UNIVERSITY

Name : .....

..... School/College

Class : ..... Section : .....

Roll No : ..... Year : .....

Subject : OOP

GW 15.10.22

OOP

class

attribute → কি আছে? } Property

task → কি করতে পারে? }

instance → object

employee → role
↳ method

- → id
- → age
- → gender
- → department
- → salary

role
- → HR
- → Developer

* Instance variable

P.T.O.

double sum(int a, int b){

return a+b;

}

* Array Creation:

int arr[] = new int [100];

x = y++

= y

= y

C.W
21.10.22

OOP

* For each loop
* We call static variable or static method as class property.
We don't use any object to access it.

```
* class Box {
    Double w, h, l;
  }
class myBox {
 psvm (...){
 Box obj = new box ();
  sop (obj.w);
  obj.w = 10;
 }
}
```

empty constructor assigns null value by default.

CW
29.10.22

OOP

```
class shape {
int height; double length, base; float radius;

shape (int height, double length,
       double base, float radius) {

    this.height = height;
    this.length = length;
    this.base = base;
    this.radius = radius;
}

void area (double x, double y) {
    double z;
    z = x * y;
    sop (z);
}
```

rectangle

P.T.O

square

```
void area(double x){
    double z = x * x;
}
```

triangle

```
void area(double x, double y){
    double ans;
    ans = 1/2 * x * y;
    sop(ans);
}
```

circle

```
void area(float x){
    float ans;
    ans = 3.1416 * x * x;
    sop(ans);
}
```

```
main class {

psvm(String[] args) {

shape obj = new shape
                (2, 5.33, 2.55, 1.2);

obj.area (5.33, 2.55);

obj.area (2.55);

obj.area (2, 5.33);

obj.area (1.2);
```

1.12.22

CT Practice

OOP

import java.util.Scanner;

Class <Class Name> {

Attributes / Variables;

Constructors ();

Methods ();

}

* class Box {

double w, h, d;

Box (double w, double h, double d) {

this.w = w;

this.h = h;

this.d = d;

}

double z

void input (double x, double y) {

double ans = x * y * z;

Vol

sout (" : ") ; }

main class {

psvm.. () {

Box obj = new box(w, h, d);

obj. input (w, h, d);

Scanner sc = new Scanner (System.in);

double w = sc.nextDouble();

w h = w ;

w d = h ;

}

GW
14.11.22

Java

OOP

[redacted]

Ep: 3

Multiple Classes

```
class MyClass {
public void MyMethod () {
sout println ("I am in the MyMethod section");
}
}
public class MainClass {
public static void main (String [] args) {
MyClass x = new MyClass ();
x.MyMethod ();
}
}
```

নতুন object create করা হলো x নামে

x দিয়ে MyMethod কে call করা হলো

P.T.:

Constructors

```
public class My.Date {
    private int date;
    private int month;
    private int year;

    public My Date () {
        date = 20;
        month = 11;
        year = 2021;
    }

    public void showData () {
        System.out.println("Day is " + date());
        System.out.println("Month is " + month);
        System.out.println("Year is " + year);
    }
```

→ Constructors

(Random method for print)

public static void main (String[] args) {

MyPrde x = new MyPrde();

x.showData();

}

(Constructor কে call করার জন্য)

Blocks

Ep:7

Block > Constructor > Method

public class Main{

pwlic String information;

public Main() { // default constructor

information = "I'm a Constructor";

sout println (information);

}

P.T.

{ (Blocks)

information = "I'm a Block";

sout println(information);

}

(Method)

public void showdata() {

information = "I'm a Method";

Sout println(information);

}

}

psvm(String[] args){

Main x = new Main();

x.showdata();

}

Output:

I'm a block

I'm a constructor

I'm a method

(Always block আগে execute হয়। ~~[illegible]~~ (Constructor & method)

This

```
public class Main {
private int num1, num2, result;
public Main(int value1, int value2){
this. num1 = value1;
this. num2 = value2;
}
```

P.T.O

```
public void addition () {
result = num1 + num2;
sout ("Summation:" + result);
}
psvm (String[] args) {
Main x = new Main (2, 3)
x.addition ();
}
```

Wrapper Class

int (Data Type)

Integer (Wrapper Class)

Public class Box {

public void boxing() {

int var1 = 100;

Integer objInt = new Integer (var1);

(Auto boxing)

int var2 = objInt.intValue();

(Auto unboxing)

System.out.print (var2);

}

}

Auto

```
psvm ( ) {
  int var1 = 100;
  Integer myIntObj = var1;
  int var2 = myIntObj;
  Sout. print ln (var2);
```

Access Modifier

Modifier	Same class	Same Package	Sub Class	Universe
Private	✓			
default	✓	✓		
Protected	✓	✓	✓	
Public	✓	✓	✓	✓

Encapsulation

```
class MyNewClass {
private int Number;
// (Setter method)
public void setNumber(int x) {
  this.Number = x;
}

// getter method
public int getNumber() {
  return Number;
}
}
```

P.T.

```
public class Main {
    psvm() {
        MyNewClass obj = new MyNewClass();
        obj.setNumber(500);
        sout println("Num: " + obj.getNumber());
```

Inheritance

```
class MyClass {
    public int x = 20;
    public int j = 50;
    public void display() {
        System.out.println("Value x: " + x);
        System.out.println("Value j: " + j);
    }
}
public class Main extends MyClass {
    public int z = 100;
}
psvm() {
    Main obj = new Main();
    obj.display();
}
```

P.T.

যদি Parent class এর variable print করা হয় child class এ তখন হবে,

sout("Value x:" + obj.x);
sout("Value y:" + obj.y);
sout("Value z:" + obj.z);

Method Overriding

→ Parent class এ child class এ same method থাকবে।

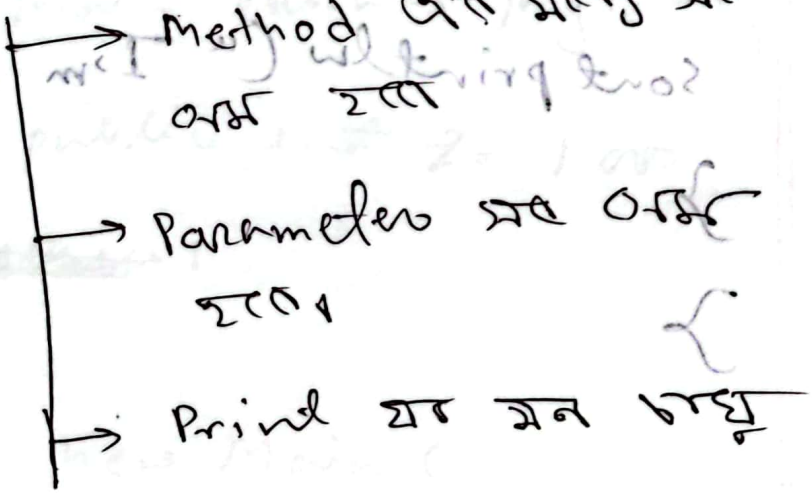
class Parent_class {
    public void display() {
        sout print ln ("I'm in the Parent class");
    }
}

P I°

```
public class child_class extends Parent_Class{

public void display(){
sout print("I'm in the child class");
}

psvm(){
child_class obj = new child_class();
obj.display();
}
}
```



Involing Overriding Methods

```
class A {
    public String name = "Fahim Bin Amin";
    public float salary = 65000;

    public void display() {
        System.out.println("Name: " + name);
        System.out.println("Salary: " + salary);
    }
}

Public class B extends A {
    public void display() {
        System.out.println("I study at DIU");
    }
}
```

```
psvm ( ) {
    B obj = new B ( );
    obj. display ( );
```

→ Overriding ↳
main class ↳
Parent class এবং
child class ↳
Same method ↳
child class ↳
method execute
করা, Parent class
↳ না।

যদি Parent class ↳ সকল child class ↳ method execute করতে চাই তখন super keyword ব্যবহার করতে হবে।

So the code: →

```
public class B extends A {
public void display ( ) {
    super. display ( );
Sout print ("_");
```

P.T.:

Invoking parent class Constructor

```
class Employee {
public String name;
public float salary;

public Employee (String name, float salary){
this.name = name;
this.salary = salary;
}
public void display() {
sout print ("Name : " + name);
sout print ("Salary : " + salary);
}
```

super(base) obj

super
mf (d2)

P.T.O.

```
public class Manager extends Employee{

    public String depName;

public Manager (String depName){
  super("Fahim", 65000);
this. depName = depName;
}

public void display(){
  super. display();
System.out("Department name:" + depName);
}

psvm(){

Manager obj = new Manager (depName: "SE");
obj.display();
}
```

Method overloading

```
public class ABC {
int a, b, c;
public void setValue() {
    a = 1; b = 2; c = 3;
sout("a:" + a + "b:" + b + "c:" + c);
}
psvm() {
ABC obj = new ABC();
obj.setValue();
```

Overload:

public void setvalue(){

}

public void setvalue(int a){

method overide

}

Output same:

a = 1

b = 2

c = 3

how if I give a value in main function :→

```
psvm() {
ABC obj = new ABC();
obj.display(100);
}
```

```
public void setvalue(int a){
this.a = a;
```

Output:

a : 100.
b = 2;
c = 3.

P.T.O

~~Constructor Overloading~~

Polymorphism

- static
- dynamic
- method overloading

Interface

* Exception Handling

{221, 222, 213, 212}

(Exception, nested class, Interface)

CT. (সর্বশেষ)

Interface

* interface Flyable {

public static final string media = "sky";

public ~~static~~ abstract void fly();

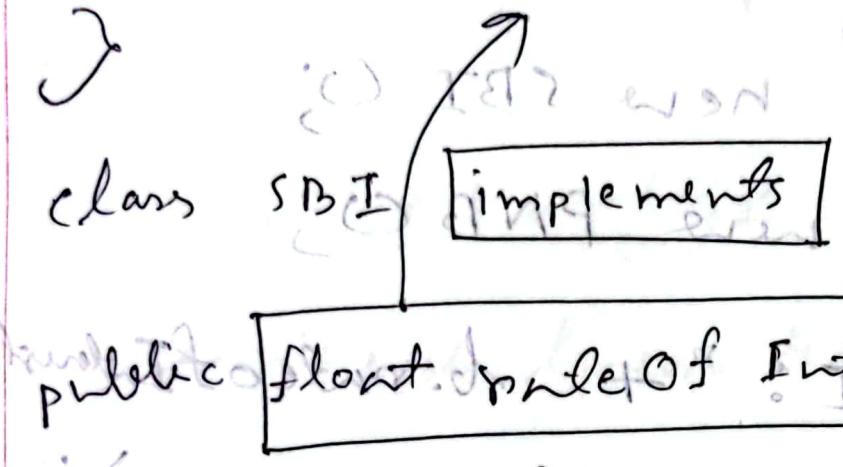
" " boolean needFuel();

}

```
// interface Bank {
    float rateOfInterest();
}

class SBI implements Bank {
public float rateOfInterest() {
return 9.15f;
}
}

class PNB implements Bank {
public float rateOfInterest() {
return 9.7f;
}
}
```



P.T.O.

```
class testInterface12{
psvm(){
Bank b = new SBI();
Bank c = new PNB();
sout("ROI: "+b.rateOfInterest());
s  ("ROI: "+c.r  ());
}
}
```

* Abstract / static / interface
* Exception / File
* Collection / Framework
  - → array list
  - → comparator
  - → nested) class
* static
  - → method
  - → block
  - → key word, final keyword
  - → gus

Comparator

```
public class student_com implements Comparator<student_comparator> {
public int compare(student_comparator a, b) {
return a.name.compareTo(b.name);
}
}
```

short_byroll implements

```
return a.rollno - b.rollno;
```