we have functions to change states.

At last we have a print function which just displays us the current state of our objects .



Inheritance

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance the information is made manageable in a hierarchical order.

Ex. All two wheelers share common characterstics from bicycle so we can say that bicycle can be the parent class and all other two wheelers can inherit the properties from bicycle and can have their special functions as well.

# child classes are known as sub class

Syntax

Class MountainBike extends Bicycle

{

-------Extra methods for mountain bike------

}

Interface – A contract

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

What is the level of access given to user in form of a contract can be called as an interface.

Ex. Netflix giving limited access according to the plan you subscribe .

So we are bind up in a contract of content that we can see.

Syntax

Public interface Bicycle

{

------------- functions-----------

}

to implement a interface we use:

class ATLASBicycle implements Bicycle

{

}

--------------------------------------------------------------------------------------------

**Lecture 5 Date: 4th February 2021**

**PACKAGES**

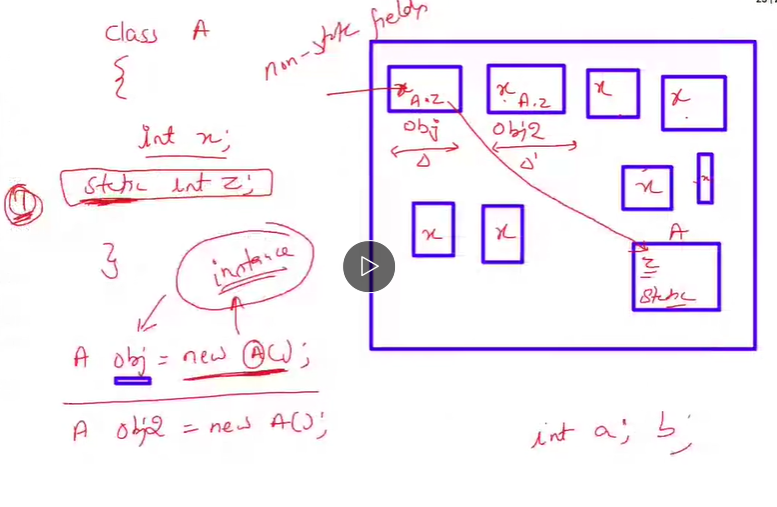
Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces.

API – application programming interface

# a string object contains state and behaviour for character strings , and other genral purpose tasks have already built packages in java.

VARIABLES

* INSTANCE VARIABLE : Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.
* STATIC VARIABLE : Static variable in Java is variable which belongs to the class and initialized only once at the start of the execution. It is a variable which belongs to the class and not to object (instance ).
* LOCAL VARIABLE : A local variable is a variable declared inside a method. A local variable is only accessible inside the method that declared it.
* PARAMETRES :  When we define a function we introduce our compiler with some variables that are being used in the running of that function. These variables are often termed as Parameters.

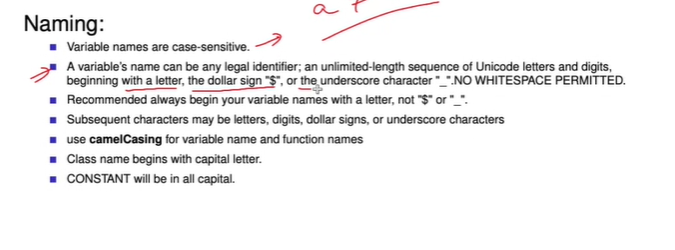
In the above in class A a x variable is defined which is copied in every object of that class , x has an instance in every object.

X – instance variable

While static z is allocated a separate memeory and it is used whenever required in and outside of the class

Z – static variable

VARIABLE NAMING RULES :



--------------------------------------------------------------------------------------------

**Lecture 6 Date: 8th February 2021**

**Is java completely object oriented programming language?**

* **Java has some primitive data types so we don’t need to create class for strings and all they are all already defined .**

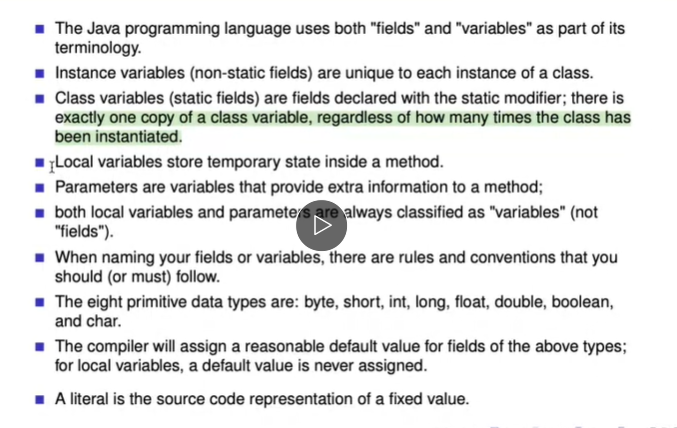
**So java is not completely object oriented programming language .**

LITERAL:

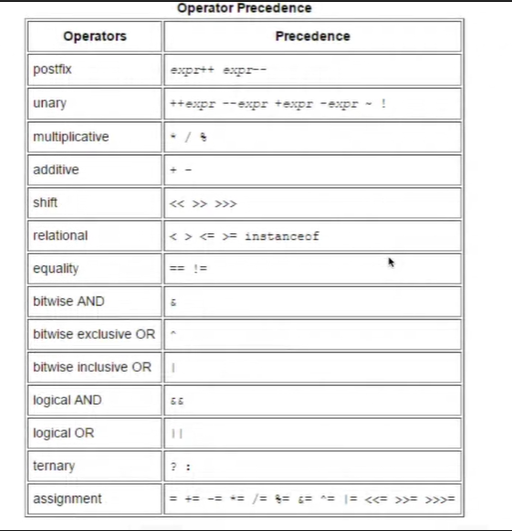
A literal is the source code representation of a fixed value.

Ex . for large integar values we use ‘l’ at its end to let java know that it is a long in data type.

Summary



OPERATORS



# operators changing their behaviour on increasing the no. of operand is called operator overloading.

Ex +a (unary operator)

a + b (airthematic operator)

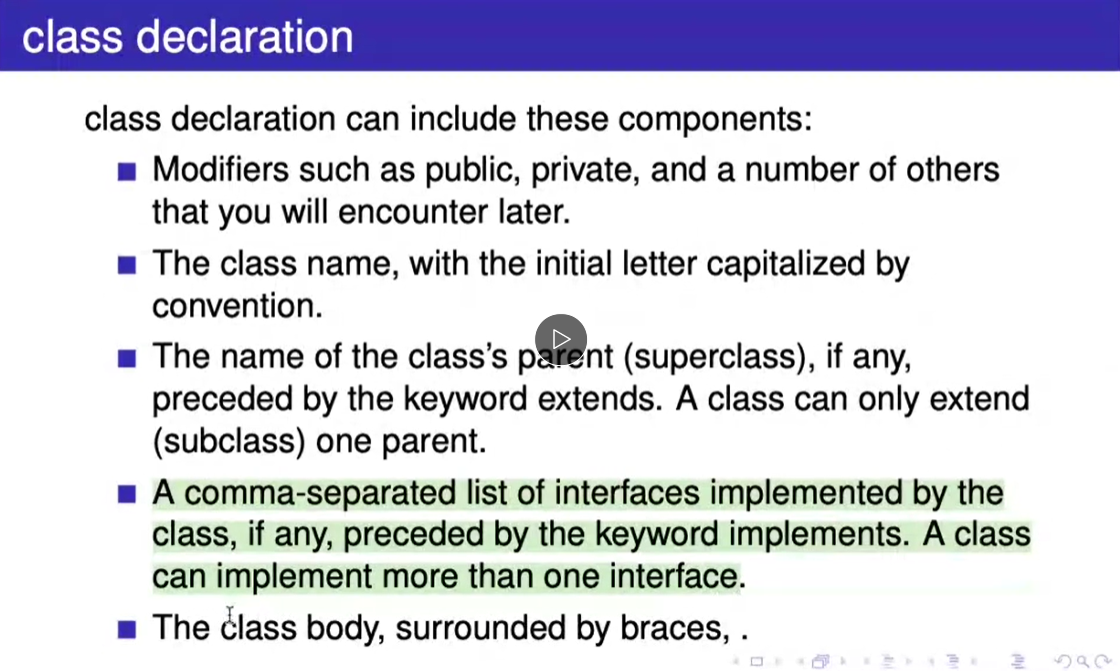
CONTROL FLOW STATEMENTS

However, JAVA provides statements that can be used to control the flow of java code. Such statements are called control flow statements.

**Lecture 7 Date: 10th February 2021**

**Hello.java**

**Declaration of class**

****

Example:

// modifers

public class Bike {

*//fields:*

public int speed;

public int gear;

*//Constructor:*

public Bike(int startSpeed, int startGear) {

this.gear = startGear;

this.speed = startSpeed;

}

*//Methods:*

public void setGear (int newValue) {

this.gear = newValue;

}

public void applyBrake(int decrement) {

this.speed -= decrement;

}

public void speedUp(int increment) {

this.speed += increment;

}

}

#Public variables, are variables that are visible to all classes. Private variables, are variables that are visible only to the class to which they belong. Protected variables, are variables that are visible only to the class to which they belong, and any subclasses.

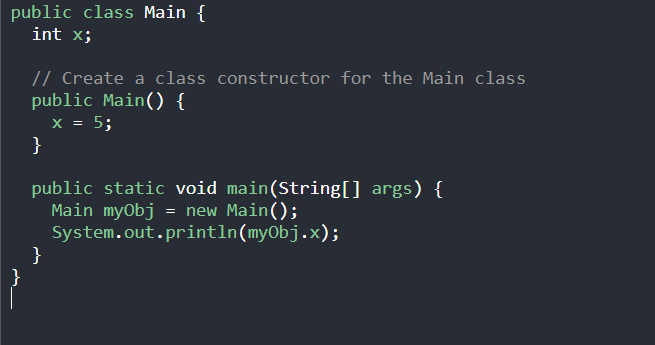
--------------------------------------------------------------------------------------------

**Lecture 8 Date: 11th February 2021**

**CONSTRUCTOR**

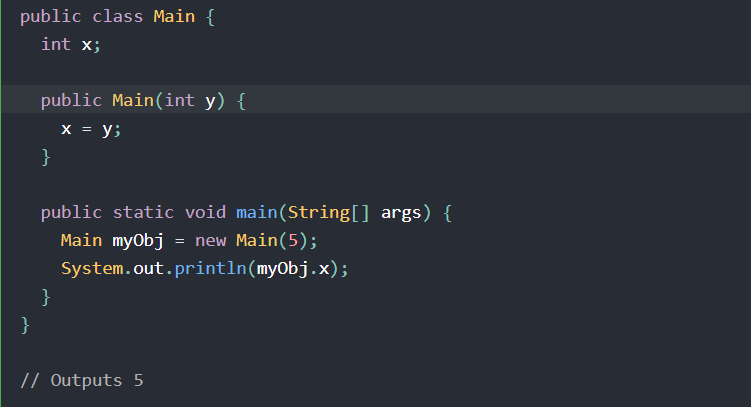
A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created.

Ex



**Output = 5**

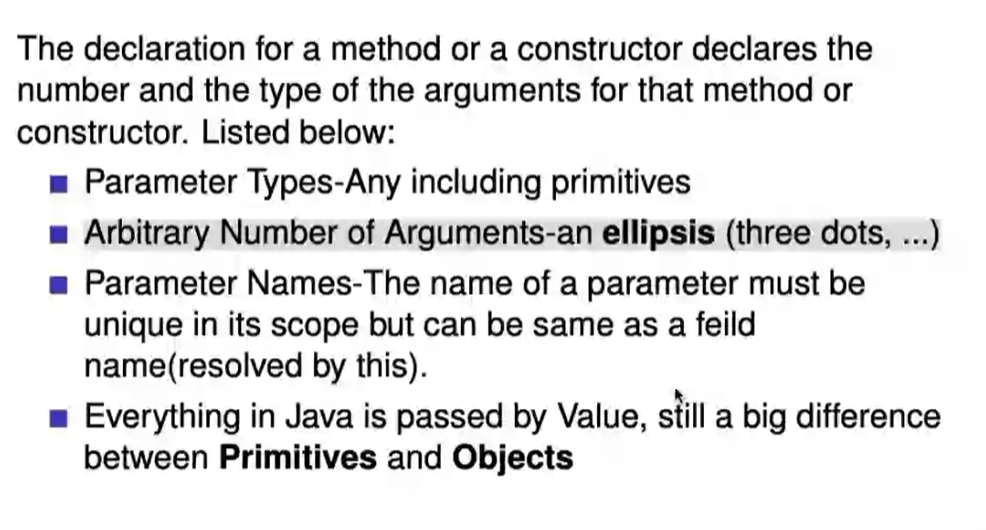
**Constructor with parameter :**



**--------------------------------------------------------------------------------------------**

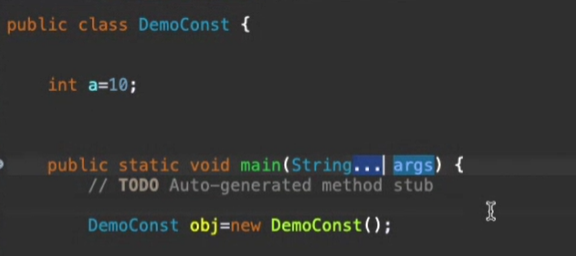
**Lecture 9 Date: 15th February 2021**

PASSING INFORMATION TO A METHOD OR A CONSTRUCTOR



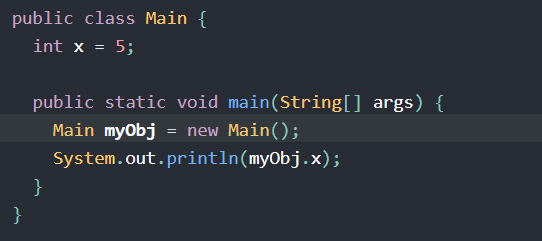
Eplisses

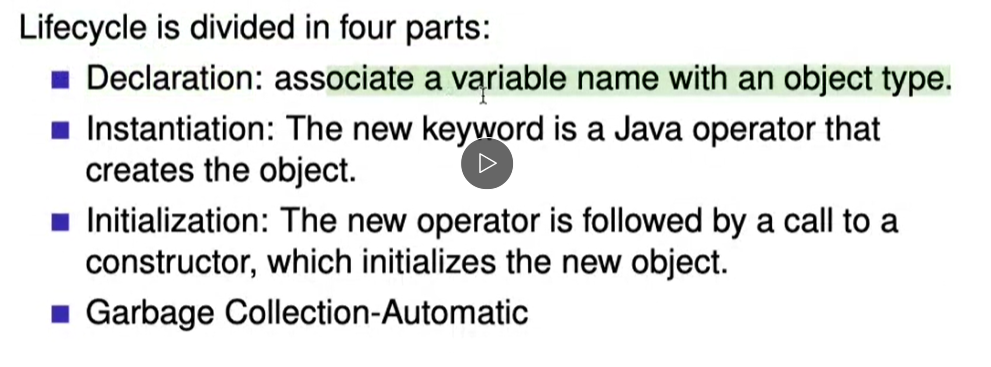
(…) Java syntax that describes an argument in a method that can take in zero or many arguments.



Create an Object:

To create an object of ‘MyClass’, specify the class name, followed by the object name, and use the keyword ‘new’ .





**Lecture 10 Date: 17th February 2021**

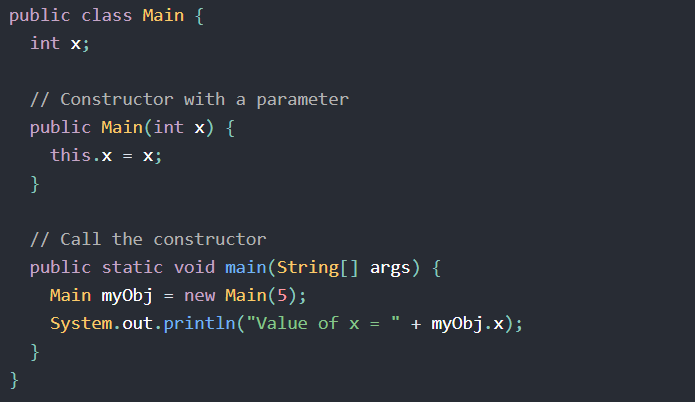
THIS keyword

The ‘this’ keyword refers to the current object in a method or constructor.

Usage of ‘this’ keyword

* Invoke current class constructor
* Invoke current class method
* Return the current class object
* Pass an argument in the method call
* Pass an argument in the constructor call

Ex .



Why is constructor declared first in a class?

* Java enforces that the call to super (explicit or not) must be the first statement in the constructor. This is to prevent the subclass part of the object being initialized prior to the superclass part of the object being initialized.

--------------------------------------------------------------------------------------------