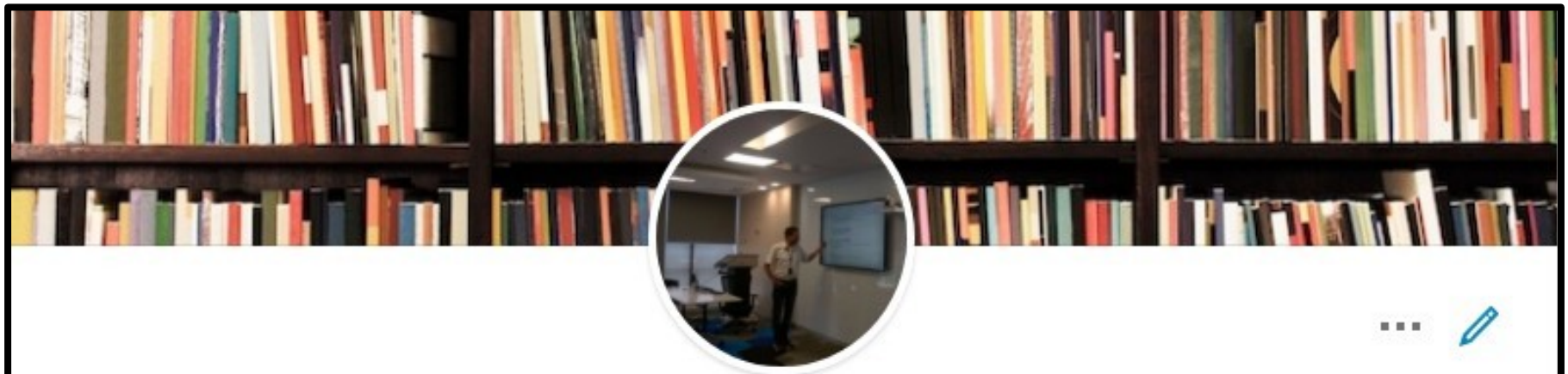
# Core Java

**Rajeev Gupta**
Java Trainer & Consultant (MTech CS)
rgupta.mtech@gmail.com



# Rajeev Gupta

FreeLancer Corporate Java JEE/ Spring Trainer

freelance • Institution of Electronics and Telecommunication Engineers IETE

New Delhi Area, India • 500+

---

1. Expert trainer for Java 8, GOF Design patterns, OOAD, JEE 7,Spring 5, Hibernate 5 ,Spring boot, microservice, netflix oss, Spring cloud, angularjs, Spring MVC, Spring Data, Spring Security, EJB 3, JPA 2, JMS 2, Struts 1/2, Web service

2. Helping technology organizations by training their fresh and senior engineers in key technologies and processes.

3. Taught graduate and post graduate academic courses to students of professional degrees.

I am open for advance java training /consultancy/ content development/ guest lectures/online training for corporate / institutions on freelance basis

**Day-1: Introdcution to Java, JVM, procedural programming, Arrays**

- Introduction to OO Concepts
- Introduction to UML
- Pillars of OO: Inheritance - Polymorphism - Abstraction - Encapsulation
- JVM basics: JDK,JRE and JVM
- Procedural programming: if else, looping, swith etc
- Array, defining usages 1D/2D arrays
- Lab Assignments

**Day 2: Object Orientation & JVM introduction**

- Creating Class, Object, constructor, init paramaters
- Static variable and method
- Concepts of packages, Access specifier
- Inheritance, Types of inheritance in Java, Inheriting Data Member and Methods
- Role of Constructors in inheritance,Overriding super Class methods, super
- Hands On & Lab

**Day 3: Advanced Class Features**

- Loose coupling and high cohesion
- composition, aggrigation, inheritance, basic of uml
  - Abstract classes and methods
  - Relationship between classes- IS-A, HAS-A, USE-A
  - Interface Vs Abstract, when to use what?
  - Final classes and methods
  - Interfaces, loose coupling and high cohesion
  - SOLID principles, Square – rectangle problem
  - Hands On & Lab

**Day-4:Strings,Wrapper classes, Immutability, Inner classes,Lambda expression, Java 5 features , IO, Exception Handling**

- String class: Immutability and usages, stringbuilder and stringbuffer
- Wrapper classes and usages, Java 5 Language Features
- AutoBoxing and Unboxing, Enhanced For loop, Varargs, Static Import, Enums
- Inner classes: Regular inner class, method local inner class, annonymous inner class
- Char and byte oriented streams
- BufferedReader and BufferedWriter
- File handling
- Object Serialization and IO [ObjectInputStream / ObjectOutputStream]
- Exception Handling, Types of Exception, Exception Hierarchy, Exception wrapping and re throwing
- Hands On & Lab

**Day 5: Introduction to Java threads, thread life cycle, synchronization, dead lock**

- Program vs process
- Thread as LWP
- Creating and running thread
- Thread life cycle
- Need of synchronization
- Producer consumer problem
- dead lock
- Hands on & Lab

**Day 6:Java Collection, Generics**

- Collections Framework introduction
- List, Set, Map
- Iterator, ListIterator and Enumeration
- Collections and Array classes
- Sorting and searching, Comparator vs Comparable
- Generics, wildcards, using extends and super, bounded type
- Hands on & Lab

**Day-7: Annotation, Java Reflection**

- Introduction to java reflection
- Java Annotation, JDK annotation, creating custom annotation, Annotation and reflection
- Introduction to Design pattern
- Hands on & Lab

**~~Day 8:~~ GOF design patterns**

- Pattern categories: Creational, Structural, Behavioral
- Creational Patterns: Singleton, Factory, Builder, Prototype
- Structural Patterns: Decorator, Facade Pattern, Proxy
- Behavioral Patterns: Iterator, Observer, Strategy,Template Method
- Hands on & Lab

# DAY -1

rgupta.mtech@gmail.com

**Day-1: Introdcution to Java, JVM, procedural programming, Arrays**

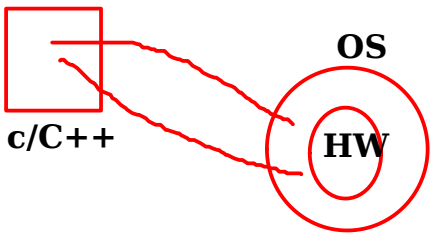
- Introduction to OO Concepts
- Introduction to UML
- Pillars of OO: Inheritance - Polymorphism - Abstraction - Encapsulation
- JVM basics: JDK,JRE and JVM
- Procedural programming: if else, looping, swith etc
- Array, defining usages 1D/2D arrays
- Lab Assignments

What is Java?

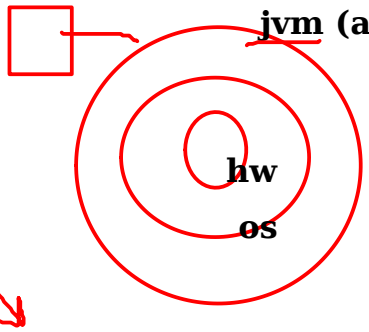
OOPL

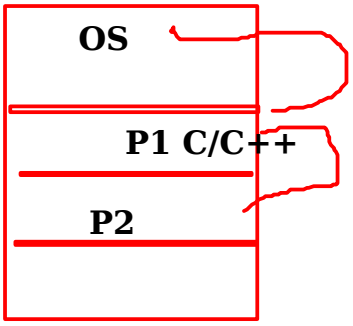
Java vs C++

java =oopl + jvm + rich set of lib




C/C++ are native hw pointer ?

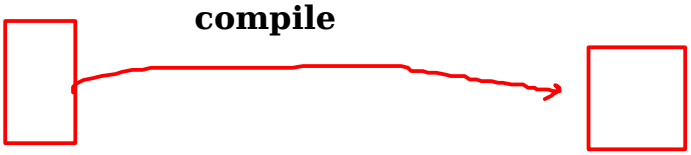



Java:1. prg for network sec
2. it should be PI

compile

:(



demo.cpp
demo.c

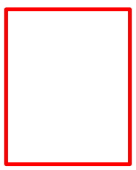
demo.exe

this ex code bind with machine info

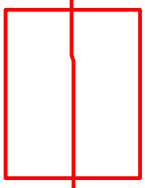
touboc
gcc

Plateform dependent

javac



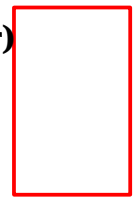
demo.java



java.class

java
(interpreter)

jvm

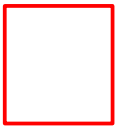


linux

:)

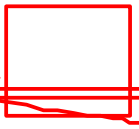
byte code?
optimized instacution set

PI



win

mac



dev phase
PI

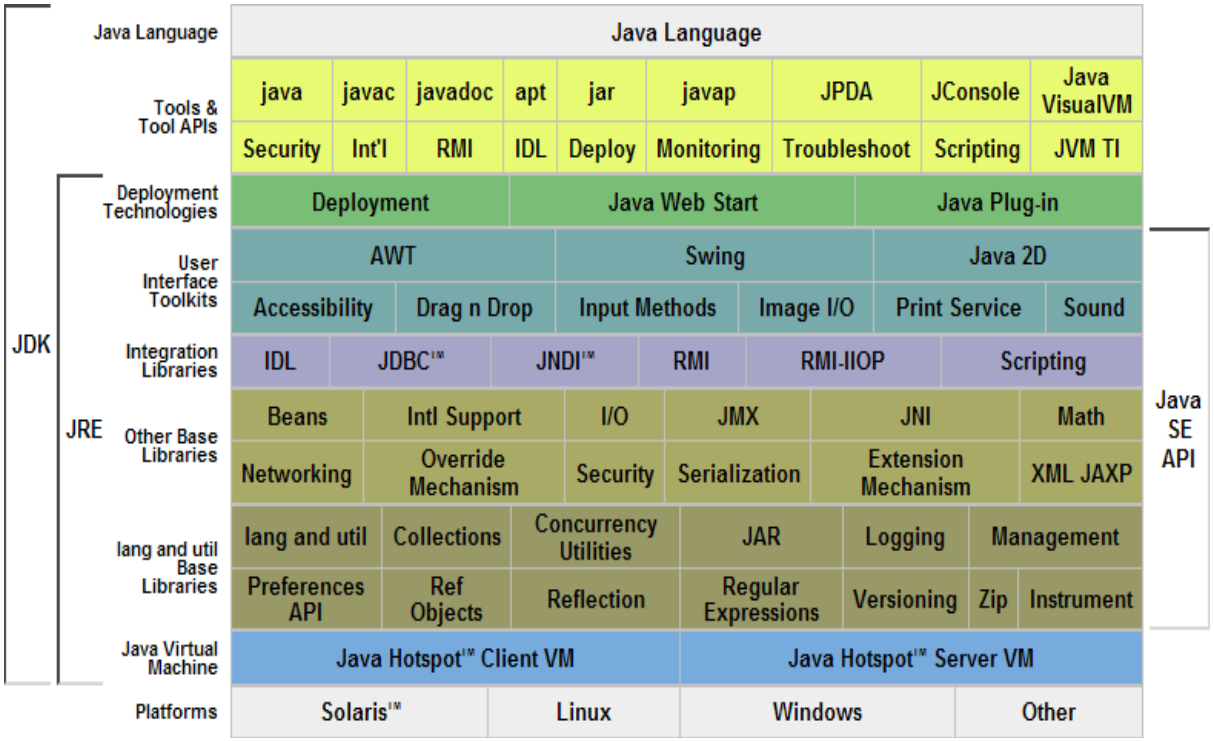
Deployment
PD

## Jdk vs JRE

# What is Java?

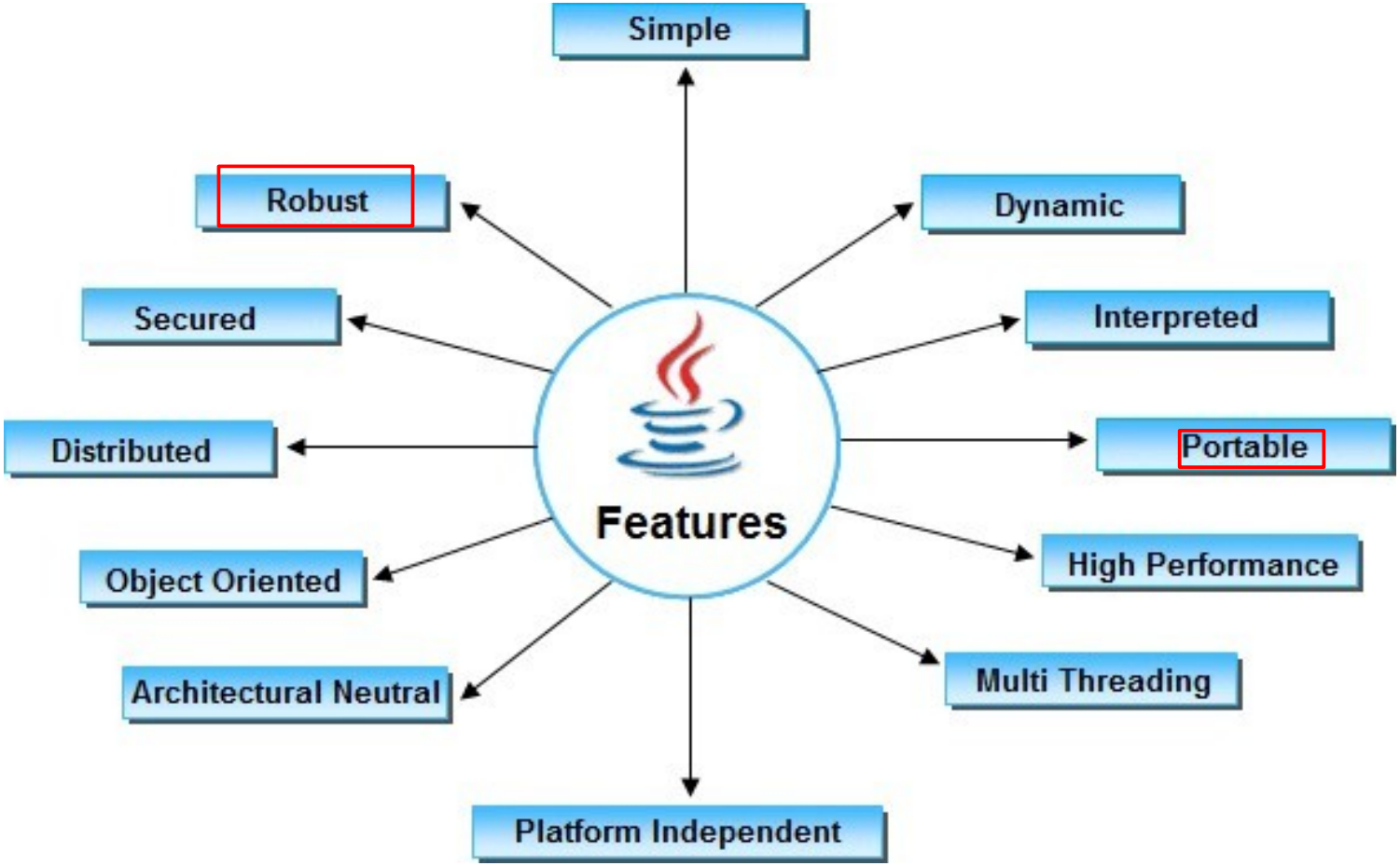
Java=OOPL+JVM+lib

How Java is different then C++?




rgupta.mtech@gmail.com

# How Sun define Java?




rgupta.mtech@gmail.com

# Versions of Java

| Java Version/Code Name | Release Date | Important Features/Code Name |
|---|---|---|
| **JDK Alpha and Beta** | 1995 | Initial release |
| **JDK1.0(OAK)** | 23rd Jab 1996 | Initial release |
| **JDK1.1** | 19th Feb 1997 | Reflection ,JDBC, Inner Classes, MI |
| **J2SE1.2(Playground)** | 8th Dec 1998 | Collection,JIT,String memory map |
| **J2SE 1.3(Kestrel)** | 8th May 2000 | Java Sound,Java Indexing,JNDI |
| **J2SE 1.4(Merlin)** | 6th Feb 2002 | Assert,regex,exception chaining |
| **J2SE 5.0(Tiger)** | 30th Sept 2004 | Generic,autoboxing,enums,varargs,for each,static import |
| **Java SE 6.0(Mustang)** | 11th Dec 2006 | JDBC4.0,Java compiler API Annotations |
| **Java SE7.0(Dolphin)** | 28th July 2011 | String in switch case, resource management in exception,catching multiple exception |
| **Java SE8.0** | 18th March 2014 | Lambad expression,Annotation on Java type,Data and Time API |

rgupta.mtech@gmail.com

# Versions of Java

| Version | Release date | End of Public Updates[5] | Extended Support Until |
|---|---|---|---|
| JDK Beta | 1995 | ? | ? |
| JDK 1.0 | January 1996 | ? | ? |
| JDK 1.1 | February 1997 | ? | ? |
| J2SE 1.2 | December 1998 | ? | ? |
| J2SE 1.3 | May 2000 | ? | ? |
| J2SE 1.4 | February 2002 | October 2008 | February 2013 |
| J2SE 5.0 | September 2004 | November 2009 | April 2015 |
| Java SE 6 | December 2006 | April 2013 | December 2018 |
| Java SE 7 | July 2011 | April 2015 | July 2022 |
| Java SE 8 (LTS) | March 2014 | January 2019 (commercial)<br>December 2020 (non-commercial) | March 2025 |
| Java SE 9 | September 2017 | March 2018 | N/A |
| **Java SE 10 (18.3)** | March 2018 | September 2018 | N/A |
| Java SE 11 (18.9 LTS) | September 2018 | March 2019 from Oracle<br>Later from OpenJDK | Vendor specific |
| Java SE 12 (19.3) | March 2019 | September 2019 | N/A |

**Legend:** Old version | Older version, still supported | **Latest version** | Future release

**Adv of java vs c++**

---

**easy and preditable !**
**C/C++**

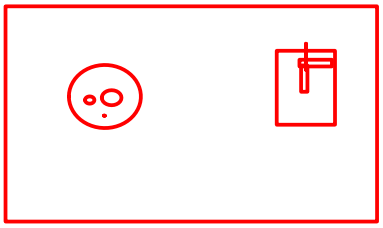
**int p[]=new int[5];**

**p[6]; AIOEx**

**C/C+**
**memory mgt is the respo**
**of dev**

**Emp *p =new ....()**

**~Emp(){**
**}**
**=> memory leak?**



**The changes of memory leak are less in java**
**as it is done by one of the comp of jvm ie called ... GC(Garbage collection)**

---

**How it works?**

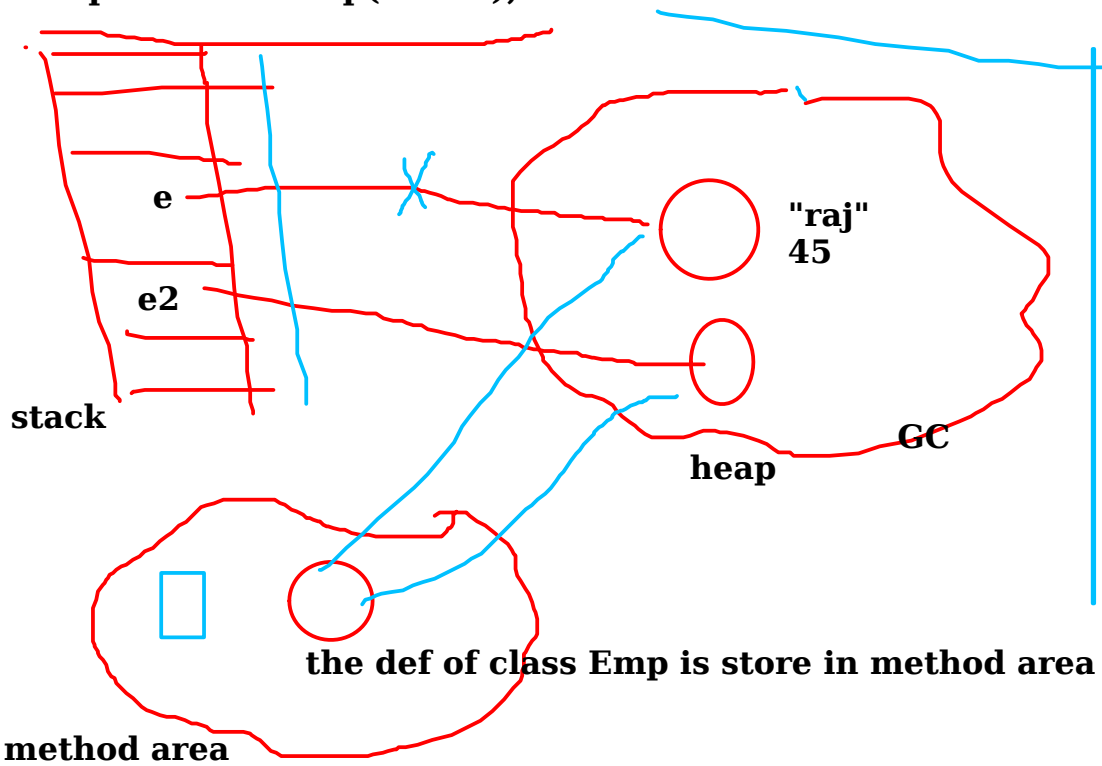
**Emp e =new Emp("raj");**

**object handler**
**ref**

**class : is a tempate**
**to create the object**

**cookie cuttor**

**Emp e2 =new Emp("ekta");**

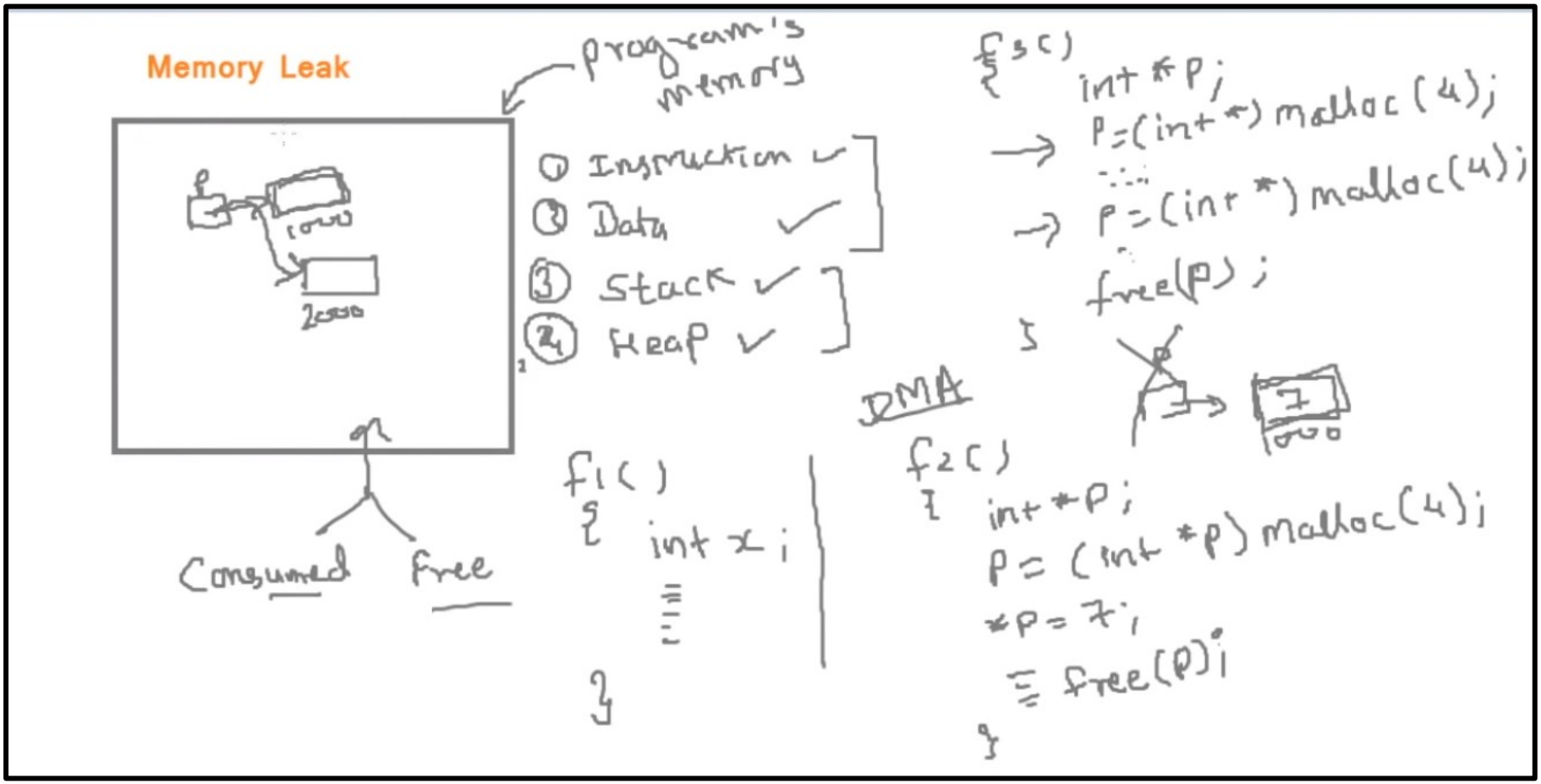



**e=null;**

**then GC will**
**remove the**
**object**
**sooner or**
**latter**

**Memory leak is still possible in java inspite of memory mgt**
**is job of GC**
**EX? on day2-day3**

# Memory management issuces in C/C++



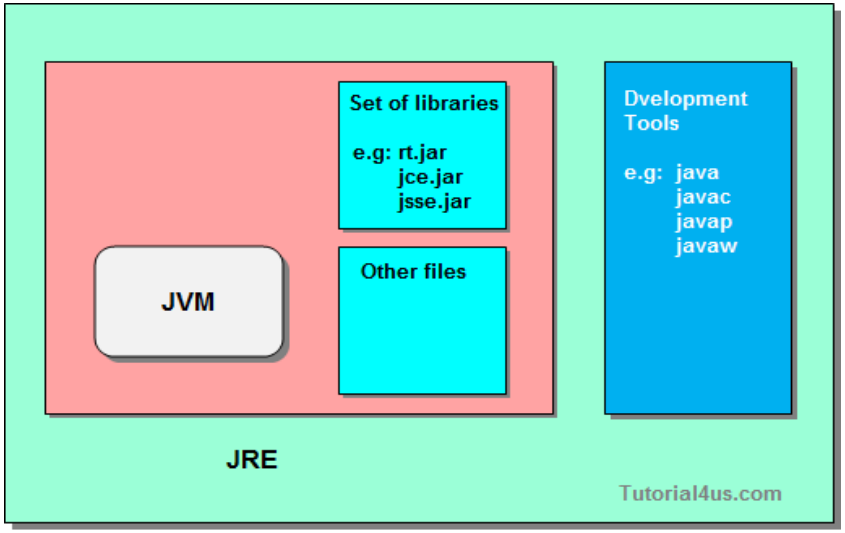
# Some common buzzwords

**JDK**

- Java development kit, developer need it
- Byte Code
- Highly optimize instruction set, designed to run on JVM  Platform independent
- Java Runtime environment, Create an instance JVM, must be there on  deployment machine.
- Platform dependent

**JVM**

- Java Virtual Machine JVM
- Pick byte code and execute on client machine  Platform dependent…




JDK

rgupta.mtech@gmail.com

JDK: dev need it

jre: deployer need it (server)

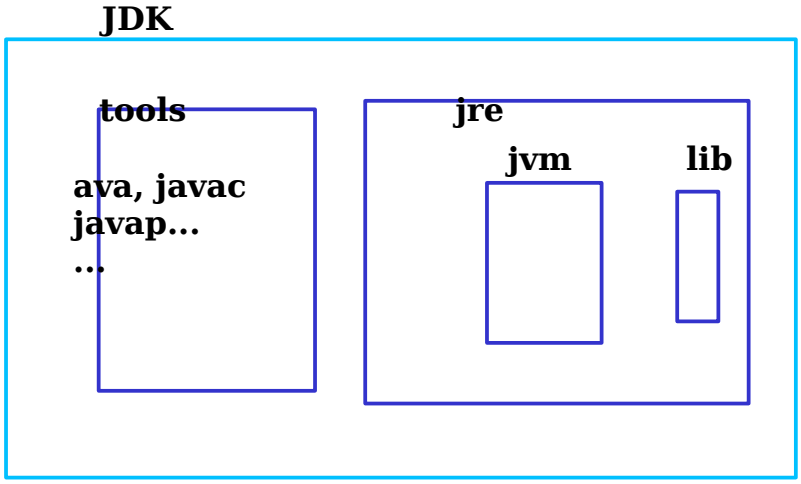
jdk = tools + jre

sw tools
java, javac
javap...

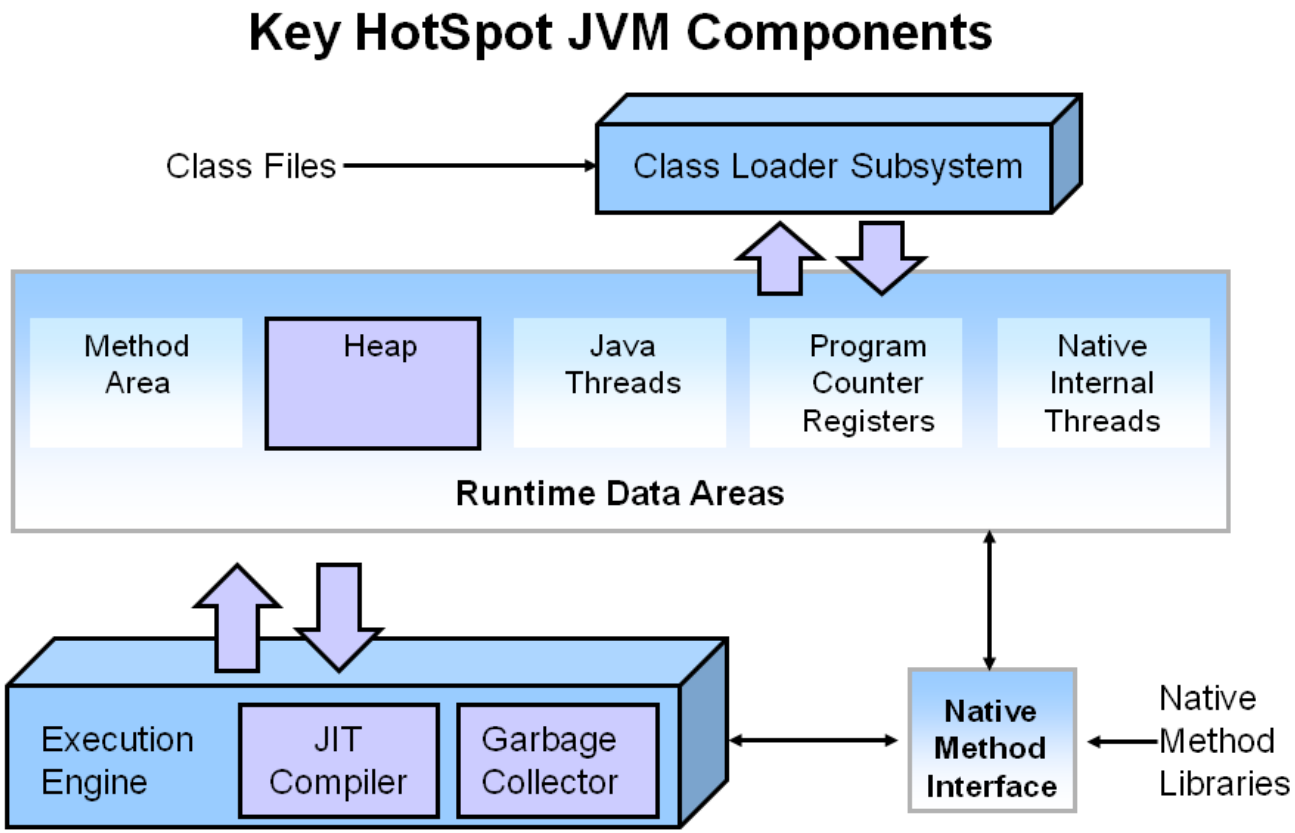
when you
run the command

java Hello

jre ---> give rise to JVM + lib



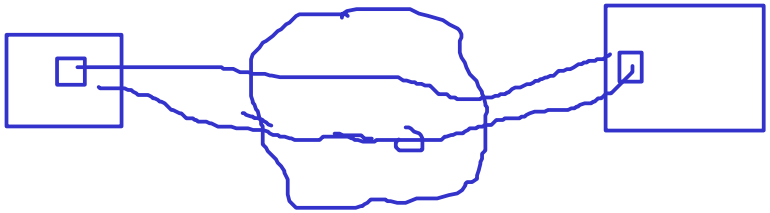
# Understanding JVM




rgupta.mtech@gmail.com

# Hello World

Applet: netspace neg + java!



## Lab set-up

- Java 1.8
- Eclipse ~~Luna~~

```
public class HelloWorld {

    public static void main(String args[])
    {
        System.out.println("welcome to the world of Java !!!");
    }
}
```

rgupta.mtech@gmail.com

**sts : aka eclispe ide + spring ide support**

**how to start it**

1. **install jdk**
2. **install maven**
3. **install sts**

# Analysis of Hello World !!!

keyword

visibility modifier

Name of the program

declare by compiler, define by user

```
public class HelloWorld {

    public static void main(String args[])
    {
        System.out.println("welcome to the world of Java !!!");
    }
}
```

need to be static

command line arguments

important class in java

predefine object of PrintWriter

function define inside this class..

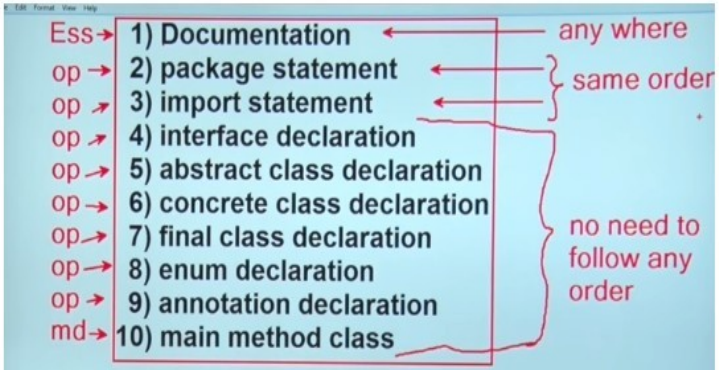
rgupta.mtech@gmail.com

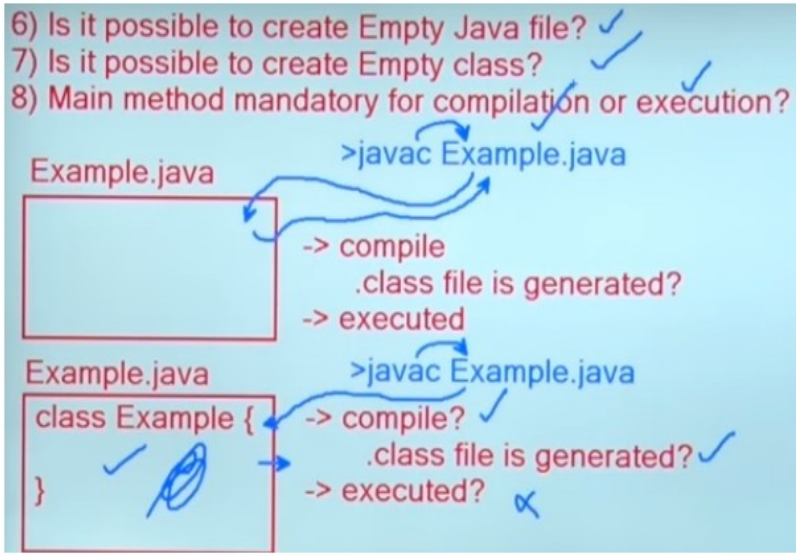
# Some basics rules about java classes

1) Java Source file structure

2) Order of placing package and import statement
3) Order of placing interface, class, enum, annotation, doc comment

4) How many package and import statements are allowed in a single java file?
5) Why only one package statement is allowed in and why multiple import statements are allowed?



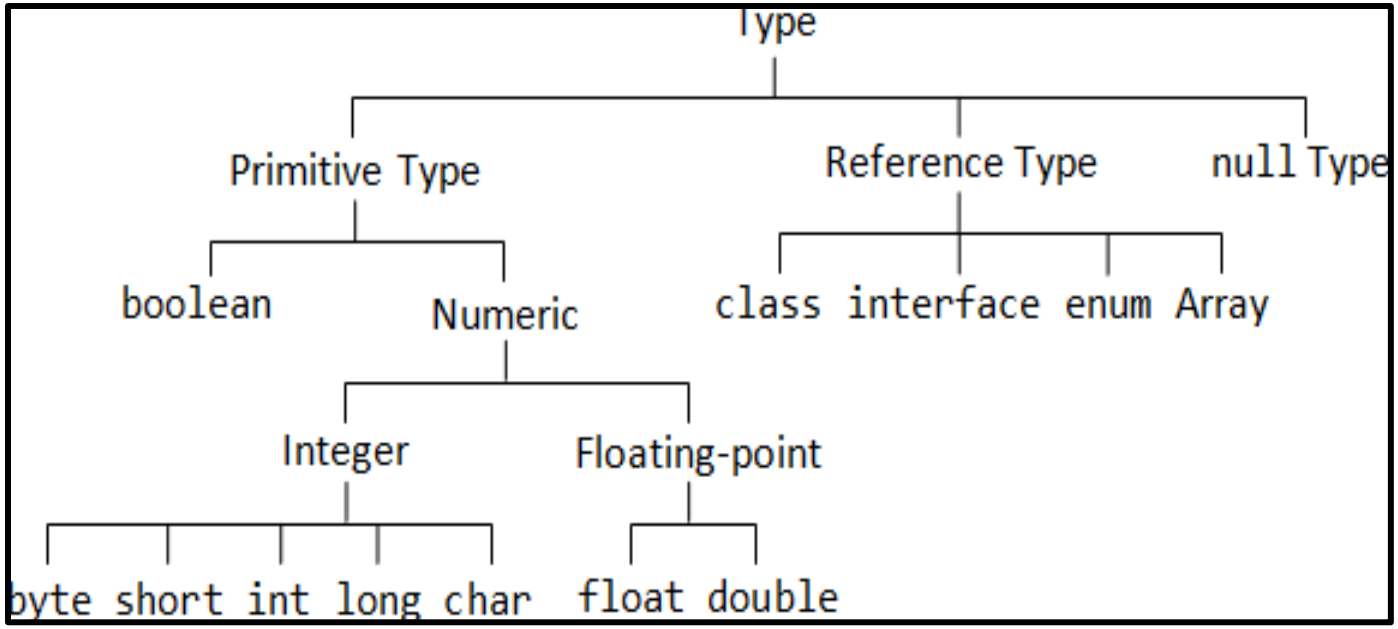




rgupta.mtech@gmail.com

# Java Data Type

## 2 type

- Primitive data type
- Reference data type




rgupta.mtech@gmail.com

# Primitive data type

| | |
|---|---|
| boolean | either true of false |
| char | 16 bit Unicode 1.1 |
| byte | 8-bit integer (signed) |
| short | 16-bit integer (signed) |
| int | 32-bit integer (signed) |
| Long | 64-bit integer (singed) |
| float | 32-bit floating point (IEEE 754-1985) |
| double | 64-bit floating point (IEEE 754-1985) |

**Java uses Unicode to represent characters internally**

rgupta.mtech@gmail.com

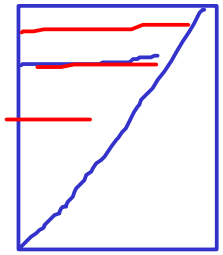
# Procedural Programming

- if..else
- switch
- Looping; for, while, do..while as usual in Java as in C/C++

**Dry run : running the code inside the brain**

- Don't mug the program/logic
- Follow dry run approach
  - Try some programms:
    - Create
    - Factorial program
    - Prime No check
    - Date calculation

```
*
**
***
****
*****
```



rgupta.mtech@gmail.com

**prime no:**
**proof by contradition?**

**logic :**
**input the no from the user**
**2. let assume no is prime no**
**isPrime=true**

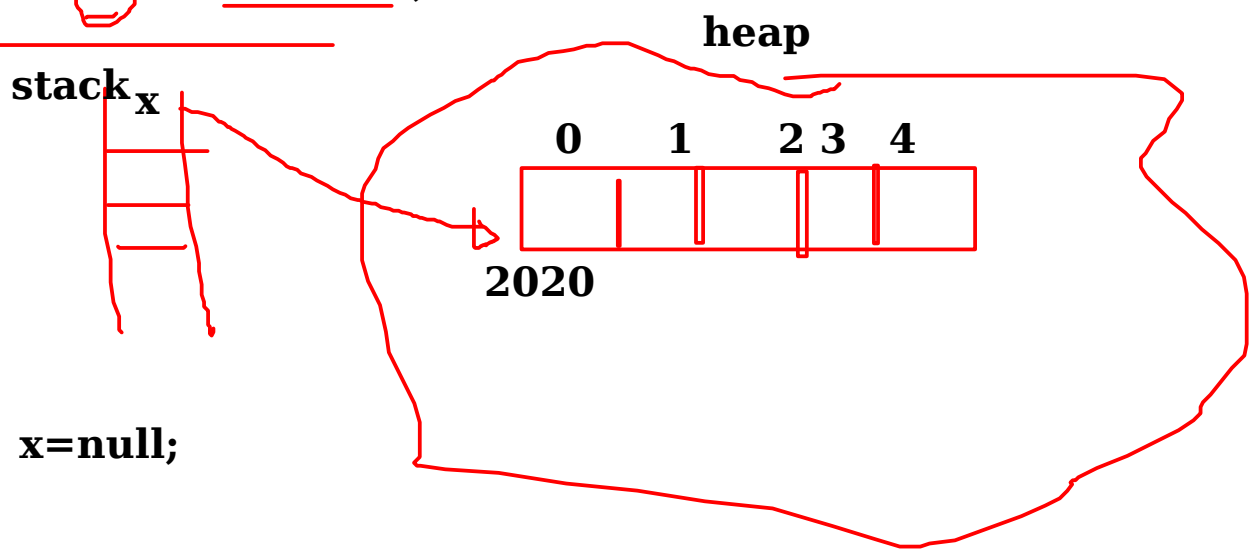
**3. 2-- (n-1) will divide the no**
**if it divided then our assumtion was wrong**
**isPrime=false**

---

**Array in java:**

***array are objects in java***

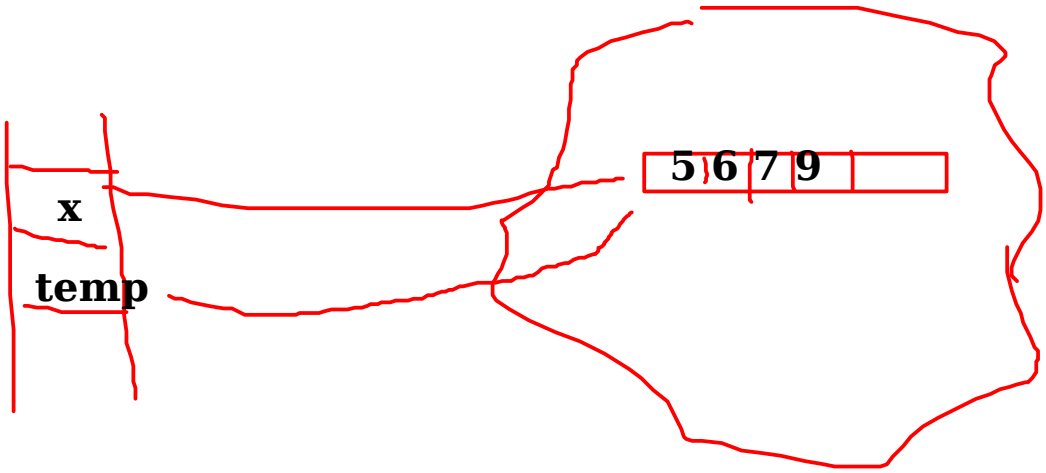
**int x[]=new int[5];**

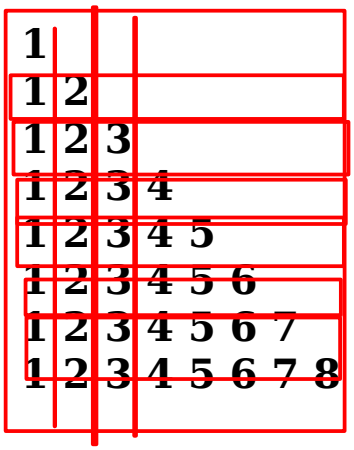


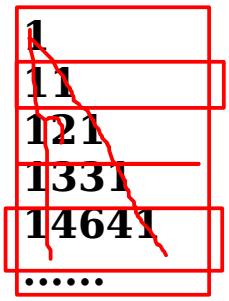

**x=null;**

**why ref is safer then pointer?**
**c++: ref vs pointer**
**safe pointer**

```
public class DemoCopyOfArray {

        public static void main(String[] args) {

                int x[]= {5,6,7,9};
                int temp[]=copyOfArray(x);
        }

        private static int[] copyOfArray(int[] x) {
                return x;
        }
}
```



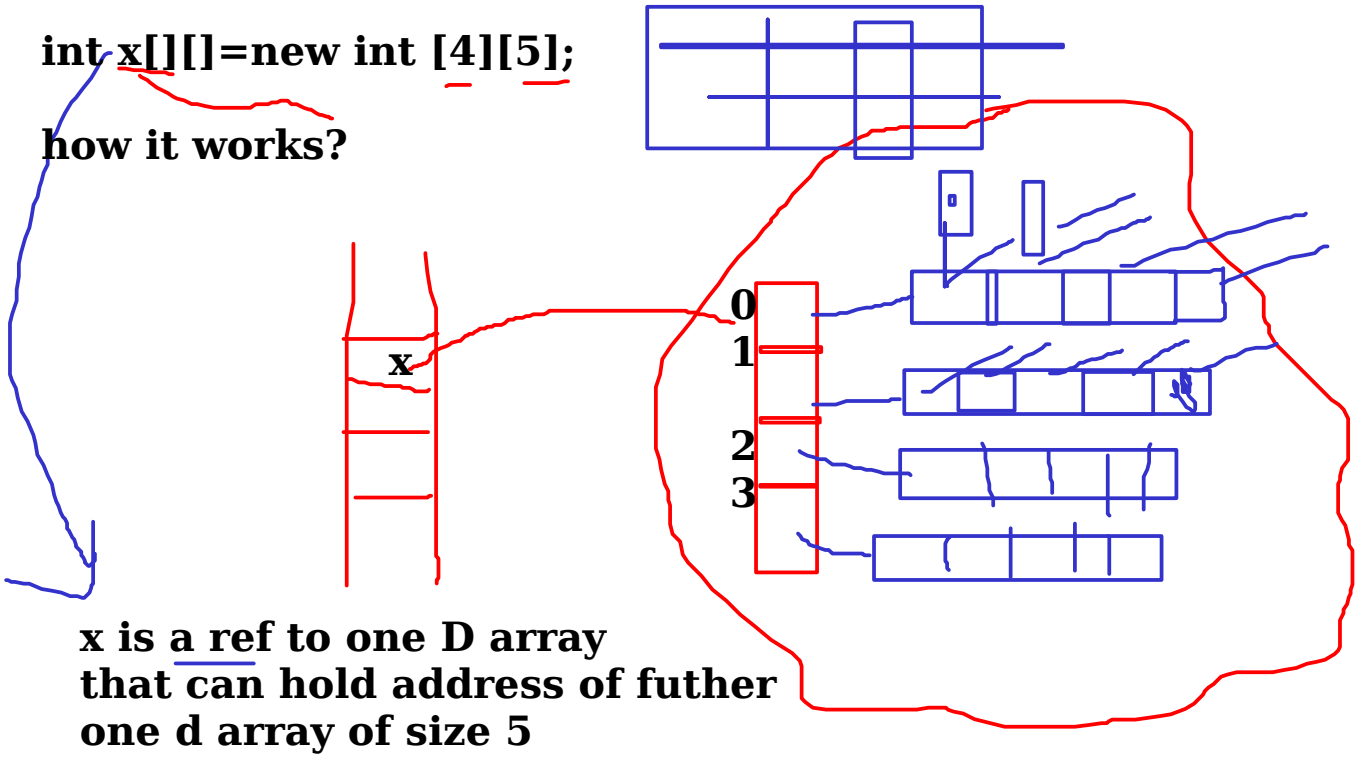




2D array

2 D array
array of objects?

int x[][]=new int [4][5];

how it works?



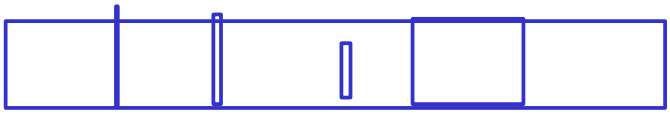

x is a ref to one D array
that can hold address of futher
one d array of size 5

int x[][][]=new int [4][5][6];

1. sorting of array
2. arr
-3 -5 4 9 8 20 -51 -4

3. add / remove the element from an arry

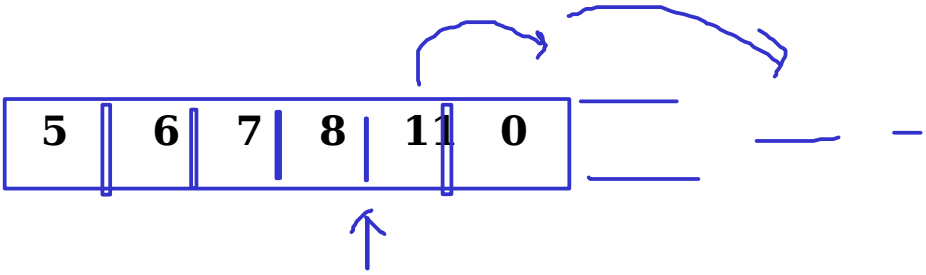
int x[]={5,6,7,8,11,0};
5,6,7,8,10,11 (shifting ...)



primitive programming

Object oriented programming!
collection api

---



**More about array:**
**array of primitives**

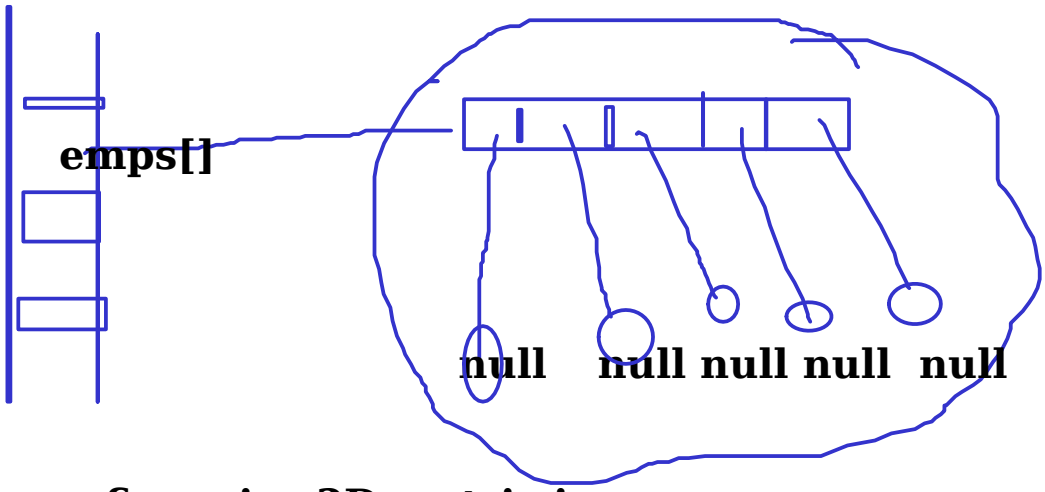
**int x[]=new int[5];**

**Array of objects?**

**Employee**

**e1 e2 e2...**

**Employee [] emps= new Employee[5];**




**wap to verfiy a give 2D matrix is**

**identity matrix (I)**

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

**today date is given**

**20 .7.21**
**21.7.21**

**31.12.2021**
**1**
**1.1.2022**

**28.2.2021**
**1**

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

**//leap year**

**//check the month**

# Procedural Programming

- if..else
- switch
- Looping; for, while, do..while as usual in Java as in C/C++

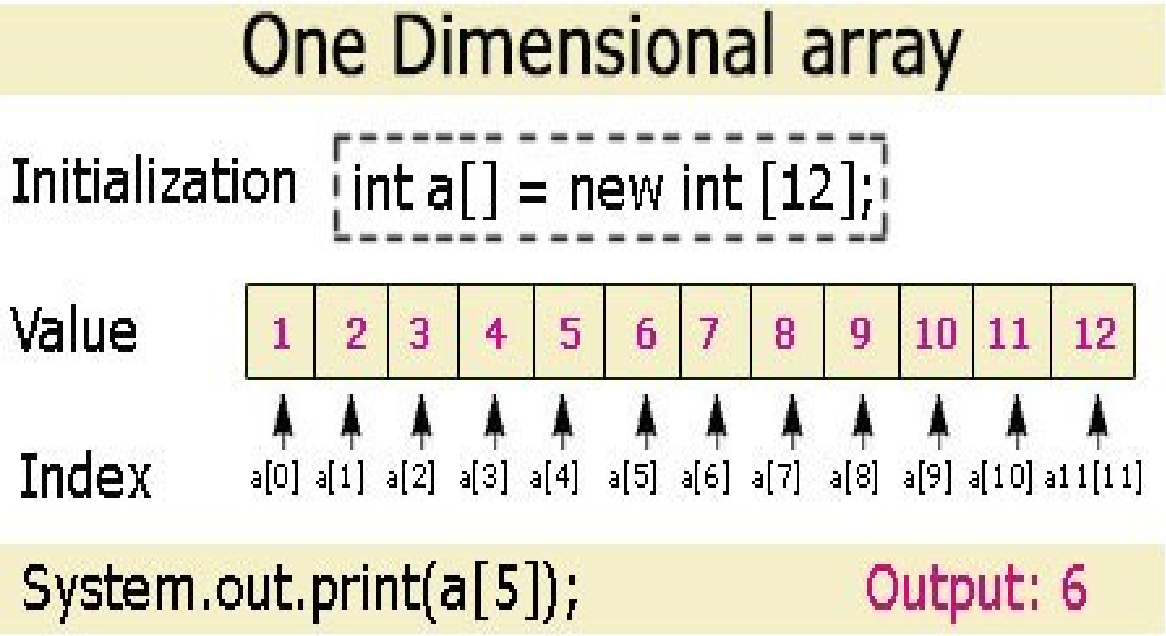
- Don’t mug the program/logic
- Follow dry run approach
  - Try some programms:
    - Create
    - Factorial program
    - Prime No check
    - Date calculation

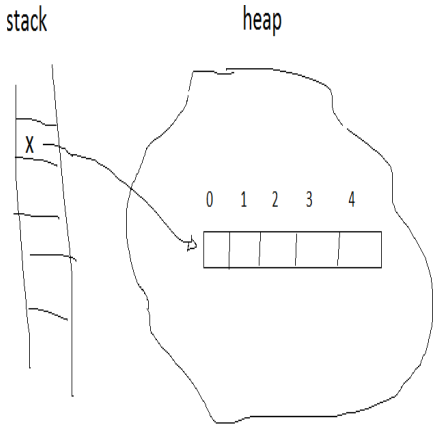
```
*
**
***
****
*****
```

rgupta.mtech@gmail.com

# Array are object in java

- How Java array different frrom C/C++ array?




Java Array: its different then C/C++

rgupta.mtech@gmail.com

# Object technologies

- OO is a way of looking at a software system as a collection of interactive objects

Reality

Tom's House

Tom

Tom's Car

Model


rgupta.mtech@gmail.com

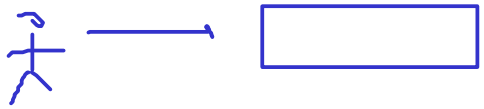
**Object Orientation?**

**is nothing but the system of object in intraction with each other**

**3 kind of relationship**

**USE A: LIGHTEST RELATION**

**m1** **M2**



**Passanger is using metro to travel from PV to Noida**

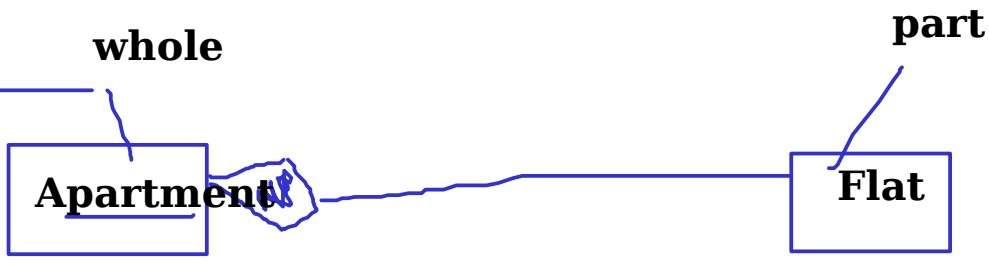
**I am using marking while teaching**

```
class Metro{

}

class Passanger {
        public void travel(Metro metro){
                metro.move();
}
}



class Trainer{
        public void teach(Marker marker){
        }

        public void eat(Food food){
        }

}

class Marker{
}
```

**use a**

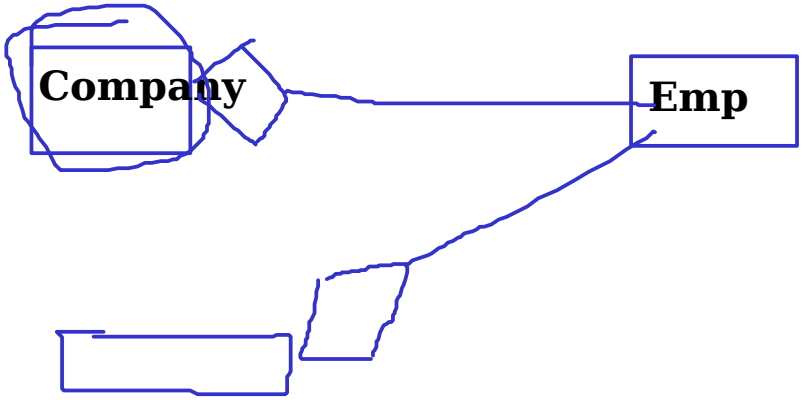
**HAS- A**
**(Association)**

**compostion :**




1. strong relationship
2. whole and part, if whole is destroyed.. part is also destroyed...

**aggigation**




**Inheritnace:**

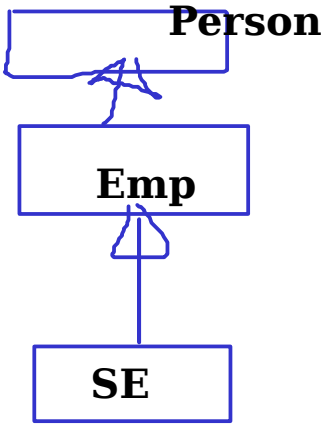
prime purpose of inhertiance is for code resuablity

code resuablity can be achive with inhertiance using compostion ...

then what is the prime purpose of inheritance?

1. extandabiilty
2. subtituablity

polymp




```
class A{
        ...
        ..
}
class B extends A{

}
```

"When to go for inheritance and when to go for compostion"

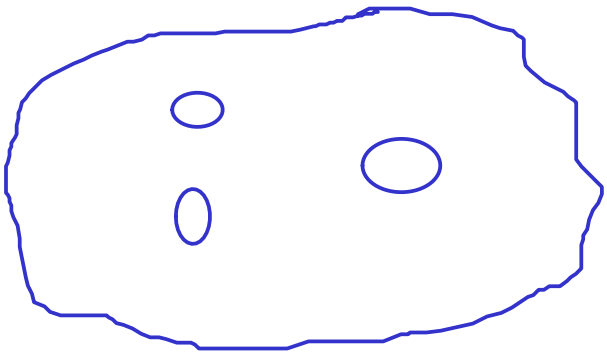
```
class RupaDairay{
        public void sellMilk(){
                give simple mike to user
        }
}
```

```
lass RupaDairayImproved extends RupaDairly{
        @Override
        public void sellMilk(){
                give p.. milk to the use
        }
}
```

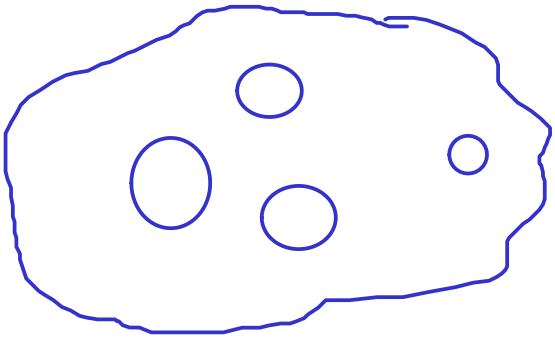
```
RupaDairy rd=new RupaDairyImproved();
rd.sellMilk();
```

**Object and class:**

**class: cookie cutter, template, category...**



**metadata**



**Object : it is run time ex of a class**

**instance**

**object :**
**identity**

**state**

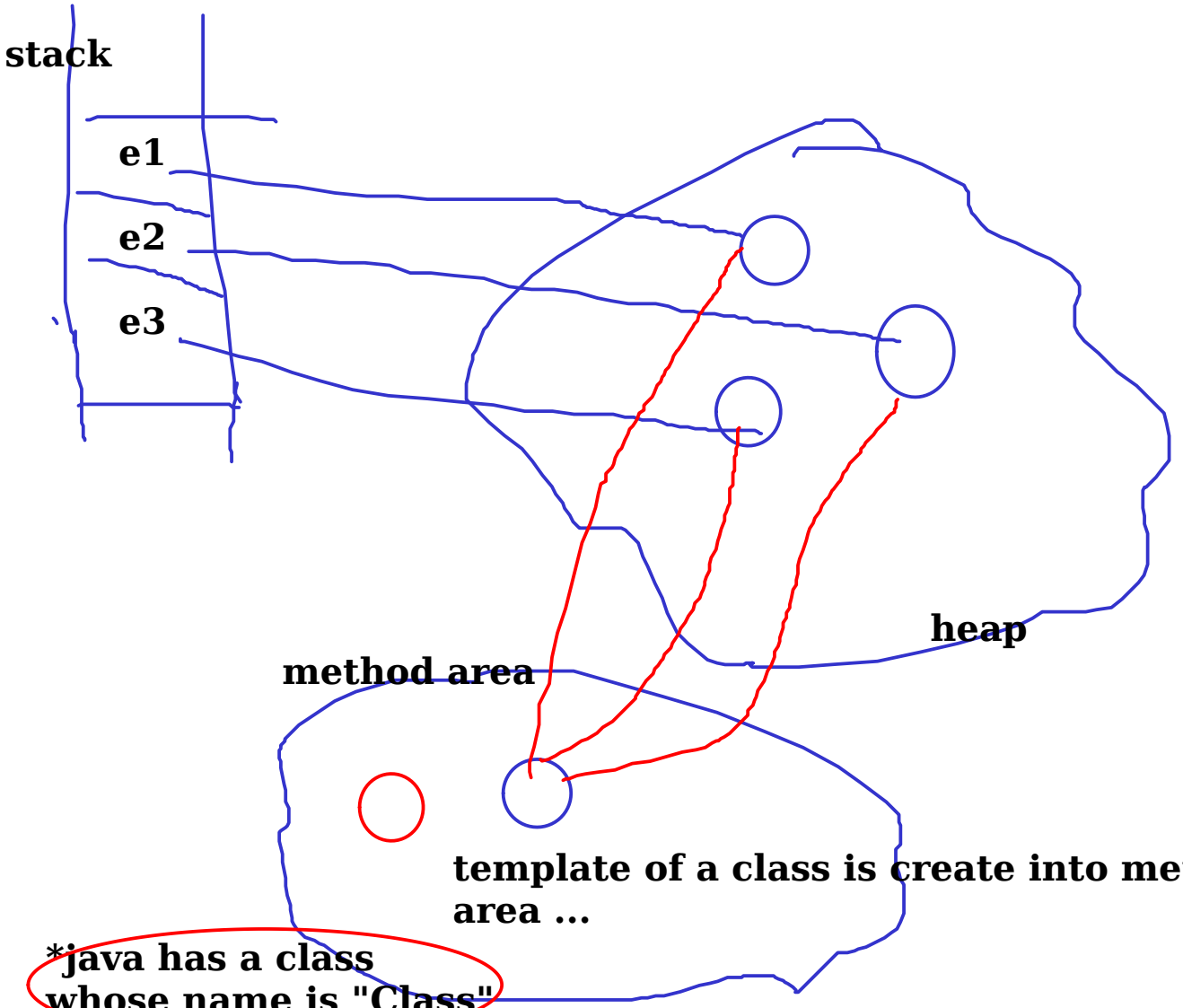
**behavour**

```
class Emp{
        id
        name
        salary

        }
```

**Emp e1=**
**new Emp(1, "raj",45);**

**Emp e1=**
**new Emp(1, "raj",45**

**Emp e1=**
**new Emp(1, "raj",45);**




**template of a class is create into method area ...**

***java has a class**
**whose name is "Class"**

# What is an Object?

- Informally, an object represents an entity, either physical, conceptual, or software.
  - Physical entity

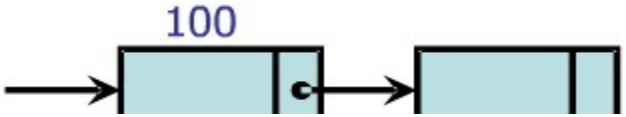
Tom's Car

  - Conceptual entity

US$100,000,000,000

Bill Gate's bank account

  - Software entity




rgupta.mtech@gmail.com

# Class

- A class is the blueprint from which individual objects are created.
- An object is an instance of a class.

Object factory

Cookie Cutter

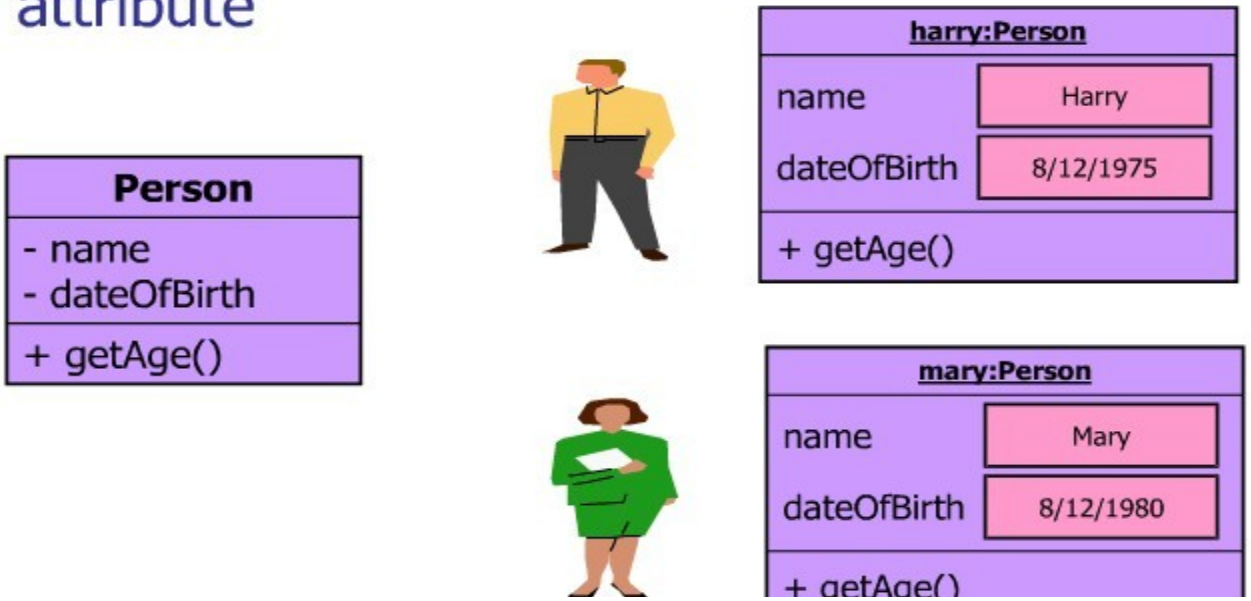
```
public class StudentTest {
  public static void main(String[] args) {
    Student s1 = new Student();
    Student s2 = new Student();
  }
}
```

```
- class -

public class Student {
  private String name;
  // …
}
```

rgupta.mtech@gmail.com

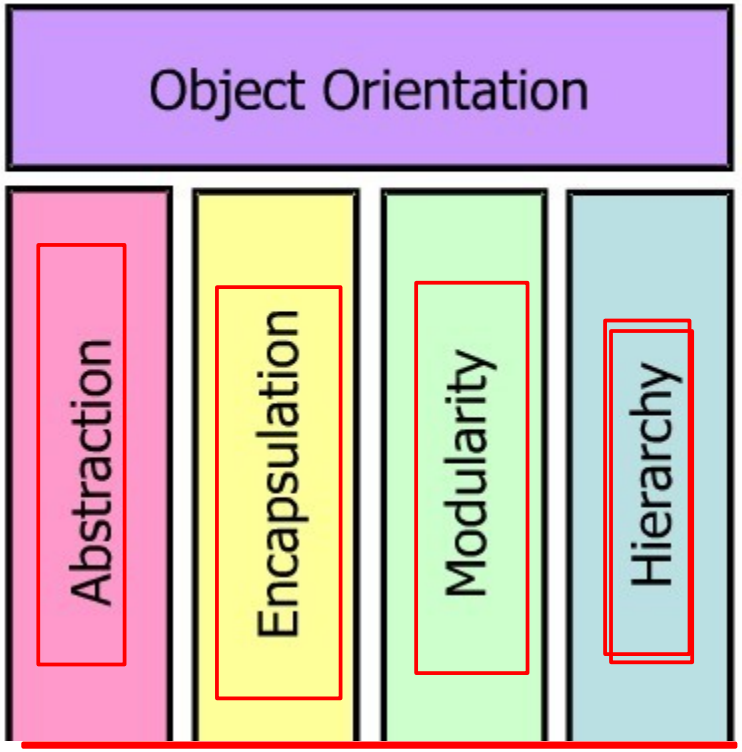
# Classes and Object

- All the objects share the same attribute names and methods with other objects of the same class
- Each object has its own value for each of the attribute



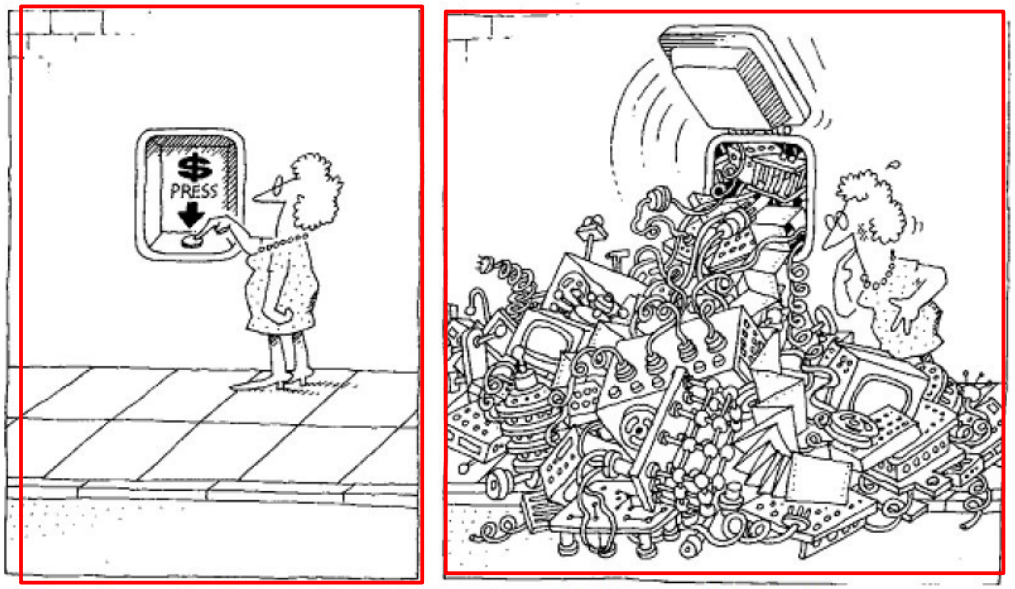

rgupta.mtech@gmail.com

# Basic principles of OO




rgupta.mtech@gmail.com

# Dealing with Software complicity:Abstraction




The task of the software development team is to engineer the illusion of simplicity.

Stack

```
AmitStack{

 public myPop(){

 }
public void myPush(int data){

}


}
```

```
RaviStack{

 public pop1(){

 }
public void push1(int data){

}


}
```

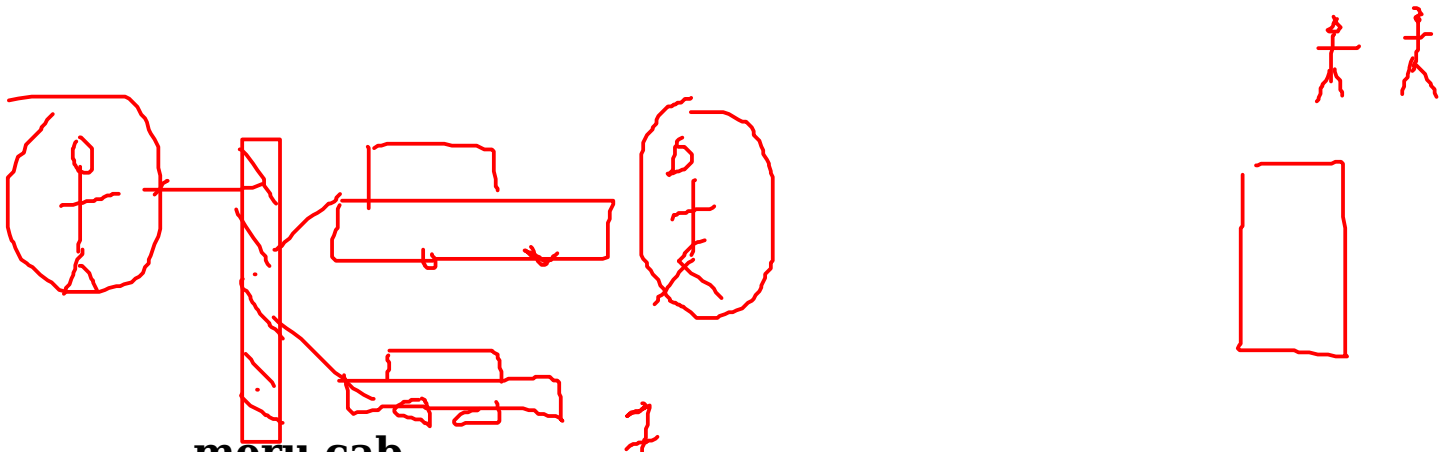
Java: we can achive abstraction
using inteerface/ abstraction class

```
interface Stack{
 public void push(int x);
 public int pop();
}
```

```
Stack s=new AmitStack();
                RaviStack();
```

"abstraction provide loose coupling and high cohesion"

**Example how : abstaction help in real life?**



**meru cab**

**noun analysis**

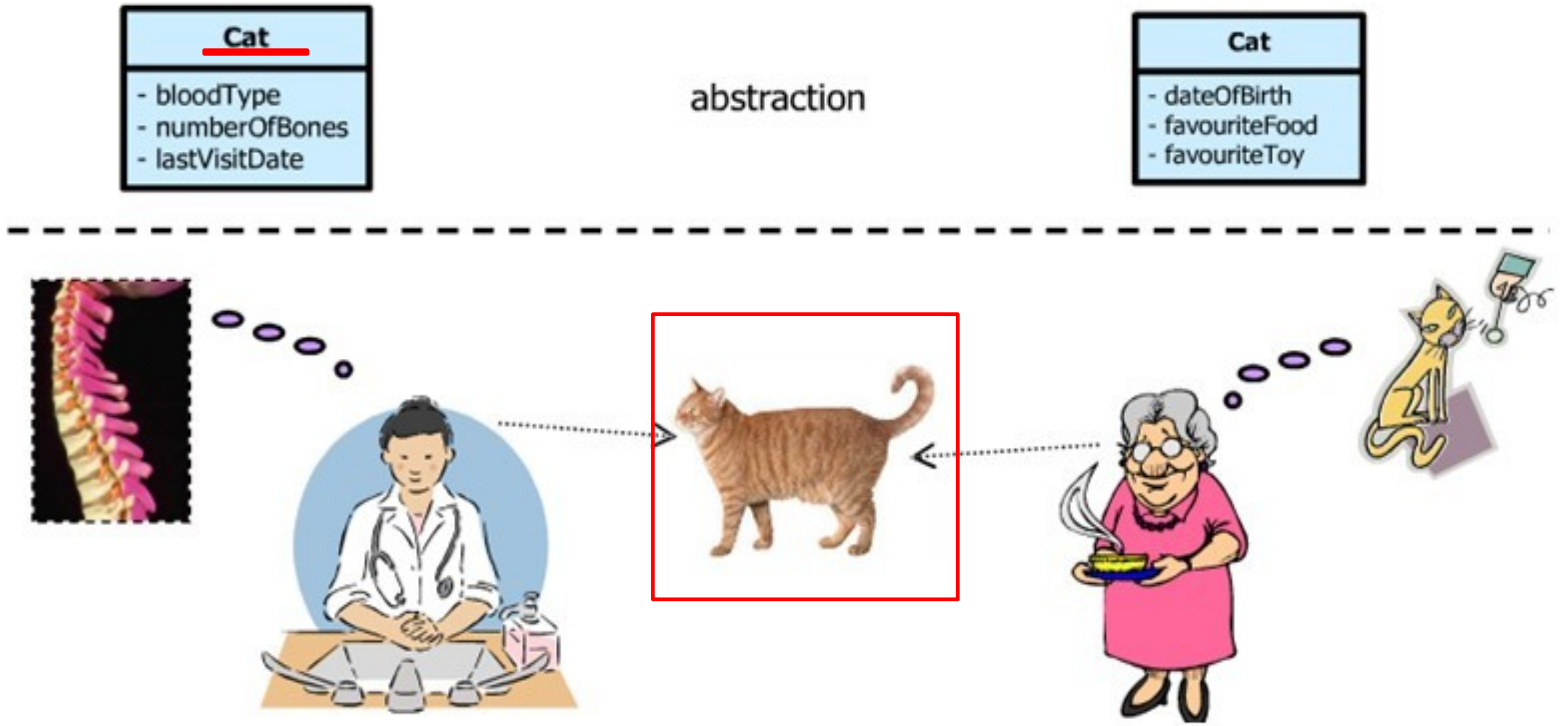
**a passanger is using metro to travel from PV to Noida**

# Abstraction

- Fundamental ways that we use to cope with complexity "abstraction arises from a recognition of similarities between certain
- objects, situations, or processes in the real world, and the decision to
- concentrate upon these similarities and to ignore for the time being the differences" -Hoare
- An abstraction denotes the **essential characteristics of an object that distinguish it from all other kinds of objects** and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.

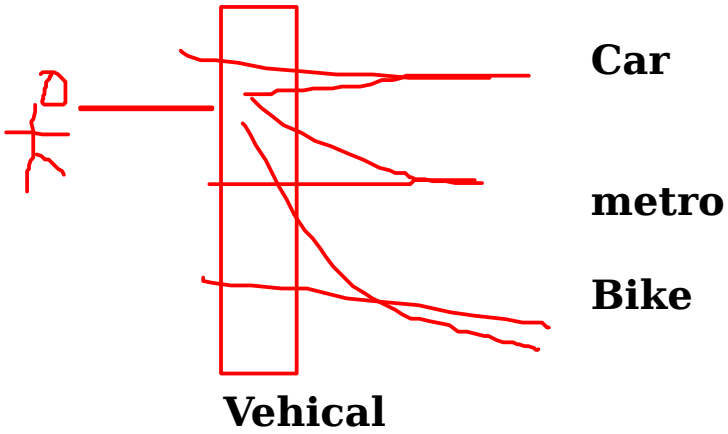
# Abstraction

- Determine the relevant properties and features while ignoring non-essential details

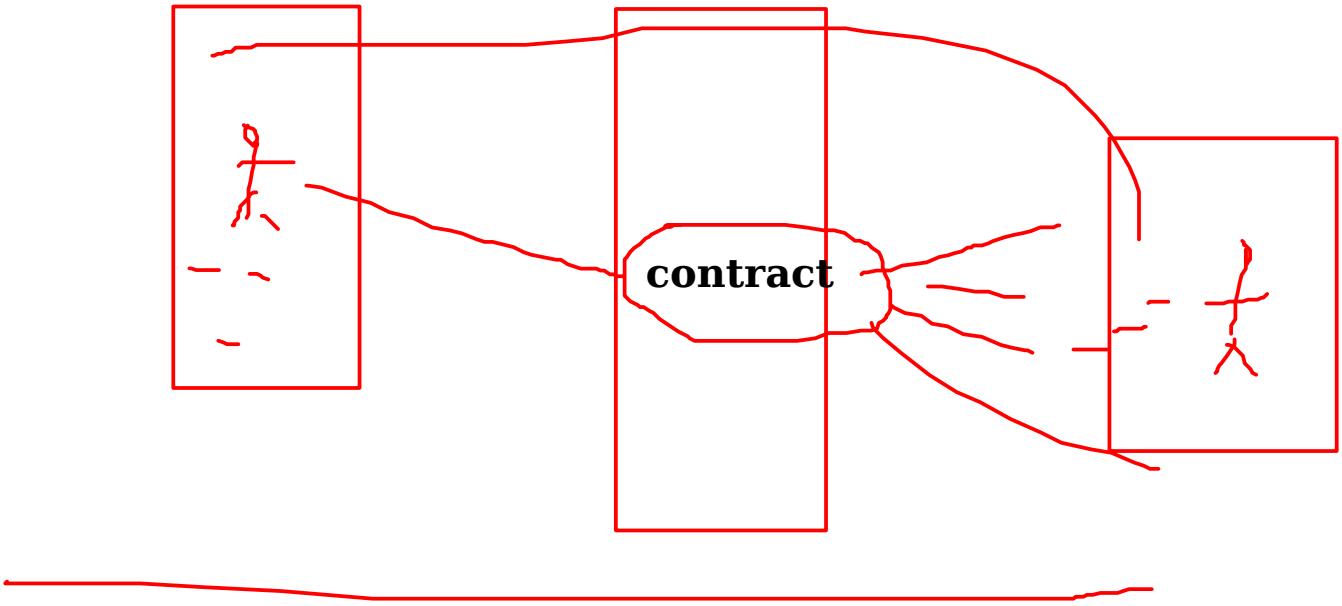



rgupta.mtech@gmail.com

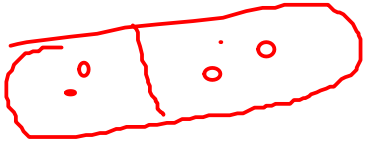
```
class Metro{
	public void move(String source, String destination) {
		System.out.println("moving in metro from "+ source + " " +
	}
}
class Car{
	public void move(String source, String destination) {
		System.out.println("moving in car from "+ source + " " +de
	}
}
class Passanger{
	private String name;

	public Passanger(String name) {
		this.name=name;
	}

	public void travel(String source, String destination, Car car) {
		System.out.println("passanger name is:" + name);
		//deligation of resp
		car.move(source, destination);
	}

}
public class TravelProblem {

	public static void main(String[] args) {
		//Metro metro=new Metro();
		Car car=new Car();
		Passanger passanger=new Passanger("raj");
		passanger.travel("PV", "Noida", car);

	}

}
```




**abstaction along with run time polymorphism provide loose coupling**



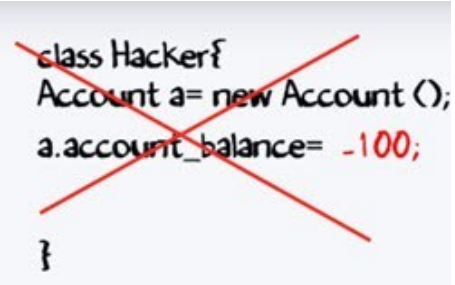
# Encapsulation

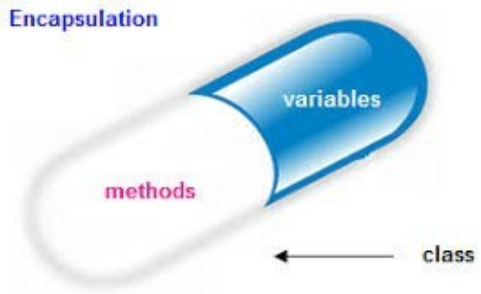


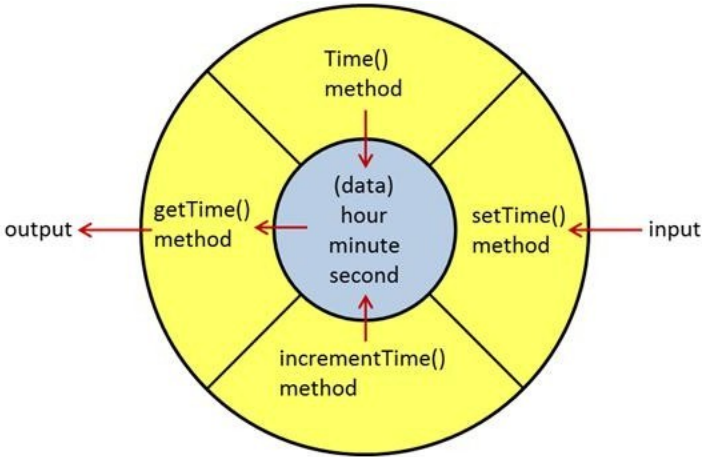
- Abstraction and encapsulation are *complementary concepts*: abstraction focuses upon the observable behavior of an object, whereas encapsulation focuses upon the implementation that gives rise to this behavior.
- Encapsulation is most often achieved through information hiding which is the process of hiding all the secrets of an object that do not contribute to its essentialcharacteristics; typically, the structure of an object is hidden, as well as the ,implementation of its methods.
- Information hiding is tool to achive encapsulation

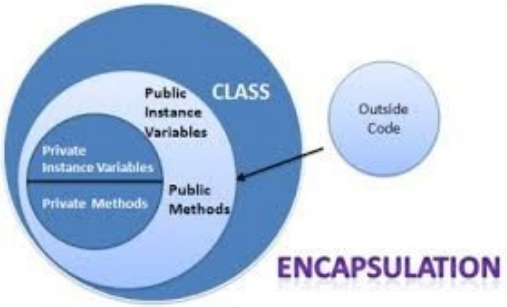
**Encapsuslation is achive by inforation hinding**

# Understanding encapsulation



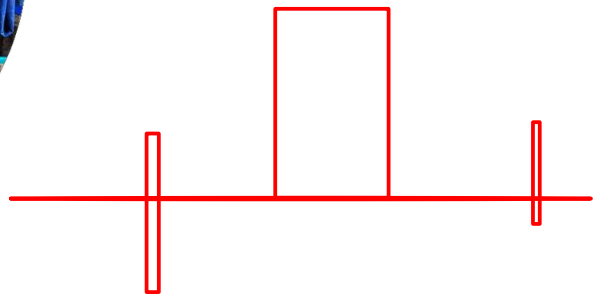



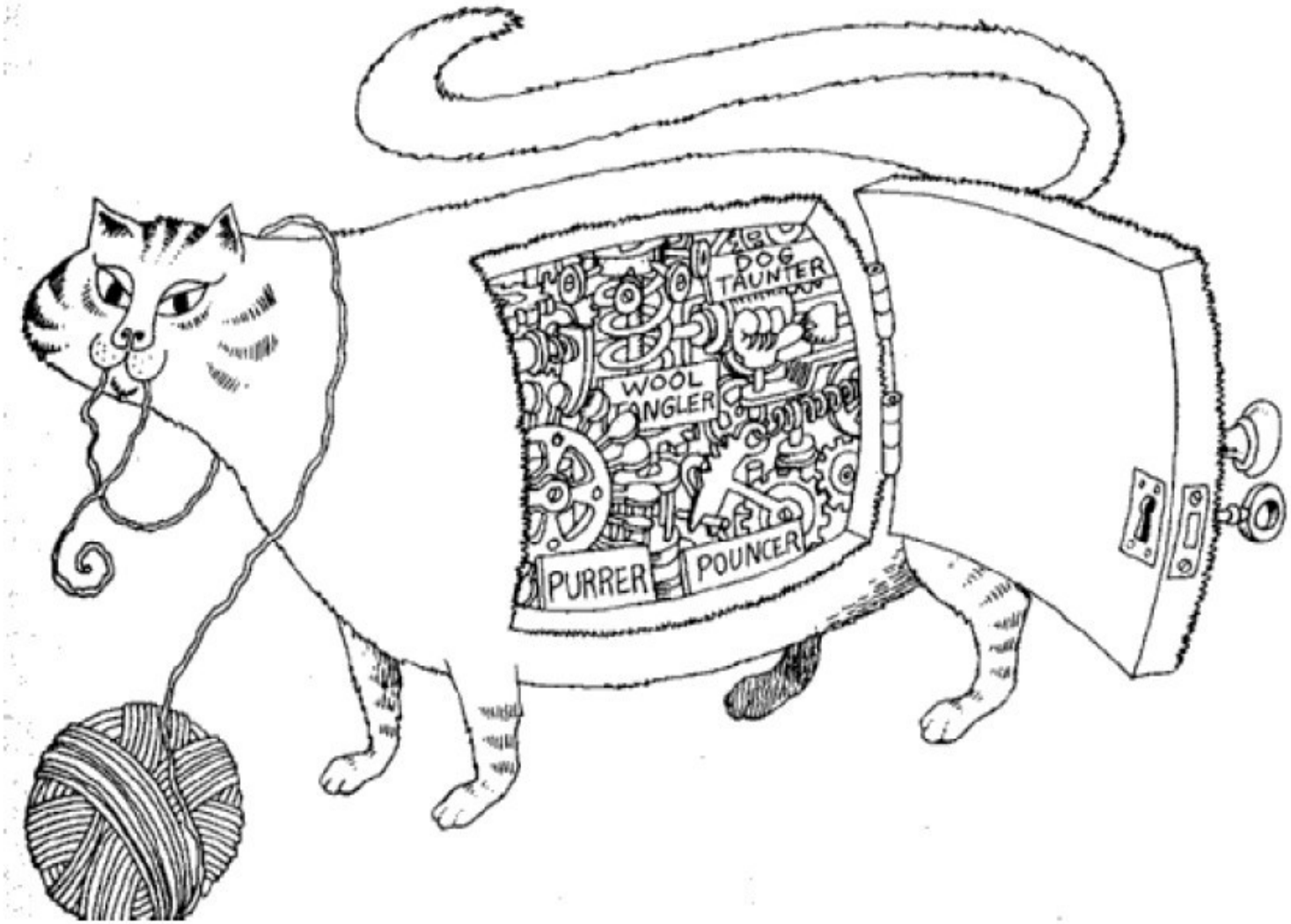
```
class Account{
	private int id;
	private String name;
	private double balance;
	private int creditScore;

	// ctr
	public Account(int id, String name, double balance, int creditSc

	}

	int getCreditScore() {
		return 8;
	}

	public void printAccount() {
		System.out.println(id);
		System.out.println(name);
		System.out.println(balance);
		System.out.println(creditScore);
	}
}
```

```
class Stud{
	private int id;
	private String name;

	public Stud(int id, String name) {
		id=id;
		name=name;
	}
	public void printStudentDetails() {
		System.out.println(id +" : "+ name);
	}
}
public class DemoThis {

	public static void main(String[] args) {
		Stud s=new Stud(1, "ravi");
		s.printStudentDetails();
	}
}
```

# Encapsulation

- Changing data in organized way, by using data hiding and appying business constraints
- Encapsulation= data hiding + constraints

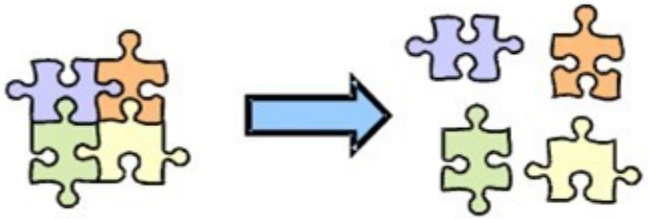


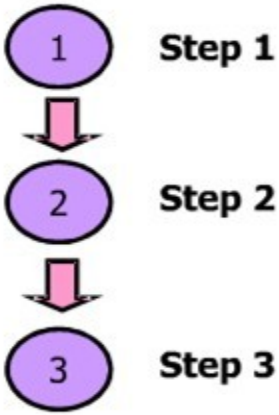



Encapsulation hides the details of the implementation of an object.

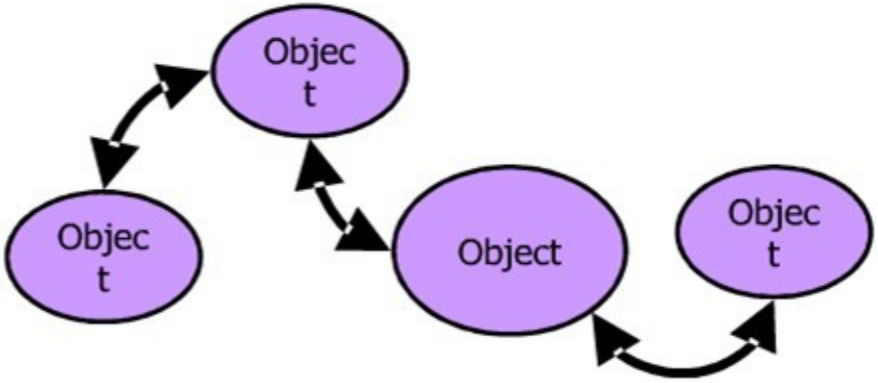
# Modularity

- Break something complex into manageable pieces
  - Functional Decomposition
  - Object Decomposition






Functional Decomposition

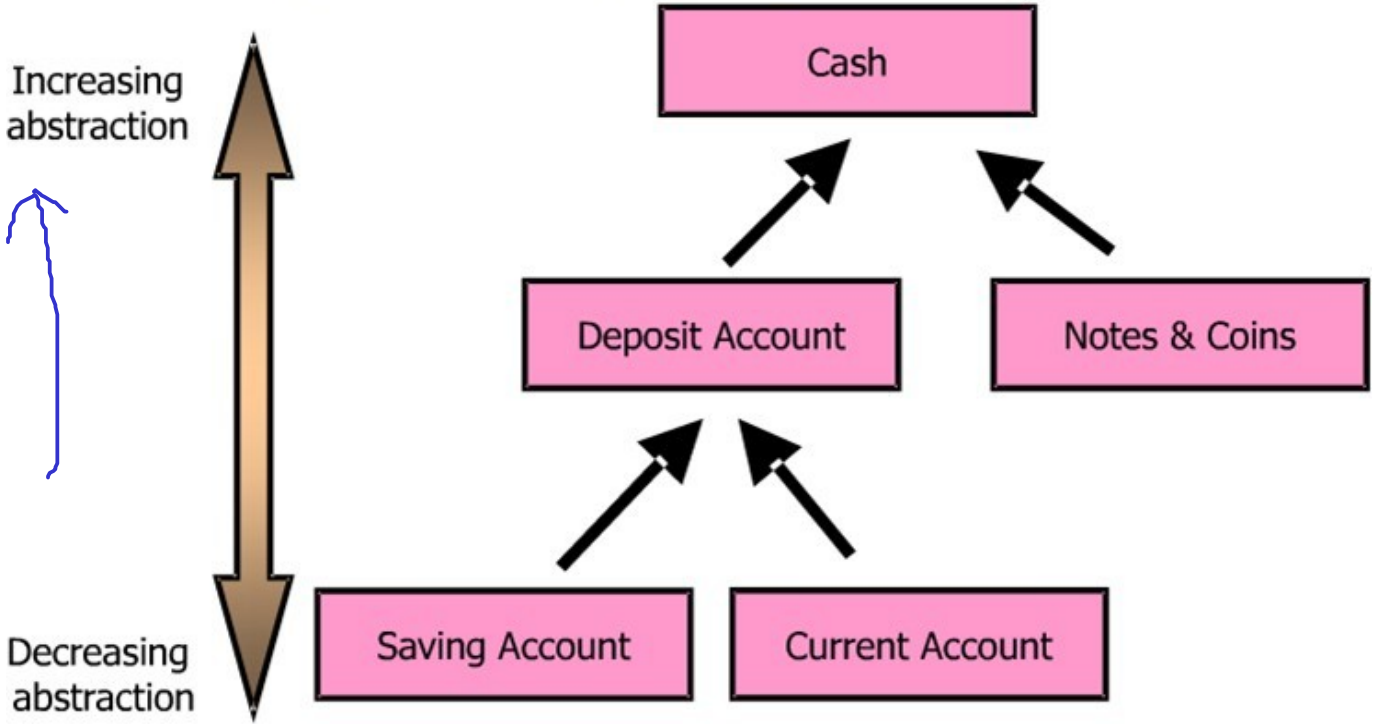



Object Decomposition

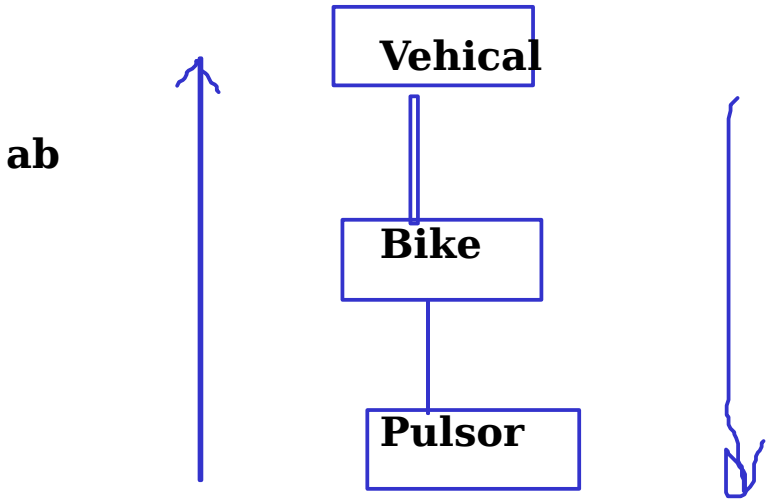
rgupta.mtech@gmail.com

# Hierarchy

- Ranking or ordering of objects




rgupta.mtech@gmail.com




**what is next?**

**we want to basics syntex?**