

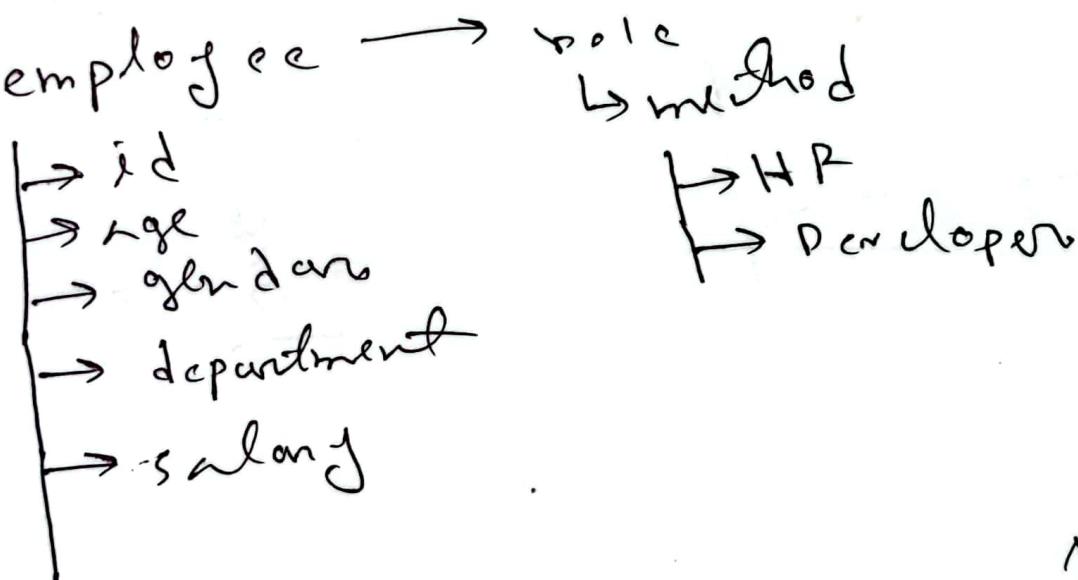
UNIVERSITY

Name :	Sandeep	
	11	School/College
Class :	11	Section :
Roll No :	Project	
Subject :	OOP	

~~GW~~
~~15.10.22~~

OOP

class (definition + behavior) user object
attribute → for state {
task → for action } }
instance → object writer {



* Instance ~~variable~~ variable

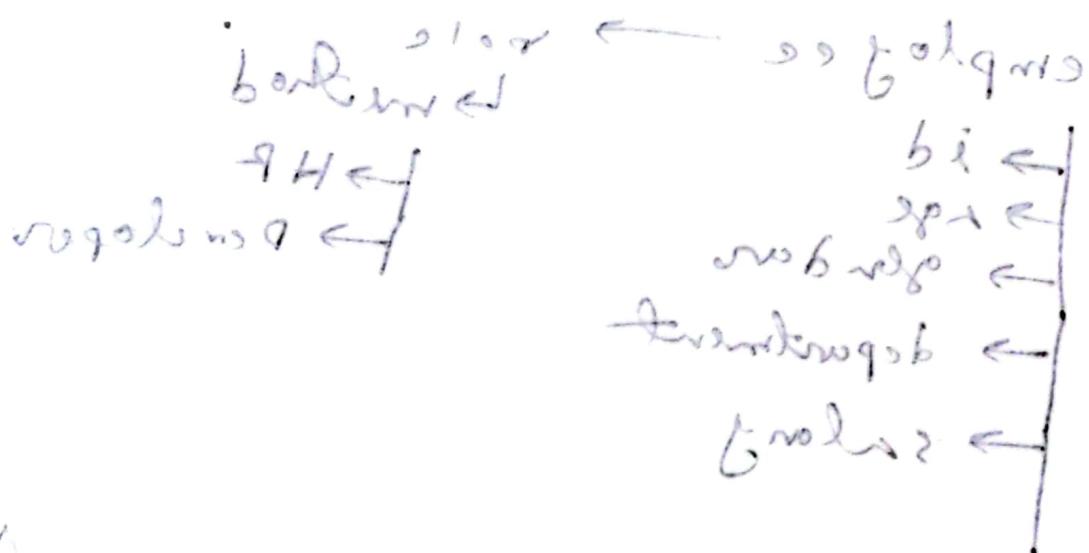
1. X
2. X
3. X

QUESTION

double sum (int & int) $\{ \text{int} \}$

$c_2 = a + b$ } $\{ \text{int} \}$ at a start
of digits
} same ones at a start

} return c_2 $\{ \text{int} \}$ = sum of digits



x_1
 x_2
 x_3
 x_4
 x_5
old ones ~~dig~~ sum of digits *

157

LW
18.10.22

00P

* Array Creation: is this - not *

or shadow pointer like x, y &
initializing in bottom write

int arr new int [100]; // memory
of size 100 bytes now available
 $x = y$

= $y - j$
= jy

} ~~not const~~
int w oldarr

{
} ~~not const~~
{C...} mV29

?() void wN = ido roll
region restrictions (kgms) ?(w. ido) go
that is to say wN = 01 - w. ido

{
L

LW
21.10.22

OOP
1-81

- * For each loop iteration through *
- * We call static variable or static method as class level func.

Property: [o] thi won thio thi
we don't use any object to ^{access} ~~access~~
it.

* Class Box {
 Double w, l, h;
}

class myBox {
 PSVM {
 Box obj = new Box();
 SOP (obj.w);
 obj.w = 10;
 }
}

empty constructor assigns null value by default.

GW
29.10.21

OOP

* $\left\{ \begin{array}{l} \text{class } \{ \text{shape} \} \\ \text{shape } \{ \end{array} \right.$ was biov

int height; double length, base, radius; front

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

this. height = height;

this. length = length = ans

this. base = base; (ans) 902

this. radius = radius;

}

{

void area(double, double) {

double z;

$z = \pi * r^2$; $r = \sqrt{\text{area}} = \sqrt{\pi * \text{radius}^2}$ = ans

SOP(z);

(ans) 902

rectangle ans

P.T.O

100

void area (double x) {
 double ans = 0.0;

square
 double z = x * x;
 ans = ans + z;

triangle
 double ans = 0.0;

 void area (double x, double y) {

 double ans = 0.0;

 ans = 1/2 * x * y;

 sop (ans);

 };

 }

}

void area (float x) { bio

 float ans; // float

 ans = 3.1416 * x * x;

 sop (ans);

 }

PSVM(\vdash obj) $\{ \text{obj} \in \text{dictA}$

shape obj $\{ \begin{array}{l} (\text{}) \text{ no rotation} \\ (\text{c}) \text{ dont sm} \end{array} \}$

(2, 5.33, 2.55 { 1.2)}

obj. area (5.33, 2.55); \star

obj. area b (2.55); \star (dwob)

obj. area (2, 5.33);

} (b \star dwob \star dwob \star dwob) + \star

obj. area (2.2);

\star $w = w.$ width

\star $ht = ht.$ width

\star $b = b.$ width

\star dwob

{ (5.33 / 2.55) \star dwob } \star ht. bior

\star 5.33 \star 2.55 \star dwob

{ ((5.33 / 2.55) \star ht. bior)

1.17.22

CT Practice

OOP

```
import java.util.Scanner;
```

Class <(class name)>

Attributes / Variables ;) m V : 9

Constructors () ;

Methods () ;

{ } ()

* <(class Box{ }> new . ido

double w, h; new . ido

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

this. w = w;

this. h = h;

this. d = d;

}

double z

void import (double x, double y){

double ans = x * y * z;

System.out.println("ans = " + ans); }

main class

PSVM -> {

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

obj. input (w, h, d);



↳ object to target

Scanner sc = new Scanner (System.
in);

double w = sc. nextDouble();

w h = v ;

v d = r ;



6/11/22

Java

OOP

~~multiple classes in one file~~

Multiple classes in one file Ep: 3

class MyClass {
 static void main (String args) {
 public void MyMethod () {
 System.out.println ("I am in the MyMethod
 section");
 }
 }
}

↳ In main() class = object
↳ In method() class = object

public class Main {
 public static void main (String args) {
 MyClass obj = new MyClass();
 obj.MyMethod();
 }
}

↳ obj → MyMethod() → Create an object
↳ obj → MyMethod() → Create an object
↳ obj → MyMethod() → Create an object
↳ obj → MyMethod() → Create an object

P.T.O.

Constructors

public class My Date { }
6

(private) int date;

private int month;

private int year;

public My Date () { }
} ← constructor

date = 20;

month = 11;

year = 2021;

} ← constructor
random method
for print

public void showDate () { }
} ← constructor

System.out.println("Date is " + date + "/" + month + "/" + year);
} ← constructor

→ month is between 1 to 12

→ year shows ("Year is 2021") → year

→ Both x and y are 2021

→ This was note

→ Note

271

```

public static void main (String[])
{
    "Hello & welcome" - writing or not in args
    MyPrinter.println (MyPrinter.kw)
    x. showButtons ();
}
} ("bottom & main" - writing or not in args)
("writing or not in args") writing level

```

A diagram showing a call from 'showButtons()' to 'MyPrinter.println()'. An arrow points from the 'showButtons()' method down to the 'MyPrinter.println()' call. A bracket labeled 'Constructor' is shown above the 'MyPrinter.println()' call, indicating it's a constructor call.

Block Ex: F

Block > Constructor > Method

```

public class Main {
    public String information;
    public Main () { (1) default constructor }
    information = "I'm the constructor!";
    System.out.println (information);
}

```

P.T.

```

    } { prints) (in Block) after void
} { 2pm information = "I'm a Block"
    send print bw (information) i
}

} { (in method) works . x
public void Showdata () {
    } { 2pm information = "I'm a Method"
    send print bw (information)
}

} { Fig Whole
} { bottom < top < whole
    } { Print whole void
psvm (String [] args) {
    } { whole in main void
Main() <--> New Main() void
    "Showdata" (P) <--> whole
}

} { whole in main void

```

P.T.O.

Output } () waitibba biav silshig

I am in block C now - Thread

"(Thread + constructor)" two

I am in method

(Always block & goes execute Σ_1)

empty (constructor) waiting

(in method)

(else) wait even block waiting

This () waitibba up

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() { Eligible
```

```
    result = num1 + num2;  $\rightarrow$  m.c1
```

```
    cout << "Summation is " << result;
```

```
}  $\rightarrow$  bants m  $\rightarrow$  m.c1
```

```
↳ knows object which is formula
```

```
PSVM (String & through) {
```

```
    Main x = new Main(2, 3)
```

```
x. addition(); int
```

```
}  $\rightarrow$  main calls adding
```

```
{ Else, even sum kvi storing
```

```
{ another kvi (sum kvi) stores adding
```

```
{ balsv = sum - diff.
```

```
{ salsv = sum + diff.
```

Q.T.1

Wrapper Class

otwA

int (Primitiv Type) } () mvg

Integer (Wrapper Class) mvg twi

~~class Integer { }~~ mvg twi

public class Integer { } mvg twi

public void boxing() { } mvg twi

int var1 = 100 ;

Integer objInt = new Integer(var1);

gewartet wird und wird (unboxing)

int var2 = objInt.intValue();

(unboxing)

System.out.print(var2);

befehl

ausgabe

P.T.

Auto

Auto oggervl

psvm () { (int, str) kri

int van1, van2; // oggervl register

Indyer myIntObj = var1;

int van2 = myIntObj; // ok

Sout. print lin(van2); // ok

Access Modifiers

Modifiers	Same class	Same Package	Sub Class	Chiverse
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() { } *{ constructor only}*

MyNewClass obj = new MyNewClass();

obj.setNumber(500);

cout << "Num: " + ~~obj.getNumber()~~ *obj.getNumber()*;

i(x + " is ok") *error*

i(t + " is ok") *error*

L.

L.

Good) for overriding with only sibling

call of this sibling ~~function~~

~~I am overriding~~

} << endl

i() with own = do with

i() ~~function~~ - i() *function*

???

Inheritance with constructor

```
class MyClass { } // class  
public int x=20; // instance variable  
public int y=50; // instance variable  
public void display() {  
    System.out.println("Value x: " + x);  
    System.out.println("Value y: " + y);  
}
```

3
3

```
public class Main extends MyClass { }
```

~~public class Main { }~~
~~public int z=100;~~
~~public void display() { }~~

```
public Main() { }
```

```
Main obj = new Main();  
obj.display();
```

3. } P.T.O.

first parent class has variable print
from child class or own class

cout < "Value of " + obj.z;

(cout < "Value of " + obj.z);
cout < "Value of " + obj.z;

Method Overriding

↳ Inheriting every class → Parent class
↳ class Parent { } and child class is
↳ public void display() { } same method

↳ If we write both
cout < "I'm in the Parent Class";

} 1st output
} 2nd output
} 3rd output

P.T.O

public void child-class extends Parent-Class
{
 ...
}

public void display() {
 System.out.println("Time in the child class");
}
} // class is "TimeIn"

PSVM: <--> private void print()

Object created
2 child-class obj = new child-class;
both are
obj. display() is being called

Method display is called
obj 2

Parameters are obj

201

Print 201 on 202

201

{

202

201

202

201

202

201

202

~~It prohibits~~
~~to make~~ ~~wrong~~ Involving Overriding Method

It's only two
classes {list} \rightarrow Suspense = ido 8
↳ B.alqib.ido

A contains public String name = "Fahim Bin Amin"
B contains

↳ float salary = 6500;

and two , it's

public void display () {

↳ System.out.print ("Name : " + name);
System.out.print ("Salary : " + salary);

log brogjw nqas ido 8
}

Public class B extends A {

public void display () {
System.out.println ("I study at UG");
}

i (" ") bina bao

```

    b.display()
}
B obj = new B();
obj.display();

```

→ Overriding ↗
 main class ↗
 Parent class ↗
 Child class ↗

child will inherit parent's method

→ ~~public void display()~~ ← Method Overriding
 same method ↗
 child class ↗
 Method Overriding
 ↗ in Parent class
 ↗ in child class

if i (class + " : " + B) brings to child
 class (class + " : " + A) → executive
 करते हैं तो super keyword का
 काम करता है

So the code: →

```

} A extends B {
public class B extends A {
    public void display() {
        super.display();
    }
    public void print() {
    }
}

```

P.T.

Involving a Parent Class Constructor

```
class Employee {  
    void print() {  
        System.out.println("Employee");  
    }  
    public String name; // instance variable  
    public float salary;  
}
```

```
public Employee (String name, float salary) {
```

```
    this.name = name; // field  
    this.salary = salary; // field
```

```
}
```

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

```
super(base) obj) // getting info
```

```
P.T.O.
```

super
mf (d)

public class Manager extends Employee{

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

```
public void display() {  
    super.display();  
    cout << " Department name: " << depName;  
}  
}
```

```
psym() {  
    cout << " Name: " << name;  
}  
Manager obj = new Manager(depName: "SE");  
obj.display();  
}
```

```
}  
super.display();  
P.T.
```

Method overloading

public class ABC { } b1ov silvng

int a, b, c;

public void set Value () { }

a = 1; b = 2; c = 3;

sent ("a" + "b" + "c" + " " + " " + " ") b1ov silvng

} b1okw
silvng

psvm { }

ABC obj = new ABC();

obj.set Value ();

silvng

5 = 0

— 5.0

ε = 0

ε = 0

PT.0

overloading

public void getvalue()
{
 System.out.println("child")
}

}
} class A
} class B extends A

public void setvalue(int a){
 System.out.println("parent")
}

method
overide

} class C
} class D extends C
} class E extends D

Output Same:

a = 2

b = 2

c = 3

eff

eff

how to give value in

main function ~~without~~

public void
setvalue(int a)

psvm () {
 ~~int a = 10;~~
 ~~int b = 20;~~
 ~~int c = 30;~~

this.a = a;

~~obj.value = 100;~~
ABC obj = new ABC();

obj.display();

Output:

A : 100.

B = 200.

C = 300.

P.T.O

Constructor Overloading

bis silder

Kennen lernen
in 2 - 3 min

Polymerphismus

- static
- dynamic

→ method overloading

(obj) fügt sich ein

intervall

• 01 : 2

int d

• E : 3

1.7.9

QW

FR =

OOP

Interface

* Exception Handling

{ 221, 222, 213, 212 }

(Exception, nested class Interface)
CT. (异常)

Interface

* interface Flyable {

public static final string media = "sky";

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

}

P.S.

~~# interface Bank {~~ float rateOfInterest(); }
~~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 test Interface2{  
    public static void main(String[] args) {  
        Bank b = new SBI();  
        Bank c = new PNB();  
        System.out.println("Bank A : " + b.roi());  
        System.out.println("Bank B : " + c.roi());  
    }  
}
```

↳ Inheritance
↳ Inheritance is a mechanism of reusing code
↳ Inheritance is a mechanism of reusing code
↳ Inheritance is a mechanism of reusing code

~~Final~~

~~OOP~~

- * Abstract / static / interface ✓
 (01.1 - 01.11) - bsw - A *
- * Exception / file - std - S *
- * Collection / framework - C *
 - array list - nr2 - N *
 - comparator - nr2 - T *
 - (05.1 - 05.8) (Kotlin) class - T *
 - (08.1 - 8) - nr2 - A *
- * static
 (11 - 08.8) - bsw - H *
 - method
 - block
 (11 - 08.8) - nr2 - G *
 - keyword, final keyword
 (1 - 08.8) - cont - V *
 - gui

Comparator

public class student -> ~~com~~ shortbyname{

implements Comparator<student> {

public int compare (student a, student b) {

return a.name.compareTo(b.name);

}

else if (a.rollno < b.rollno) {

return a.rollno - b.rollno;

else {

if (a.gpa > b.gpa) {

N #

(blit to)

GPA *
GPA increasing

GPA increasing straight

GPA highest