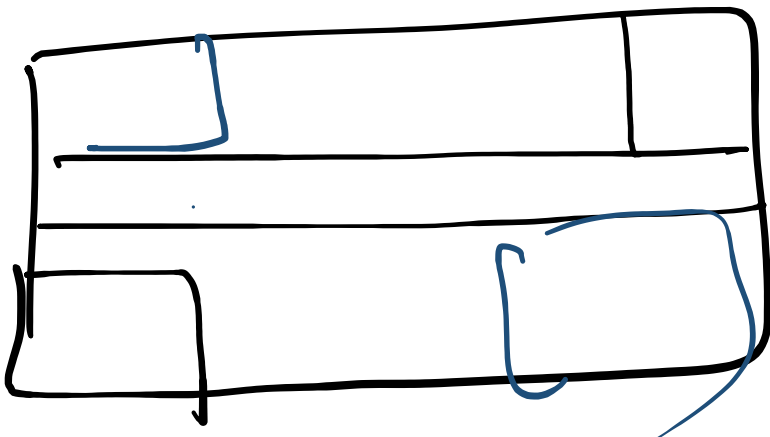


class → blue print



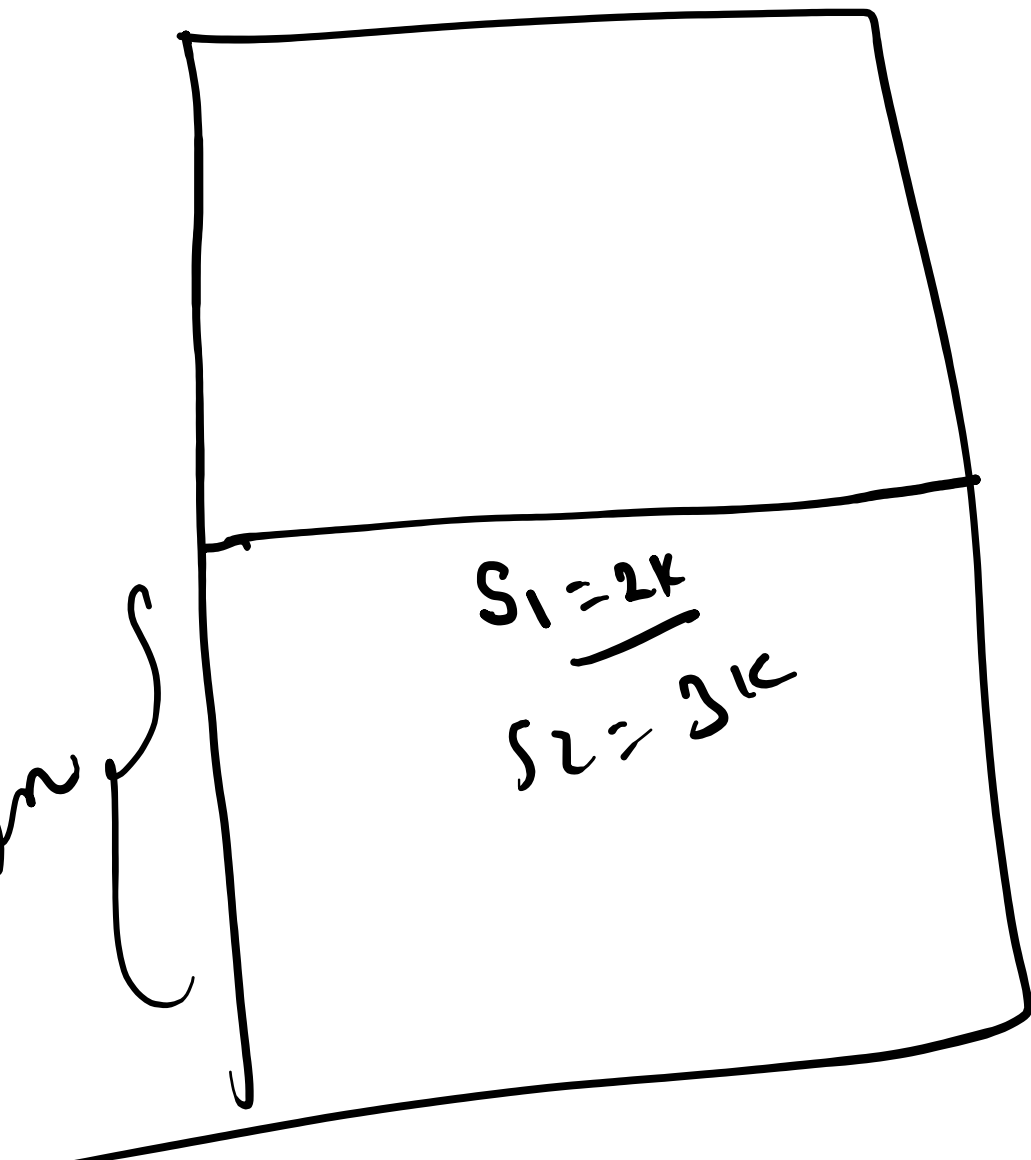
```
public class Student {
    String name;
    int age;

    public void Intro_yourSelf() {
        System.out.println("My name is " + name + " age is " + age);
    }
}

public class StudentClient {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "Kaju"; ✓
        s1.age = 19;
        Student s2 = new Student(); ✓
        s2.name = "Raju"; //
        s2.age = 45;
        s1.Intro_yourSelf(); ✓
        s2.Intro_yourSelf(); ✓
    }
}
```

Student Class

main



name  
age  
Intro ✓  
Heap

name = Kaju  
age = 19

name → Raju  
age → 45

S1.compareTo(S2) > 0  
S1 > S2  
S1.equals(S2)

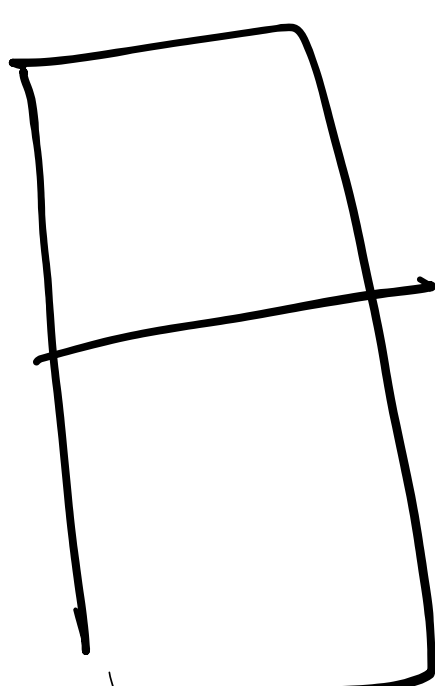
```
public class Person {
    String name = "Kaju";
    int age = 20;

    public Person() {
        // TODO Auto-generated constructor stub
    }

    // public Person(String name) {
    //     // TODO Auto-generated constructor stub
    // }

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

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    Person p = new Person("Raju", 27);
}
```



① memory allocated  
② parsing  
③ constructor  
name → Kaju  
age → 20  
27

**Encapsulation**

- It is the process of **Grouping** the **data** with its related **methods** into single unit(Class).
- To **prevent unauthorized access** to data members of object.
- Advantages of Encapsulation**
  - Security
  - Code flexibility
  - Modularity
  - maintainability

Get  
up date

Encapsulation

Error

Exception

↳ Abnormal behav

