Object Oriented Programming in Java

folder OOPS

Main.java ⇒ source code

```java
package OOPS;

public class Main {
    Run | Debug
    public static void main(String[] args){
        System.out.println(x: "Hello World");
    }
}
```

Compilation of SourceCode

javac OOPS/Main.java

Compiler

↳ Compilation errors → no compilation

↳ Syntax errors

↳ logical errors

line by line

↳ top to bottom

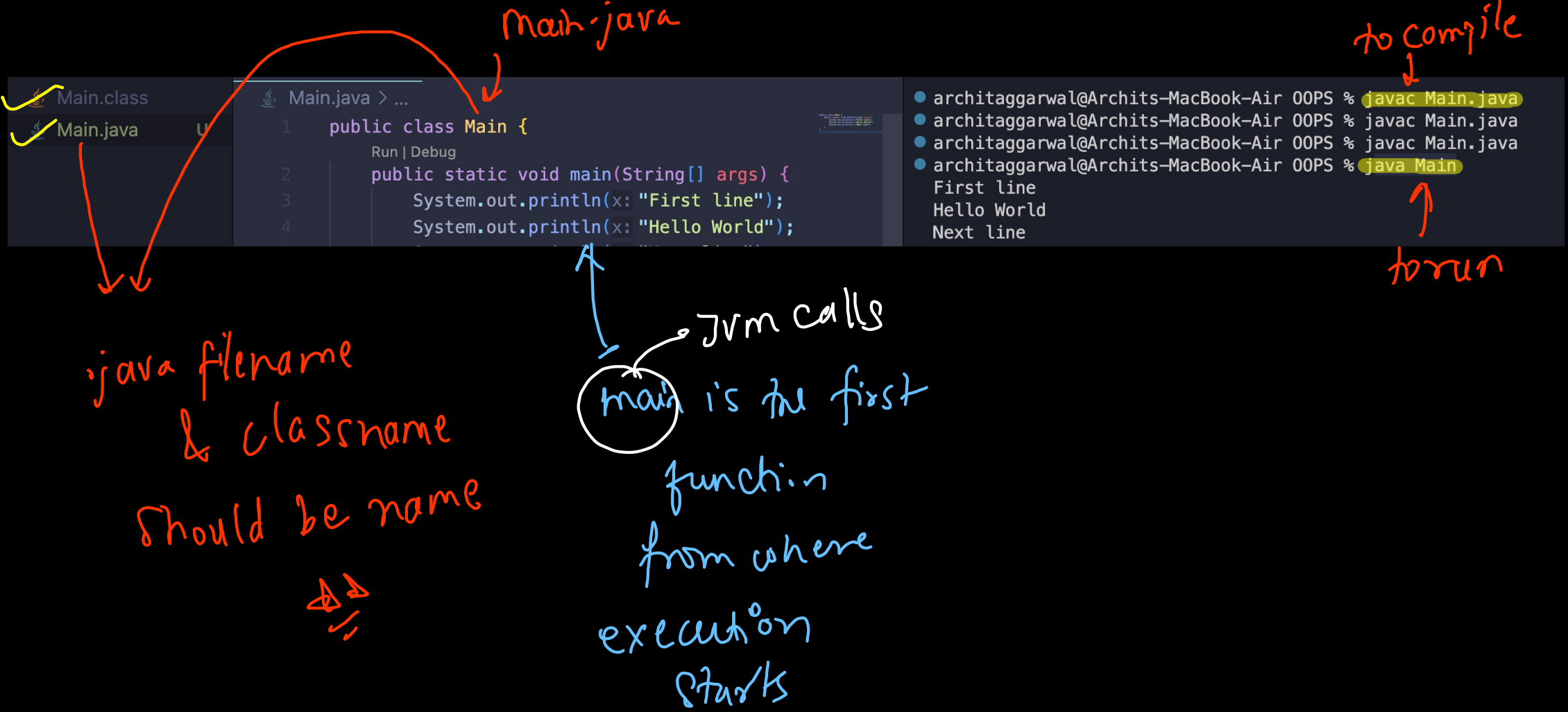
↳ left to right

Main.class ← file which can be executed!

{byte code}

or any platform

↳ Java platform Independent


Main.class
Main.java
Main.java > ...
1 public class Main {
Run | Debug
2 public static void main(String[] args) {
3 System.out.println(x: "First line");
4 System.out.println(x: "Hello World");
Main.java
to compile
architaggarwal@Archits-MacBook-Air OOPS % javac Main.java
architaggarwal@Archits-MacBook-Air OOPS % javac Main.java
architaggarwal@Archits-MacBook-Air OOPS % javac Main.java
architaggarwal@Archits-MacBook-Air OOPS % java Main
First line
Hello World
Next line
to run
.java filename & classname should be name
JVM calls main() is the first function from where execution starts

Problem without OOPS

→ data of an entity is not related to other data.
(properties) (real world)

→ Data inconsistency can arise

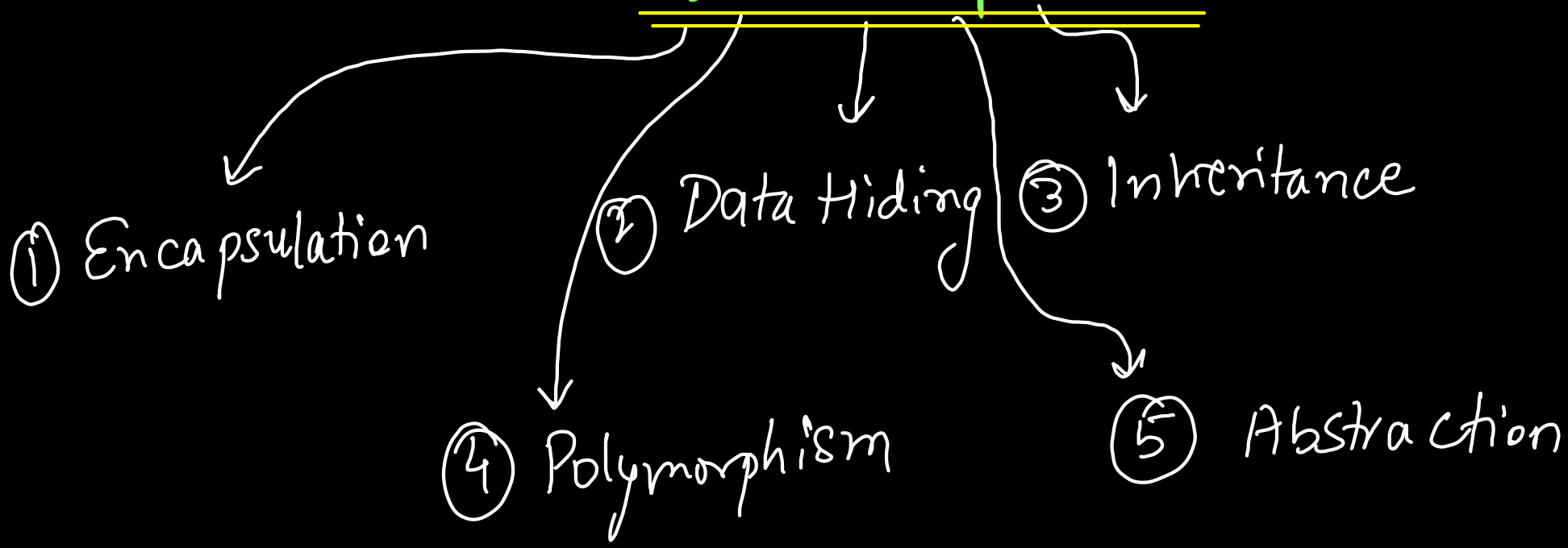
→ lines of code will increase
↳ debugging becomes harder

→ Can't provide functionalities / behavior to the data

→ less security

→ less reusability of code

5 Pillars of OOPS



$$^5C_2 = \frac{5!}{2!\,3!}$$

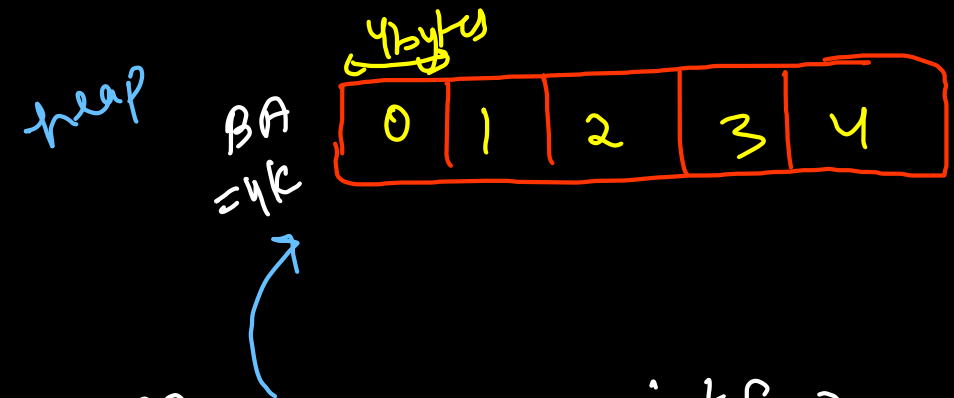
```java
int nfact = factorial(n: 5);
int rfact = factorial(n: 2);
int nmrfact = factorial(n: 3);

System.out.println(nfact / (rfact * nmrfact));
```

```java
public static int factorial(int n) {
    int fact = 1;
    for (int idx = 1; idx <= n; idx++) {
        fact = fact * idx;
    }
    return fact;
}
```

Less Reusability in Procedural Programming

① → object oriented programming principle

② → no concept of pointers ⇒ enhances security



arr[2] = 4K + 4+4
= 4K08

arr[5] ⇒ array index out of bound exception

③ Java is platform independent

④ Java is both compiled and interpreted language!

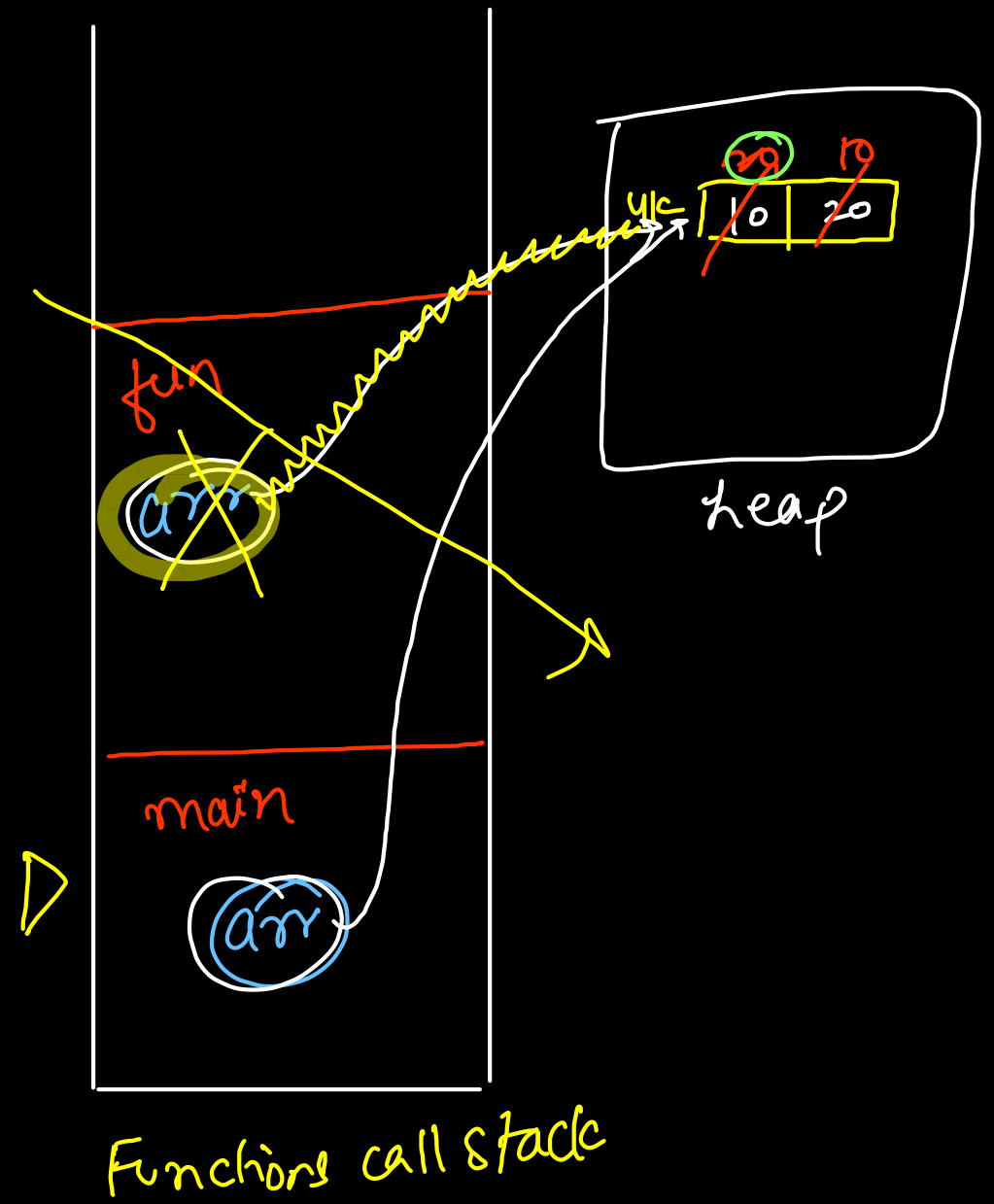
main

```
int[] arr = new int[2];

arr[0] = 10
arr[1] = 20

fun(arr);
Syso( arr[0] + arr[1])
```

(arr[0] → 20, arr[1] → 10)

fun

```
arr[0] = 20
arr[1] = 10
```



```java
public class Student {
    int rollNo;
    String name;
    double marks;
    char gender;
}
```

properties or data members

Class : code

⇓

text segment

```java
public class Main {
    Run | Debug
    public static void main(String[] args)
    {
        Student s1 = new Student();

        System.out.println(s1.name);
        System.out.println(s1.rollNo);
        System.out.println(s1.gender);
        System.out.println(s1.marks);
    }
}
```

→ object / Instance (memory allocated)

→ null

→ 0

→ '\0'

→ 0.0

Arrays of Objects

```java
Student s2 = new Student();

s2.name = "Babita";
s2.rollNo = 2;
s2.gender = 'F';
s2.marks = 30;

System.out.println(s2.name);
System.out.println(s2.rollNo);
System.out.println(s2.gender);
System.out.println(s2.marks);
```

```java
Student[] arr = new Student[3];

arr[0] = s1;
arr[1] = s2;

arr[2] = new Student();
arr[2].name = "Chetan";
arr[2].rollNo = 3;
arr[2].gender = 'M';
arr[2].marks = 95;

System.out.println(arr[2].name);
System.out.println(arr[2].rollNo);
System.out.println(arr[2].gender);
System.out.println(arr[2].marks);
```

Memory Allocation of object

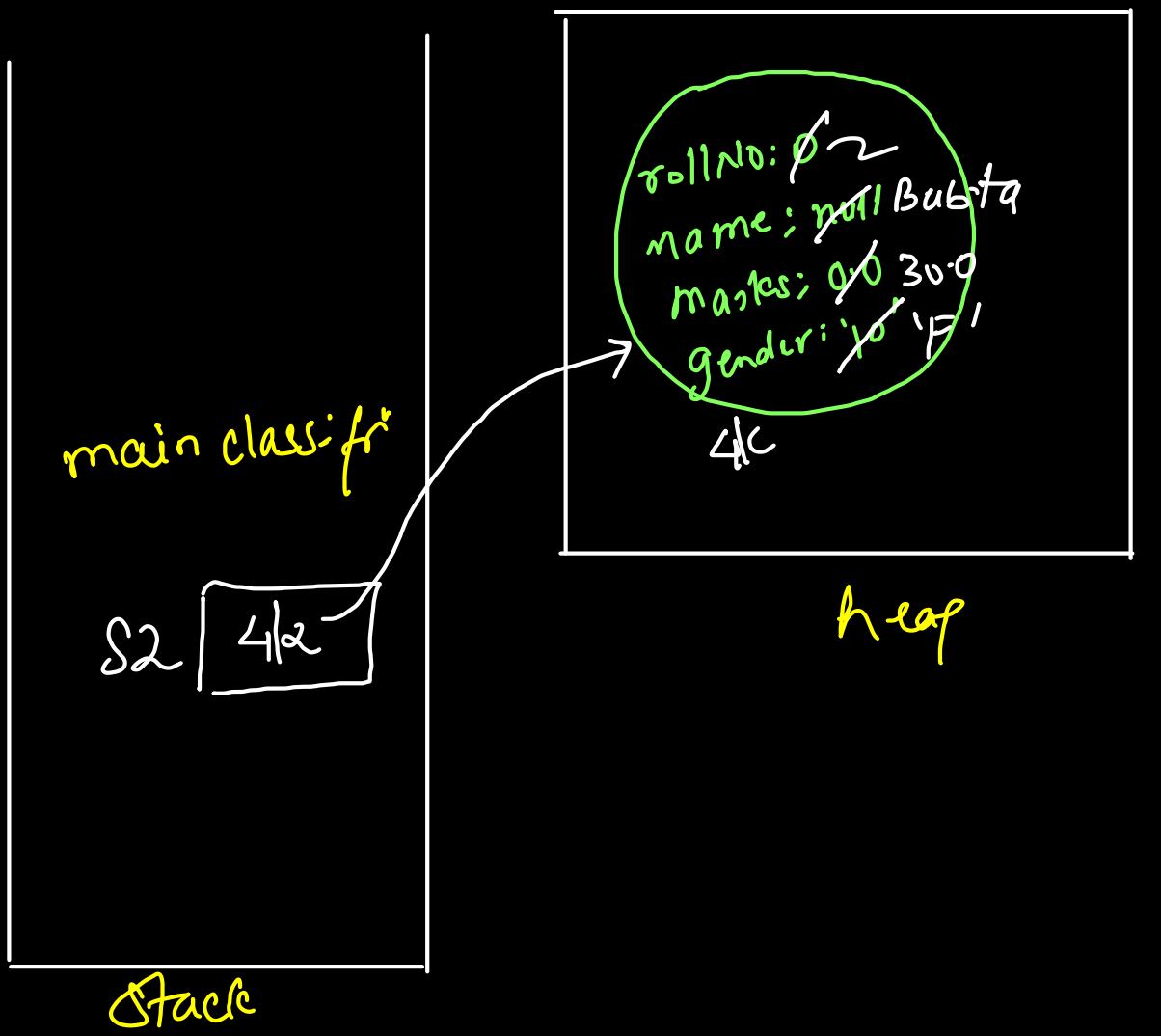
```java
public class Student {
    int rollNo;
    String name;
    double marks;
    char gender;
}
```

properties

main class {

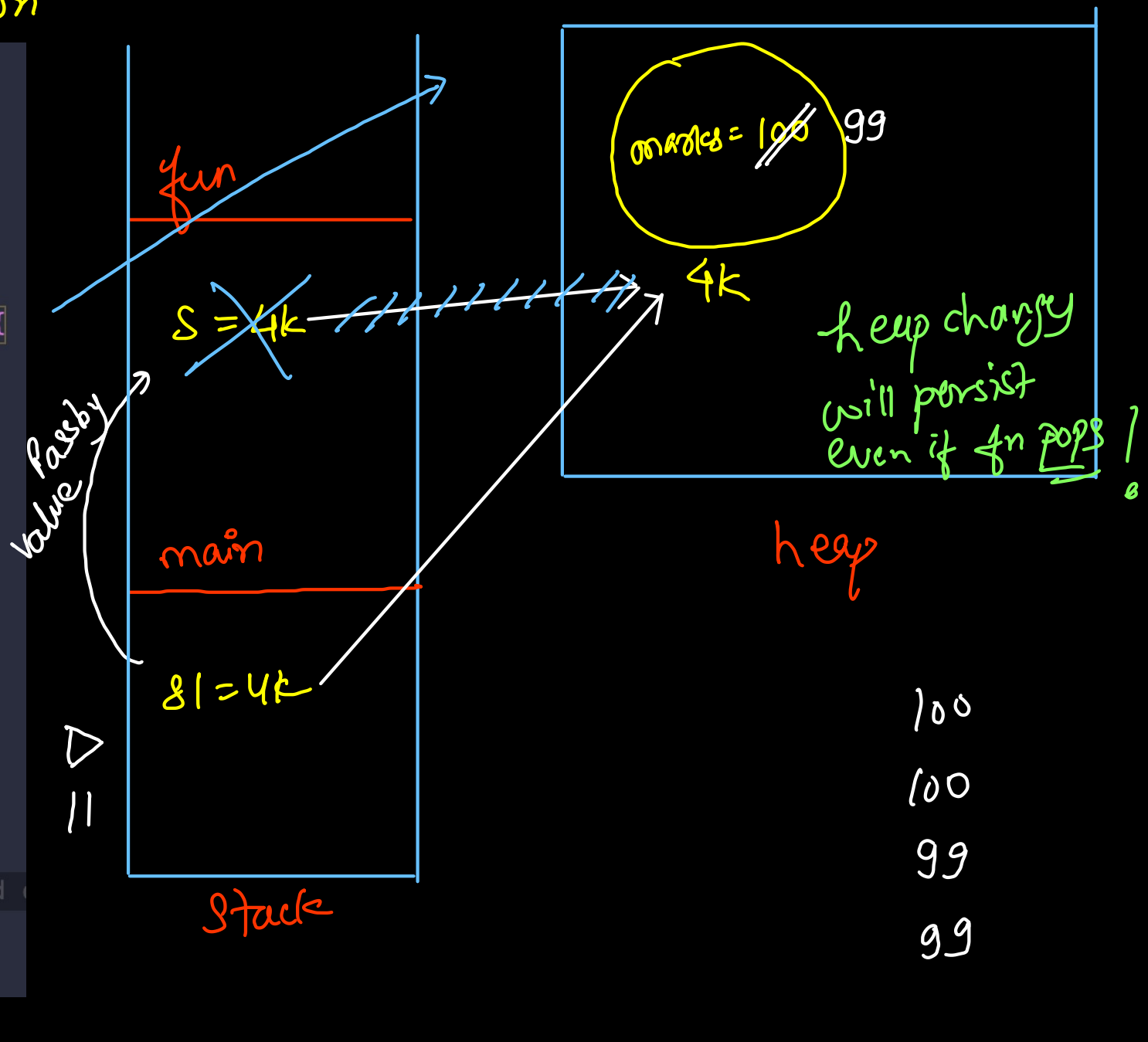
```java
Student s2 = new Student();

s2.name = "Babita";
s2.rollNo = 2;
s2.gender = 'F';
s2.marks = 30;

System.out.println(s2.name);
System.out.println(s2.rollNo);
System.out.println(s2.gender);
System.out.println(s2.marks);
```

}



Passing object to a function

```java
public static void fun(Student s) {
    System.out.println(s.marks);→100
    s.marks = 99;
    System.out.println(s.marks);→99
}

Run | Debug
public static void main(String[] args) {
    Student s1 = new Student();

    // Set the properties
    s1.name = "Archit";
    s1.rollNo = 1;
    s1.gender = 'M';
    s1.marks = 100;

    // Get the properites
    // System.out.println(s1.name);
    // System.out.println(s1.rollNo);
    // System.out.println(s1.gender);
    System.out.println(s1.marks);→100

    fun(s1);        You, now • Uncommitted

    System.out.println(s1.marks);→ 99
```



```java
public static void swap(Student x, Student y) {
    Student z = x;
    x = y;
    y = z;
}

Run | Debug
public static void main(String[] args) {
    Student s1 = new Student();
    s1.marks = 100;
    s1.gender = 'M';

    Student s2 = new Student();
    s2.marks = 75;
    s2.gender = 'F';

    System.out.println(s1.marks + ", " + s1.gender);
    System.out.println(s2.marks + ", " + s2.gender);

    swap(s1, s2);

    System.out.println(s1.marks + ", " + s1.gender);
    System.out.println(s2.marks + ", " + s2.gender);
}
```

x = 100, M ; y = 75, F

75, F ; 100, M

1. SWAP GAME (Single Choice) *

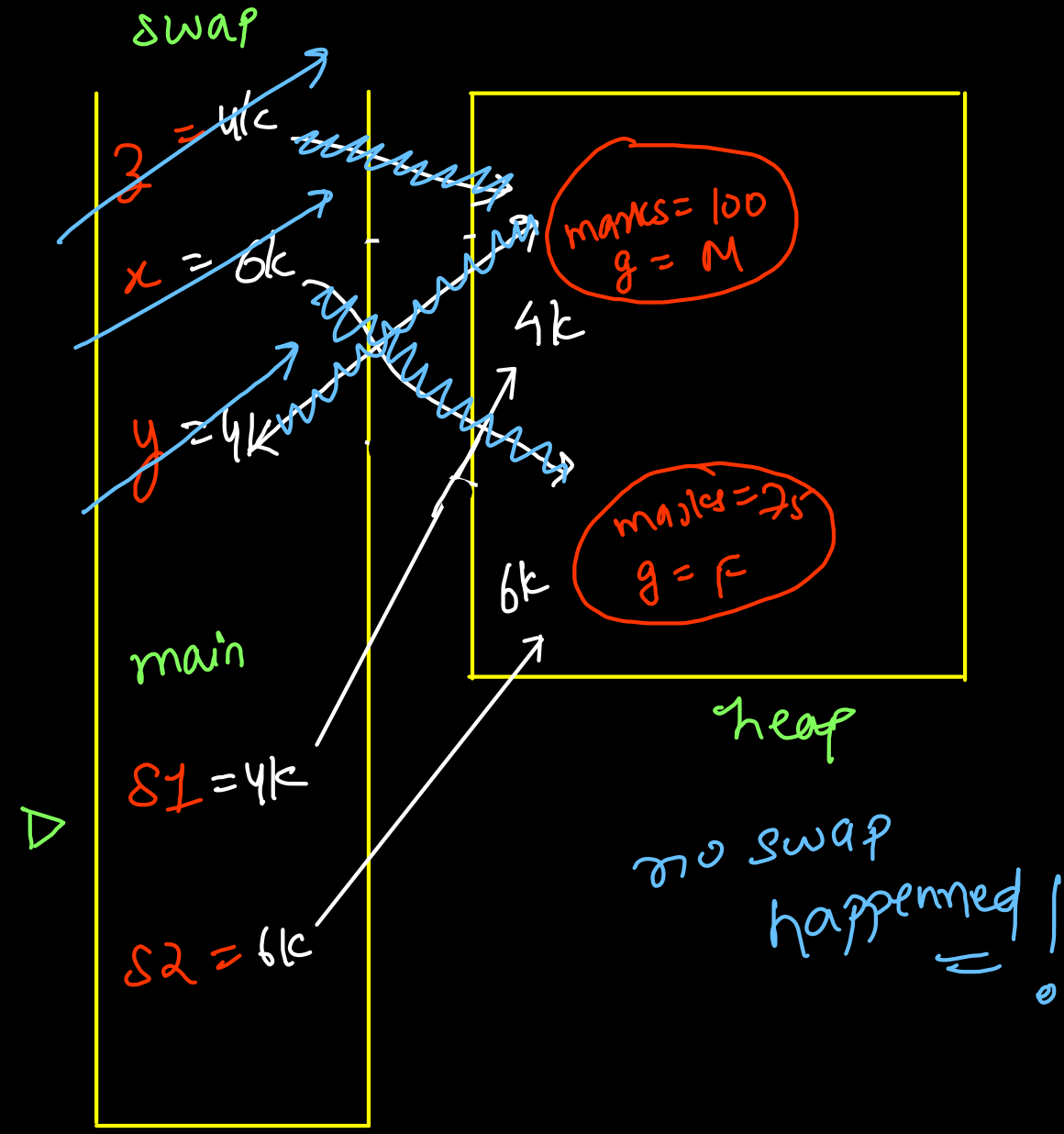
- 100 75 75 100
- ✓ 100 75 100 75
- 100 75 75 75
- 100 75 100 100

100 M

75 F

100 M

75 F



no swap happened !

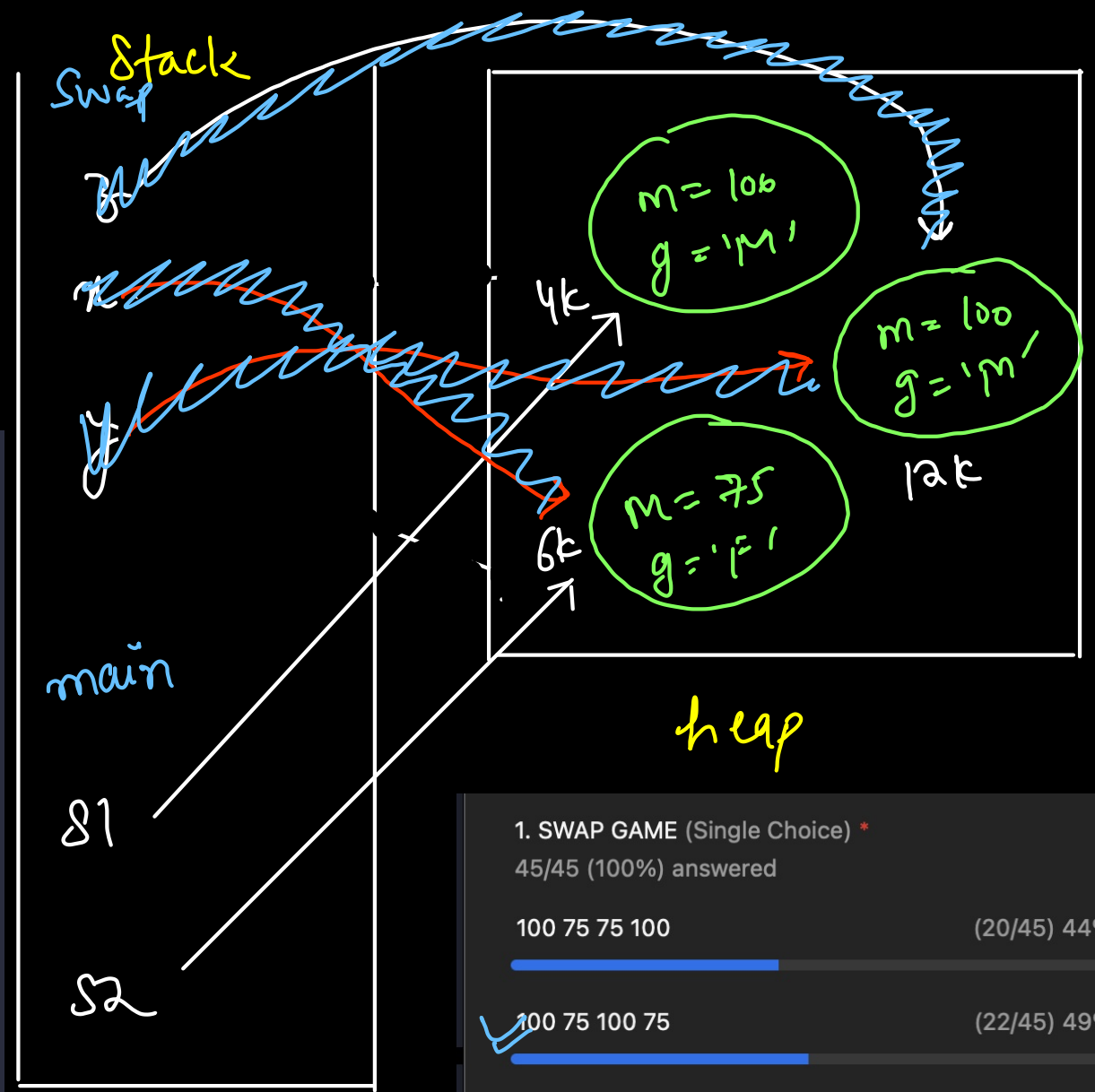
```java
public static void swap2(Student x, Student y) {
    Student z = new Student();
    z.marks = x.marks;
    z.gender = x.gender;

    x = y;
    y = z;
}
```

x.marks = 75
y.marks = 100

```java
public static void main(String[] args) {
    Student s1 = new Student();
    s1.marks = 100;
    s1.gender = 'M';

    Student s2 = new Student();
    s2.marks = 75;
    s2.gender = 'F';

    System.out.println(s1.marks + ", " + s1.gender);
    System.out.println(s2.marks + ", " + s2.gender);

    // swap1(s1, s2);
    swap2(s1, s2);

    System.out.println(s1.marks + ", " + s1.gender);
    System.out.println(s2.marks + ", " + s2.gender);
}
```

100 M
75 F

100 M
75 F

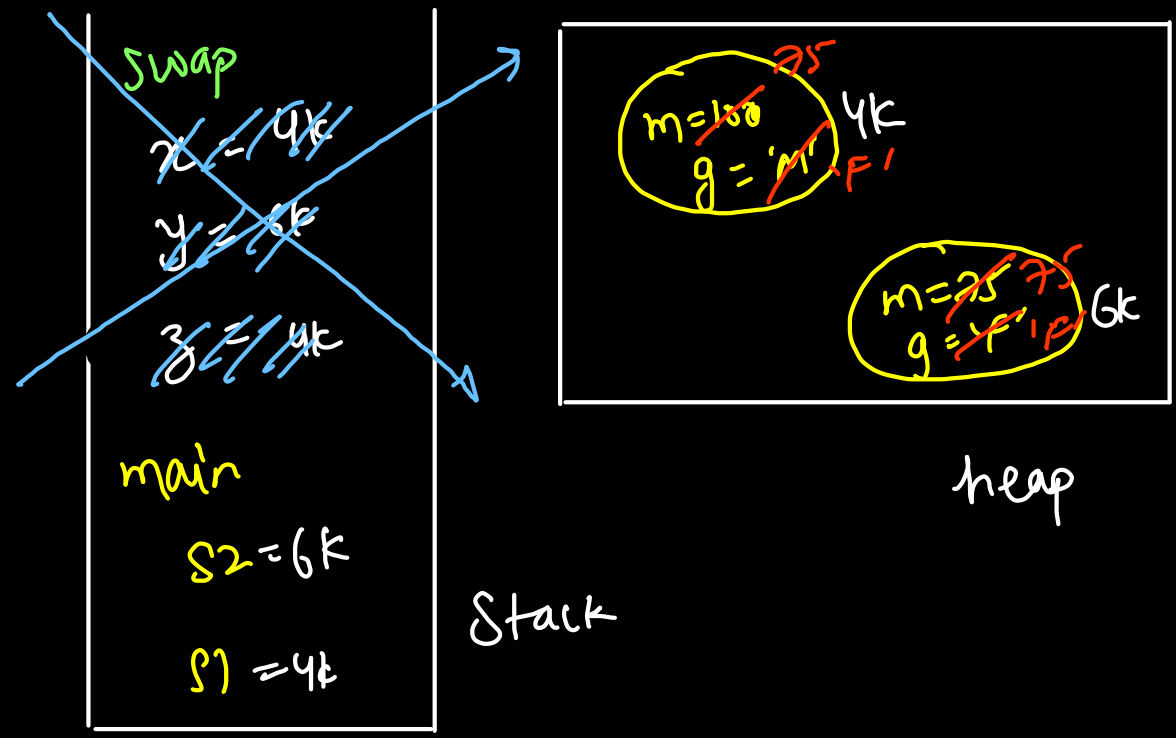


1. SWAP GAME (Single Choice) *
45/45 (100%) answered

100 75 75 100	(20/45) 44%
100 75 100 75	(22/45) 49%
100 75 75 75	(1/45) 2%
100 75 100 100	(2/45) 4%

```java
public static void swap3(Student x, Student y) {
    Student z = x;

    x.marks = y.marks;
    x.gender = y.gender;

    y.marks = z.marks;
    y.gender = z.gender;
}
```

```java
public static void main(String[] args) {
    Student s1 = new Student();
    s1.marks = 100;
    s1.gender = 'M';

    Student s2 = new Student();
    s2.marks = 75;
    s2.gender = 'F';

    System.out.println(s1.marks + ", " + s1.gender);
    System.out.println(s2.marks + ", " + s2.gender);

    // swap1(s1, s2);
    swap2(s1, s2);

    System.out.println(s1.marks + ", " + s1.gender);
    System.out.println(s2.marks + ", " + s2.gender);
}
```



1. SWAP GAME (Single Choice) *

42/42 (100%) answered

100 75 75 100	(21/42) 50%
100 75 100 75	(10/42) 24%
100 75 75 75	(9/42) 21%
100 75 100 100	(2/42) 5%

```java
public static void swap4(Student x, Student y) {
    Student z = new Student();
    z.marks = x.marks;
    z.gender = x.gender;

    x.marks = y.marks;
    x.gender = y.gender;

    y.marks = z.marks;
    y.gender = z.gender;
}
```

3rd copied object

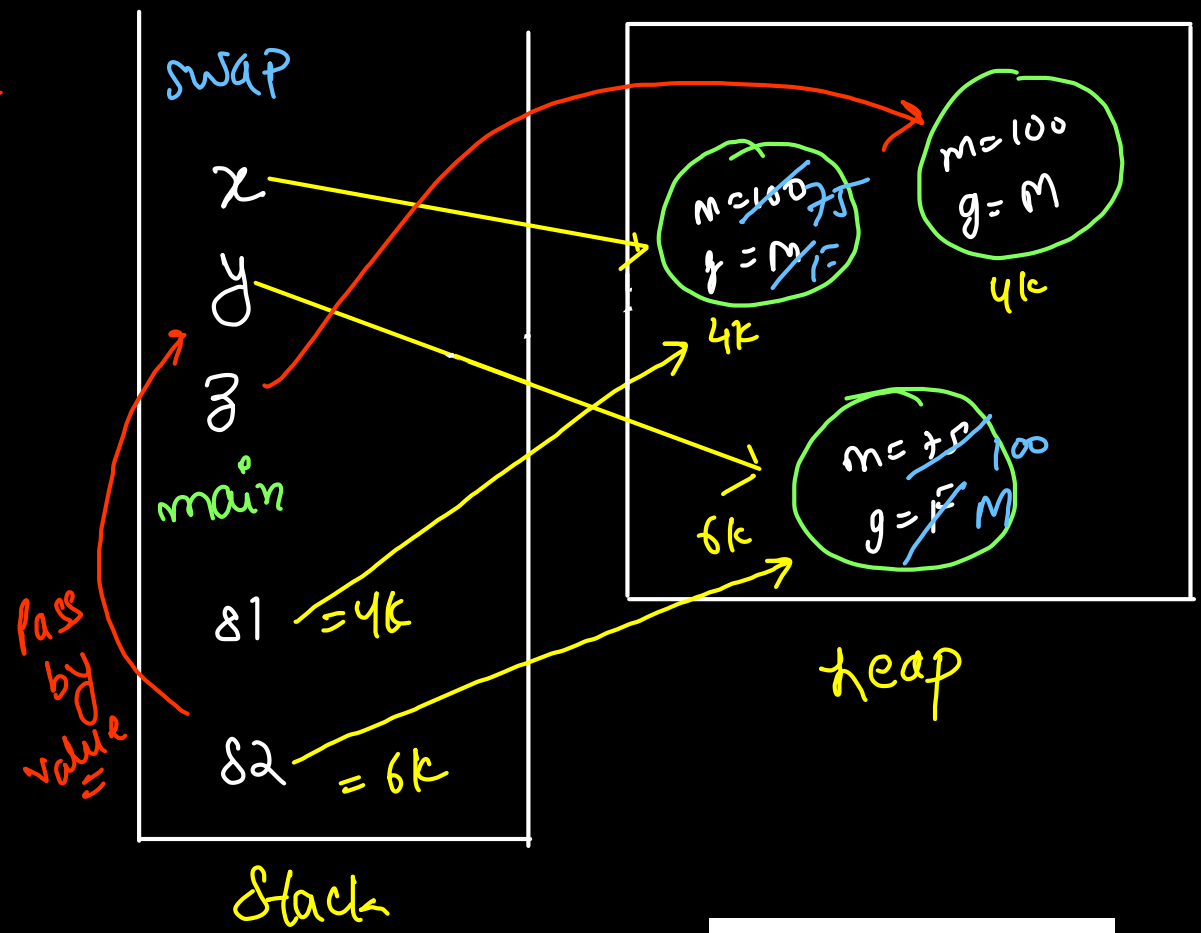
swap the data inside heap

```java
public static void main(String[] args) {
    Student s1 = new Student();
    s1.marks = 100;
    s1.gender = 'M';

    Student s2 = new Student();
    s2.marks = 75;
    s2.gender = 'F';

    System.out.println(s1.marks + ", " + s1.gender);
    System.out.println(s2.marks + ", " + s2.gender);

    // swap1(s1, s2);
    swap2(s1, s2);

    System.out.println(s1.marks + ", " + s1.gender);
    System.out.println(s2.marks + ", " + s2.gender);
}
```

100 → M

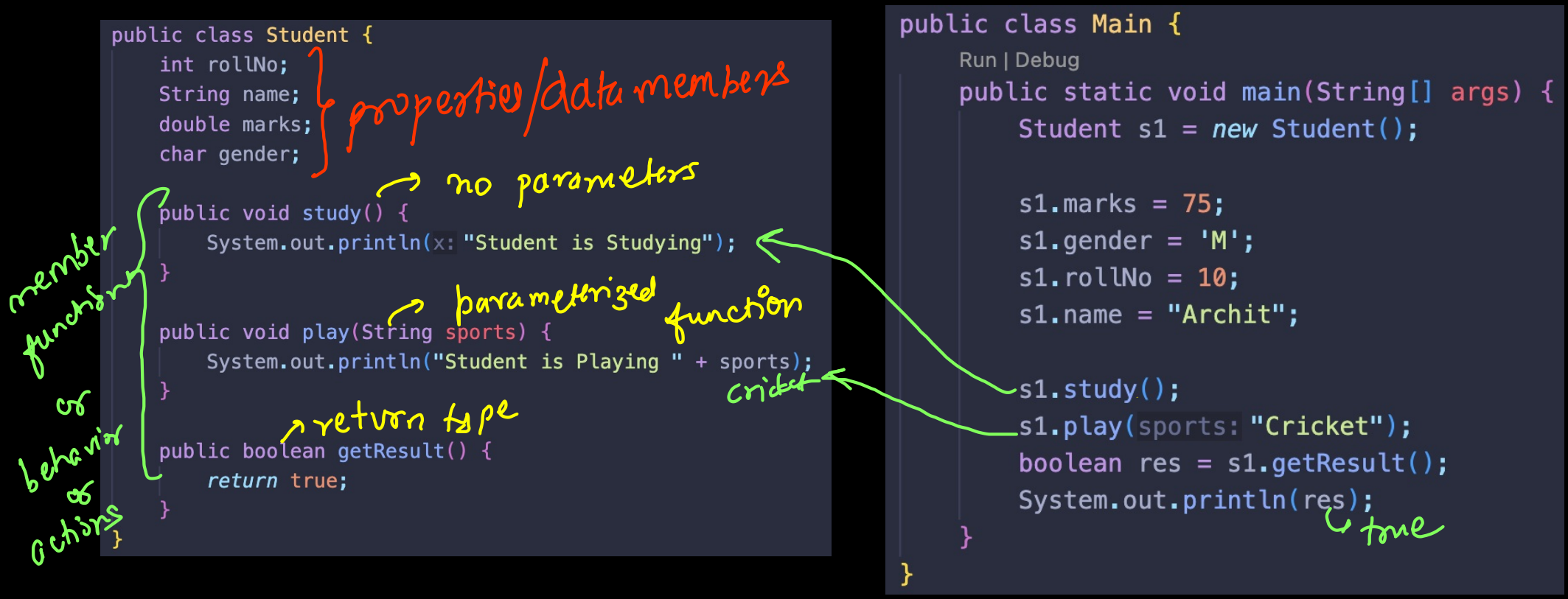
75 → F

100 → M



1. SWAP GAME (Single Choice) *

- 100 75 75 100
- 100 75 100 75
- 100 75 75 75
- 100 75 100 100


public class Student {
int rollNo;
String name;
double marks;
char gender;
properties/data members
no parameters
public void study() {
System.out.println(x: "Student is Studying");
}
parameterized function
public void play(String sports) {
System.out.println("Student is Playing " + sports);
}
cricket
return type
public boolean getResult() {
return true;
}
}
member functions or behaviour or actions
public class Main {
Run | Debug
public static void main(String[] args) {
Student s1 = new Student();
s1.marks = 75;
s1.gender = 'M';
s1.rollNo = 10;
s1.name = "Archit";
s1.study();
s1.play(sports: "Cricket");
boolean res = s1.getResult();
System.out.println(res);
true
}
}

```java
public class Student {
    int rollNo;
    String name;
    double marks;
    char gender;

    public void study() {
        System.out.println("Student is Studying");
    }

    public void play(String sports) {
        System.out.println("Student is Playing " +
        sports);
    }

    public boolean getResult() {
        if (marks >= 33)
            return true;
        return false;
    }

    public void setRollNo(int newRollNo) {
        rollNo = newRollNo;
    }
}
```

not secure

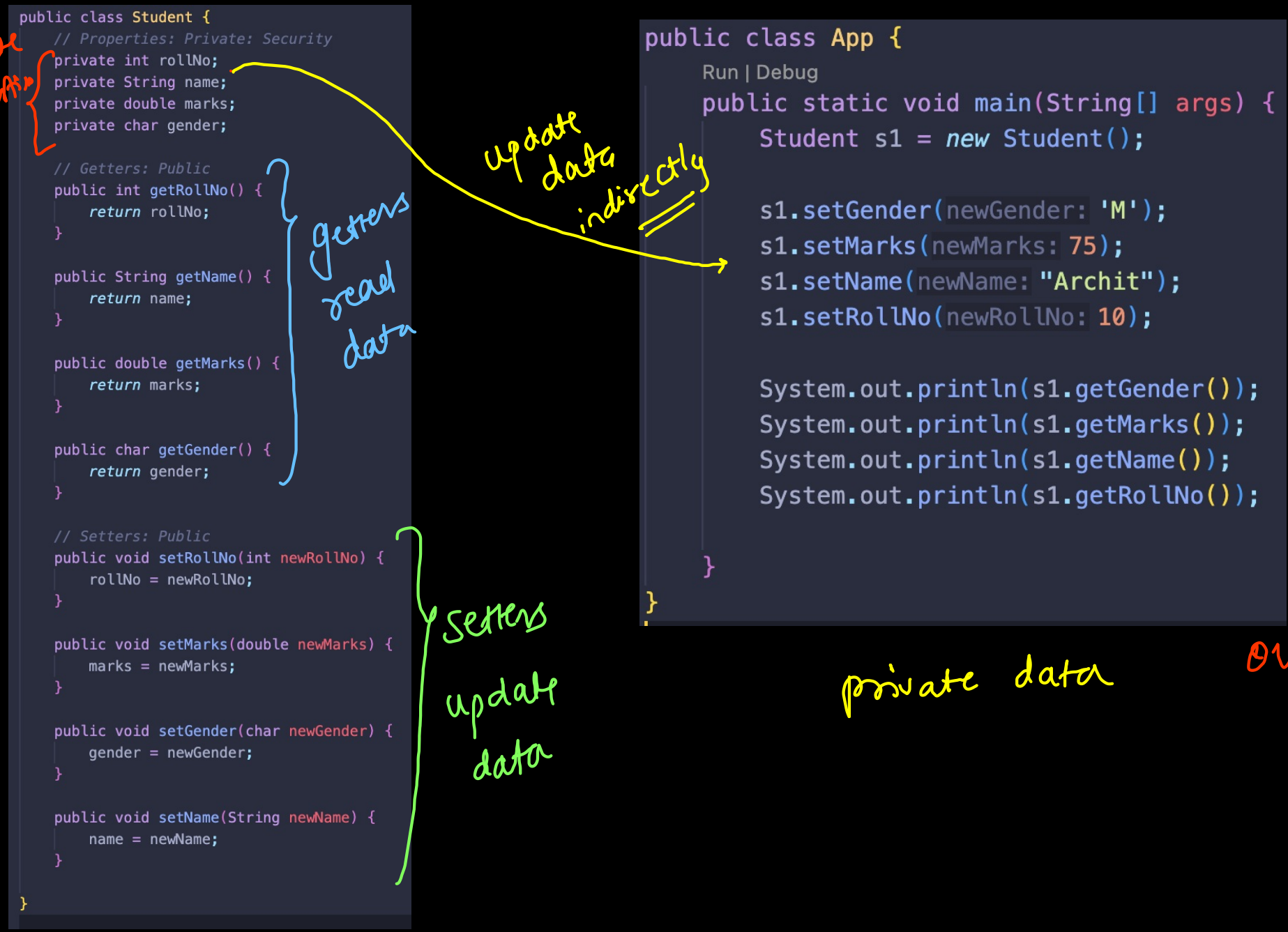
directly update

App

```java
public static void main(String[] args) {
    Student s1 = new Student();

    s1.marks = 20;
    s1.gender = 'M';
    s1.rollNo = 10;
    s1.name = "Archit";

    s1.study();
    s1.play(sports: "Cricket");
    boolean res = s1.getResult();
    System.out.println(res);

    s1.setRollNo(newRollNo: 12);
    System.out.println(s1.rollNo);
```

getter / fetch / read

setter / update / put


public class Student {
// Properties: Private: Security
private int rollNo;
private String name;
private double marks;
private char gender;
// Getters: Public
public int getRollNo() {
return rollNo;
}
public String getName() {
return name;
}
public double getMarks() {
return marks;
}
public char getGender() {
return gender;
}
// Setters: Public
public void setRollNo(int newRollNo) {
rollNo = newRollNo;
}
public void setMarks(double newMarks) {
marks = newMarks;
}
public void setGender(char newGender) {
gender = newGender;
}
public void setName(String newName) {
name = newName;
}
}
private properties
getters read data
setters update data
update data indirectly
public class App {
Run | Debug
public static void main(String[] args) {
Student s1 = new Student();
s1.setGender(newGender: 'M');
s1.setMarks(newMarks: 75);
s1.setName(newName: "Archit");
s1.setRollNo(newRollNo: 10);
System.out.println(s1.getGender());
System.out.println(s1.getMarks());
System.out.println(s1.getName());
System.out.println(s1.getRollNo());
}
}
private data
Outside access ✗
within class ✓

```java
public class Student {
    // Properties: Private: Security
    private int rollNo;
    private String name;
    private double marks;
    private char gender;

    // Getters: Public
    public int getRollNo() {
        return rollNo;
    }

    public String getName() {
        return name;
    }

    public double getMarks() {
        return marks;
    }

    public char getGender() {
        return gender;
    }

    // Setters: Public
    public void setRollNo(int newRollNo) {
        rollNo = newRollNo;
    }

    public void setMarks(double newMarks) {
        marks = newMarks;
    }

    public void setGender(char newGender) {
        gender = newGender;
    }

    public void setName(String newName) {
        name = newName;
    }

}
```

private properties

→ model { Data Layer } Backend

getters read data

setters update data

```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();

        s1.setGender(newGender: 'M');
        s1.setMarks(newMarks: 75);
        s1.setName(newName: "Archit");
        s1.setRollNo(newRollNo: 10);

        System.out.println(s1.getGender());
        System.out.println(s1.getMarks());
        System.out.println(s1.getName());
        System.out.println(s1.getRollNo());

    }
}
```

→ View { Application layer } Frontend

private data

Outside access ✗

within class ✓

```java
public void setMarks(double newMarks, String password) {
    if (password.equals(anObject: "1234jaijai") == true)
        marks = newMarks;
}
```

```java
public static void main(String[] args) {
    Student s1 = new Student();

    s1.setGender(newGender: 'M');
    s1.setMarks(newMarks: 70, password: "1234jaijai");
    s1.setMarks(newMarks: 75, password: "12345678");
    s1.setName(newName: "Archit");
    s1.setRollNo(newRollNo: 10);

    System.out.println(s1.getGender());
    System.out.println(s1.getMarks());
    System.out.println(s1.getName());
    System.out.println(s1.getRollNo());

}
```

```java
public class Student {
    // Properties: Private: Security
    private int rollNo = 1;
    private String name = "Anonymous";
    private double marks = 100.0;
    private char gender = 'N';
```

default values
{Dynamic Initialization}

```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();

        // s1.setGender('M');
        // s1.setMarks(70, "1234jaijai");
        // s1.setMarks(75, "12345678");
        // s1.setName("Archit");
        // s1.setRollNo(10);

        System.out.println(s1.getGender());
        System.out.println(s1.getMarks());
        System.out.println(s1.getName());
        System.out.println(s1.getRollNo());

    }
}
```

N
100.0
Anonymous
1

```java
public class Student {
    // Properties: Private: Security
    private int rollNo;
    private String name;
    private double marks;
    private char gender;
    String[] subjects;

    public void setDefaultProperties(int n) {
        rollNo = 1;
        name = "Anonymous";
        marks = 100.0;
        gender = 'N';
        subjects = new String[n];
    }
```

```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setDefaultProperties(n: 3);

        // s1.setGender('M');
        // s1.setMarks(70, "1234jaijai");
        // s1.setMarks(75, "12345678");
        // s1.setName("Archit");
        // s1.setRollNo(10);

        System.out.println(s1.getGender());
        System.out.println(s1.getMarks());
        System.out.println(s1.getName());
        System.out.println(s1.getRollNo());

        s1.subjects[0] = "C++";
        s1.subjects[1] = "Java";
        s1.subjects[2] = "Python";

        for (String s : s1.subjects) {
            System.out.print(s + " ");
        }
    }
}
```

```java
public class Student {
    private int rollNo;
    private String name;
    private double marks;
    private char gender;

    Student() {
        rollNo = 1;
        name = "Anonymous";
        marks = 100.0;
        gender = 'N';
    }
```

special function

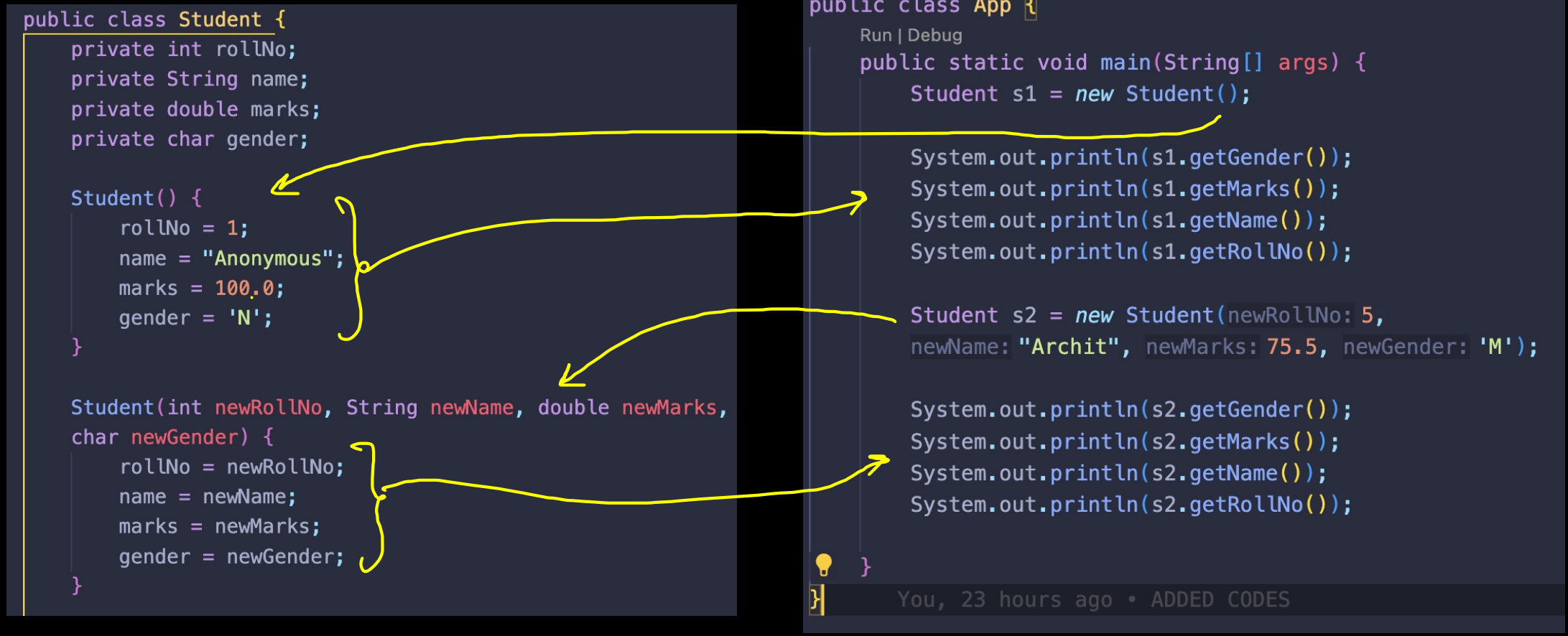
```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();

        System.out.println(s1.getGender());
        System.out.println(s1.getMarks());
        System.out.println(s1.getName());
        System.out.println(s1.getRollNo());
    }
}
```

constructor calling

String str = "archit"
↳ intern pool

String str = new String("archit")
↳ intern pool

object

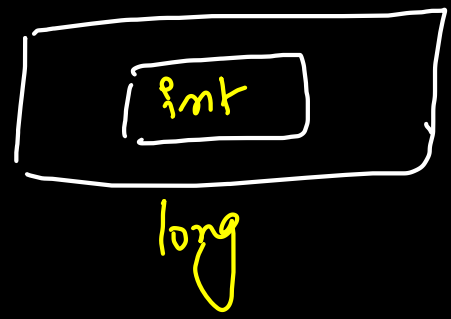

public class Student {
private int rollNo;
private String name;
private double marks;
private char gender;
Student() {
rollNo = 1;
name = "Anonymous";
marks = 100.0;
gender = 'N';
}
Student(int newRollNo, String newName, double newMarks,
char newGender) {
rollNo = newRollNo;
name = newName;
marks = newMarks;
gender = newGender;
}
public class App {
Run | Debug
public static void main(String[] args) {
Student s1 = new Student();
System.out.println(s1.getGender());
System.out.println(s1.getMarks());
System.out.println(s1.getName());
System.out.println(s1.getRollNo());
Student s2 = new Student(newRollNo: 5,
newName: "Archit", newMarks: 75.5, newGender: 'M');
System.out.println(s2.getGender());
System.out.println(s2.getMarks());
System.out.println(s2.getName());
System.out.println(s2.getRollNo());
}
}
You, 23 hours ago • ADDED CODES

If there is no explicit constructor, java compiler will provide an implicit constructor (with no body).

else if there is any explicit constructor, java compiler will not provide implicit constructor.

```
long phNo = 9319117889
  ↓
int ph = (int)phNo;
```

explicitly




```
int marks = 50;
  ↓
long lmarks = marks;
```

implicit

```java
// Non-Parameterized or Default Explicit Constructor
public Student() {
    rollNo = 1;
    name = "Anonymous";
    marks = 100.0;
    gender = 'N';
}

// Non-Parameterized or Default Implicit Empty Constructor
// Provided by Compiler
// public Student(){}

// // Parameterized Constructor
public Student(int newRollNo, String newName, double
newMarks, char newGender) {
    rollNo = newRollNo;
    name = newName;
    marks = newMarks;
    gender = newGender;
}
```

You, now • Uncommitted changes

Student s1 = new Student()

∝ not provided

Student s2 = new Student (5, "Archit", 75, 'M');

```java
// Non-Parameterized or Default Explicit Constructor
public Student() {
    rollNo = 1;
    name = "Anonymous";
    marks = 100.0;
    gender = 'N';
}

// Non-Parameterized or Default Implicit Empty Constructor
// Provided by Compiler
// public Student(){}

// // Parameterized Constructor
public Student(int newRollNo, String newName, double
newMarks, char newGender) {
    rollNo = newRollNo;
    name = newName;
    marks = newMarks;
    gender = newGender;
}
```

You, now • Uncommitted changes

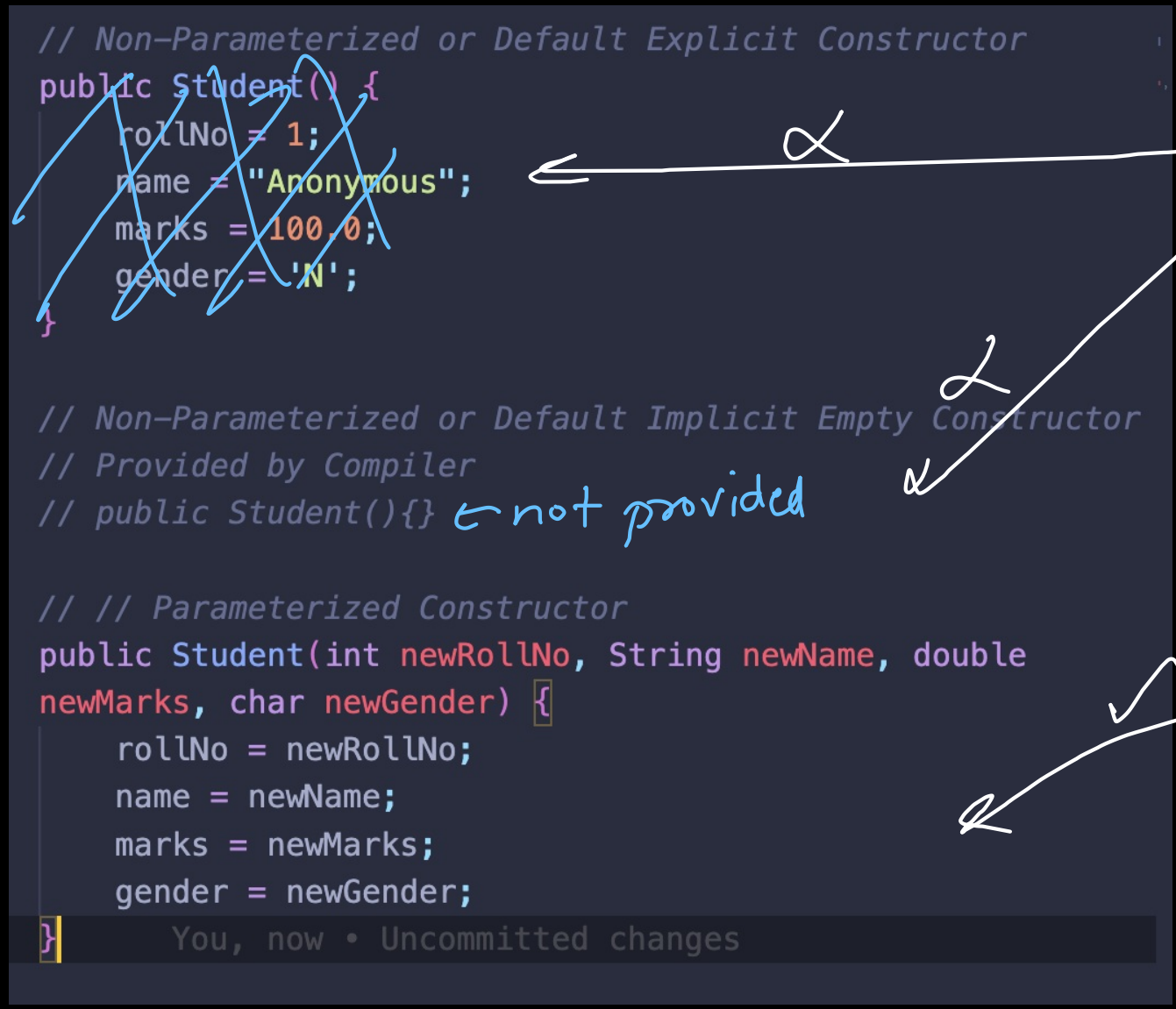
Student s1 = new Student()

∝ not provided

Student s2 = new Student (5, "Archit", 75, 'M');

not provided

not compile


// Non-Parameterized or Default Explicit Constructor
public Student() {
rollNo = 1;
name = "Anonymous";
marks = 100.0;
gender = 'N';
}
// Non-Parameterized or Default Implicit Empty Constructor
// Provided by Compiler
// public Student(){} ← not provided
// // Parameterized Constructor
public Student(int newRollNo, String newName, double
newMarks, char newGender) {
rollNo = newRollNo;
name = newName;
marks = newMarks;
gender = newGender;
}
You, now • Uncommitted changes
new Student();
not Compile
new Student(----);

no explict constructor

```java
// Non-Parameterized or Default Explicit Constructor
public Student() {
    rollNo = 1;
    name = "Anonymous";
    marks = 100.0;
    gender = 'N';
}

// Non-Parameterized or Default Implicit Empty Constructor
// Provided by Compiler
// public Student(){}

// // Parameterized Constructor
public Student(int newRollNo, String newName, double
newMarks, char newGender) {
    rollNo = newRollNo;
    name = newName;
    marks = newMarks;
    gender = newGender;
}     You, now • Uncommitted changes
```

this will be provided

rollno=0, name=null, gender='\0', marks=0.0

new Student();

new Student(----);

not compile

Types of Constructor

1. Default implicit Constructor
2. Default explicit Constructor
3. Parameterized constructor
4. Copy constructor

Overloading of Constructors

1. Constructor name (for name) should be same
2. Any one must be true :-
   - Exact Match:
     - (2.1) Number of parameters are different
     - or
     - (2.2) Datatypes are different (order/types)
3. Try for implicit conversions
4. otherwise ambiguous

```java
public Student(int newRollNo) {
    rollNo = newRollNo;
}

public Student(int newRollNo, String newName) {
    rollNo = newRollNo;
    name = newName;
}

public Student(char newGender, double newMarks) {
    marks = newMarks;
    gender = newGender;
}

public Student(String newName, int newRollNo) {

}

public Student(String newName, double newMarks) {

}

public Student(String newName, float newMarks) {

}
```

```java
public Student(int newRollNo) {
    rollNo = newRollNo;
    System.out.println(x: "Integer Constructor");
}

public Student(short newRollNo) {
    rollNo = newRollNo;
    System.out.println(x: "Short Constructor");
}
```

```java
Student s3 = new Student(newRollNo: 5); // integer
// System.out.println(s3.getRollNo());

Student s4 = new Student((short) 5); // short
```

```java
public Student(int newMarks) {
    marks = newMarks;
}
```

```java
Student s3 = new Student(newMarks: 5); // integer
System.out.println(s3.getRollNo());

char ch = 'A';
Student s4 = new Student(ch);
System.out.println(s4.getRollNo());

long phoneNo = 9319117889l;
Student s5 = new Student(phoneNo);
System.out.println(s5.getRollNo());
```

4 bytes → 4 bytes

Exact match

'A' → 2 bytes

2 bytes → 4 bytes

↳ implicit typecasting

implicit conversion

65

8 bytes

8 bytes → 4 bytes

Compilation error

loss of conversion (no implicit conversion)

```java
Student s3 = new Student(newMarks: 5); // integer
System.out.println(s3.getMarks());

char ch = 'A';
Student s4 = new Student(ch);
System.out.println(s4.getMarks());

long phoneNo = 93191178891;
// Student s5 = new Student(phoneNo);
Student s5 = new Student((int) phoneNo);
System.out.println(s5.getMarks());
```

```java
public Student(Student other) {
    rollNo = other.rollNo;
    other.rollNo = -1;
    // rollNo = other.getRollNo();
}

public Student(){
    rollNo = 1;
}
```

allowed?

```java
public static void main(String[] args){
    Student ram = new Student();

    // System.out.println(s2.getRollNo());

    // Employee other = new Employee();
    Student shyam = new Student(ram);

}
```

S1 & S2 are objects of same class

If we are creating S2 object & passing S1 in it, we should not be allowed to access S1's private properties in S2.

The reason is that access modifiers work on **class level** and not on **object level**. Two different objects of the same class can access each others private members.

Encapsulation will be violated

That's strange, why doesn't it work on object level?

This is a good question. Smalltalk, for instance, does in fact enforce access control on object level.

Here are three reasons:

1

Efficiency

Enforcing it on class level allows the compiler to catch all violations in compiletime. Had it been on object level on the other hand, the compiler would have had to do one of the following things: - Guard every field access with something like `if (obj != this) throw IllegalAccessException();` which would slow down execution, or - defensively reject anything but `this.i` (like `obj.i`) since it's in general impossible to tell if `obj == this` in all executions. This would have been annoying in cases it was clear to the programmer that `obj` did in fact equal `this`.

2

Locality

The field you're accessing (even if it's of a different object in runtime) is declared in the file you're editing. If you mess something up by modifying another objects state, the fault is in the file you have in front of you. There's no spooky action at a distance.

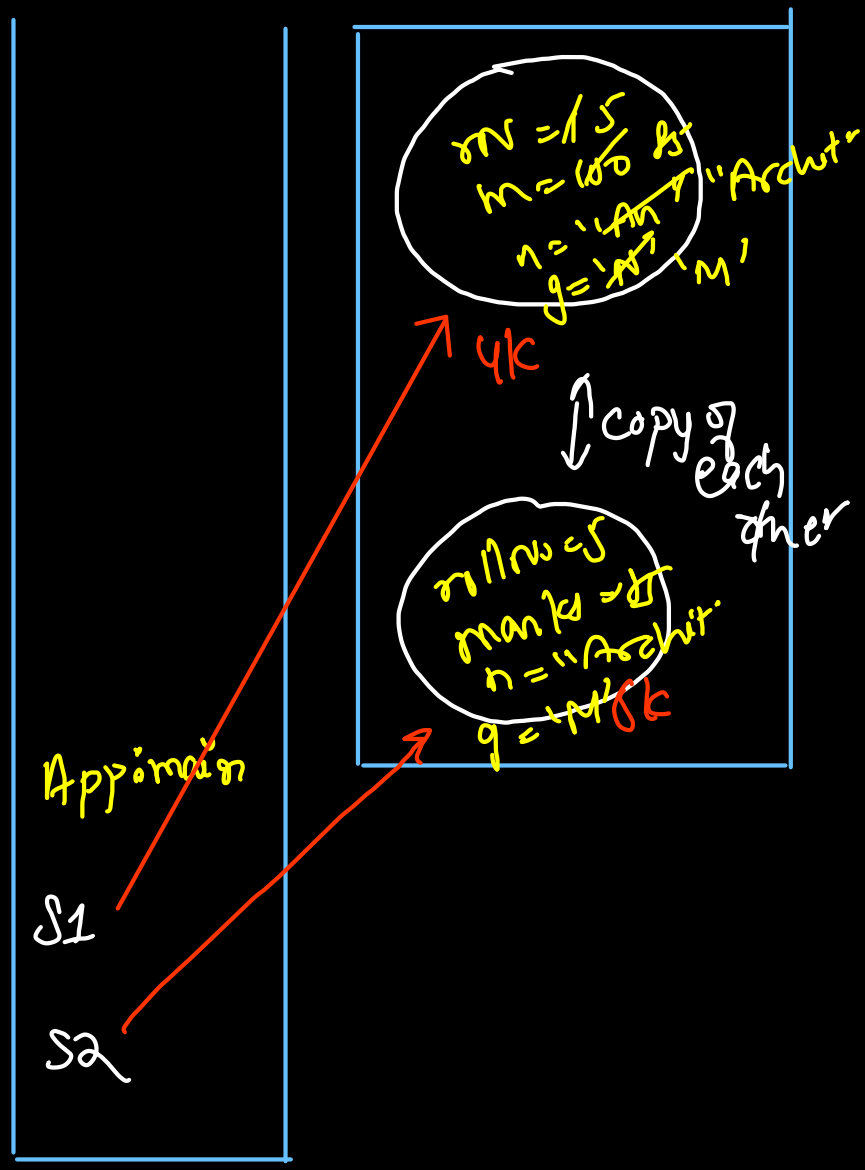
3

Convenience

Some methods such as `equals`, `clone` and copy constructors would be tricky to write without giving up encapsulation.

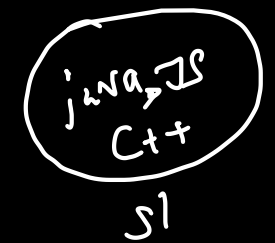
```java
public class Student {
    private int rollNo;
    private String name;
    private double marks;
    private char gender;

    // Getters: Public
    public int getRollNo() {…

    public String getName() {…

    public double getMarks() {…

    public char getGender() {…

    // Setters: Public
    public void setRollNo(int newRollNo) {…

    public void setMarks(double newMarks) {…

    public void setGender(char newGender) {…

    public void setName(String newName) {…

    // Copy Constructor
    public Student(Student other) {
        rollNo = other.rollNo;
        marks = other.marks;
        name = other.name;
        gender = other.gender;
    }

    public Student() {
        rollNo = 1;
        marks = 100;
        name = "Anonymous";
        gender = 'N';      You, 2 minutes ago
    }
}
```

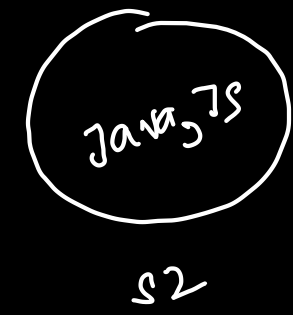
```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setRollNo(newRollNo: 5);
        s1.setMarks(newMarks: 85);
        s1.setName(newName: "Archit");
        s1.setGender(newGender: 'M');

        Student s2 = new Student(s1);
        // s2.setRollNo(s1.getRollNo());

        System.out.println(s2.getRollNo());
        System.out.println(s2.getMarks());
        System.out.println(s2.getName());
        System.out.println(s2.getGender());

    }
}
```



```java
public class Student {
    private int rollNo;
    private String name;
    private double marks;
    private char gender;
    ArrayList<String> subjects;

    // Copy Constructor
    public Student(Student other) {
        rollNo = other.rollNo;
        marks = other.marks;
        name = other.name;
        gender = other.gender;
        subjects = other.subjects;
    }

    public Student() {
        rollNo = 1;
        marks = 100;
        name = "Anonymous";
        gender = 'N';
        subjects = new ArrayList<>();
    }
```

```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setRollNo(newRollNo: 5);
        s1.setMarks(newMarks: 85);
        s1.setName(newName: "Archit");
        s1.setGender(newGender: 'M');

        System.out.println(s1.subjects);
        s1.subjects.add(e: "Java");
        s1.subjects.add(e: "Javascript");
        System.out.println(s1.subjects);

        Student s2 = new Student(s1);
        System.out.println(s2.subjects);

        s1.subjects.add(e: "C++");
        System.out.println(s1.subjects);
        System.out.println(s2.subjects);
    }
}
```





```java
public class Student {
    private int rollNo;
    private String name;
    private double marks;
    private char gender;
    ArrayList<String> subjects;

    // Copy Constructor
    public Student(Student other) {
        rollNo = other.rollNo;
        marks = other.marks;
        name = other.name;
        gender = other.gender;
        subjects = other.subjects;
    }

    public Student() {
        rollNo = 1;
        marks = 100;
        name = "Anonymous";
        gender = 'N';
        subjects = new ArrayList<>();
    }
```

Shallow copy

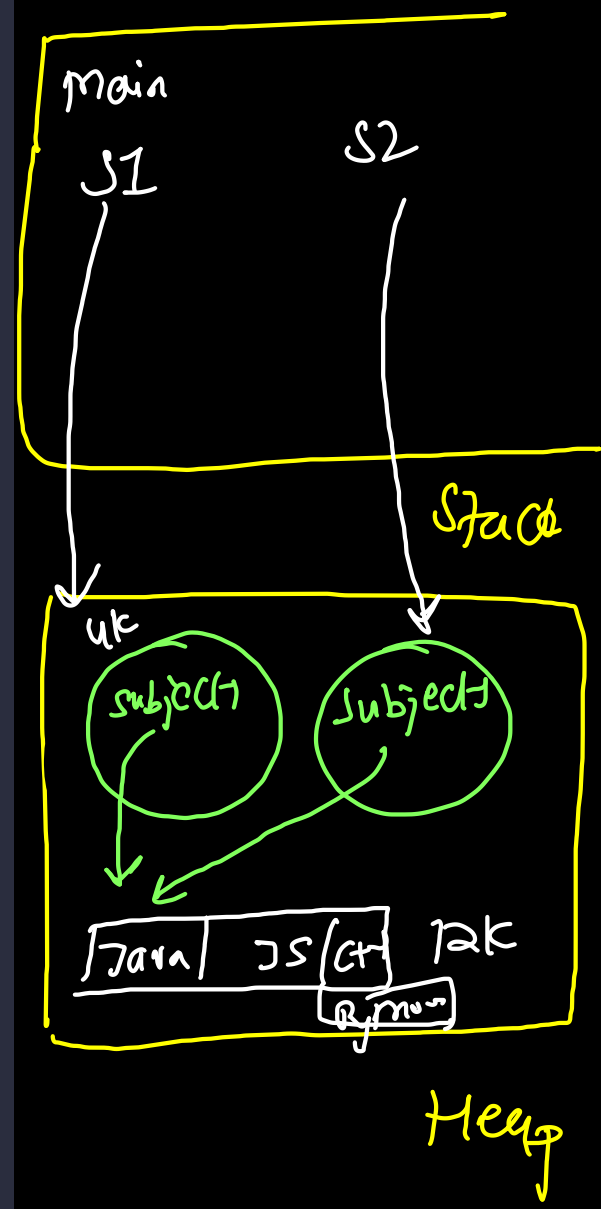
Reference Variable

Only copied

```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setRollNo(newRollNo: 5);
        s1.setMarks(newMarks: 85);
        s1.setName(newName: "Archit");
        s1.setGender(newGender: 'M');

        System.out.println(s1.subjects);
        s1.subjects.add(e: "Java");
        s1.subjects.add(e: "Javascript");
        System.out.println(s1.subjects);

        Student s2 = new Student(s1);
        System.out.println(s2.subjects);

        s1.subjects.add(e: "C++");
        System.out.println(s1.subjects);
        System.out.println(s2.subjects);

        s2.subjects.add(e: "Python");
        System.out.println(s1.subjects);
        System.out.println(s2.subjects);

    }
}
```

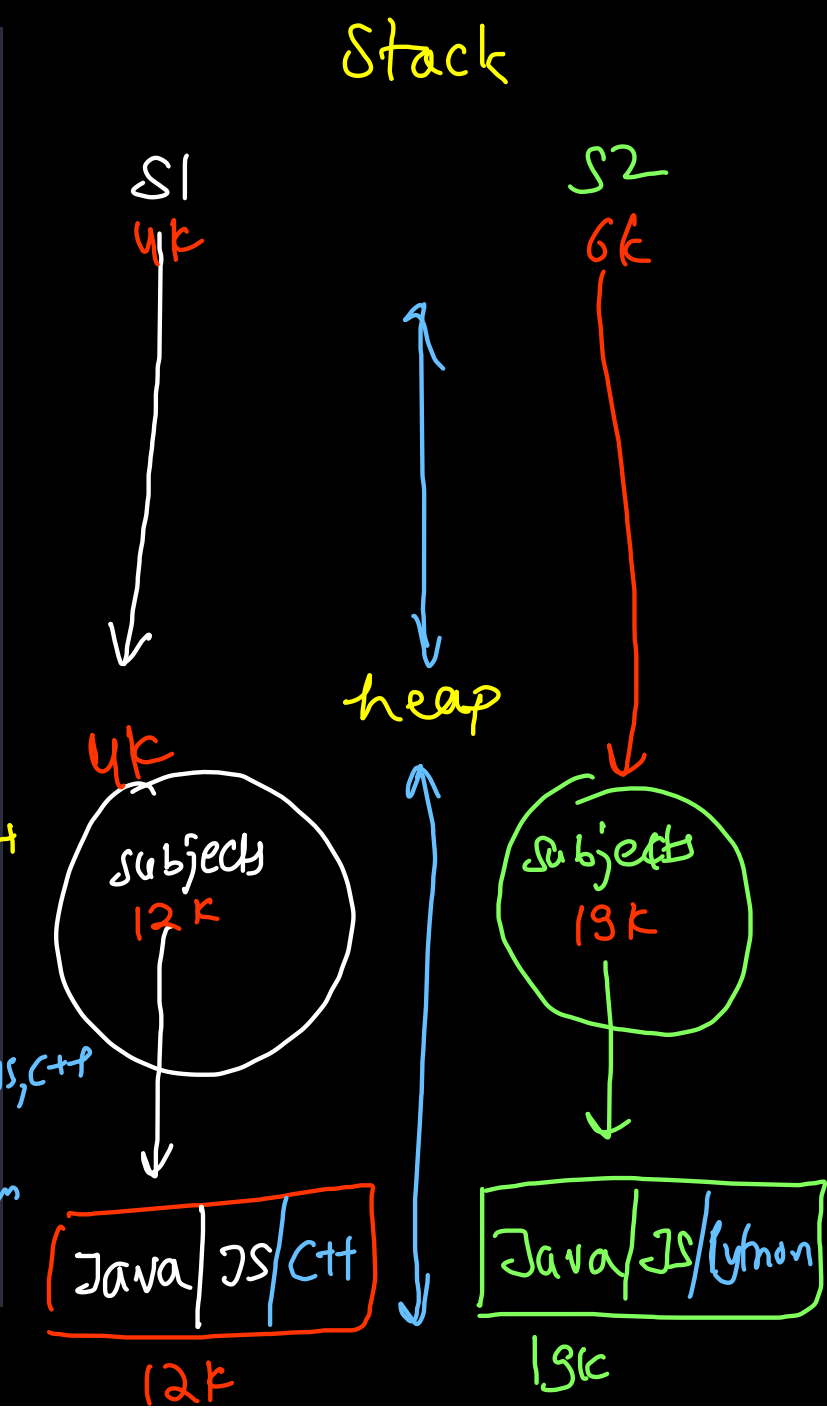
[]

Java, JS



```java
public class Student {
    private int rollNo;
    private String name;
    private double marks;
    private char gender;
    ArrayList<String> subjects;

    // Copy Constructor: Shallow Copy
    // public Student(Student other) {
    // rollNo = other.rollNo;
    // marks = other.marks;
    // name = other.name;
    // gender = other.gender;      You, 1 minute
    // subjects = other.subjects;
    // }

    // Copy Constructor: Deep Copy
    public Student(Student other) {
        rollNo = other.rollNo;
        marks = other.marks;
        name = other.name;
        gender = other.gender;

        subjects = new ArrayList<>();
        for (String subject : other.subjects) {
            subjects.add(subject);
        }
    }

    public Student() {
        rollNo = 1;
        marks = 100;
        name = "Anonymous";
        gender = 'N';
        subjects = new ArrayList<>();
    }
```

```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setRollNo(newRollNo: 5);
        s1.setMarks(newMarks: 85);
        s1.setName(newName: "Archit");
        s1.setGender(newGender: 'M');

        System.out.println(s1.subjects);
        s1.subjects.add(e: "Java");
        s1.subjects.add(e: "Javascript");
        System.out.println(s1.subjects);

        Student s2 = new Student(s1);
        System.out.println(s2.subjects);

        s1.subjects.add(e: "C++");
        System.out.println(s1.subjects);
        System.out.println(s2.subjects);

        s2.subjects.add(e: "Python");
        System.out.println(s1.subjects);
        System.out.println(s2.subjects);

    }
}
```



```java
public class Student {
    private int rollNo;
    private String name;
    private double marks;
    private char gender;

    public Student(int rollNo, String name, double
    marks, char gender){
        rollNo = rollNo;
        name = name;
        marks = marks;
        gender = gender;
    }
```

parameters

```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int rollNo = scn.nextInt();
        String name = scn.next();
        double marks = scn.nextDouble();
        char gender = scn.next().charAt(index: 0);

        Student s1 = new Student(rollNo, name, marks,
        gender);

        System.out.println(s1.getGender());
        System.out.println(s1.getMarks());
        System.out.println(s1.getRollNo());
        System.out.println(s1.getName());
    }
}
```

You, 1 minute ago • Uncommitted changes

```java
public class Student {
    private int rollNo;
    private String name;
    private double marks;
    private char gender;

    public Student(int rollNo, String name, double
    marks, char gender) {
        // Parameter = Parameter
        // rollNo = rollNo;
        // name = name;
        // marks = marks;
        // gender = gender;

        // Property = Parameter
        this.rollNo = rollNo;
        this.name = name;
        this.gender = gender;
        this.marks = marks;
```

to resolve name space conflict

```java
You, 14 minutes ago | 1 author (You)
public class App {
    Run | Debug
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int rollNo = scn.nextInt();
        String name = scn.next();
        double marks = scn.nextDouble();
        char gender = scn.next().charAt(index: 0);

        Student s1 = new Student(rollNo, name, marks,
        gender);

        System.out.println(s1.getGender());
        System.out.println(s1.getMarks());
        System.out.println(s1.getRollNo());
        System.out.println(s1.getName());
    }
}
```

You, 14 minutes ago • Uncommitted changes

Application of this keyword

Memory mapping using this

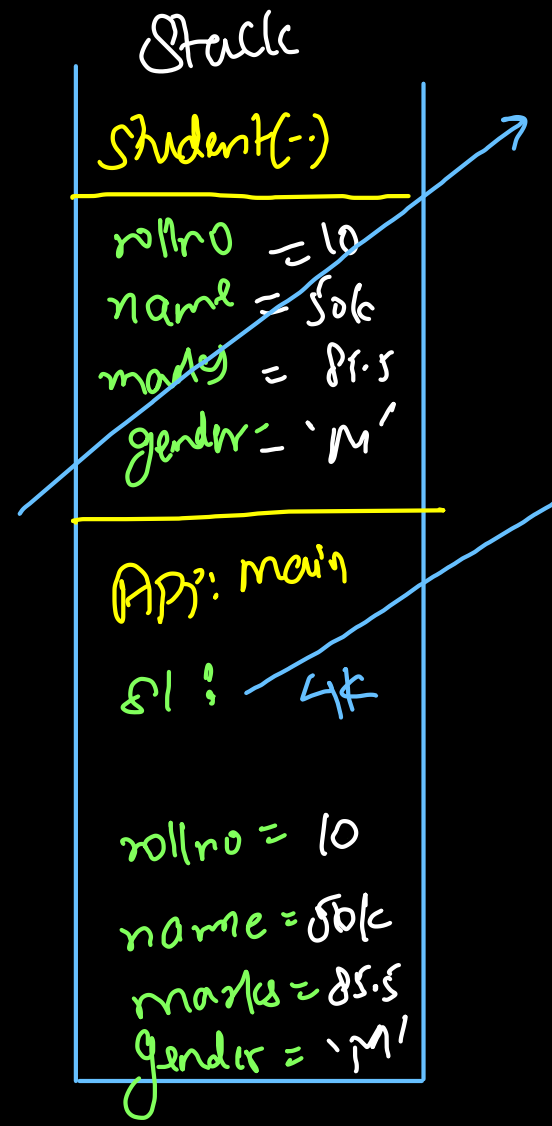
↳ refers to the current object

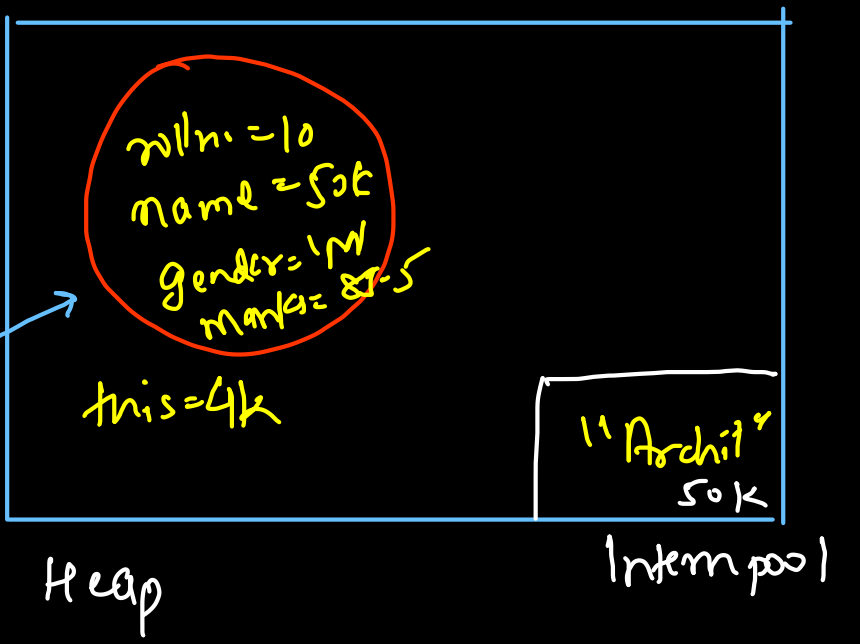
```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int rollNo = scn.nextInt();
        String name = scn.next();
        double marks = scn.nextDouble();
        char gender = scn.next().charAt(index: 0);

        Student s1 = new Student(rollNo, name, marks,
        gender);

        System.out.println(s1.getGender());
        System.out.println(s1.getMarks());
        System.out.println(s1.getRollNo());
        System.out.println(s1.getName());
    }
}
```

```java
public class Student {
    private int rollNo;
    private String name;
    private double marks;
    private char gender;

    public Student(int rollNo, String name, double
    marks, char gender) {
        // Property = Parameter
        this.rollNo = rollNo;
        this.name = name;
        this.gender = gender;
        this.marks = marks;
    }
}
```

this

4k

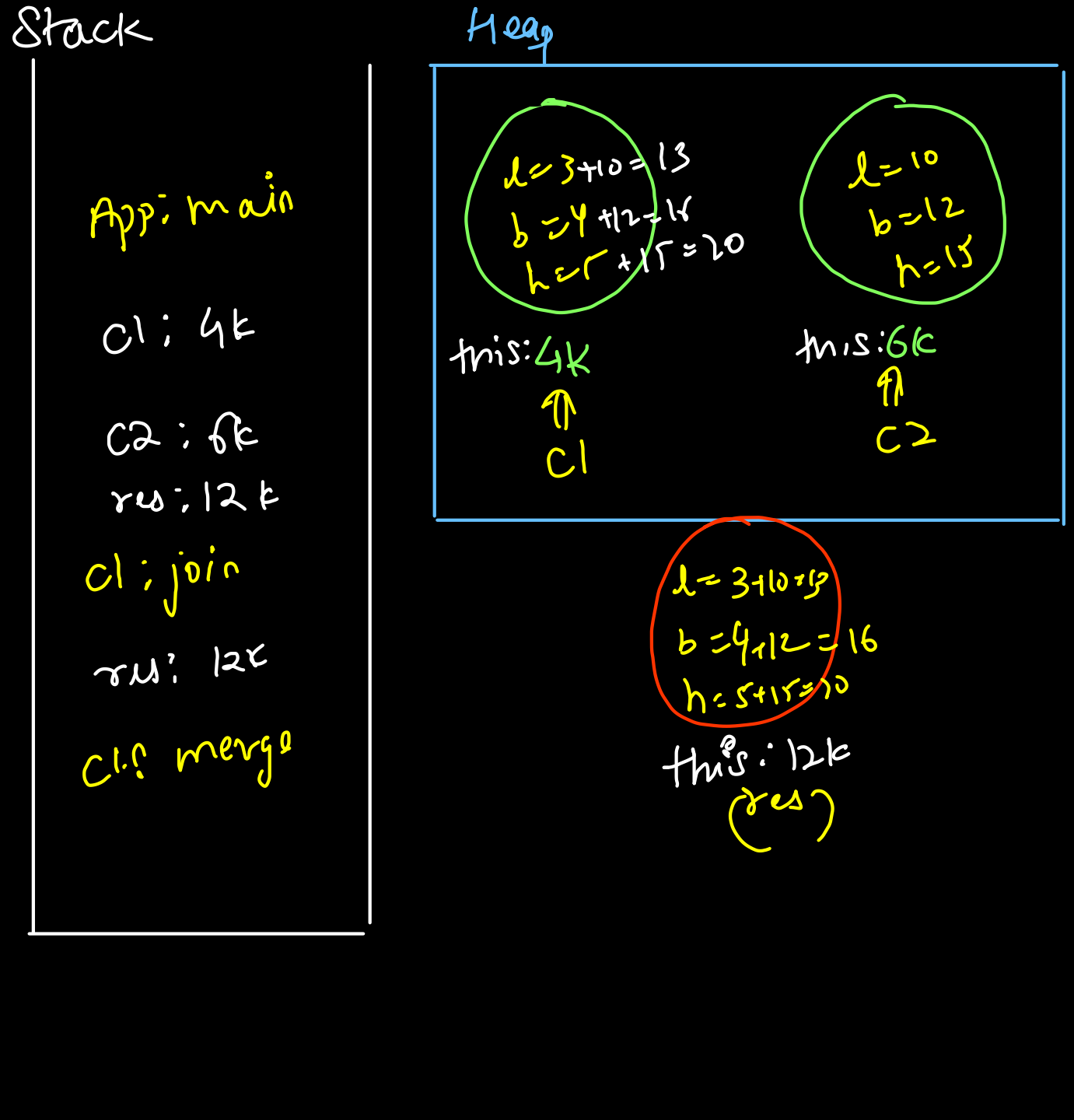




```java
public class Cuboid {
    int length;
    int breadth;
    int height;

    Cuboid(int length, int breadth, int height) {
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }

    Cuboid() {

    }

    // Not Modifying the original (c1 & c2) cuboids
    Cuboid join(Cuboid other) {
        Cuboid res = new Cuboid();
        res.length = this.length + other.length;
        res.breadth = this.breadth + other.breadth;
        res.height = this.height + other.height;
        return res;
    }

    // Modifying the original (c1) cuboid
    Cuboid merge(Cuboid other) {
        this.length += other.length;
        this.breadth += other.breadth;
        this.height += other.height;
        return this; // Return This
    }
}
```

this = c1
other = c2
res =

```java
class App {
    Run | Debug
    public static void main(String[] args) {
        Cuboid c1 = new Cuboid(length: 3, breadth: 4, height: 5);
        Cuboid c2 = new Cuboid(length: 10, breadth: 12, height: 15);

        Cuboid res = c1.join(c2);
        System.out.println(res.length + " " + res.breadth + " " + res.height);

        c1 = c1.merge(c2);
        System.out.println(c1.length + " " + c1.breadth + " " + c1.height);

    }
}
```



```java
public class Cuboid {      You, 17 minutes ago • Ad
    int length;
    int breadth;
    int height;

    // Parameterized Constructor: Cuboid
    Cuboid(int length, int breadth, int height) {
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }

    // Default Explicit Constructor: Unit Cube
    Cuboid() {
        this.length = 1;
        this.breadth = 1;
        this.height = 1;
    }

    // Parameterized Constructor: Cube
    Cuboid(int side){
        this.length = side;
        this.breadth = side;
        this.height = side;
    }

    // Parameterized Constructor: Rectangle
    Cuboid(int length, int breadth){
        this.length = length;
        this.breadth = breadth;
        this.height = 0;
    }
```

cuboid(1);

cuboid(side,side,side)

cuboid(l,b,0);

Unit cube = cube(side=1)

Cube(side) = cuboid(side, side, side)

Rectangle(l,b) = cuboid(l, b, 0)

```java
public class Cuboid {
    int length;
    int breadth;
    int height;

    // Parameterized Constructor: Cuboid
A   Cuboid(int length, int breadth, int height) {
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }

    // Default Explicit Constructor: Unit Cube
B   Cuboid() {
        this(side: 1);
        // this.length = 1;
        // this.breadth = 1;
        // this.height = 1;
    }

    // Parameterized Constructor: Cube
C   Cuboid(int side) {
        this(side, side, side);
        // this.length = side;
        // this.breadth = side;
        // this.height = side;
    }

    // Parameterized Constructor: Rectangle
D   Cuboid(int length, int breadth) {
        this(length, breadth, height: 0);
        // this.length = length;
        // this.breadth = breadth;
        // this.height = 0;
    }
```

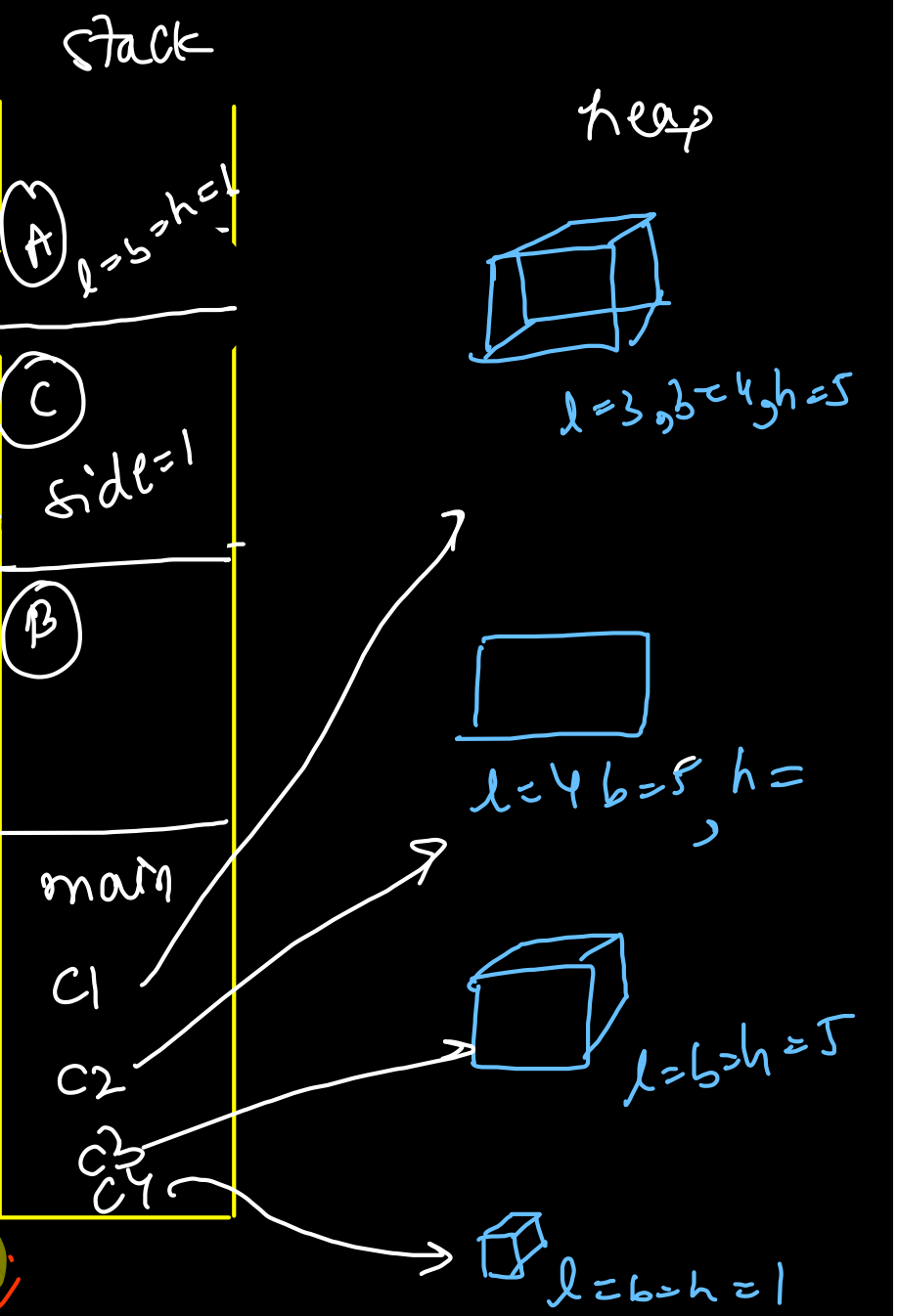
Constructor Chaining !!

Cuboid c1 = new Cuboid(3,4,5)
↳ A

Cuboid c2 = new Cuboid(4,5);
↳ D → A

Cuboid c3 = new Cuboid(5);
↳ C → A

Cuboid c4 = new Cuboid();
↳ B → C → A



Applications of this

(1) Accessing properties

(2) Accessing functions

(3) Constructor chaining

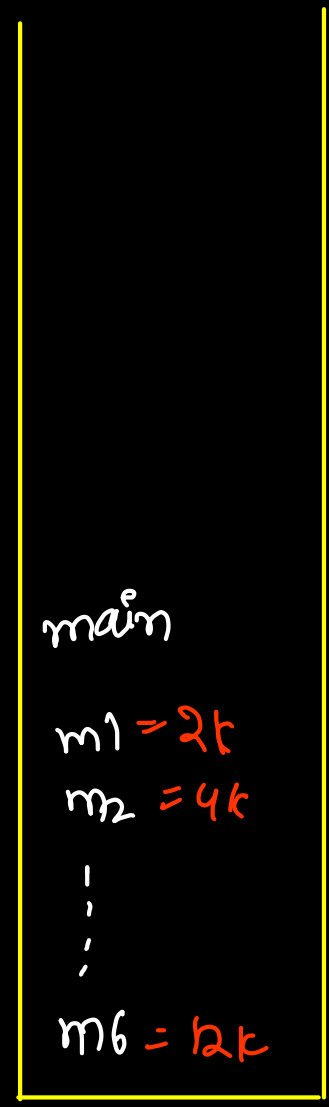
(4) Return current object

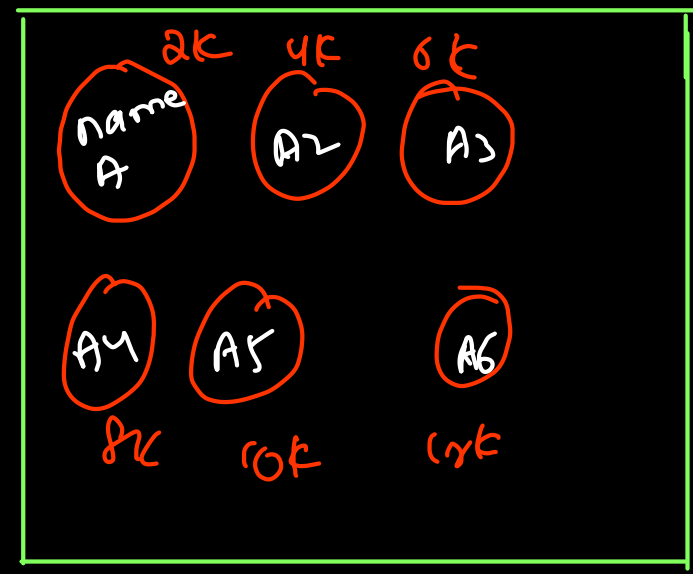
(5) Pass current object as parameter

separate (non-static) object

static (class)

```java
class Movie {
    String name;
    static String language = "Hindi";
    static int numberOfMovies = 0;

    public Movie(String name) {
        this.numberOfMovies++;
        this.name = name;
    }
}

public class App {
    Run | Debug
    public static void main(String[] args) {
        Movie m1 = new Movie(name: "Avengers");
        Movie m2 = new Movie(name: "Avengers Age of Ultron");
        Movie m3 = new Movie(name: "Avengers Infinity War");
        Movie m4 = new Movie(name: "Avengers End Game");
        Movie m5 = new Movie(name: "Avengers Kang Dynasty");
        Movie m6 = new Movie(name: "Avengers Secret Wars");

        System.out.println(m1.numberOfMovies);
        System.out.println(m6.numberOfMovies);

        m1.language = "Telugu";
        System.out.println(m2.language);
    }
}
```





numberOfMovies = 0 / 12

language = "Hindi" / "Telugu"

6

```java
public static void main(String[] args) {
    Movie m1 = new Movie(name: "Avengers");
    System.out.println(m1.numberOfMovies);
    System.out.println(Movie.numberOfMovies);

    Movie m2 = new Movie(name: "Avengers Age of Ultron");

    System.out.println(m1.numberOfMovies);
    System.out.println(Movie.numberOfMovies);

    Movie m3 = new Movie(name: "Avengers Infinity War");

    System.out.println(m1.numberOfMovies);
    System.out.println(Movie.numberOfMovies);

    Movie m4 = new Movie(name: "Avengers End Game");
    System.out.println(m1.numberOfMovies);
    System.out.println(Movie.numberOfMovies);

    Movie m5 = new Movie(name: "Avengers Kang Dynasty");
    System.out.println(m1.numberOfMovies);
    System.out.println(Movie.numberOfMovies);

    Movie m6 = new Movie(name: "Avengers Secret Wars");

    System.out.println(m1.numberOfMovies);
    System.out.println(Movie.numberOfMovies);

    m1.language = "Telugu";
    System.out.println(m2.language);

    // Access Static Properties using ClassName
    // Even if no object is there
    System.out.println(Movie.numberOfMovies);
    System.out.println(Movie.language);
}
```

```java
class Movie {
    String name;
    static String language = "Hindi";
    static int numberOfMovies = 0;

    public Movie(String name, String language) {
        numberOfMovies++;
        this.name = name;
        Movie.language = language;
    }
}
```

```java
class Movie {
    private String name;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    private static String language = "Hindi";
    private static int numberOfMovies = 0;

    public Movie(String name) {
        numberOfMovies++;
        this.name = name;
        // Movie.language = language;
    }

    public void setLanguage(String language) {
        Movie.language = language;
    }

    public String getLanguage() {
        return Movie.language;
    }

    public int getNumberOfMovies() {
        return Movie.numberOfMovies;
    }
}
```

Movie.setLanguage → static

```java
Movie av = new Movie(name: "Avengers");

av.setName(name: "Young Avengers");
System.out.println(av.getName());

av.setLanguage(language: "English");
System.out.println(av.getLanguage());
```

non static

we cannot call non static methods w/o object using classname

```java
class Movie {
    private String name;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    private static String language = "Hindi";
    private static int numberOfMovies = 0;

    public Movie(String name) {
        numberOfMovies++;
        this.name = name;
        // Movie.language = language;
    }

    public static void setLanguage(String language) {
        Movie.language = language;
    }

    public static String getLanguage() {
        return Movie.language;
    }

    public static int getNumberOfMovies() {
        return Movie.numberOfMovies;
    }
```

```java
Movie.setLanguage(language: "Hindi");
System.out.println(Movie.getLanguage());

Movie av = new Movie(name: "Avengers");

av.setName(name: "Young Avengers");
System.out.println(av.getName());

av.setLanguage(language: "English");
System.out.println(av.getLanguage());
```

```java
public void nonStaticFun() {
    // name, genre, language, numberOfMovies
    // After Object Creation

    // Non Static Properties Accessible
    System.out.println(this.name);
    System.out.println(this.genre);

    // Static Properties Accessible
    System.out.println(Movie.language);
    System.out.println(Movie.numberOfMovies);

}

public static void staticFun() {
    // name, genre, language, numberOfMovies
    // Before Object Creation

    // No this keyword inside static
    // System.out.println(this.name);
    // System.out.pritnln(this.genre);

    // Static Propeties Accessible
    System.out.println(Movie.language);
    System.out.println(Movie.numberOfMovies);
}
```

```java
Movie.staticFun();

// Movie.setLanguage("Hindi");
// System.out.println(Movie.getLanguage());

Movie av = new Movie(name: "Avengers");

av.nonStaticFun();
av.staticFun();
```

```java
public class ipl {
    static HashMap<String, Integer> map = new HashMap<>();

    static ipl() {
        // Initialize Static Properties Using Constructor
        map.put(key: "CSK", value: 0);
        map.put(key: "DC", value: 0);
        map.put(key: "MI", value: 0);
        map.put(key: "GT", value: 0);
    }
}
```

accessible using class name

ipl obj = new ipl();

ipl.ipl();

syntax ✗

contradiction

→ do not need object

→ create object

Static Constructor not possible

```java
public class ipl {
    static HashMap<String, Integer> map = new HashMap<>();

    static {
        // Class is loaded: it will run
        map.put(key: "CSK", value: 0);
        map.put(key: "DC", value: 0);
        map.put(key: "MI", value: 0);
        map.put(key: "GT", value: 0);

        System.out.println(x: "Hello IPL Class Static
        Block");
    }

    ipl() {
        // Initialize Static Properties Using Constructor
        System.out.println(x: "Hello IPL Object");
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Hello Main Function");
        ipl obj = new ipl();
        System.out.println(obj.map);
    }
}
```

1

3

2

it will only run once/ class loading

↳ class only gets loaded only once

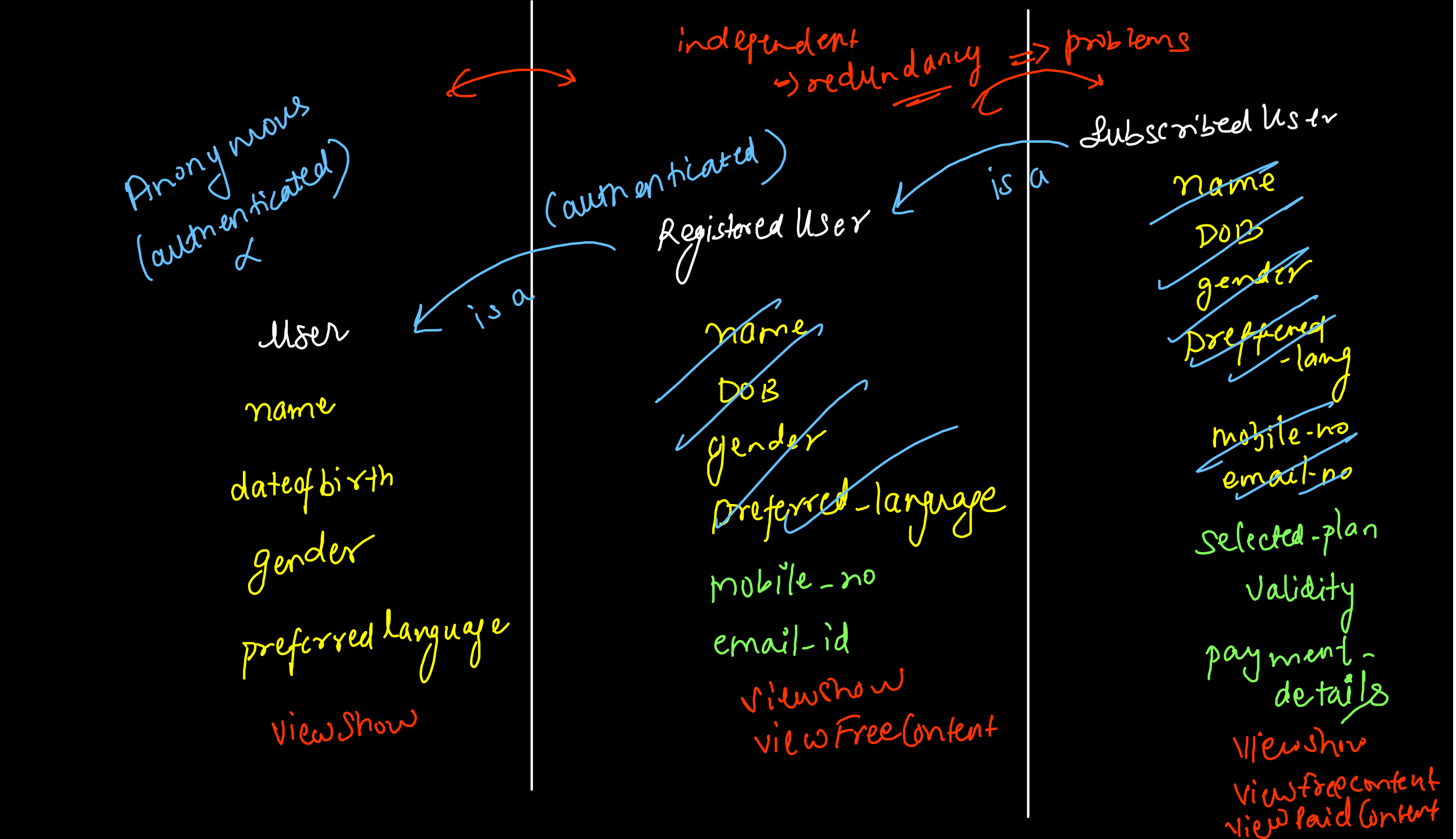
1st which will execution

```java
public class ipl {
    static HashMap<String, Integer> map = new
    HashMap<>();

    static {
        // Class is loaded: it will run
        map.put(key: "CSK", value: 0);
        map.put(key: "DC", value: 0);
        map.put(key: "MI", value: 0);
        map.put(key: "GT", value: 0);

        System.out.println(x: "Hello IPL Class
        Static Block");
    }

    ipl() {
        // Initialize Static Properties Using
        Constructor
        System.out.println(x: "Hello IPL Object");
    }

    // public static void main(String[] args) {
    //     System.out.println("Hello Main
    Function");
    //     ipl obj = new ipl();
    //     System.out.println(obj.map);
    // }
}
```

Compilation ✓

nothing will run

{ error: no main method found }


independent
=> redundancy => problems
Anonymous (authenticated) & (authenticated)
Registered User
Subscribed User
is a
is a
User
name
dateofbirth
gender
preferred language
ViewShow
name
DOB
gender
Preferred-language
mobile-no
email-id
viewshow
viewFreeContent
name
DOB
gender
Preffered-lang
mobile-no
email-no
Selected-plan
Validity
payment-details
Viewshow
Viewfreecontent
viewPaidContent

User.java U

```java
public class User {
    private String name;
    private char gender;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public char getGender() {
        return gender;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public void viewShowListing() {
        System.out.println
        (x: "Unauthenticated user");
        System.out.println(x: "View Show
        Listings");
    }
}
```

RegUser.java U

```java
public class RegUser extends User {
    private String emailId;
    private long phoneNo;

    public String getEmailId() {
        return emailId;
    }

    public void setEmailId(String
    emailId) {
        this.emailId = emailId;
    }

    public long getPhoneNo() {
        return phoneNo;
    }

    public void setPhoneNo(long phoneNo) {
        this.phoneNo = phoneNo;
    }

    public void viewFreeShows() {
        System.out.println
        (x: "Authenticated User");
        System.out.println(x: "View Free
        Shows");
    }
}
```

App.java M

```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        User u1 = new User();
        u1.setName(name: "Hrishi");
        u1.setGender(gender: 'M');

        System.out.print(u1.getName() + " ");
        System.out.print(u1.getGender());
        u1.viewShowListing();

        RegUser u2 = new RegUser();
        u2.setName(name: "Archit");
        u2.setGender(gender: 'M');
        u2.setEmailId(emailId: "archit.
        aggarwal023@gmail.com");
        u2.setPhoneNo(phoneNo: 9319117889l);

        System.out.print(u2.getName() + " ");
        System.out.print(u2.getGender() + " ");
        System.out.print(u2.getEmailId() + " ");
        System.out.print(u2.getPhoneNo());

        u2.viewShowListing();
        u2.viewFreeShows();
    }
}
```