OOPs

Tuesday, 20 February 2024 8:36 PM

↳ Object Oriented Programming
- → Encapsulation
- → Abstraction
- ↳ Inheritence
- → Polymorphism

↳ Consider everything as an object

Java Runtime Environment

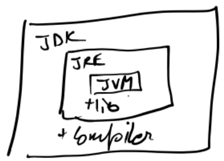
JVM + lib → JRE ⇒ JDK
JRE + Compiler

JVM ↓ Java Virtual Machine

JDK ↓ Java Development Kit

Java?
↳ C, C++ : .exe → C++
↳ Platform dependent

↳ JVM ✓ .java ✗ → Byte code
↳ .class ⇒ Machine code
↳ platform independence



Client	Provider
JVM ✓	✓
Compiler ✗	✓
Lib ✓	

↳

If C, why did we need C++ or Java?
↓
Procedural Language
↳ Everything is a function

int carnel = ..
int age = ..
cal cc = ..

Car {
name
age :
cc :

10

Interact with real life → fun
Student {
int . ..

int carno = " "

→ CC
→ Type size

?? Modeling things is tough

now

↳ object

int rollno;

String name = " "

}

s1.roll
s2.roll
s3

Student { → class

String name;
int age;
}

Student class
↳ used by client

D.S

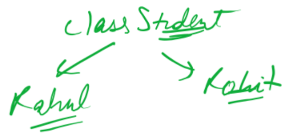
primitive? or non?

→ objects
↳ Kartik
↳ object

Class is a blueprint which defines the object
↳ data members → properties
↳ data function → behaviour

Object is a physical entity of the class

Encapsulation → wrapping up relevant info (members, function) under a single unit
↳ class



Student s2 = new Student();
↳ s2.name = Rahul
s2.age = 24
Student s3 = new Student();
↳ Reference to 15K

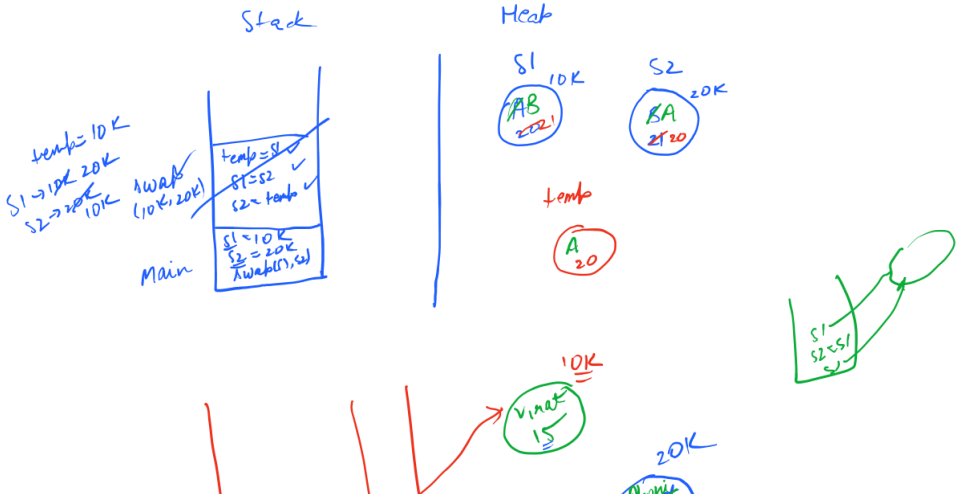
s2 → Rahul 24 → Instance

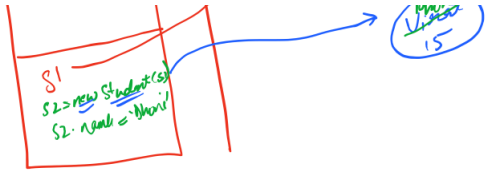
new A/B/C x y z

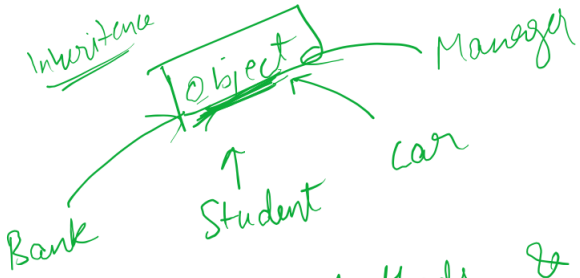
10K → used to bind the properties to current object
this.age;
↳ keyword which is used to point to a particular memory address.

10K
Student s = new Student();
↓
constructor

Heap
10K







Methods & Functions

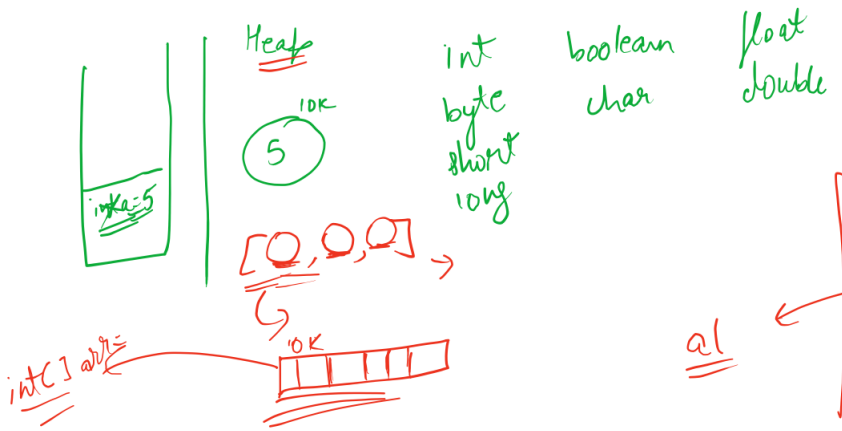
↳ Functions that belong to a class

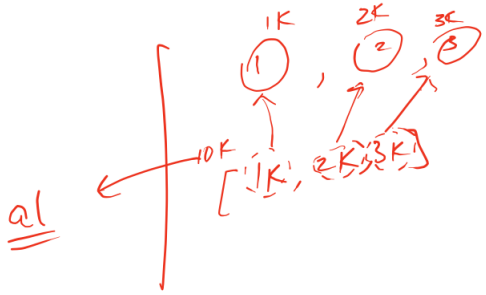
↳ OOPs concept

↳ associated with class

↳ Any set of instruction

Wrapper ... Class ... for ... primitive data type





Object / String → null

int → 0

boolean → false

Constructor returns the instance of the object