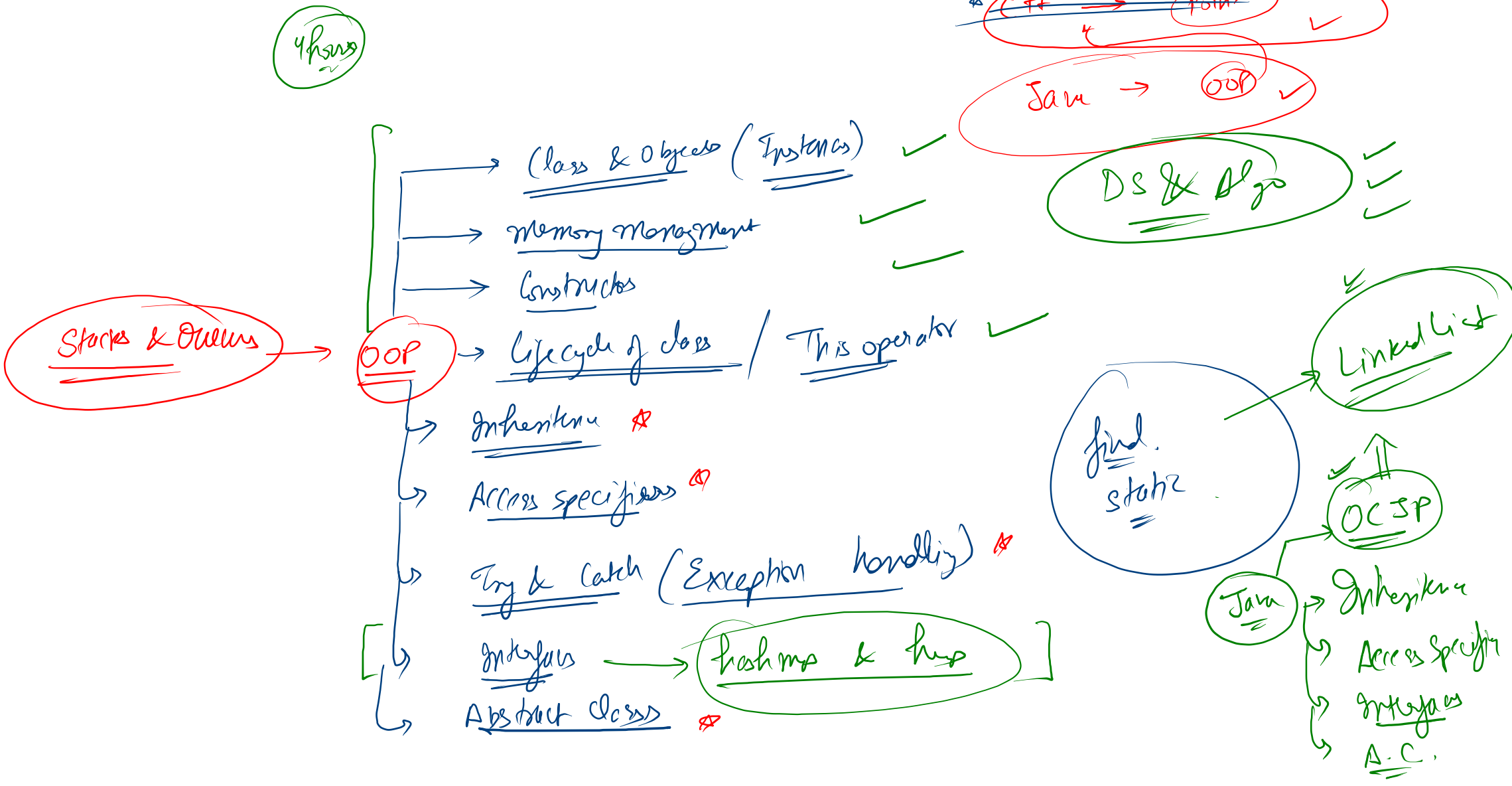

4 hours
C++ → Pointers
Java → OOP
DS & Algo
Stacks & Queues
OOP
Class & Objects (Instances)
Memory Management
Constructor
Lifecycle of class / This operator
Inheritance
Access specifiers
Try & Catch (Exception handling)
Interfaces → Hashmap & Map
Abstract class
find.
static
LinkedList
OCSP
Java
Inheritance
Access Specifiers
Interfaces
A.C.

NOTES

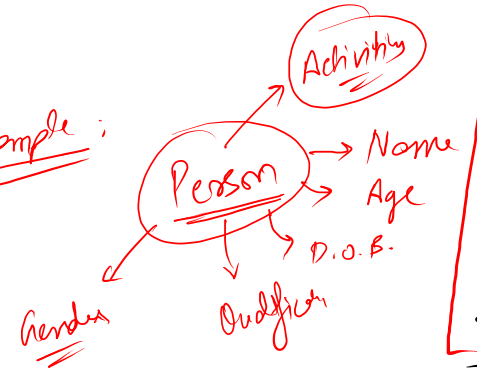
Class → Blueprint / Map

Collection of data members & member function

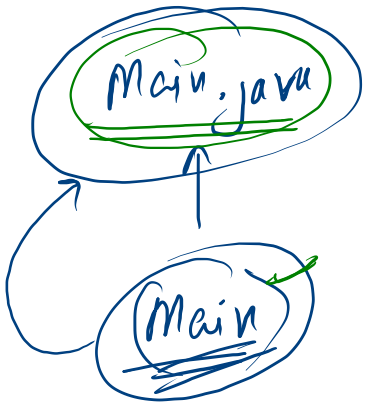
Used for defining a system or real world object

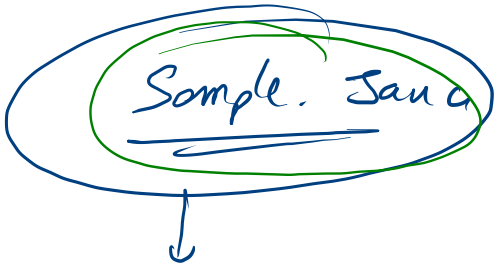
used to define a heterogeneous data type

Example:




```
p s   class Person {

    String Name;
    int age;

    void sayHi(){
        print(name + " -> " + age);
    }
}
```





run (Compile + Execute)

Sample.jav

class Sample {

psvm main(S args){

}

}

```java
public class Solution{
    public static class Person{
        String name;
        int age;
        void saysHi(){
            System.out.println(name+" --> "+age+" says hi.");
        }
    }
    Run | Debug
    public static void main(String args[]){
        // System.out.println(Person);

        Person p1 = new Person();
        p1.name = "abc";
        p1.age = 1;
        p1.saysHi();
        // System.out.println(p1.name);
    }
}
```

Data Members

Ref

Instance

Heap

name : abc
age : 0 1

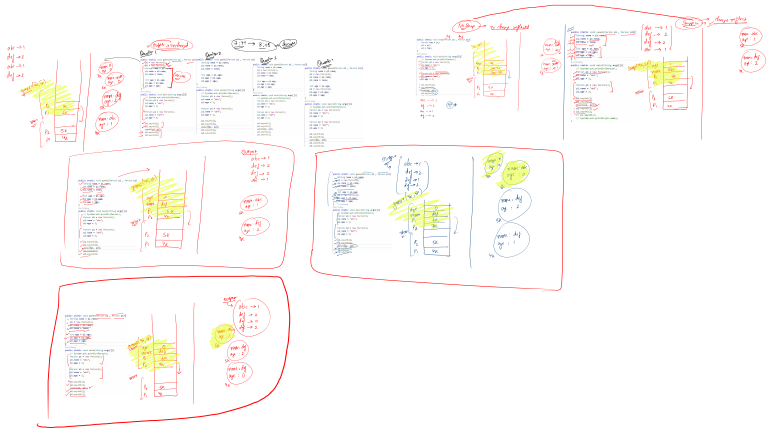
4K

abc <--> 1 says hi

main

P1 | 4K

object → Ref + Instance → Space allocation → data members



Constructor

- Special member func which is called at the time of instance creation.
- no return type
- has same name as class
- automatically called
- Types
  - Default
  - Parametrized

NOTE

- Java takes care/provides default constructor if user forgets to do so.
- * Java will not provide a self implemented default constructor if user has defined a parameterized constructor i.e. we have to implement default constructor ourselves.

	Default	Parameterized	Conclusion
1	X	X	: Java will provide default constructor
2	✓	X	: User has defined default constructor
3	X	✓	: Java will not provide default constructor i.e. user will have to define if needed.
4	✓	✓	: both default + parameterized const. are defined by user, no conflict case.

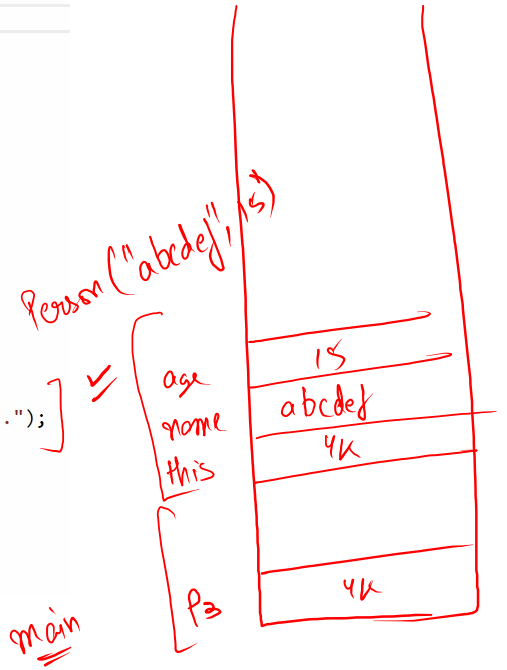
Introduction to this Keyword

local var update ✗ data memb
↳ name same

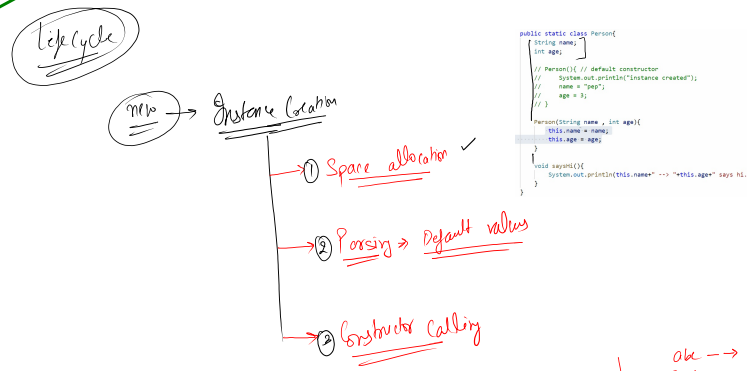
Practice
↳ data members
↳ this

```java
public static class Person{
    String name;
    int age;

    // Person(){ // default constructor
    //     System.out.println("instance created");
    //     name = "pep";
    //     age = 3;
    // }

    Person(String name, int age){
    this.name = name;
    this.age = age;
    }

    void saysHi(){
        System.out.println(name+" --> "+age+" says hi.");
    }
}
Run | Debug
public static void main(String args[]){
    Person p3 = new Person("abcdef",15);
    p3.saysHi();
```

this →



name : abcdef
age : 15
4K




```java
public static class Person{
    String name;
    int age;

    // Person(){ // default constructor
    //     System.out.println("instance created");
    //     name = "pep";
    //     age = 3;
    // }

    Person(String name , int age){
        this.name = name;
        this.age = age;
    }

    void saysHi(){
        System.out.println(this.name+" --> "+this.age+" says hi.");
    }
}
```

```java
public static class Person{
    String name = "abc";
    int age = 100;

    Person(){ // default constructor
        this.saysHi();
        System.out.println("instance created");
        name = "pep";
        age = 3;
    }

    // Person(String name , int age){
    //     this.name = name;
    //     this.age = age;
    // }

    void saysHi(){
        System.out.println(this.name+" --> "+this.age+" says hi.");
    }
}
Run | Debug
public static void main(String args[]){
    // Person p3 = new Person("abcdef",15);
    // p3.saysHi();
    // System.out.println(Person);
    Person p1 = new Person();
    p1.saysHi();
```