"Java"

→ pure Object Oriented

Date - 12-01-2025

```
public clam Main {
    public static void main (String [] angs) {
        System.out.println ("Vanendra University");
    }
}
```

Date : 14-01-2025

```
public clam Main {
    public static void main (String[] angs){
    int a=3;
    int b=5;
System.out.println ("Sum" + a+b) ;    // Sum 35
```

↳ String এরসাথে যা যোগ করব সেটাও String হয়ে যাবে।

```
System.out.println ("Hello" + "World") ; // Hello World
```

↳ ২ টা String, Single দিতে হবেএবং " " মাঝে দিতে হবে।

```
System.out.println ("Sum" + (a+b)) ;  // Sum 8
```

↳ potentiality বাড়ায়েদিচ্ছেতাই আগে () ভিতরের কাজ করবে, তারপর যেটা Stringএরসাথে মিলে Stringহবে

```
System.out.println (a+b);    // 8
    }
}
```

concact করতে / যোগ করতে আমরা clam use করি

viba

comment :— Multi line comment /*

Single line comment */

1 byte – 8 bit

boolean → bit
char → byte

- public class Main {

  public static void main(String[] args){

  byte b = 7; → 1 byte = 2^8
  (Data Type, name, variable)
  127 এর মধ্যে যে কোনো num রাখতে পারব

  Short s = 258; → 2 byte 2^{16}

  int i = 1024; → 4 byte

  long l = 9875321; → 8 byte

⊛ সবাই int num ই store করতে পারবে এবং store করবে শুধু memory space এর কম বেশি, মানে যতটুকু দরকার ততটুকু use করবে ঐ type এর মধ্যে, যদি 127 এর কম সংখ্যা হয় তবে byte এ কাজ করতে পারব, যদি তার চেয়ে বড় সংখ্যা হয় তবে Short এ কাজ করতে পারব, তার চেয়ে বড় হলে int এ।

  boolean bn = true;
  Char ch = 'B';
  float f = 3.95f;
  double d = 98.9753;

  float এ f দিতে হয় নইলে by default double হিসেবে নেয়

premetive data Type :- ৮ প্রকার : byte, Short, int, long, float, double, boolean, char.
→ basic

Non-premetive Data Type :-
Class
Object
Array
String
Interface

এগুলো premetive, non premetive এগুলো হিসেবে কাজ করে

String আগেই Capital ও লিখতে হবে।

int a = 5;

String s = "CSE"; → premative হিসেবে কাজ করছে

String batch = new String ("34b") → non-premative হিসেবে কাজ করছে

19.01.2025

Scanner in = new Scanner (System.in);

int n = in.nextInt();

System.out.println(n);

double d = in.nextDouble();

Obj ছাড়া input তৈরি করা যাবেনা

Math m = new Math()

line input নেয়া শেখাবে

String s = in.nextLine()

Declare

Initialization

Square s1,s2,s3; // Declare

s1 = new Square(); // Initialization

initialize না করলে obj দিয়ে কাজ করা যাবেনা। new keyword use না করলে Initialize হচ্ছেনা

Take an integer input n. Check the number is positive or negative.

Q. : Take an integer input n. Check the number is positive or negative.

⇒

```
public clars main {
    public static void main (String [] args){
    int n;

    Scanner in = new Scanner (System.in);
    int n = in.nextInt ();
    System.out.println (n);          // এইলাইন না দিলেও হবে, In মানেই nextLin
    if (n>0){
     System.out.println ("positive");
    }
    else{
    System.out.println ("negative");
    }
  }
}
```

Date: 21.01.2025

```
# int i;
for(int i=1; i<=10 ; i++){
    System.out.println(i);
}
System.out.println(i);
```

loop এর {} মধ্যে declare করেছি তাই int i কে বাইরে declare করতে হবে যেন loop এর বাইরে থেকে access করতে পারে
i কে print করলাম

1
2
3
4
5
6
7
8
9
10
11 হবে loop এর বাইরে হলে

1. Show the even numbers within 1 to 21.

⇒
```
public class Main {
    public static void main(String[] args){

~~Scanner in = new Scan~~
~~int n, 21 int n;~~
~~for (int 1;~~
for( int i=1; i<=21 ; i++){

        if (i%2 == 0){
    System.out.println ("i ~~Even number~~");
    }
  else {
System.out.~~println ("odd")~~;
  }
}
```

1. Show the even numbers within 1 to 21.

⇒
```
public class Main {
    public static void main(String[] args){
    for (int i=1; i<=21 ; i++){
        if (i % 2 == 0){
    System.out.println (i );
    }
    }
}
```

```
for (char c ='a'; c<='z' ; c++) {
    if(c== 'a' || c == 'e' || c =='i' || c=='o' || c=='u'){
    Sout ("vowel") ;
    }
else {
Sout ("Constant")
    }
}
```

*Infinity loop → `for ( ; ; ){ break; }`

`for (int i=0; i<=10; i--){ break; }`

infinity থামানোর জন্য use করি break.

"i এর মান 100 পেলেই break করে দেবে"

- int i = 5;

```
for( ; ; ){
    i++;
if (i == 100){

    break;
  }
}
```

Show 1 to 99 except 50.

```
int i;
for (i = 1 ; i <= 99 ; i++ ){
if (i == 50){
        continue;
 }
  System.out.println (i);

}
```

continue যেখানে থাকবে loop সেখান থেকে throw করবে

26.01.2025

Convert the following for loop to while loop without Changing the output.

initialization condition increment/decrement

```
for(int i=5; i<=25 ; i++){
    System.out.println(i);
}
```

⇒

```
public class Main {
    public static void main(String[],args){
    int i=5;
    while (i<=25){
    System.out.println(i);
        i++;
      }
    }
}
```

for loop এ আমরা চাইলে initialize করে নিতে পারি increment/dec করে নিতে পারি while এর মতো কিন্তু while শুধু condition ই থাকবে। while loop for এর মতন নিয়ন্ত্রণযোগ্য না

```
public class Main {
    public static void main (String[] args){
        Employee e = new Employee ();
        e.setID (1);
        e.SetName ("Nahid");
        System.out.println (e.getID());
        System.out.println (e.getName());
    }
}
```

class name obj keyword constructors

```
Square s = new Square();
Circle c = new Circle();
Math m = new Math();
```

Extra লিখেছি

declare — Initialize

কাজে দিয়েছি

class name বড় হাতের অক্ষরময়

Square S; → memory তে জায়গা assign করে declare

S = new Square (); → memory তে জায়গা

02-02-2025

Write a java program that will have a class Student. It contains the attributes id, name, batch. Also provide the set and get methods for all attributes.

Another class Main will have the main method to run the program

Ans:

```
public class Student {
    public static void Main(String[] args){

private int id, batch;
private String name;
private int batch;
 public void setID (int i){
        id = i;
{
public void setName (String n){
    name = n;
{
public void setBatch (int b){
        batch = b;
{
public int getID (){
        return id;
    {
public String getName (){
        return name;
{
public int getBatch(){
    return batch;
{ }
```

```
public class Main {
    public static void main (String[] args) {
        Student s = new Student ();

        s.setID(2);
        s.setName ("Tasnim");
        s.setBatch(34);
    System.out.println(s.getID());
    System.out.println(s.getName());
    System.out.println(s.getBatch());
    }
}
```

constructor

যদি salary থাকে 5% বাড়বে তাহলে

```
    int newSalary =
s.getSalary() + 5/100 * s.getSalary();
}
```

টাকা বাড়ানোর পর total salary কত আসবে সেটা

```
int extraSalary =
        5/100 * s.getSalary();
```

extra salary কতহবে?

Encapsulation
bind করে → রেখে রাখে

this keyword :-

```
public void setID (int id){
this.id = id ;
}
```

parameter

this → এই this id বলতে student class এর id কে বোঝায়।

"Encapsulation"

Date 04-02-2025

private
private

→ is a Mechanism to bind related code and data together. the attribute should be private and the method should be public

OOP Charecteristics :-

public 1. Encapsulation
public 2. Inheritance
public 3. Polymorphism
public 4. Abstruction

~~For privacy policy~~

For security purpose we declare attribute as private and method should be public.

Write an example program that will reflect Encapsulation mechanism.

```
public class Student {
private int id ;
private String name ;
    public void setID (int i){
        id = i ;
    }
}
```

```
    public void setName (String n){
        name = n;
    }
    public int getID(){
        return id;
    }
    public String getName (){
        return name;
    }
}
public class Main {
    public static void main (String [] args){
        Student s = new Student ();
        s.setID (05);
        s.setName ("Tasnim");
        System.out.println (s.getID ());
        System.out.println (s.getName ());
    }
}
```

```
public class Square {
    private int length;

    public void setLength (int length) {
        this.length = length;
    }
    public int getLength () {
        return length;
    }
}

public class Main {
    public static void main (String[] args) {

        Square s = new Square ();
        s.setLength (5);
        System.out.println (s.getLength());

    }
}
```

"Constructors"

```
public class Square{
    private int length;

    public Square(){
        System.out.println("Object Created");
    }

    public Square(int length){
        this.length = length;
    }
}
```

default constructors

Constructor has no returntype

→ parameterised constructor.

Square s1 = new Square(10)

এখন run করবে কিন্তু এখানে parameter দিলে প্রথমেও দিতে হবে default রাখা যাবেনা

যদি একবার parameterised বানায়ে নেই তবে পরেও parameter দিতে হবে. default থাকলে যদি দেই তবে আগেও default বানায়ে নিতে হবে।

* A class Square class has an attribute length. One default constructor that will set the length 0. One parameterized Constructor that will receive an argument and set that to lengths. The setLength and getlength method must have there. And another method getArea that will calculate and return area of the Square.

⇒

```
public class Square{
    private int length;
```

```
public Square () {
    length = 0;
}

public Square (int length) {
    this.length = length;
}

public int setLength (int length) {
    this.length = length;
}

public int getLength () {
    return length;
}

public int setArea ( int area) {
    this.area = area;
}

public int getArea () {
    return length * length;
}
}

public class Main {
public static void main ( String[] args) {
    Square s1 = new Square ();
    Square s2 = new Square (5);
    s1.setLength (10);
```

```
System.out.println (s1.getLength ());
System.out.println (s1.getArea ());
}
}
```

```
s2.getLen();
s2.getArea();
```

প্রথমে length 0 দিছিলাম পরে 10 দিয়েছি
তাই ২টা Square হবে ২টা object দিব

পরীক্ষায় Main চাইলে দিবে নইলে দিবে time waste করা No Need.

Date: 16.02.2025

```
public class Person {
    private int age;
    private String name;
Public int setAge(int age){
    = this.age = age;
}
get =
```

extends Person

```
public class Student {
    private int id;
    private String session;
    private String semester;
    private char section;
    private String dept;
set -

get -
```

Java তে inheritance করতে extends use করব → code reuseability

```
public class Teacher extends person{
    private int salary;
    private String designation;
    private String dept;
set =
get =
```

Inheritance :- Single
Multiple
MultiLevel → A → B → C
Heirarchy → A → B, A → C
Hybrid

Java তে inheritance 4 প্রকার। Multiple হয়না

Multiple Java তে support করেনা

১ টি class এর property অন্য class এ inherite করা

Inheritance is a process by which the attributes of one class can be acquired by another class.

Access Specifiers:

৪টা
- Default
- public
- private
- protected

class, attribute, methode গুলার আগে access specifier দেই।

Default টা public এর মতো কিছুটা কাজ করে।

* Relationship or Connectivity between Encapsulation and Inheritance.

⇒ Encapsulation is about whether the data is hidden, not whether the code looks similar.
But what inheritance achieves is essentially that, it makes sure the code is the same plus when we modify the code of A class we're modifying the inherited behavior of B, And that's what we want, that's why we have inheritanc

Employee {

Developer {

role;

team Name;

project Name;

Executive {

এমন ২/৩ টা class বানায় করবে

Data :- 18-02-2023

```
Public clam Employee{
    private int id ;
    private string name;
public  void setID(int id){
        this. id= id;
    }
public void setName (String name){
    this. name =name;
  }
public   int getID (){
      return id;
  }
public  String getName(){
      return name;
  }
```

```
public clam Main {
  public static void main(String[]args){
  {
Employee [] e =new Employee [3]

  e [0] . setID (5);
  e[0] . setName ("Ni");
  e[1] . setID ("10");
  e[1] . setName ("Ri");
System. out. println (e[0]. getID ());
System. out. println (e[0]. getName ());
```

int x,y,z ;

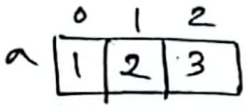
int [] a = new Int [3]

x = 1

y = 2

z = 3

System. ... (x)



int [] a = new Int [3]

a [0] = 1 ;

a [1] = 2 ;

a [2] = 3 ;

Student
{
int id
int
}

Suggestion Date : 04.03.20xx

Write a program that will calculate the factorial of a number.

→
```
public static void main (String [] args){

    public clas Main {
     int i ;
     int n ;= 1;

    for ( int i = 1; i <= 3 ; i++ ){

       n = n * i ;

    }
  System.out.println ( n );
```

1*1*2*3
=6
3 এর factorial

⇒
```
int i ;
int n = 1 ;
int num = 5 ;
for ( i = 1 ; i <= num ; i++ ){

    n = n*i ;
  }
Sout (n)
}
```

কিছু না দিলে default 0 ধরবে।

ternary Operator
?:

public class Numeric {

public static boolean isPositive (int n) {

return n >= 0 ? true : false;

}

public static boolean isEven (int n) {

return n % 2 == 0 ? true : false;

}

}

public class Main {

psvm (String[] args) {

Numeric n = new Num ();

যদি static মেথডগুলো তাহলে এই line লাগবেনা

System.out.println (n.isPositive (5)); // true

" " (n.isEven (5)); // false

true
false

Static মিলে, numeric.n এমন মিল হবে

Sort (numeric.isPositive)

}

}

static মিলে obj create করতে হবেনা, class name . object name দিলে

" or . mathode " "

Static না দিলে

Static দিলে

```
# public class Numeric {
   public static boolean isPositive(int n){
      return ——
   }
  public static boolean isEven (int n){
  return ——
}
public class Main{
—— main (String [] args) {
Sout —— (Numeric. isPositive (5)); //true
Sout — (Numeric. isEven (5)); //false
```

Static variable memory তে একবার নেবে ৩টা

সব class এর জন্য একটা dept ই হবে অনেকগুলো নিতে হবেনা

```
# public class Student {

    id
    name
    static dept
```

```
public static void main (String [] args) {
```

* static keyword have some important role in Java programming – Explain

Suggestion for Mid

Class
Object
Encapsulation
Inheritance
Constructors
this
static
Array
String
attribute
Methode
Continue
break
Array of Object

for
while
if
else
Access Specifier
Data type
String function
Declare & initialization
Naming Convention
input
output
Type casting (String থেকে int, double থেকে int...
Operators → (Ternary operator, arithmetic operator...)

* valid invalid identifiers (name + Example)
*