Topic Name : POP vs OOP

Day : .

Time : Date : / /

Paradigm and its types

some common paradigm :

→ Imperative → procedural

→ Declarative → object-oriented

→ structured

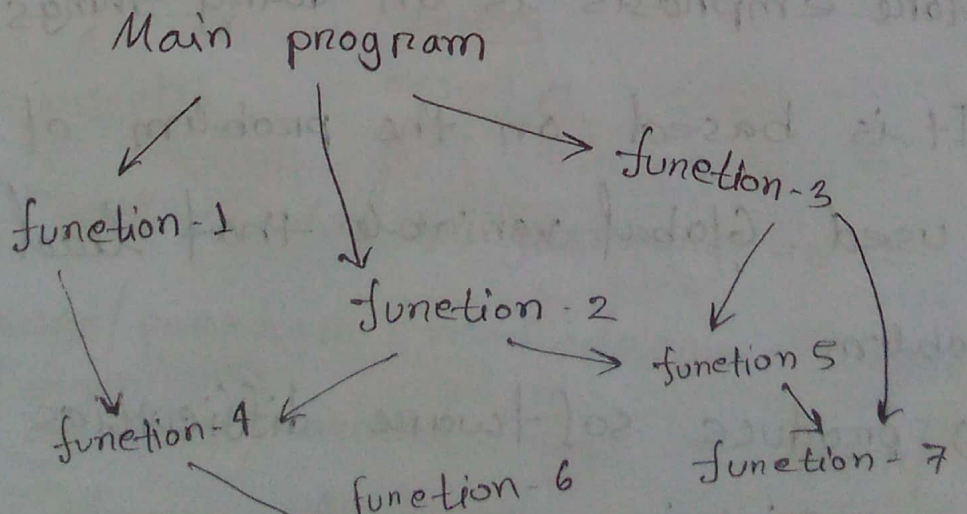
A paradigm is a way of doing something (like programing)

* pop → procedural oriented programming → other name

structured " "

→ A program in procedural is a set of instruction

→ The program are devided into sub program called functions.



→ what have to do to solve this problem?

Topic Name : Day : Time : Date : / /

problems in pop:

→ Global variable which causes problems

→ uncontrolled modification (like: when a function uses a global variable the variable becomes changed and a new function can't uses the actual value because the value is already deviated by function)

in global → $x=5$, after using a function its value now $x=3$, now a new function can't get $x=5$, because its value changed by a function it can use $x=3$.

→ problem of enhancement & maintainability

→ code re-usability

Major characteristics:

→ More emphasis is on doing things

→ It is based on the problem at hand.

→ used Global variable that lead following problem

(i) produce software difficulties to maintain

Topic Name : _____ Day : _____

Time : Date : / /

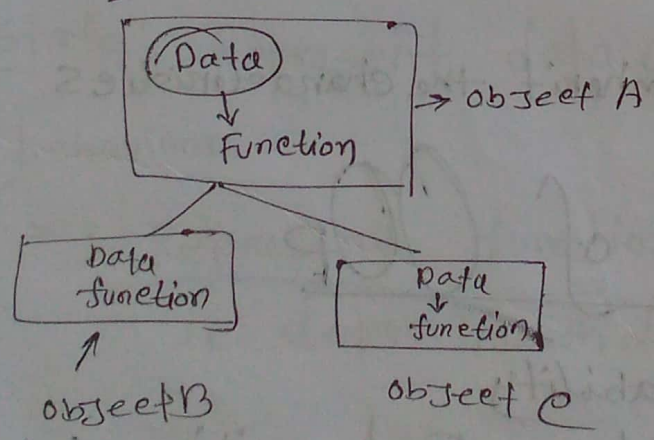
(ii) The risk of unwanted access & modification increases.

→ It will not result in reusable software

→ It employs top down approach in program design

→ works well in small system.

OOP



⇒ Object oriented programming is a programming language/paradigm / methodology in which program are (organized / oriented) around data or object rather than function & logic.

Topic Name : Day : Time : Date : / /

→ OOp refers to type of computer programming in which programmer define not only the data structure but also the types of operation (i.e. means function here) that can be applied to the data structure

→ The data structure becomes an object that include both data & function

→ programmer can create relationship ~~among~~ between objects.

→ object can inherit the characteristics from other object.

Advantages of OOP

→ code reusability

→ Data Redundancy [not writing a code that is no longer needed]

→ code maintanance

→ security ⟶ using abstraction, encapsulation

→ Design Benifit

→ polymorphism flexibility

API = Application programming Interface.

cls- 3

Topic Name : 05-01-22

Day :

Time : Date : / /

Basic feature of oop:

→ class
→ object
→ Inheritance
→ Encapsulation → main feature
→ polymorphism
→ Abstraction

Object:

→ Object is an instance of class

→ The object is an entity which has state & behaviour

→ object has the state & behaviour

→ : state: Represent data (value) of object. ie. Balance

: behaviour

↳ : Represent functionality of an object ie. deposit, withdraw

Ex:

Pen

State: color, brand, size (noun)

behaviour: writing → (verb)

In terms of ~~data~~ software:

state : data

behaviour : functionality on data

class:

→ A class is a blueprint / prototype from which object are created.

→ A class is a blueprint, idea & prototype for creating object.

→ A class is a group of object which have common properties.

Example:

→ sketch (prototype) of house

→ It contains all details about floor, door, windows etc.

→ Based on this description, we can create house.

→ Here, House is the object.

Topic Name :

Day :

Time : Date : / /

```
Class student {
    string name;
    int ID;
    int age;
    };
```

object :

```
class Name    objectName = new
                  className();
```

```
student object1 = new student();
    object1.name = " Rahim ";
    object1.ID = 50;
    object1.age = 20;
```

Encapsulation :-

- The wrapping up of a data & function into a single unit is called Encapsulation
- The data is not acessible outside the class

Topic Name : Day : Time : Date : / /

Abstraction → ~~Represent~~ Define the representation of essential feature without including background details or explanation.

→ Inheritance: Refers to a process in which object of one class acquires the property of another class

Polymorphism: It means the ability to take more than one form.

The complete reference Java, H. Schift

Programming with Java latest edition. E. Balagurusamy

Java how to program Deitel & Deitel

Topic Name : Day : Time : Date : / /

Basic feature / properties of Java:

→ compiled & interpreted → source code (Java) → byte code (compiling) → Machine code (interpretation)

→ platform independent → JVM → byte code (readable)

→ object oriented

→ Distributed

→ High performance

→ simple & familiar

→ Multithreading → একসময় অধিক program at a time compile করতে পারে

Application :

→ Mobile application → swing, AWT, JavaFx

→ Desktop " → (No. need of internet)

→ web-based " → (Must have ")

course → Scientific " → JEE, servetlet, Jsp

→ Gaming "

→ cloud-based "

→ Business "

Topic Name : _____ Day : _____ Time : Date : / /

সবকিছুই token

→ variable
→ Token
→ Datatype
→ Identifier
→ operator

→ control statement (if, else, switch...)
→ loop statement (while, do, while, for)

Definition for exam

Token: Token is basically smallest element that can Identified by the compiler.

Token → keyword → reserved words: int, ch
→ Identifier → variable বা function এর নামগুলো যেগুলো user দিবে
→ constant
→ String
→ operator → +, -
→ special symbol → (), {},

Topic Name : _____ Day : _____ Time : Date : / /

Identifier: → is a sequence of character

→ are programmer designed token.

→ used for naming class, method, variable, object, packages etc..

Keyword: reserved word / Identifier

→ we can't use them as identifier

Ex: int, if, switch etc

variable: A variable is an Identifier that denotes a storage location used to store data value.

→ Their value changes during the execution of program.

→ Every variable must be declared before used.

→

→

Topic Name : _____ Day : _____ Time : Date : / /

Rules: → Variable names may ~~contains~~ consists of alphabet, digit, underscore, & dollar sign

→ upper case & lower case are distinct

case sensitive
↳ a, A are ~~chang~~ different

→ It should not be keyword

→ ~~white~~ space not allowed white

→ variable name can be ~~only~~ any length.

→ They must not begin with a digit.

camelCase is better for variable Naming

Ex: firstName rather than firstname

Topic Name : _____ Day : _____ Time : Date : / /

Data types:

Data type specifies the size & type of values that can be stored in variable.

Data type

- primitive
  - Ex: int, float, character, boolean, integer
- Non-primitive
  - Ex: class, interface, array, string

Operator: An operator is a symbol that tells the computer to perform certain mathematical or logical operation.

→ used in program to manipulate data & variable.

CamScanner

Topic Name : Day : Time : Date : / /

→ Arithmatic operator : +, -, /, *

→ Relational " : ==, !=, >, <, >=, <=

→ logical " : &&, ||, !

→ Assignment " : =, +=, -=, *=

→ Increment / decrement " : ++, --

→ conditional " → ? :

→ Bitwise " → &, |, <<, >>

→ special " → .

program structure

Topic Name : _____ Day : _____ Time : Date : / /

→ Documentation section
→ package statement
→ import statement
→ interface "

এইগুলো সবসময় না ও থাকতে পারে।

→ class definition
→ main method declaration

always থাকবে

Packages create না করলে defaul package কাজ করবে।

→ এখানে, variable declare করলে by default public static final হয়ে

→ method → default হয়ে always

→ abstract method.

Abstract method এ শুধু declaration থাকবে।

interface ও class প্রায় একই but not same

Topic Name : Day : Time : Date : / /

access modifier

keyword

class name

Not changeable

```
public class Hello{
    public static void main(String args[])
    {
        System.out.println("Hello to Java");
    }
}
```

class here

method তে public

class

object

method

এটা change able

object create ছাড়াই আমরা এই গুলো access করতে পারব

Topic Name : _____ Day : _____ Time : Date : / /

→ The keyword public is an access modifier

→ The keyword static means that a method is accessable and usable without using object creation.

→ Void means main method does not return any value.

JDK = Java development kit

JRE = " Run-time Environment

import Java.util.*;

↓ packages

* দিলে util ~~এর~~ package এর সব class include হয়ে যাবে।

CamScanner

+ qv

Topic Name : Day : Time : Date : / /

```
import Java.util.Scanner;
public class Math1{

public static void main (String args[])
{
  Scanner tv = new Scanner (system.in);
  int a, b, c, sum, avg;
  a = tv.nextInt();
  b = tv.nexInt();
  c = tv.nexInt();

  sum = a+b+c;

system.out.println("sum = " + sum);

avg = (a+b+c)/3;

system.out.println(" avg = " + avg);
}
}
```

Topic Name : Day :
Time : Date : / /

```
~~Package mbstu;~~
import java.util.*;
public class max {
public static void main (String[] args){
  int a,b,c;max;
  Scanner z = new Scanner(System.in);
   a= z.nextInt();
   b= z.nextInt();
   c= z.nextInt();
  if (a>b && a>c)
  System.out.println("max = "+a);
  else if (b>a && b>c)
   System.out.println(" max= "+b);
   else
   System.out.println("max = "+c);
  }
}
```

Topic Name : _____ Day : _____ Time : Date : / /

```
import Java.util.*;
public class odd_even {

public static void main (String[] args){

Scanner tv = new Scanner (system.in);

int a, b;

a= tv.nextInt();

b= tv.nextInt();

if (a%2==0)

~~System.out.println("Even" "~~ o

system.out.print line (" a+" is Even);

else

System.out.println (a+" is odd");

if (b%2==0)

System.out.println (b+" is even);

else

system.out.println (b+" is odd");

}}
```

Topic Name : _____ Day : _____

Time : Date : / /

Static হলে object create করা(ও হয় না।

class {

~~public static void main(String arg[ ])~~

```
import java.util.*;
Public class Summation {

int totalsum (int n) {
    int sum=0;
  for (int i=1; i<=n; i++)
    { ~~int~~ sum = sum+i;
    }
  return sum;

public static void main (String args[]) {
    int n;
  Scanner tv= new scamer(system.in);
    ~~tv~~ ~~tv~~ n = tv.nextInt();
  Summation obj = new Summation();

  System.in.println(" Sum = ",+obj.summation);
  }}
```

Topic Name : Day :
Time : Date : / /

10 may class

Problem :

* write a java program that initialize a variable balance, account number, Account holder Name.

→ credit to balance 10000 tk using method name
creditAccount();

→ debit from Account 5000 tk using method Name
debitAccount()

→ show the balance with all information using method showBalance();

* **Static keyword:**

→ static keyword used in Java mainly for memory management;

→ static member function belongs to class instead of specific instance/object;

→ Static member can access without object;

→ static keyword in java can be used in :

1→ variable
2→ method
3→ nested class
4→ block

Topic Name : Day : Time : Date : / /

Nested class

```
class Test {
int x = 20;
static class Hello {
  void show () {
   print(x);
   }
  }
}
public static void m_____
{
Test. Hello obj = new Test. Hello ();
      obj. show ();
}
```

18-5-22

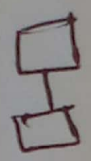
Topic Name :

Day :

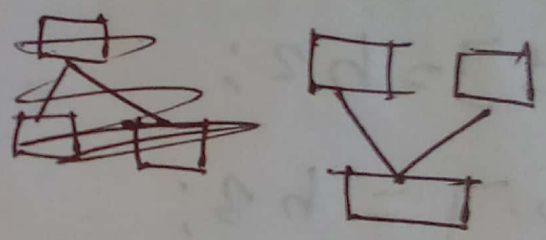
Time :

types of inheritance:

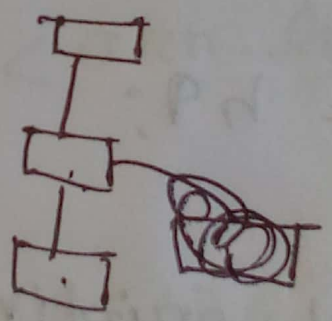
1. ~~Mu~~ single



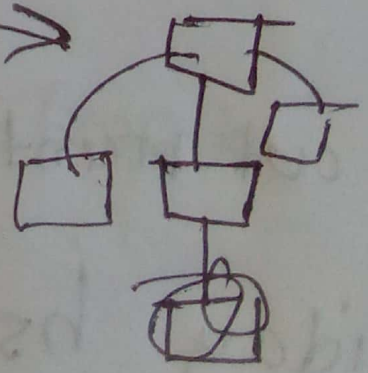
2. Multiple



3. multilevel



4. Hierarchical



5. Hybrid

(1)

```
class parentName
{
  String n = "Touhid";
  } Public class name extends parentName
public static void main (String [] ags)
{
  name obj = new name();
    System.out.println("Name = " + obj.name);
}
}
```

(2)

```
class val
{
  int a = 5, b = 7;
  }
public class sum extends val
{
public static void main (String[] ags)
    sum o = ne sum();
    system.out.println(" Sum : "(+ obj. a+ obj. b))
  }
}
```

Topic Name :

Time :

Multilevel:

```
1: class name 1
{
  String Name = "Touhid";
}
class Roll extends name 1
{
  String Roll = "IT-20020";
  } public class Main extends Roll{
public static void main (String [] args)
{
  Main obj = new Main();

System.out.println("Name : " + obj.Name);
  system.out.println("ID : " + obj.ID);
}
```

Topic Name : Day : Time : Date : / /

Hierarchical:

1:

```
class bird
{
  void b()
  {
    System.out.println("Hi, I am a bird");
  }
class crow extends bird
{
  void c()
  {
    System.out.println("and my name is crow.");
  }
}
class cuckoo extends bird
{
  void ck()
  {
    System.out.println(" and my name is cuckoo");
  }
}
public class h1{
```

Topic Name :

Day :

Time :

Date : / /

```
Public static void main (String[] args)
{
  crow obj = new crow();
   Obj.b();
    obj. e();

   cuekoo obj1 = new cuekoo();
    obj1. b();
    obj1. ek();
  }
}
```

polymorphism in Java ~~can be defined~~ is the ability ~~to take~~ of an object to take many forms.

Topic Name: polymorphism: poly = many, morphs = forms

Day: 30-may-21

Time: Date: / /

types of polymorphism:

* compile time polymorphism is a polymorphism that is resolved during the compilation process.
* Runtime " " → is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

Ex:

→ method overloading → একই নামে function থাকবে তবে parameter ভিন্ন

→ refers to defining different methods in a class with same name but different parameter list

Method overriding → parent ও subclass এ একই signature এ method থাকবে

Return type doesn't matter

Signature must be same as the parent class and return type also.

→ when subclass has the same ~~the~~ method as the parent class. ~~both class method have same signature~~

Method signature → :

→ is the combination of the method name and the parameter list.

→ Method overloading:

① parameter এর সংখ্যা ভিন্ন থাকবে { Sum(int x), Sum(int x, int y) }

② parameter এ datatype ভিন্ন হবে (Sum(int x), Sum(float x))

③ datatype sequence (sum(int x, float y), sum(float y, int x))

④

```
class d-Sem{

static sum (int a, ~~int~~ float b)
    {
     float s = a+b;
    System.out.println(s);
    }

ic sum (float a, int b)
    {
     float S = a+b;
    ~~sout~~ System.out.println(S);
    }

public static void main(String [] args)
{
   sum (5, 7.6);
   sum (5.55, 9);
}}
```

Topic Name : _____ Day : _____

Time : Date : / /

Method overriding:

→ Method name must same as parent class & child class.

→ Return type must be same for all

→ final method can't be overridden

→ static " can't be overridden

→ private method can't be ".

```
→ class person{
    void show()
      {s.out ("I am from person class show");
      }

class student extends person{
    void show()
      {
        s.out("I am from student class show");
      }
```

Topic Name : _____ Day : _____ Time : Date : / /

```
psvm ()
{
    student ob = new Student ();
    St. show();   [show from student
    person pt = new person ();
    pt.show();   [show from ~~sto~~ person
    ~~student~~ person ~~Person~~ pt2 = new ~~Student~~ Student ();
    pt2. show ();
```

→ student class এর object

run-time polymorphism

~~Super~~ sub class এর সমান ~~subclass~~ super এর Reference রাখতে পারব

```
    Person (pt2) = new Student ();
```

→ Reference of subclass

~~Student ps = new person();~~

এরকমও পারে

Topic Name : _____ Day : _____ Time : Date : / /

* super keyword:

→ super is used to refer parent class object

→ used to invoke parent class variable

→ used to " " " method

→ " " " " " ~~method~~ constructor

```
class person {
    int n = 20
    void show () {
        s.out("I am from pangon show)
    }
}
class student extends person {
    int n = 10;
    void show () {
        s.out ("I am from student show);
    }
```

```
void display(){
  S.out("~~I am from display~~"); X="+super.x);
  super.show();
  }
public static void main(String[] args)
{
  Student st = new Student();
  st.display();
  }
```

X=20

I am from person show

Topic Name : Day : Time : Date : / /

final keyword :

⇒ final keyword used to restrict the user.

⇒ variable → final int n=100; → চেঞ্জ করতে চাইলে compile time error হবে
→ method →
→ class

→ final int n; এরকম ভ্যারিয়েবল initialize করতে হয় constructor use করে। এটি blank final variable

* final method
* final n কে subclass এ override করা যাবে না।
* class → final হলে ওই class কে inherit করা যাবে না।
* final static int n;

static {
n=100;
}

variable যদি static হয় তো static variable final হলে static variable কেবল static block দিয়ে initialize করতে হয়।

Topic Name : _____ Day : _____

Time : Date : / /

Integer, Float, Double

wrapper class

→ এরা final class

→ method overloading এর দুইটি

→ " overriding " 1

→ super keyword এর use [var, meth, constructor]

→ final এর ক্ষেত্রে variable দিয়ে একটি

এক্ষেত্রে, blank final variable

final static "

→ final method কে overrule করা যায় না

এর implemen

→ " class কে inherit করা " "

এর ক্ষেত্রে

31 may - 22

Topic Name: Abstraction

Day:

Time: Date: / /

→ Abstraction refers to the representation of essential feature without including background details.

→ Abstraction is a process of hiding the implementation details and showing only functionality to users.

→ ~~Abstract~~

→ Abstract class → (0-100)% achive করা যায়

→ Interface → use করে 100% Abstraction achive করা যায়

abstract class এ final, non final, static, non-static সব ব্যবহার করা যায়।

concrete class = Normal class

Normal class কে abstract ~~subclass~~ class inherit করতে পারে।

Topic Name : Day: Time : Date : / /

~~class person~~

A abstract class person {

abstract void display ();

void show () {

}

}

class student extends person

{ abstract class কে inherit করলে 3২ class এর সকল abstract method এর implementation or body 3২ sub class এ থাকতেই হবে। তবে subclass টা যদি abstract হয় তবে আর সব abstract method এর implementation/body থাকা বাধ্যতামূলক নয়। ~~উক্ত subclass এ~~

void display () {

s.out (" ");

}

}

abstract class কে আরেক abstract class inherit করতে পারবে।

তবে 3২ subclass এর নতুন sub class এ parent class এর abstract method এর body থাকা must.

we can't create object of Abstract class.

there could be abstract method inside abstract class.

Abstract + method = abstract method

Abstract method এর body থাকবে না অর্থাৎ শুধু declaration থাকবে তবে definition থাকবে না।

Normal method এ body/implementation থাকে।

Abstract class এ Normal and abstract method থাকতে পারে।

concrete class এ শুধু normal method থাকে তবে abstract method থাকবে না।

Topic Name : Day : Time : Date : / /

Interface: → we can achive 100% abstraction

→ looks like class but not class but (works ~~as~~ like a class)

~~* interface~~

interface interfaceName{

int x = 20;

void show();

void display();

}

→ there will be no normal.

~~Abstraction~~ interface এর variable গুলো automatic public static final হয়ে যায়।

Ex:

public static final int x = 20;

method এর আগে automatic public abstract বসে যায়।

Ex: public abstract void show();

নিজে লিখে দিলে আর compiler automatic এইসব prefix add করবে না

(interface can't be instantiated
interface এর object create করা যাবে না।)

Topic Name : Time : Date : / /

```
interface Shape {
  int x = 20;
  void show();
  void display();
}
```

class interface কে extends করবে না। implements করবে

```
class Rectangle implements Shape {
  void show() {
  }
  void display() {
  }
}
```

এখানে, interface এর abstract method এর body/implementation থাকতে হবে।

একটি class একই টাইপের একাধিক interface কে implements করতে পারে।

ex: class Rectangle implements shape, circle.

we have to add public ~~when~~ in method in implementation ~~happ~~

Topic Name : _____ Day : _____

Time : Date : / /

multiple inheritance: through interface

~~class~~ interface A {

void show();

}

interface B {

void show();

}

class e implements A, B {

যেকোনো একটি show থাকবে

}

interface অন্য interface কে extends করতে পারবে

interface circle extends shape, Savare {

}

দুইটিকে extends করেছে একটিকেও করা যাবে।

we have to add public when we implement a method

Topic Name : Diamond problem

Day :

Time : Date : / /

Interface : multiple inheritance

```
interface Rectangle{
    void show();
    }

interface circle{
    void show ();
    }

class shape implements Rectangle, circle {
public static void main (){

    shape
public void show (){
        s.out. ("Hi");
        }
    psvm (){
        Shape obj = new Shape ();
        obj. show ();
        }
```

default method!

Diamond problem

[যেকোনো একটির show();
interface হতে হবে বলে]

Topic Name : _____ Day : _____

Time : Date : / /

packages:

→ ~~Pa~~ Package in Java is a mechanism to encapsulate a group of class, ~~as~~ sub package & interfaces

→ package keyword used create package in Java.

→ a class can have only one package declaration

→ Every class is part of ~~sum~~ some package;

Advantage;

→ preventing harming conflict

→ Better organization

→ providing controlled access

→ Reusability

sub package :- ?

A package under a package

Topic Name : Day : Time : Date : / /

* Package!

```
package IET;

class student{
    int n=20;
    void show (){
    {
    }

class Employee{
    void display()
    {
    }
}
```

new file in another package:

class person

```
package ICT21;
import ICT.*;

class person extends student {

psvm(){
    student st = new student();
    st.show();
}
```

→ ভবিষ্যতে particular class কেও import করা যায় → import IET.student;

এর মধ্যে সকল class কেই import করবে।

→ import এর নীচের লাইন এ

→ উপরে import না করলে

object create এর সময়

PackageName.ClassName = new PackageName.ClassName();

Ex: Iet.Student obj = new Ie.student

Topic Name : _____ Day : _____ Time : Date : / /

sub package homework;

Access modifier:

→ default → when we don't declare access modifier → Same package এ access করা যাবে।

→ public → দুনিয়ায় সব package থেকেই access করা যাবে।

→ private → 3টি class এই কাজ.

→ protected → Same package and this package's class's Subclass of another package.

Ex:

```
package Iet;
class p{
protected void show(){
    }
}

package Iet2;
import Iet.*;
class student extends p{
    we can access show() here; as it is
    subclass of p and show is
    protected
}
```

Topic Name : Day : Time : Date : / /

* **this** → current class or current instance

```
class person {
    int x, y, z;
    person ( int x , int y)
    {
        this. x = x;
        this.y = y;
    }

    void show () {
        s.out (x,y)
    }
    void display () { this. show ();}
}
class student extends person {
    psvm ( ) {
    person p = new person(10, 20);
```

this using method

Topic Name : _____ Day : _____

Time : Date : / /

```
Class person (int x, int y){
    this.x = x;
    this.y = y
    }

person ( )
    {
    this (5,7);
    }
```

~~this and super is similar here~~

এই class এর constructor এ দুটি মান pass করবে।

```
person p = new person ( );
```

Topic Name : * ~~60~~

Day :

Time : Date : / /

06-06-22

Exception handling:

একটি try এর মধ্যে অনেকগুলো exception থাকতে পারবে।

finally block always execute হবে exception থাকুক বা না থাকুক

catch block শুধু exception থাকলেই execute হবে।

throw vs throws

~~Interface~~

multiple inheritance?