Core Java

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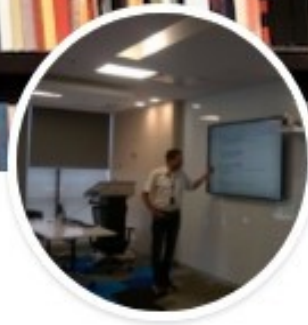
Rajeev Gupta

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
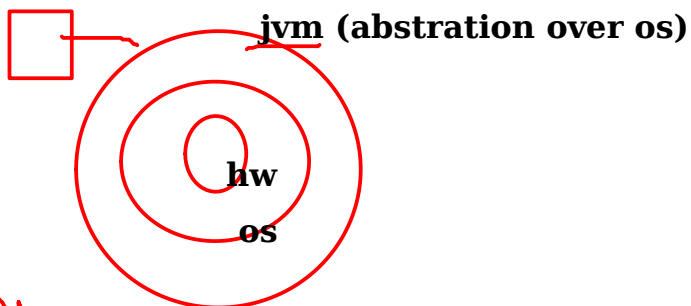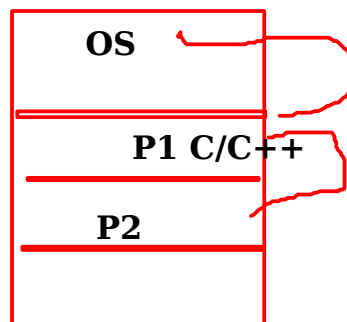- Lab Assignments

What is Java?

OOPL

Java vs C++

java = oopl + jvm + rich set of lib

c/C++

OS

HW
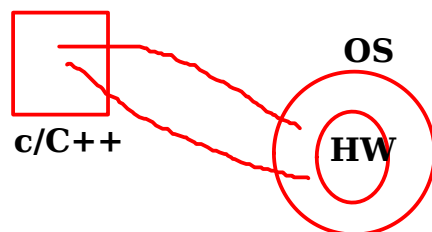
C/C++ are native hw pointer ?

jvm (abstration over os)

hw

os

OS
P1 C/C++
P2

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++?

Java Language	Java Language								
Tools & Tool APIs	java	javac	javadoc	apt	jar	javap	JPDA	JConsole	Java VisualVM
	Security	Int'l	RMI	IDL	Deploy	Monitoring	Troubleshoot	Scripting	JVM TI
Deployment Technologies	Deployment			Java Web Start			Java Plug-in		
User Interface Toolkits	AWT			Swing			Java 2D		
	Accessibility		Drag n Drop		Input Methods		Image I/O	Print Service	Sound
Integration Libraries	IDL	JDBC™		JNDI™		RMI	RMI-IIOP	Scripting	
Other Base Libraries	Beans	Intl Support		I/O	JMX		JNI		Math
	Networking	Override Mechanism		Security	Serialization		Extension Mechanism		XML JAXP
lang and util Base Libraries	lang and util	Collections		Concurrency Utilities	JAR		Logging	Management	
	Preferences API	Ref Objects		Reflection	Regular Expressions		Versioning	Zip	Instrument
Java Virtual Machine	Java Hotspot™ Client VM				Java Hotspot™ Server VM				
Platforms	Solaris™		Linux		Windows		Other		

rgupta.mtech@gmail.com

How Sun define Java?
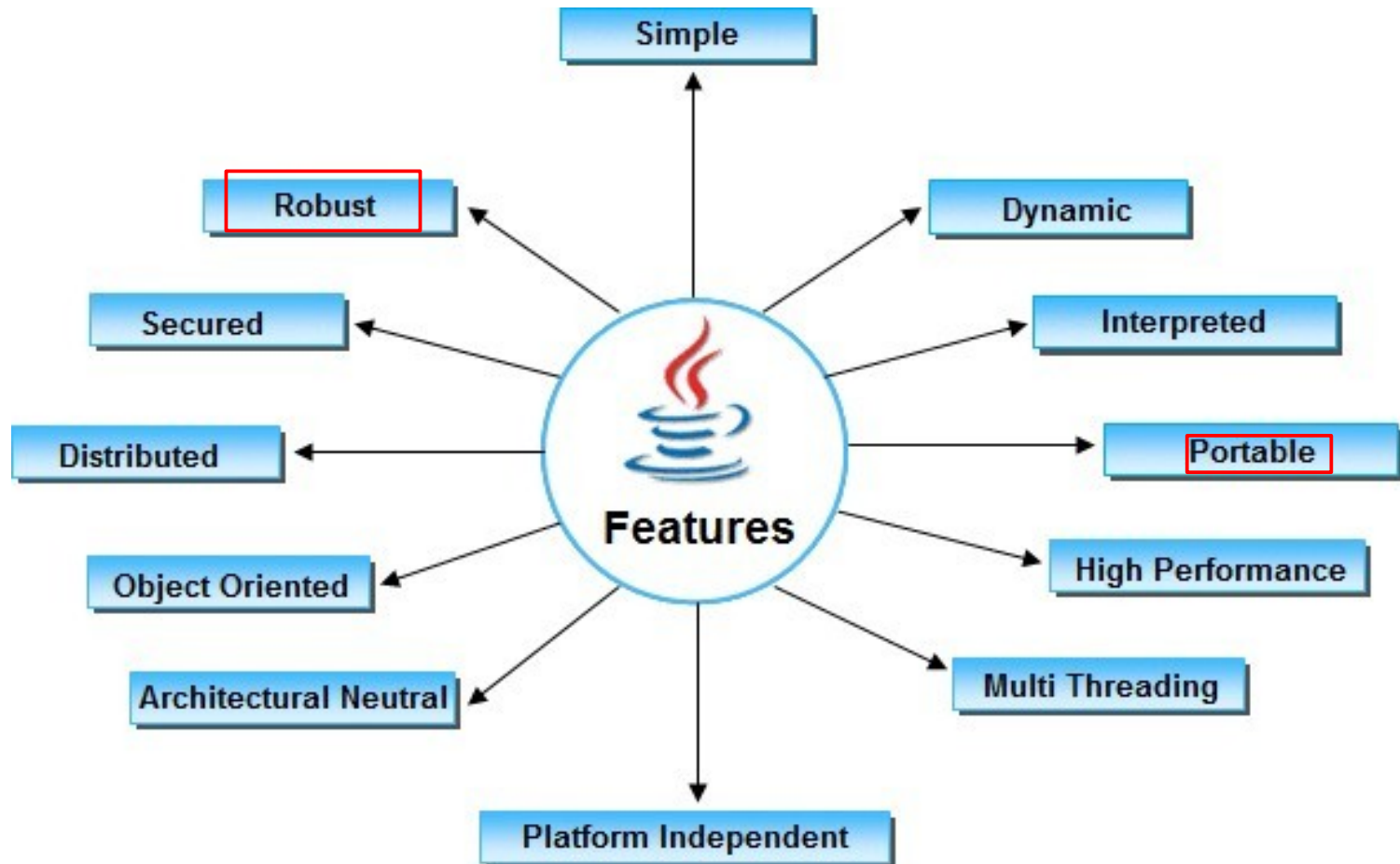
Versions of Java

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

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) 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 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];
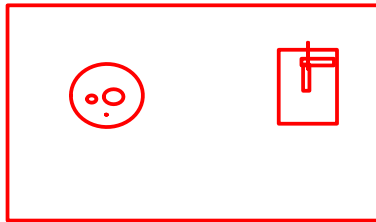
p[6]; AIOEx

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)
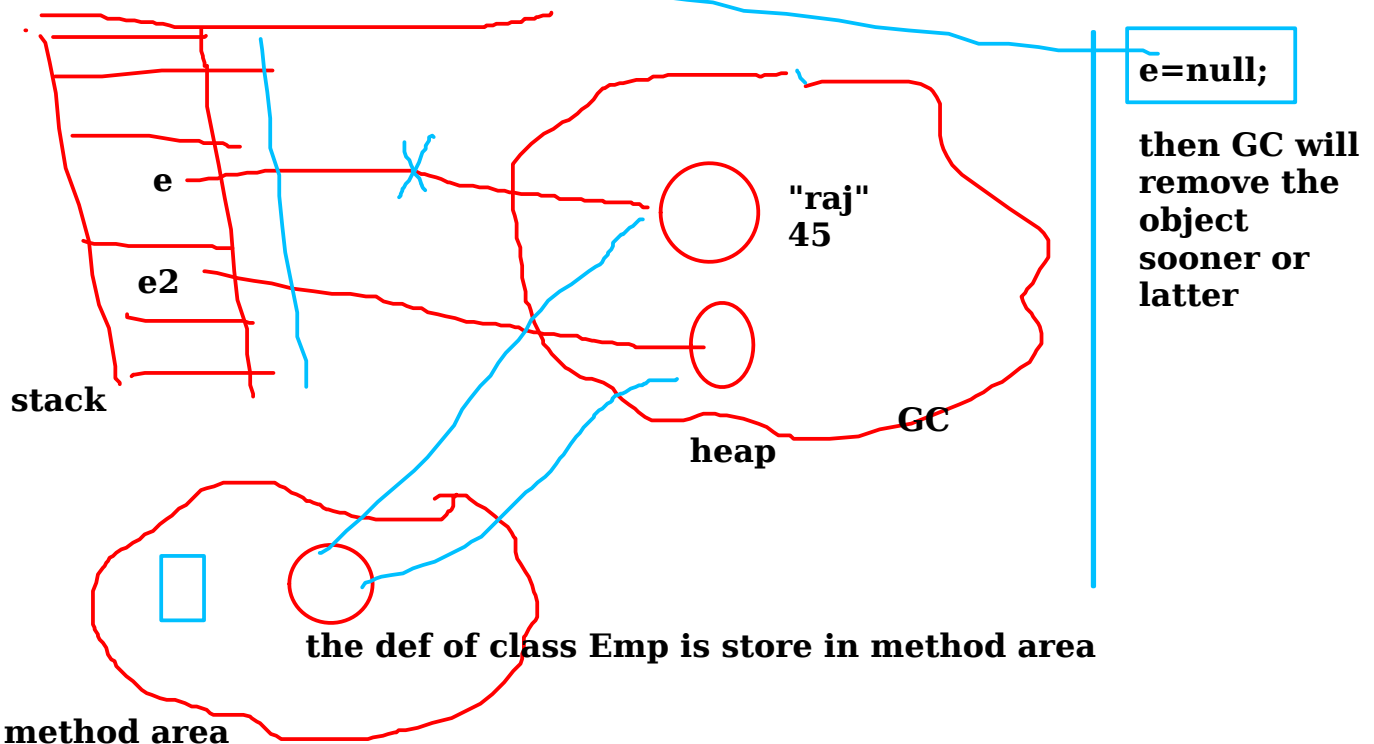
_____

How it works?

Emp e =new Emp("raj");

Emp e2 =new Emp("ekta");

object handler
ref

class : is a tempate
to create the object

cookie cuttor

e=null;

then GC will
remove the
object
sooner or
latter

e

e2

"raj"
45

stack

GC

heap

the def of class Emp is store in method area

method area

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++
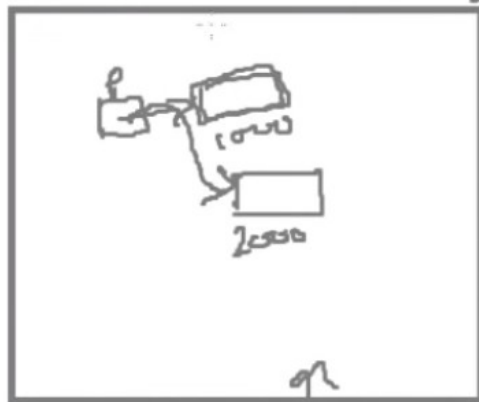
Some common buzzwords

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…

rgupta.mtech@gmail.com

JDK
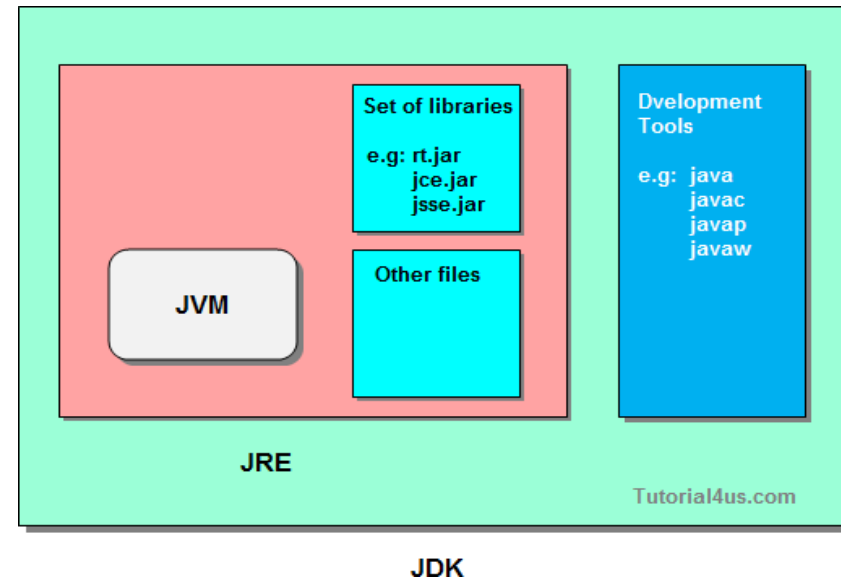
JDK: dev need it

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

JDK

tools

**ava, javac
javap...
...**

jre

jvm

lib
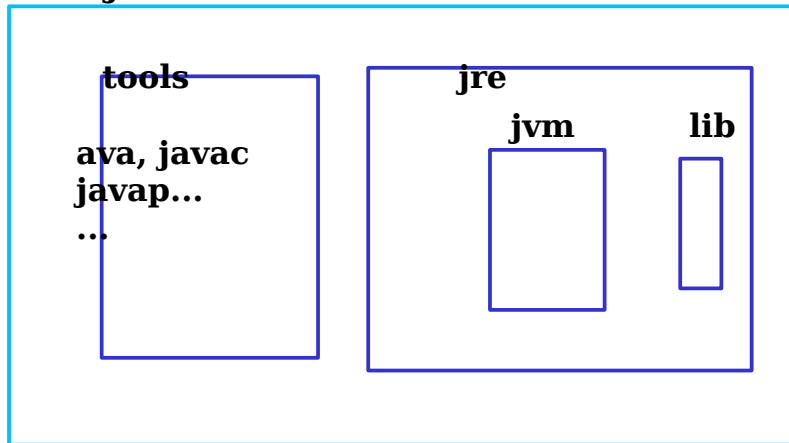
Understanding JVM

Key HotSpot JVM Components

Class Files → Class Loader Subsystem

Runtime Data Areas:
- Method Area
- Heap
- Java Threads
- Program Counter Registers
- Native Internal Threads

Execution Engine
- JIT Compiler
- Garbage Collector

Native Method Interface ← Native Method Libraries

rgupta.mtech@gmail.com
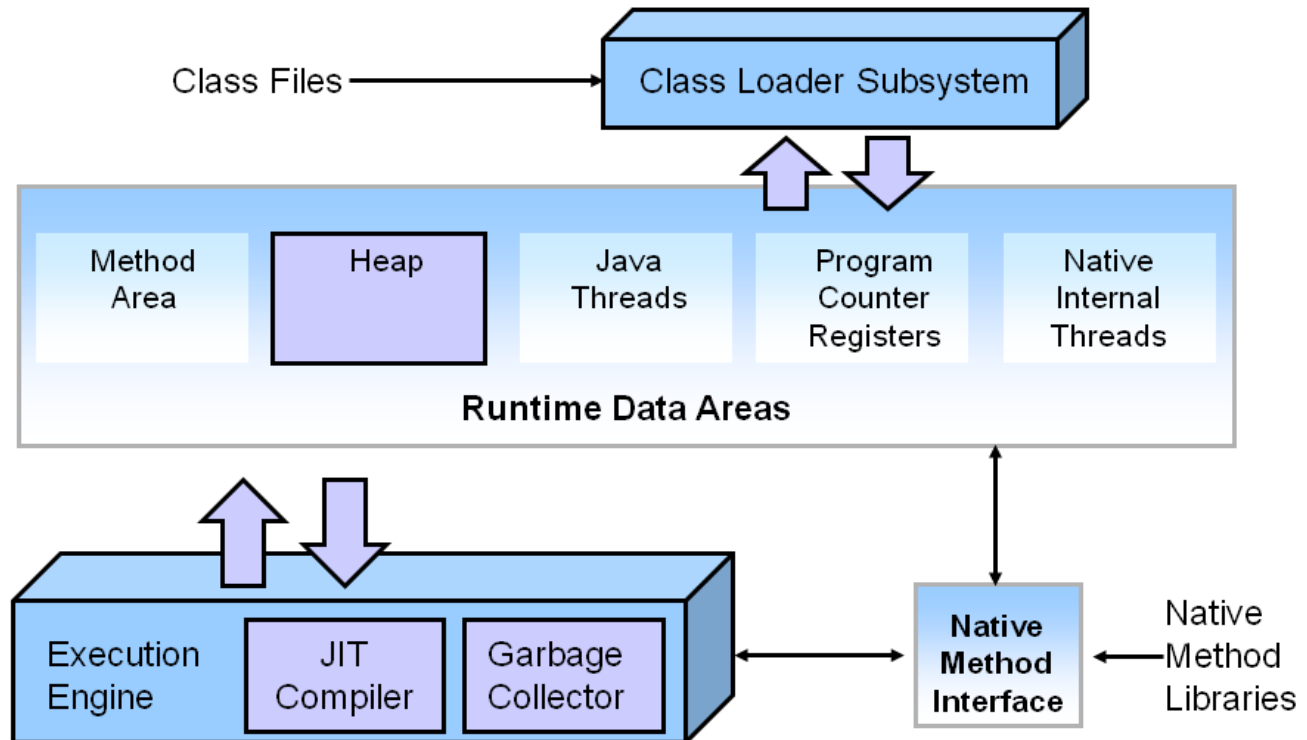
Hello World

Applet: netspace neg + java!

Lab set-up

❖ Java 1.8

❖ Eclipse ~~Luna~~
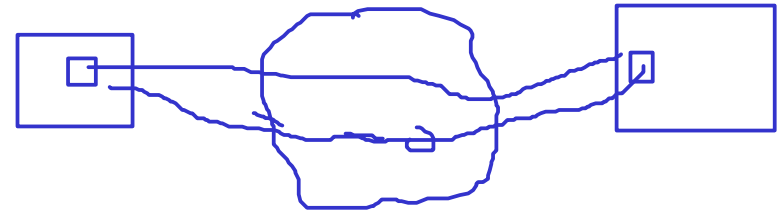
```
public class HelloWorld {

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

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

Name of the program

visibility modifier

declare by compiler, define by user

```java
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
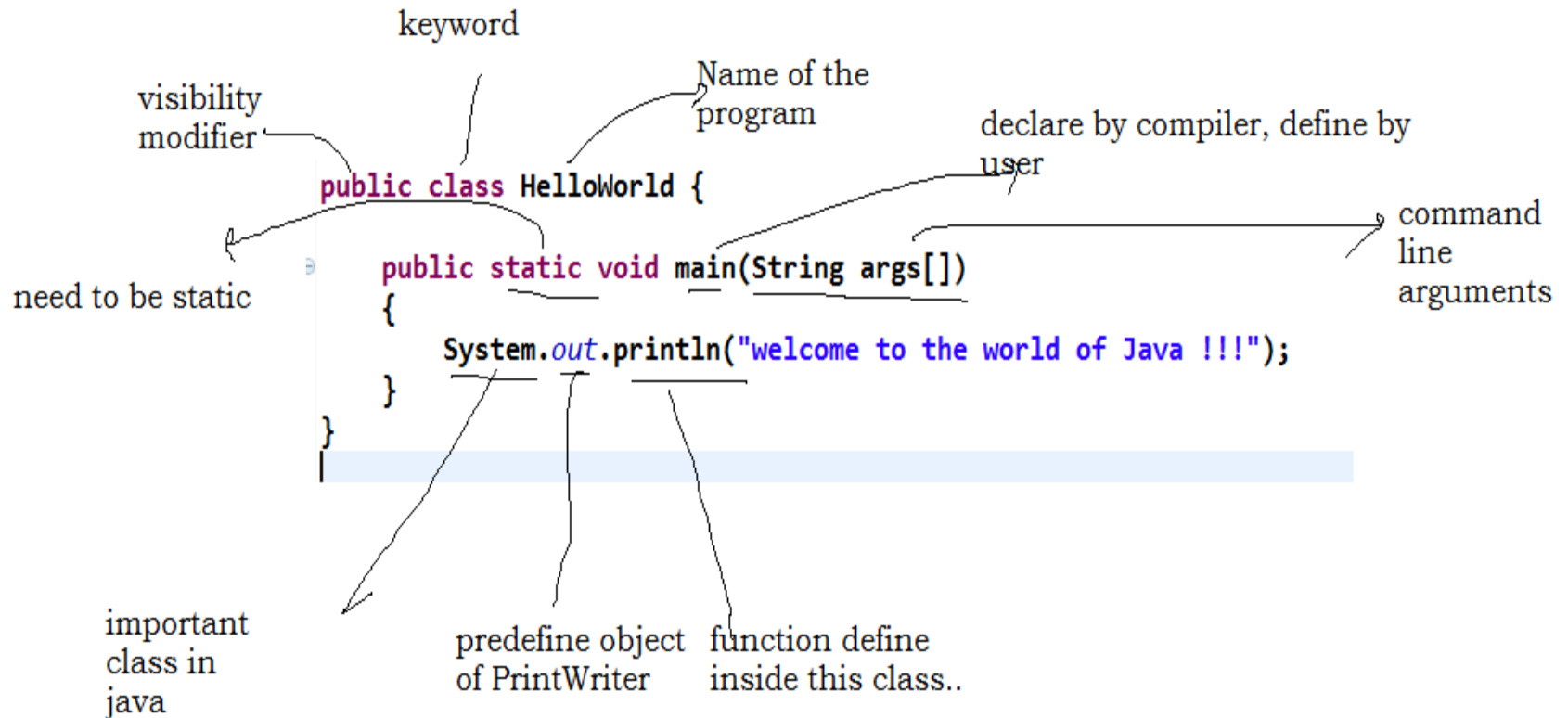
Some basics rules about java classes

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?
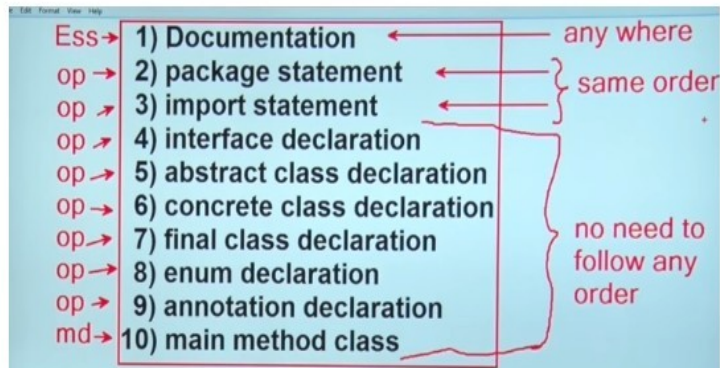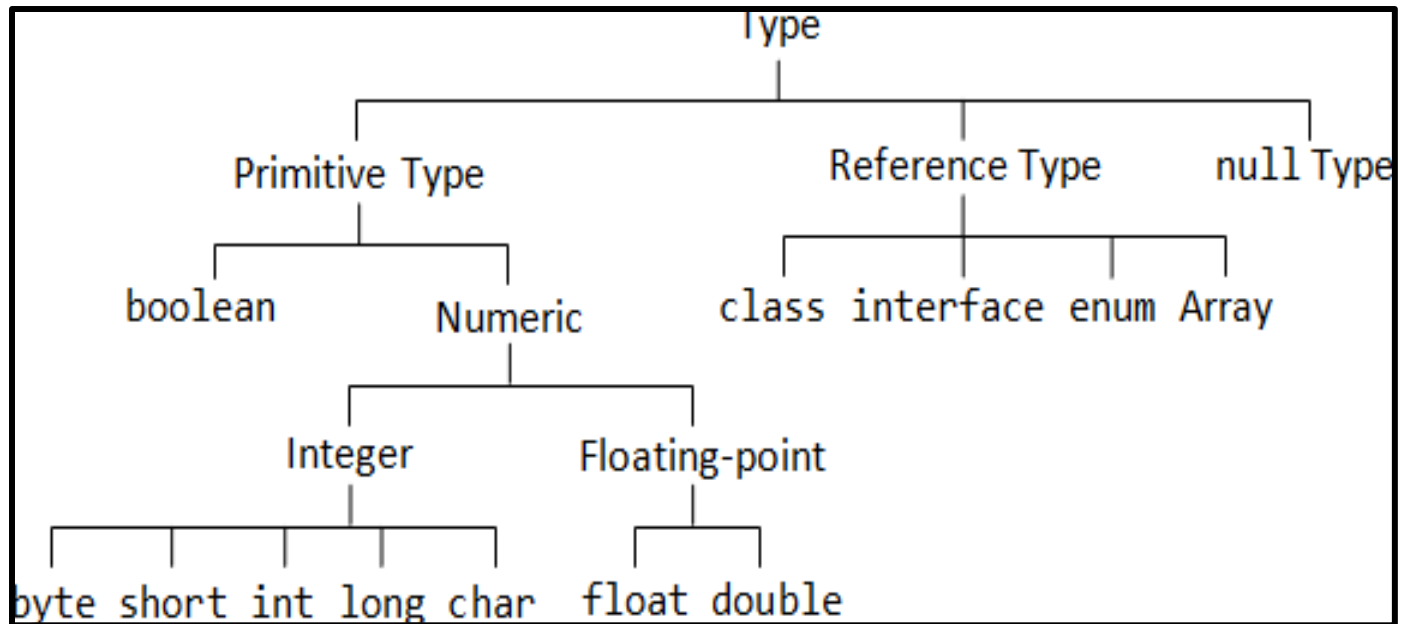
6) Is it possible to create Empty Java file? ✓
7) Is it possible to create Empty class? ✓
8) Main method mandatory for compilation or execution? ✓

Example.java

>javac Example.java
-> compile
   .class file is generated?
-> executed

Example.java

class Example {

}

>javac Example.java
-> compile? ✓
   .class file is generated? ✓
-> executed? ✗

Ess → 1) Documentation ← any where
op → 2) package statement ← } same order
op → 3) import statement ←
op → 4) interface declaration
op → 5) abstract class declaration
op → 6) concrete class declaration
op → 7) final class declaration } no need to follow any order
op → 8) enum declaration
op → 9) annotation declaration
md → 10) main method class

Java Data Type

2 type
- Primitive data type
- Reference data type

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

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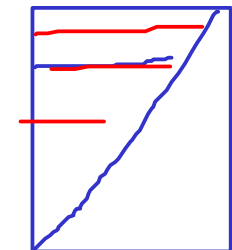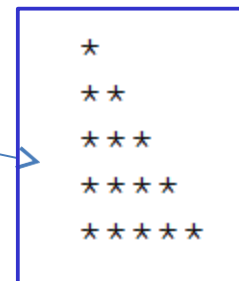
prime no:
proof by contradition?

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];
                                              heap
_____
stack**x**

0	1	2 3	4

2020

x=null;

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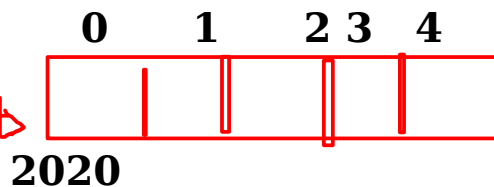
why ref is safer then pointer?
c++: ref vs pointer
        safe pointer

```java
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;
    }
}
```

x

temp

| 5 | 6 | 7 | 9 | |
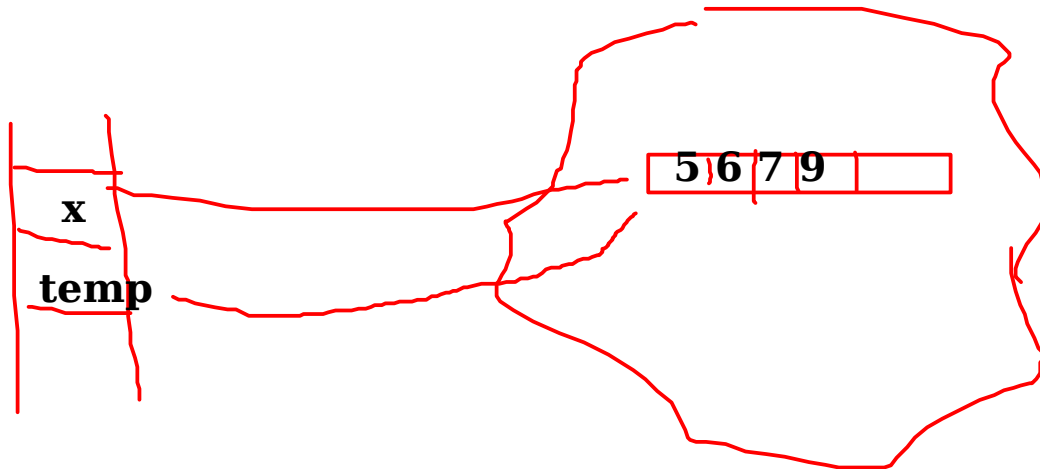
1							
1	2						
1	2	3					
1	2	3	4				
1	2	3	4	5			
1	2	3	4	5	6		
1	2	3	4	5	6	7	
1	2	3	4	5	6	7	8

1
11
121
1331
14641
......

2D array

2 D array
array of objects?

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

how it works?

0
1
2
3

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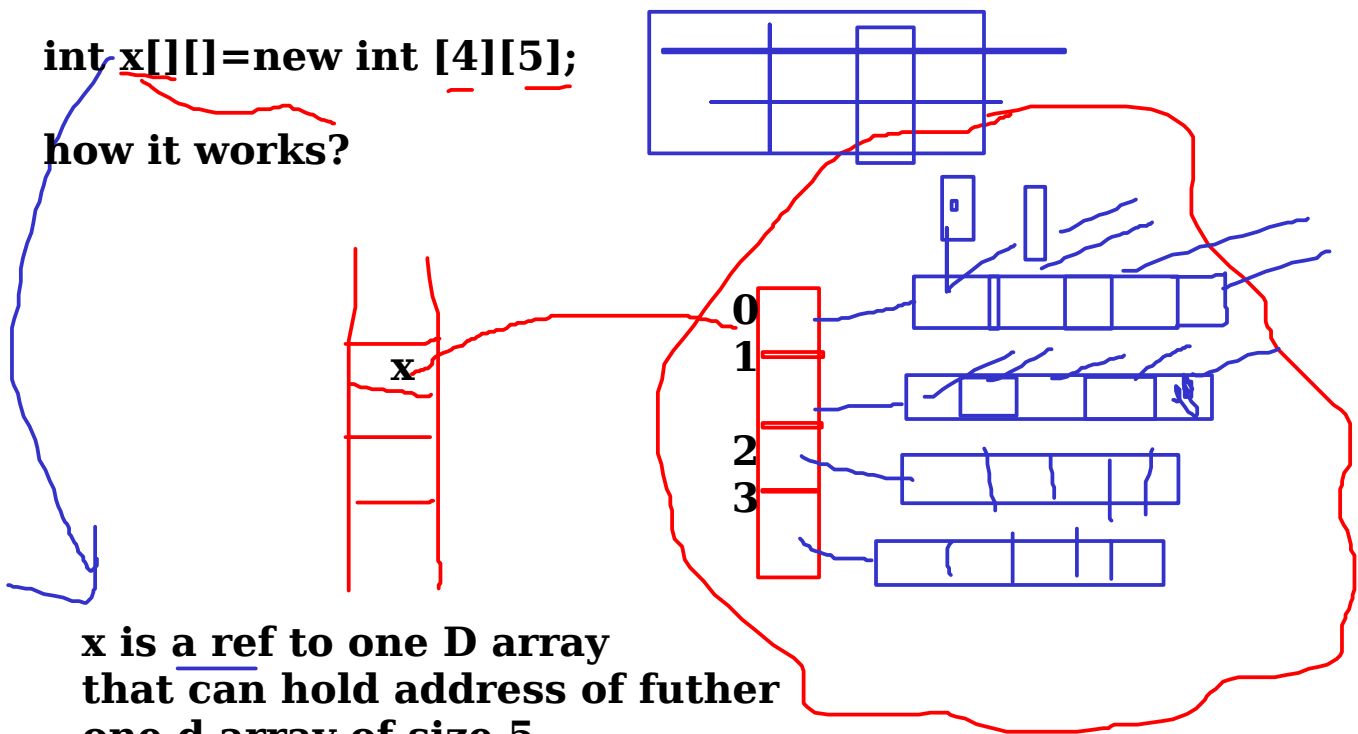
x is a ref to one D array
that can hold address of futher
one d array of size 5

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 ...)
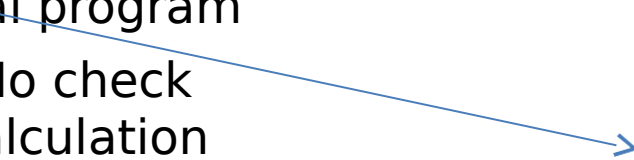
primitive programming

Object oriented programming!
collection api

_____

| 5 | 6 | 7 | 8 | 11 | 0 |

10
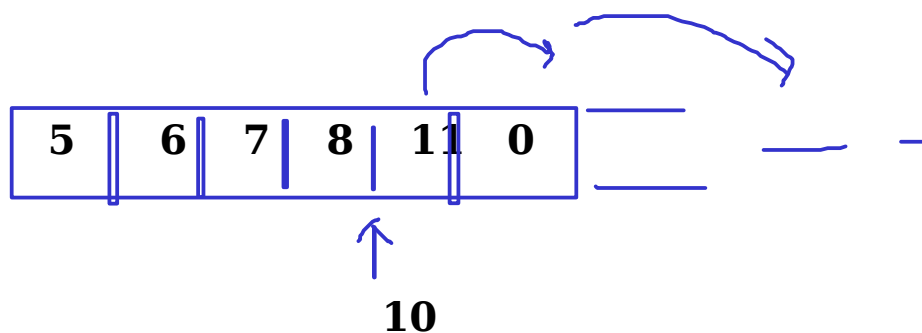
Procedural Programming

❖ 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

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

Array are object in java

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

One Dimensional array

Initialization   int a[] = new int [12];

Value | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

Index   a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a11[11]

System.out.print(a[5]);   Output: 6

stack   heap
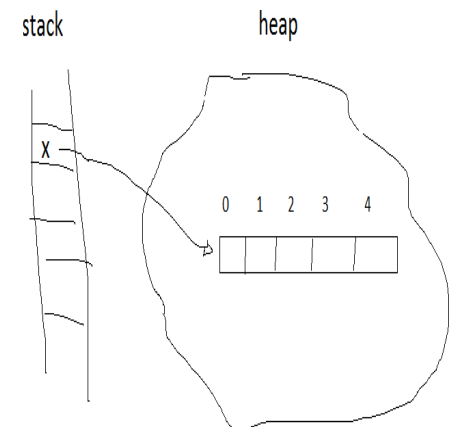
Java Array: its different then C/C++

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

Tom's House :House ← lives in — Tom:Person — drives → Tom's Car:Car

rgupta.mtech@gmail.com
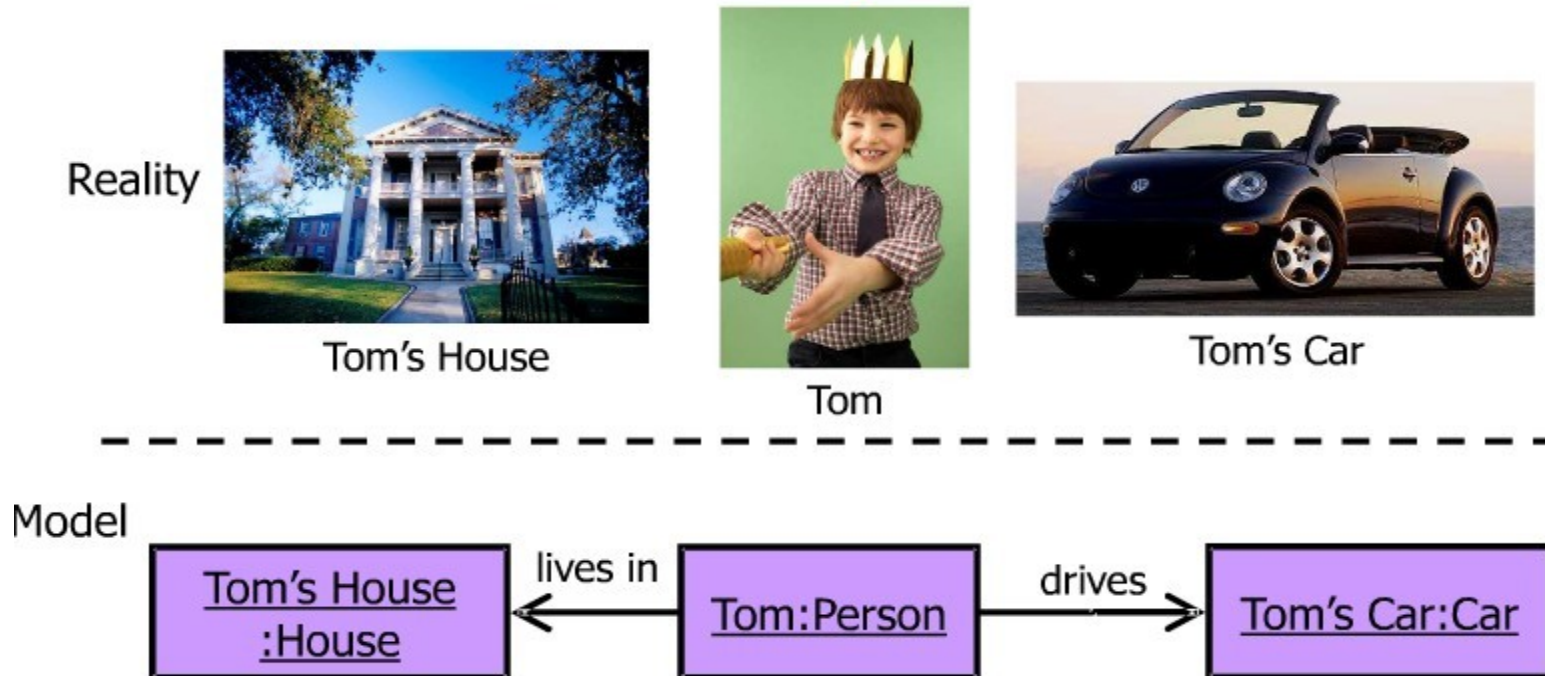
What is an Object?

- 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

    100
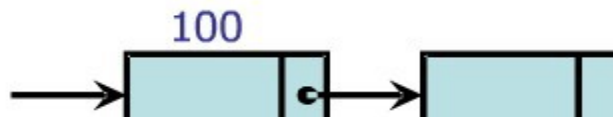
Class

- 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;
    // …
}
```

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

Person

- name
- dateOfBirth

+ getAge()

harry:Person

| name | Harry |
| dateOfBirth | 8/12/1975 |

+ getAge()

mary:Person

| name | Mary |
| dateOfBirth | 8/12/1980 |

+ getAge()

Basic principles of OO
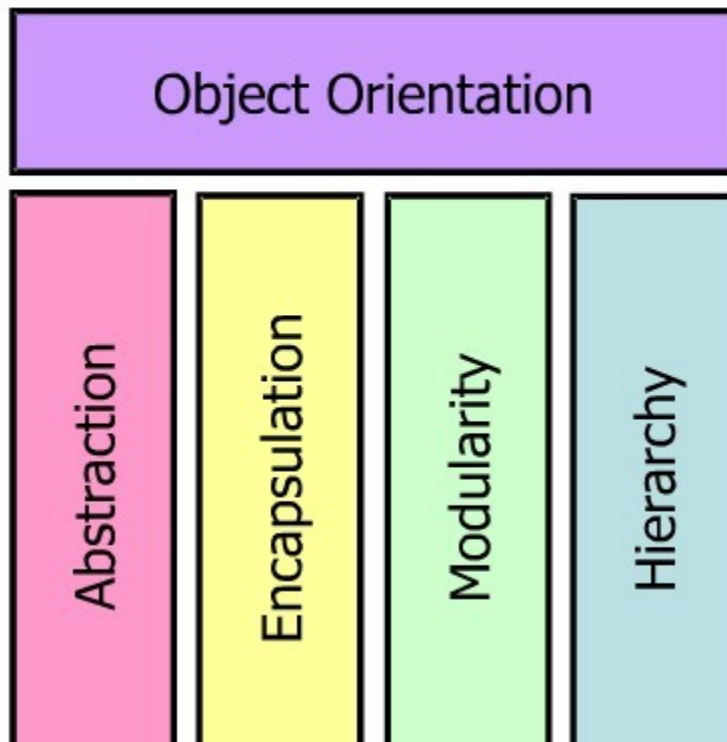
Dealing with Software complicity:Abstraction
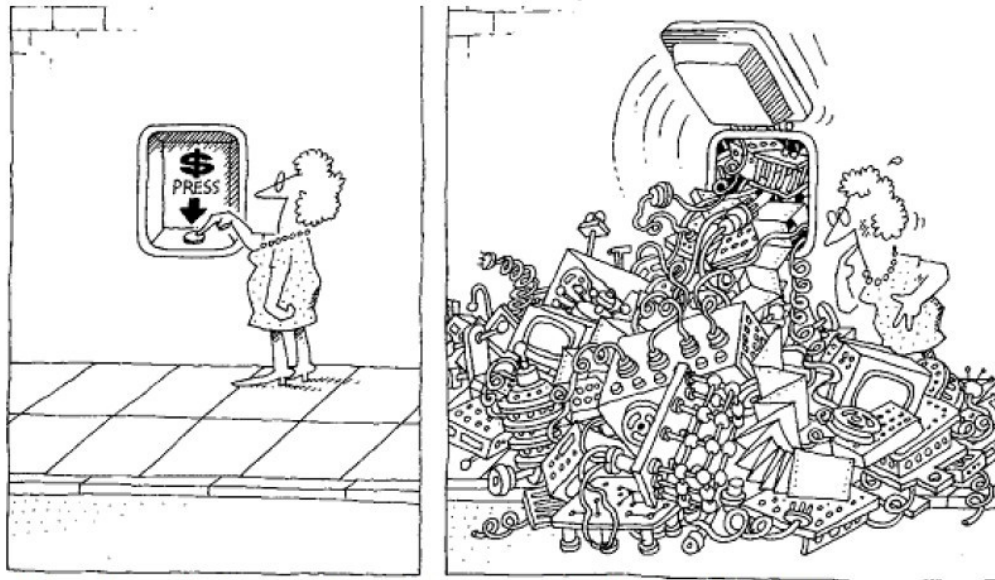
The task of the software development team is to engineer the illusion of simplicity.

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
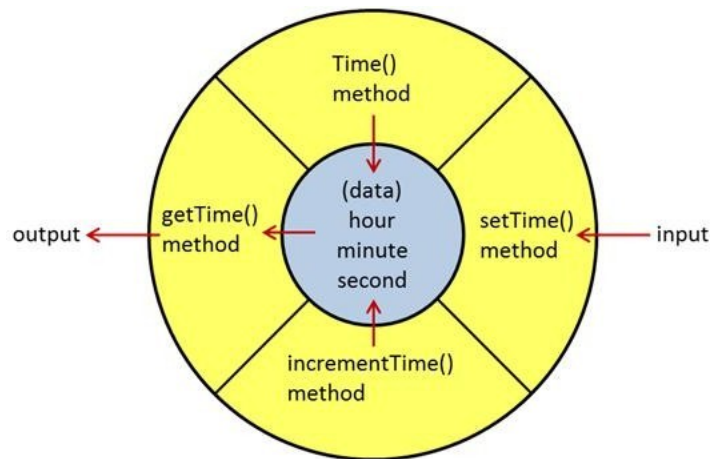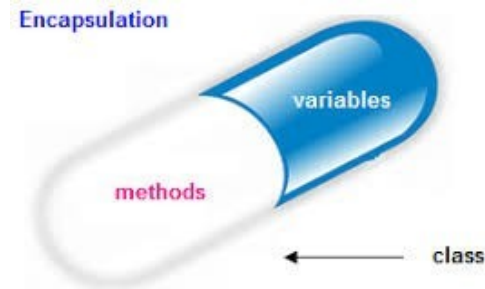
<u>Encapsulation</u>

❖ 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
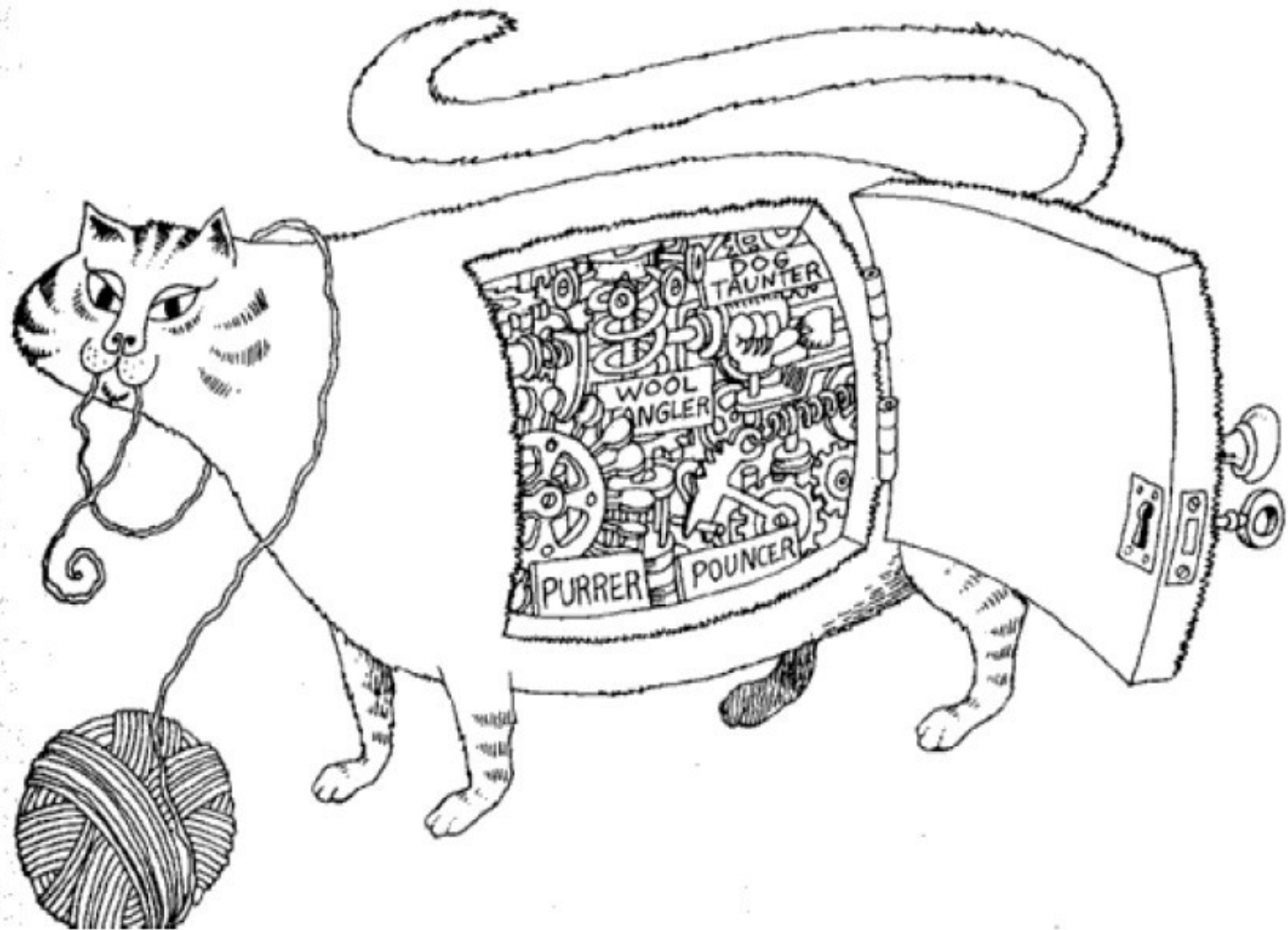
Understanding encapsulation
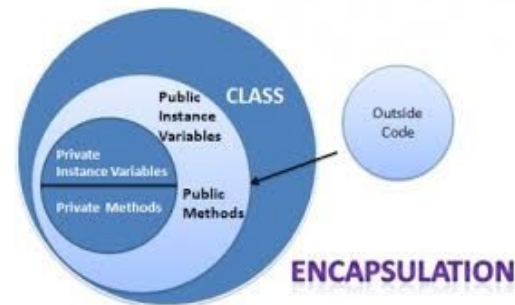
Encapsulation

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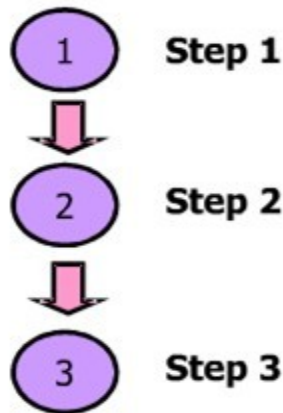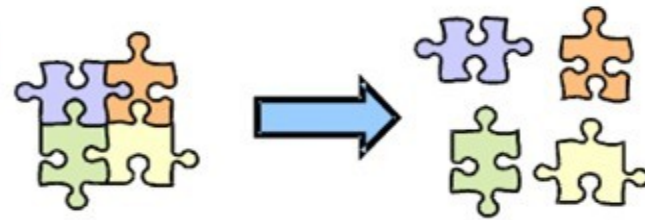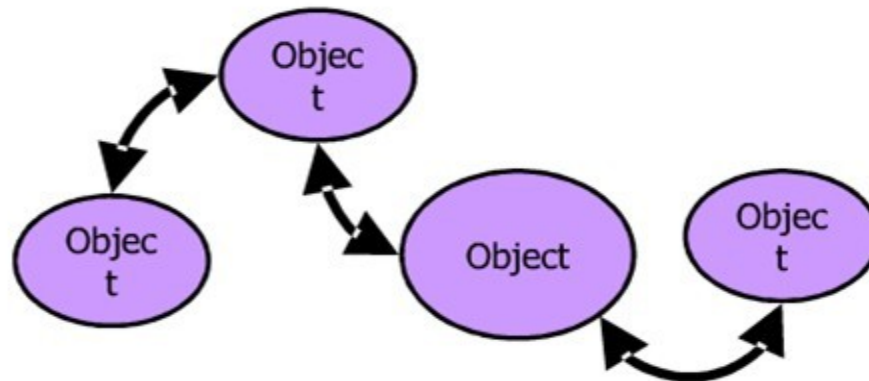- Encapsulation= data hiding + constraints

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

Hierarchy

- Ranking or ordering of objects

Increasing abstraction

Decreasing abstraction

Cash

Deposit Account

Notes & Coins

Saving Account

Current Account