OOPs 1 : Introduction

- Programming Paradigms
- Procedural Programming
- Object Oriented Programming
- Accen Modifiers

Rajat Sharma
sharath r
Saurabh Ruikar
Khushi Raj
Shrikanth
Madhan Kumar M S
Akansh Nirmal
Vishal Mosa
Pankaj
Rajendra
Purusharth A
Vasanth
Rajeeb Kumar Nath
suyash gupta
Sushant Patil
Shani
Venkata Sribhavana Nandiraju
Shradha Srivastava
Gowtham Sankaran
Abhishek Sharma
Vimal
Sanket Giri
Nikhil Pandey

100%

Avg ~ 64% ⟶ next target 70%

Programming Paradigms

what is a programming paradigm ?

Structured way of writing code.

Eg ⟶ Dominos pizza have a structured way of making pizza.

what would happen w/o programming paradigm ?

- Difficult to maintain
- Hard to test
- Difficult to read etc.....

Types of Programming Paradigms

1> Imperative Programming

⟶ You tell the computer how to do a task

Eg

```
int a = 10
int b = 20
int total = a+b
print(total)
```

2> Procedural Programming

Splitting the entire logic into small procedures.

methods
functions

Eg

```
int a = 10
int b = 20
int total = add(a,b)
print(total)

int add ( int a, int b) {
    return a+b
}
```

procedure

3> Declarative Programming

You tell what to do, but not how to do it.

Eg — SQL

```
select * from user ;
```

4> Object Oriented Programming — OOP

Later today.

Procedural Programming

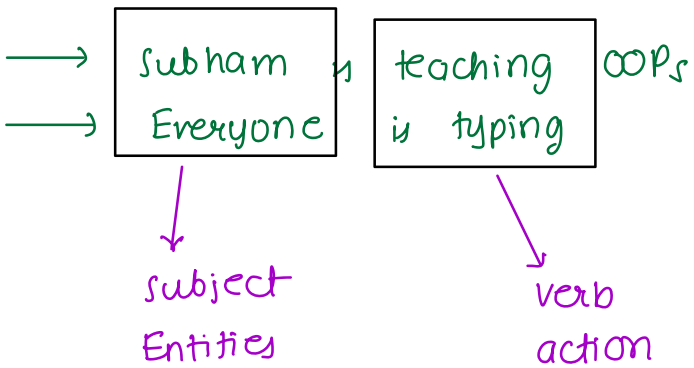
Splits entire code into small procedures or functions

Execution of a program starts from a specific procedure, what is it?

```
int add ( int a, int b) {
    return a+b
}

int sub ( int a, int b) {
    return a - b
}

main() {
    int x = add ( 10, 20) ;
    int y = sub ( 20, 40);
}
```

Problems with procedural programming

QUIZ: Type one line on what's going on in class right now?



In real word Entities perform action.

```
void printStudent( String name,
                   String batch,
                   double psp )
{
    print(name)
    print(batch)
    print(psp)
}
```

combine them in a struct.

```
struct Student {
    string name
    string batch
    double psp
}
```

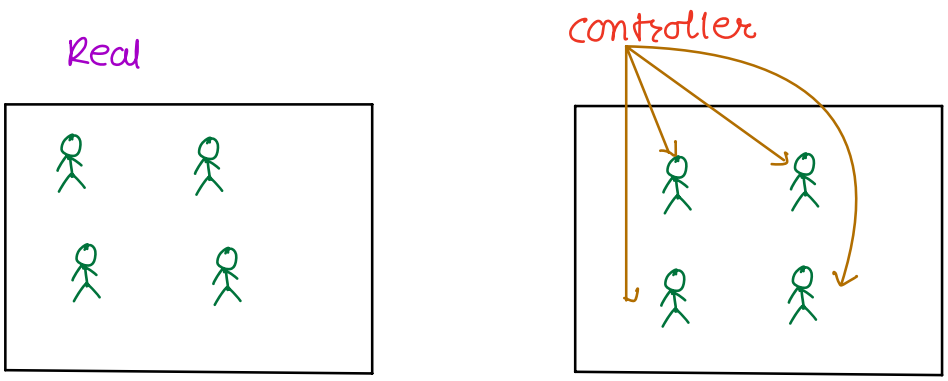
```
void printStudent ( student student) {

    print ( student.name )
    print ( student. batch )
    print ( student. psp )

}
```

action is being performed on student.

Issues with struct

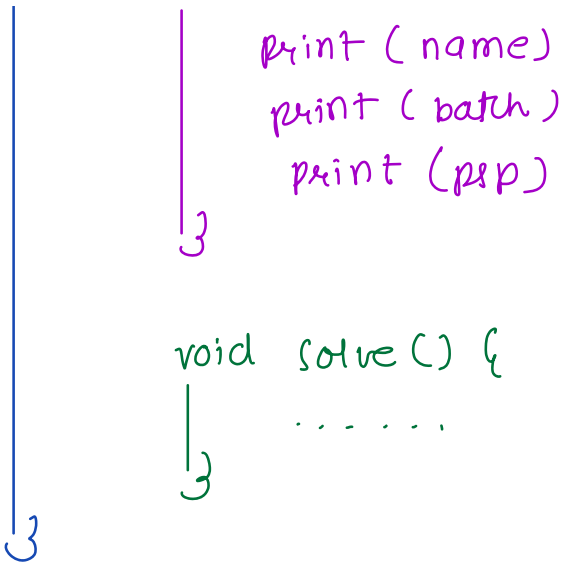
→ all the properties are public

→ structs donot support methods.



How a student looks in OOP world ?

```
class Student {
    string name
    string batch
    double psp

     void print () {
```



cons of procedural programming ?

—— Not secure { everything is public }

—— less maintainable & reusable.

—— hard to debug.

Object Oriented Programming

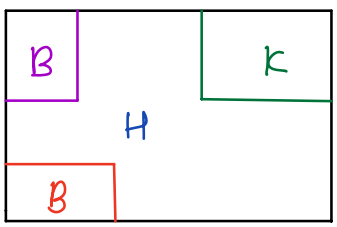
Entities + behaviour.

QUIZ: what is a class ?

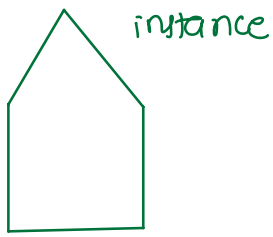
Blueprint of an idea.

doesn't take space significant space

present in memod area of JVM



Blueprint.

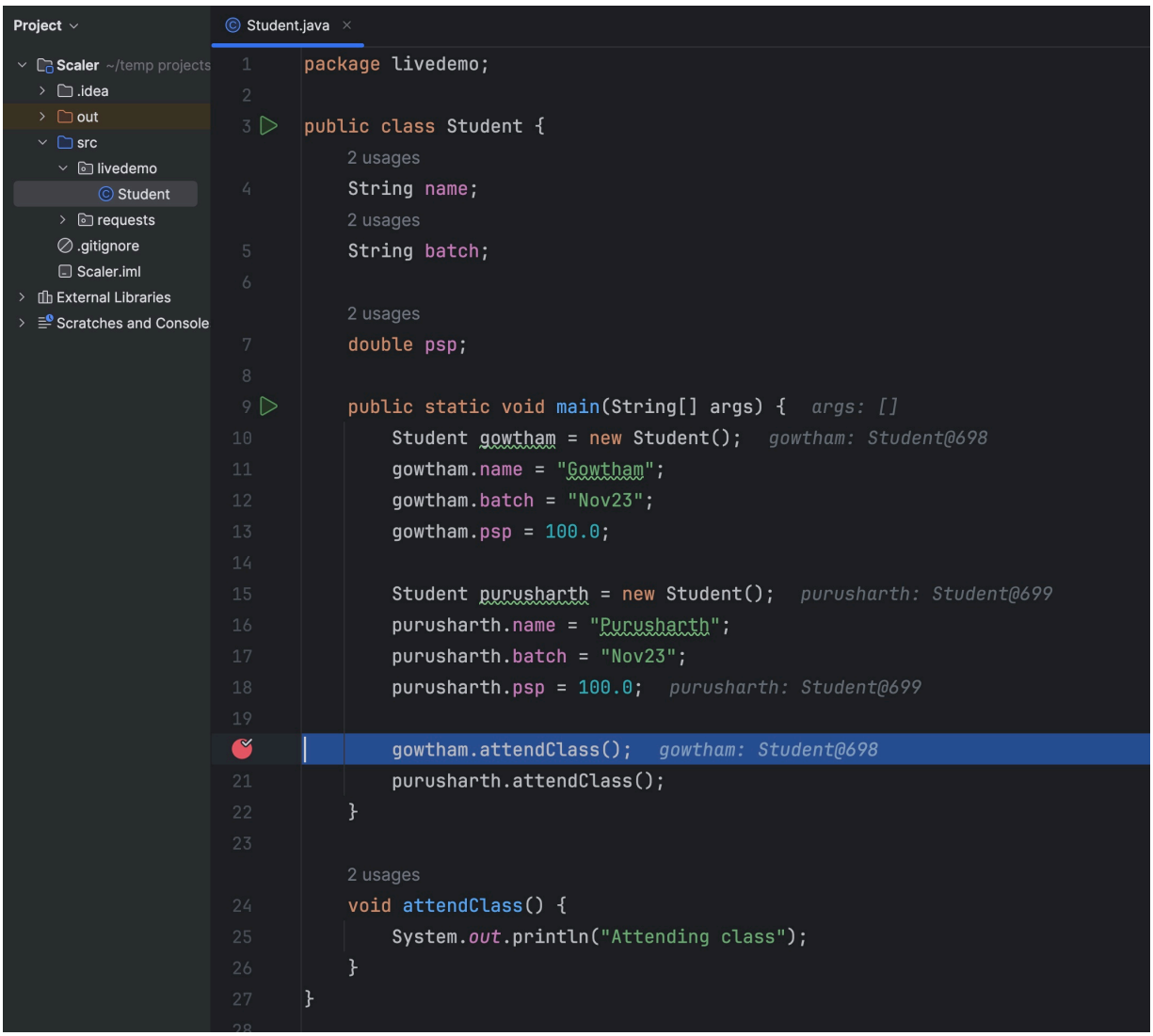


Object —— Instance of a class

takes space in heap memory.

A class can have many instance(s)

Classes & Objects {live demo}



Break : 10 : 25

Pillars of OOPs

How many pillars of OOPs are there ?

There are 3 pillars and 1 principle.

Java : The Complete Reference.

Principle — Fundamental foundation

Pillar — Support for holding thing together

principle of a student ? ⟶ discipline

How to be a disciplined student.

⟶ solving Q on time

⟶ attending class & contest

Encapsulation

Inheritance

Polymorphism

Abstraction

QUIZ: what do you understand by abstraction ?

→ Representing in terms of an idea.

Design Scaler ⟶ Ideas

Student , Mentor , Instructor

⟶ what all thing we will repsent in the programming word ?

class ⟶ attributes and behaviors / methods

Encapsulation

QUIZ: Does encapsulation contain a word we use in daily life ?

Capsule ⟶ Holds various types of medicine together

⟶ protects the medicine from outside environment.

QUIZ: what do we really store in programming ?

⟶ attributes } class

⟶ methods }

```
class Student {
    String name

    void tellName {
    }
}
```

How to protect class from outside env ?

Access modifier

Types of Access Modifiers ?

- Public — can be access by every one
- Private — can be accessed by no one outside of class.
- Protected — Can be access by classes in same package as well as sub class. child class.
- Default — Can be accessed within same package

	class	package	subclass (same package)	Sub class diff package	world
public	✓	✓	✓	✓	✓
protected	✓	✓	✓	✓	
default	✓	✓	✓		
private	✓				

---

this keyword

```
class Student () {
    String batch ;
    double psp ;
    void changeBatch ( String batch ) {
        this . batch = batch
    }

    void compare ( Student s2 ) {
        this . psp  compare with  s2 . psp .
                                  >=
    }
}
```

Examples

```java
package livedemo;

public class AccessModifierExample {
    3 usages
    public int publicVar = 10;
    2 usages
    protected int protectedVar = 20;

    2 usages
    int defaultVar = 30;

    1 usage
    private int privateVar = 40;

    1 usage
    private void privateMethod() {
        System.out.println("Private method");
    }

    3 usages
    public void publicMethod() {
        System.out.println("Public method");
    }

    2 usages
    protected void protectedMethod() {
        System.out.println("Protected method");
    }

    2 usages
    void defaultMethod() {
        System.out.println("Default method");
    }

    public static void main(String[] args) {
        AccessModifierExample accessModifierExample = new AccessModifierExample();
        System.out.println(accessModifierExample.publicVar);
        System.out.println(accessModifierExample.protectedVar);
        System.out.println(accessModifierExample.defaultVar);
        System.out.println(accessModifierExample.privateVar);

        accessModifierExample.publicMethod();
        accessModifierExample.protectedMethod();
        accessModifierExample.defaultMethod();
        accessModifierExample.privateMethod();
    }

```

```java
package livedemo;

public class OutsideClass {
    public static void main(String[] args) {
        AccessModifierExample accessModifierExample = new AccessModifierExample();
        System.out.println(accessModifierExample.publicVar);

        // Child class as well as any class in the same package can access protected variables
        System.out.println(accessModifierExample.protectedVar);

        // within the same package, default variables can be accessed
        System.out.println(accessModifierExample.defaultVar);

        // CTE since private variables can only be accessed within the same class
        // System.out.println(accessModifierExample.privateVar);
        // accessModifierExample.privateMethod();

        accessModifierExample.publicMethod();
        accessModifierExample.protectedMethod();
        accessModifierExample.defaultMethod();
        // accessModifierExample.privateMethod();
    }
}

```

```java
package outside;

import livedemo.AccessModifierExample;

public class OutsidePackageClass {
    public static void main(String[] args) {
        AccessModifierExample accessModifierExample = new AccessModifierExample();
        System.out.println(accessModifierExample.publicVar);

        // Child class as well as any class in the same package can access protected variables
//        System.out.println(accessModifierExample.protectedVar);

        // within the same package, default variables can be accessed
//        System.out.println(accessModifierExample.defaultVar);

        // CTE since private variables can only be accessed within the same class
        // System.out.println(accessModifierExample.privateVar);
        // accessModifierExample.privateMethod();

        accessModifierExample.publicMethod();
//        accessModifierExample.protectedMethod();
//        accessModifierExample.defaultMethod();
        // accessModifierExample.privateMethod();
    }
}

```

Static keyword

→ Static variables — If a variable is anociated with class rather than instance of a class.

Eg : Integer . MIN_VALUE .
static.

→ static methods — No need to create instance of a class to call static methods.

Eg : Math.pow System.out Math.max

```
class StaticEx {
    static int cnt = 0;

    static void increase() {
        cnt++
    }
}
```

common across all instances of this class.

```
staticEx.increase()   // cnt = 1
staticEx.increase()   // cnt = 2
```

Scope of a variable

Class level / static scope — common across all instances of class

Instance scope — variables whose scope is limited to instance of a class

Method scope — scope level is within the method

Block scope — { } variables scope is within braces.

```java
package livedemo;

public class ScopeDemo {
    // Class level variable
    1 usage
    public static int classVar = 10;

    // Instance level variable
    4 usages
    int instanceVar = 20;

    1 usage
    public static void setClassVar(int classVar) {
        ScopeDemo.classVar = classVar;
    }

    public static void main(String[] args) {
        // Block level
        if (true) {
            int blockVar = 40;
//            System.out.println("Block level variable: " + blockVar);
        }

        // Accessing class level variable
        ScopeDemo scopeDemo1 = new ScopeDemo(); // instance 1
        ScopeDemo scopeDemo2 = new ScopeDemo(); // instance 2
        ScopeDemo.setClassVar(100);
        scopeDemo1.instanceVar = 200;
        scopeDemo2.instanceVar = 300;
        System.out.println(scopeDemo1.instanceVar);
        System.out.println(scopeDemo2.instanceVar);

        scopeDemo1.method();

    }

    1 usage
    public void method() {
        // Method level variable
        int methodVar = 30;
        System.out.println("Method level variable: " + methodVar);

    }

}
```