09.01.20

(*) Java doesn't allow non-member function

(*) " " support structure

Architecture of Java

(*) Command line argument.

f.cpp → a.out / f.out.
↓ ./a.out

(*) Java is platform Independent.

(*) Java interpreter is called JRE.

JDK → Java Development Kit
JVM → Java Virtual Machine
JRE → Java Runtime Environment

~~typecasting~~

float f = 3.5

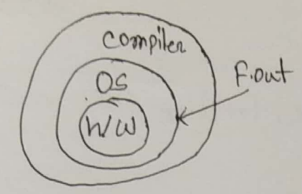
(*) Java type sensitive language.

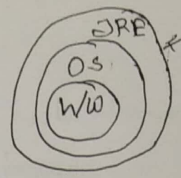
Program: file name ~~test~~ Test.java

```
1. public class Test
   {
      public static void main(String args[])
      {
         System.out.print("Hello");
      }
   }
```

javac Test.java → complie java file.
after compilation → Test.class
java Test → show output / execute

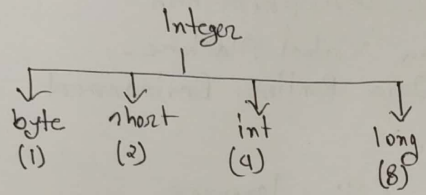
(*) In Java program is not a process, Java itself.





(20) in Java if a class is there in a files
" C/C++ " a " " " " 1 files

(21) Protected & ~~public~~ private class innot defined.



char (2)

(24) Java character is called unicode

boolean
~~true~~
False

2. class Test
{
public static void main (String args[])
{
int i = -3
if (i)
print ("i");
}
}

bitwise Operator in Java:

|
&
~
^
<<
>>
>>> → unsigned right shift.

(*) explicit casting

(*) pointer is not supported in Java.

(*) destructor is not available in Java / operator overloading also.

(*) Java by default all methods are virtual.

(*) Difference between command line argument in Java & C/C++.

Java test +2 .5]
↑ ↑
args[0] args[1]

```
byte b1=2, b3=3, b4;
b4 = b1 + b3;
b4 = (byte) (b1+b3)
```

3.
```
class Test
{
  public static void main(String args[])
  {
    int i, j, k;
    i = Integer.parseInt(args[0]);
    j =
    k = i+j;
  }
}
```

Integer is class
parseInt is function name.

(*) How to create object

```
A *p
p = new A
```
{ in CPP

Cl.

```
class Complex
{
private int x, i;
}
class Test
{
    main (String args[])
    {
        complex O1, O2, O3;
        O1 = new Complex ();
        O2 = new Complex (2, 3);
    }
}
```

Q.N

—.—

(5) Function Overloading:

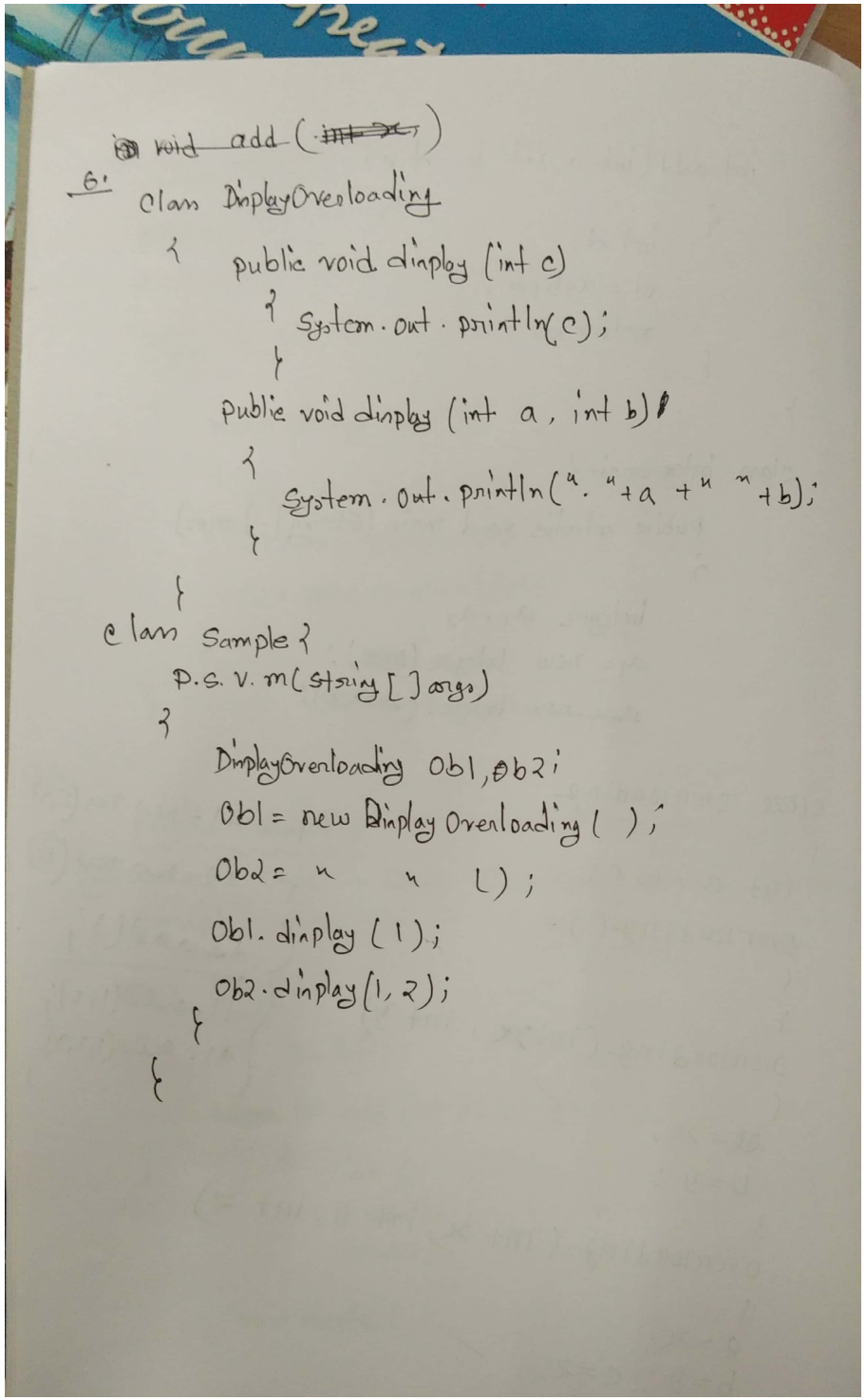
(a) Write a code in Java where using function overloading 14.01.2020
concept.

```
class Integer{
    int a, b, c;
    int add (int a, int b)
    {
        int c;
        c = a+b;
        return c;
    }
void display (
```

```
int add(int a, int b, int c)
    {
        int d;
        d = a+b+c;
        return d;
    {
}
class Integermain{
    public static void main (String[] args)
    {
        Integer a1, a2
        a1 = new Integer (5,2);
        a2 = new Integer (5,2,3);
```

```
class overloading
{
 int a, b, c;
 overloading()
 {
 }
 overloading (int x, int y)
 {
  a = x;
  b = y;
 }
 overloading (int x, int y, int z)
 {
  a = x;
  b = y; c = z;
 }
```

```
Test t1 = new Tes(2,3)
    t2 = new test(...)
{
  t1.add();
  t1.add(1,2);
  t1.add(1,2,3);
}
```



```
void add(
6. class DisplayOverloading
   {
      public void display (int c)
        { System.out.println(c);
        }
      public void display (int a, int b)
        {
          System.out.println(" " + a + " " + b);
        }
   }
class Sample {
   P.S.V.M(String[] args)
   {
      DisplayOverloading ob1, ob2;
      ob1 = new DisplayOverloading ( );
      ob2 = "  "  ( );
      ob1.display (1);
      ob2.display(1, 2);
   }
}
```

② Constructor Overloading

```
class Complex
{
    int r, i;

    Complex ( )
    {
    }

    Complex (int a, int b)
    {
        r = a;
        i = b;
    }

    int add (Complex ob)
    {
        Complex ob1;
        ob1.r = r + ob.r;
        ob1.i = r + ob.i;
        return ob1;
    }

    void display ( )
    {
        SOP( r "+" i );
    }
}

class complexmain()
{
    psvm (String [] args)
    {
        Complex ob1, ob2, ob3;  ob3 = new complex();
        ob1 = new Complex(5, 7);
        ob2 = new Complex (3, 4);
        ob3 = ob1.add (ob2);
    }
}
```

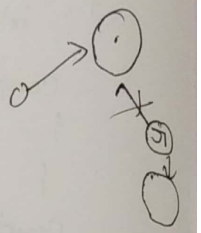
this.r = r

16.01.2020

A 0
A *p
P = new A.

(*) in Java Object name itself in a pointer.

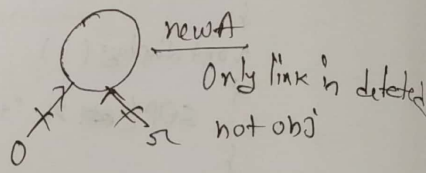
A = 0; → Reference
0 = new A();
A = π ; → reference
π = 0;
π = new A



A *p, *q;
P = new A;
q = P;
free(P); / delete(P)
q → f(); dangling pointer.

A 0; → reference
0 = new A()
A π; π = 0;
0 = null;
p = null;

Unreferenced object



(*) Java is basically multithreaded environment.

Java Test (*) in Java program is a thread.
↑ actual process
↑ thread.

(*) System defined thread → garbage collector

(*) We can manually call garbage collector.

System.gc();
↓
static function

int a[10]; → static array

int * p;

p = malloc( ) . . . ; → C

p = new int[10]; → C++

(*) in Java no static array in there.

int a[10]; X

int a[];

a = new int[10];

2d array:

int a[][] = new int [4][5];

when, int a[][];

a = new int[4][];

for(i=0; i<4; i++)

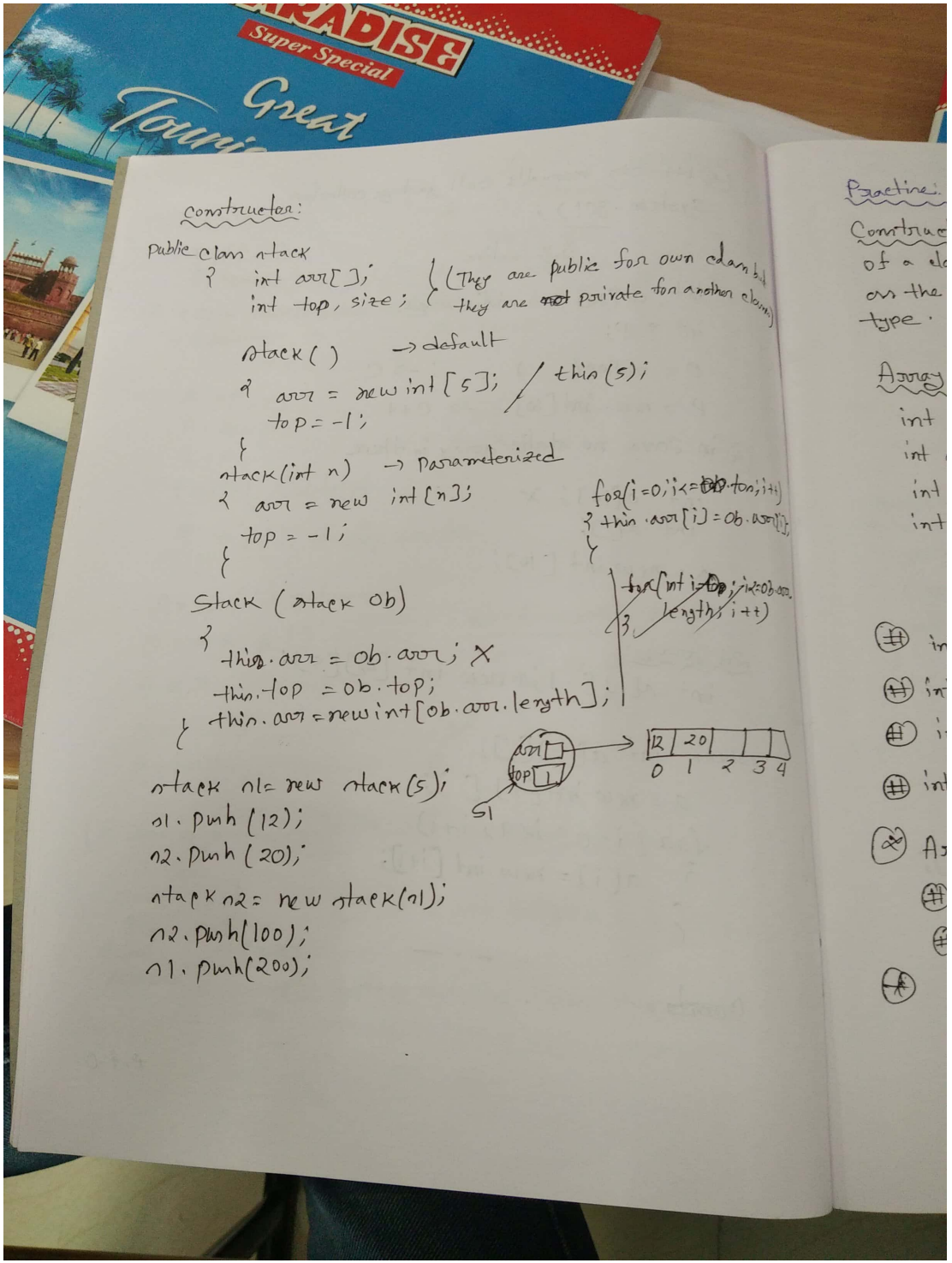
{ a[i] = new int[i+1];

{

- - -

Combra.

P.t.O

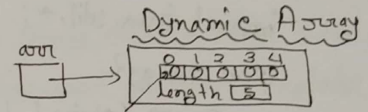


Practice:

Constructor: Constructor is a member function of a class. The name of the constructor is name as the name of the class. Constructor has no return type.

Array in JAVA:

```
int arr[5];  ✗
int arr[];  -> reference variable.
int []arr = new int[5];  -> 5 size array
int arr[] = new int[5];
```



arr[0] =
arr.length -> array size.

(#) int arr[] = new int[]{2,4,6,8,10};

(#) int arr[] = new int[2]{2,4,6,8,10}; (Error)

(#) int arr[] = new int[5]{2,3,4,5,8}; Error: can't mention size & value together.

(#) int arr[] = {2,3,5,4,8}

(*) Array is not blank.

(#) int []arr = new int[3];

(#) S.O.P("arr[0] =" + arr[0]);

(*) public class Exam {
    psvm (String []args) {
        int arr[0];
        arr[0] = 25;
        arr[1] = 50;
    }
}

Find Error?

Ans: Compilation error bez array is not ~~creating~~ made.

(2) Initialize after declaration:

```
Public class Array {
    psvm (String [] args)
    {   int arr[];
        arr = new int[3];
        arr[0] = 10;
        arr[1] = 20;
        arr[2] = 30;
    } }
```

User input Array:

```
import java.util.*;
class Array{
    psvm (string [] args)
    {   int [] arr = new int[5];
        Scanner sc = new Scanner(System.in);
        SOP("Enter 5 numbers");
        for(int i=0; i<5; i++)
            arr[i] = sc.nextInt();

        for(int i=0; i<5; i++)
            System.out.println("arr[" + i + "] = " + arr[i]);
    }
}
```

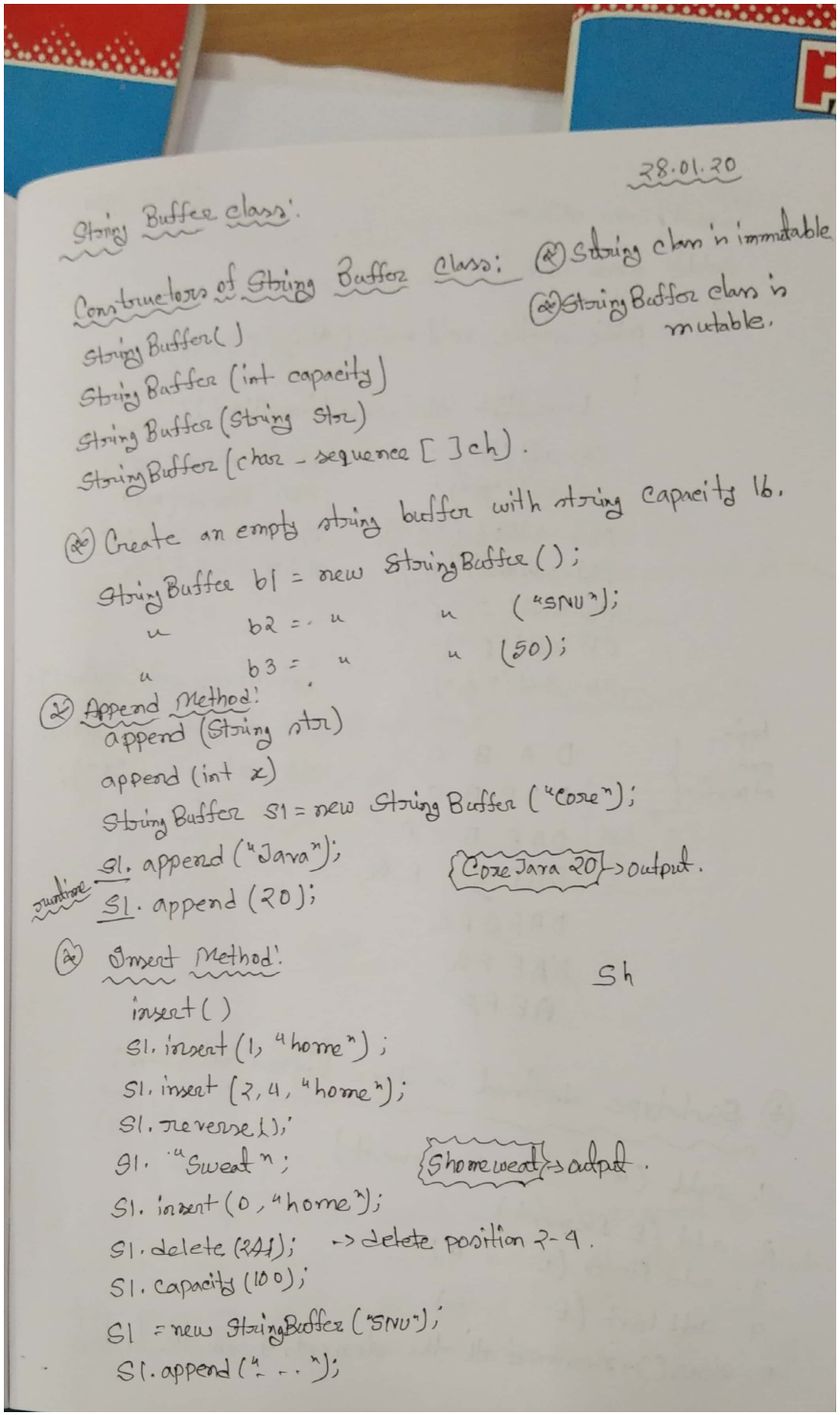
21.01.20

String class:

```
String S= new String ("Hello");
S.O.P (S);
String S1 = new String ("world");
S.O.P (S1+S2);
char [ ]C = {'a', 'b', 'c'};
String S2 = new String (c); → abc
boolean equals (String Str) → case sensitive.
int length()
char charAt (int index);
```

Assignment:

Write at least 10 methods in String class & String Buffer class methods (which are not discussed in class), Explain with example.



28.01.20

String Buffer class:

Constructors of String Buffer class: (1) String class is immutable
(2) StringBuffer class is mutable.

StringBuffer()
StringBuffer (int capacity)
StringBuffer (String str)
StringBuffer (char - sequence [] ch).

(1) Create an empty string buffer with string capacity 16.

StringBuffer b1 = new StringBuffer();
" b2 = " " ("SNU");
" b3 = " " (50);

(2) Append Method:
append (String str)
append (int x)
StringBuffer S1 = new StringBuffer ("Core");
S1. append ("Java");
S1. append (20);
{Core Java 20} -> output.

runtime

(3) Insert Method:
insert()
S1. insert (1, "home");
S1. insert (2, 4, "home");
S1. reverse();
S1. "Sweat";
S1. insert (0, "home");
S1. delete (2, 4); -> delete position 2-4.
S1. capacity (100);
S1 = new StringBuffer ("SNU");
S1. append ("- -");

{Shomeweat} -> output.

Sh

04.02.2020

```
import java.util.*
Public class Test
{
    public static void main (String [] args)
    {
        LinkedList obj = new LinkedList();
        obj.add('A');
        obj.add("B");
        obj.addLast("C");
        obj.addFirst("D");
        obj.add(2, "E");
        obj.add("F");
        obj.add("G");

        obj.remove("B");
        obj.remove(3);
        obj.removeFirst();
        obj.removeLast();

        int size = obj.size();
        Obj element = obj.get(1);
        obj.set(2, "y");
```

begin, end, at position

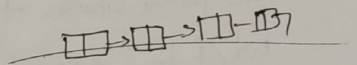
D A B C
D A E B C
D A E B C F
D A E B C F G
D A E C F G
A E F G
A E F

(A) Prototype defined in Java linked list class:

1. add (int index, E element)
2. add (E element)
3. add first (E e)
4. add last (E e)
5. clear() → removed all the elements from the list.

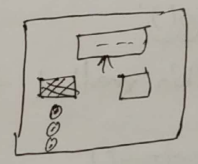
6. contain(object o) returns true if this list contain the specified element.

get (int index)
getfirst ()
getlast ()
remove (int index)
removefirst ()
removelast ()
size() returns no of elements in the list
set(int index, E element) → Set the element at the index position.



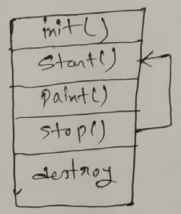
(#) Search value

```
boolean contain(Object ob)
{
    node cur, temp;
    cur = head;
}
```



Applet:

Life cycle of Applet:



⇒ init(): In this method Applet object is created by the browser. This method is called before all other methods. Programmer can utilize this method to initialize objects, variables, setting background and fore.ground color in GUI.

⇒ start()

In init() method even though applet object is created but it is in active state. To make the applet active the init method calls start() method. In start() method applet becomes active.

⇒ Paint()

Paint() method takes object of java.awt.Graphics class Graphics object as parameter. This class includes many methods of drawing necessary to draw applet window. This is the place where the programmer can write his code what he expects from the applet.

⇒ Stop():

In This method the applet becomes temporarily inactive.

⇒ Destroy():

This method is called just before an applet object is garbage collected.

06.02.2020

```
class triangle
{
    private int x,y,z;
    void setX(int x)
    { this.x = x; }
    int getX()
    { return x; }
}
```

Const is not available in Java.

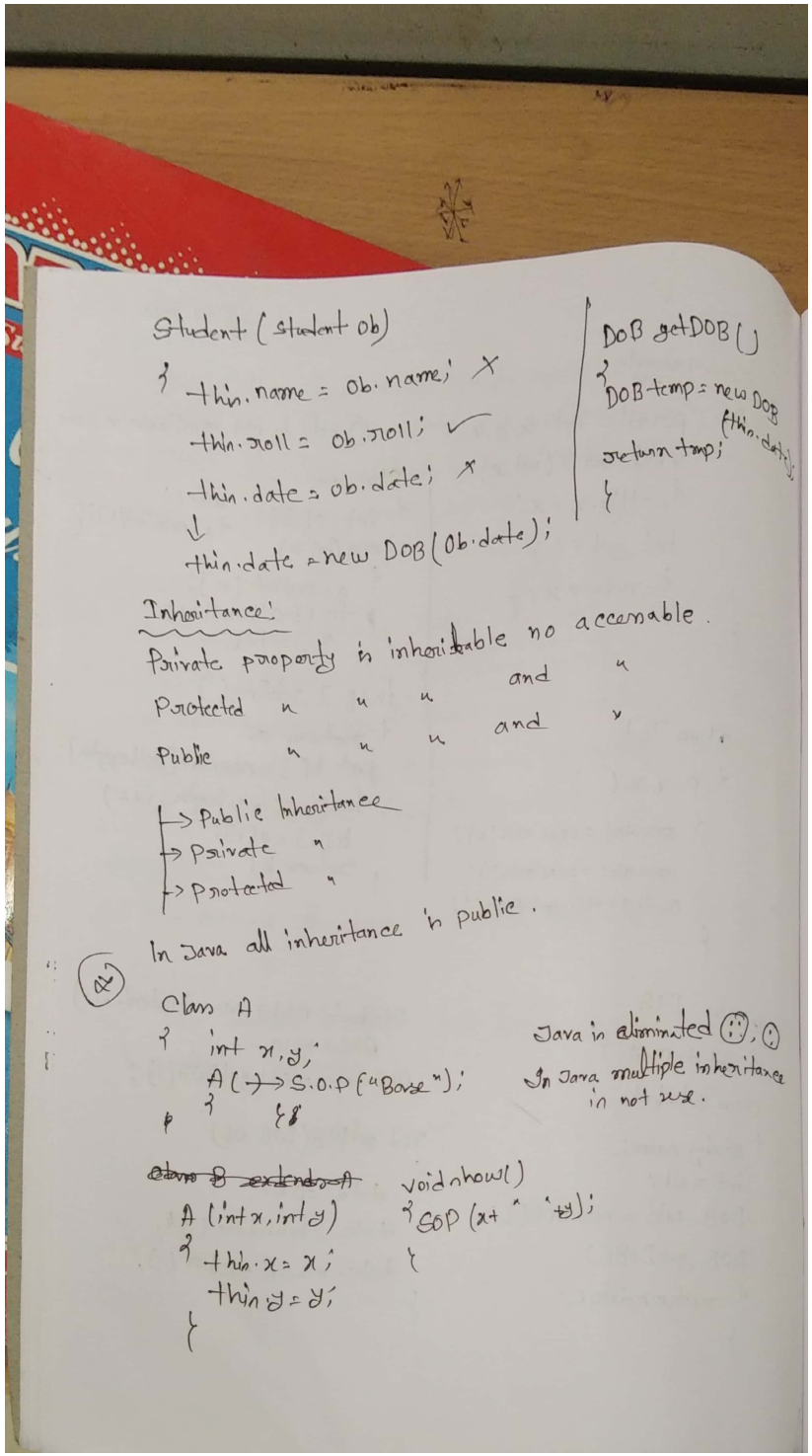
```
class Test
{ psvm()
  { arr ob1 = new arr(5);
    arr ob2 = new arr();
    ob2.a = ob1.getArr();
  }
}
```

```
class arr
{ int a[];
  arr()
  { a = new int[1]; }
  arr(int n)
  { a = new int[n];
    for(i=0;i<n;i++)
      a[i] = i+1;
  }
  int[] getArr()
  { return a;
    int b[] = new int[a.length];
    for(i=0;i<a.length;i++)
      b[i] = a[i];
    return b;
  }
```

```
class DOB
{ int d,m,y; }
class Student
{ String name;
  int roll;
  DOB date = new DOB(..);
  DOB getDOB()
  { return date;
  }
```

```
Student ob1 = new Studen(...)
  ob2 = new ....
ob2.date = ob1.getDOB();

void setDOB(DOB ob)
{
  date = ob;
  date = new date();
  date.m = ob.getM();
}
```



```
Student (student ob)
{  this.name = ob.name;  ✗
   this.roll = ob.roll;  ✓
   this.date = ob.date;  ✗
   ↓
   this.date = new DOB(ob.date);
```

```
DOB getDOB()
{
DOB temp = new DOB
                (this.date);
return temp;
}
```

Inheritance:

Private property is inheritable no accessable.
Protected " " " and "
Public " " " and "

- Public Inheritance
- Private "
- Protected "

In Java all inheritance is public.

(42)

```
Class A
{  int x,y;
   A() → S.O.P("Base");
}  {}
```

Java is eliminated ☹;☺
In Java multiple inheritance is not use.

```
~~class B extends A~~
   A(int x, int y)
   {  this.x = x;
      this.y = y;
   }
```

```
void show()
{SOP(x+" "+y);
{
```

```
class B Extends A
{  int z;
   B()
   { super(1,2);
     z=5;
   }
   B(int x, int y, int z)
   { super(x,y);
     this.z = z;
   }
   void show()
   { SOP(z);
     super.show();
   }
```

```
class Test
{
   PSVM(...)
   {
     B Ob = new B();
     A Ob1 = new A();
     B ob2 = new B();
```

(*) What is the final?

```
Ob1.show();
Ob2.show();
```

✓ Ob1 = new B(); (because child is a base)

✗ Ob2 = new A();

final method, private method, static are static.

(*) Super always refer to base class. It is a pointer. In constructor we must write super in the 1st statement.

— . —

13.02.2020

Static Block:

```
class Test
{
   static int x=10, z;
   int y;
   static{
     S.O.P. ("block 1!"); → z = ++x;
   }
   static {
     S.O.P ("block 2");
   }
   PSVM(String[] args)
   {
     S.O.P ("main");
   }
}
```

We can also initialize a static variable inside constructor but then it will be dependent that why we use static block.

final:

(i) final method, (ii) final variable, (iii) final class.

(iv) static method & private method can be overridden

```
class A
{
    void f()
    {
      s.o.p("Af");
    }
    private void g()
    {
      s.o.p("Ag");
    }
}

class B extends A        // keyword
{
    void f()
    {
      s.o.p("Bf");
    }
    void g()
    {
      s.o.p("Bg");
    }
}

class Test
{
  psvm(String[] args)
  {
    A a = a;
    a = new A();
    a.f();  -> Af
    a.g();  -> Ag
    a = new B();
    a.f();  -> Bf
    a.g();  -> Ag
  }
}
```

```
B b = new B();
b.f();  -> Bf
b.g();  -> Bg.
```

System -> class.

Java
- lang
- IO
- util -> package.

java.lang - object.java → filename
object.class.

package → having predefined function classes.

protected void finalize() → last function to be called by the garbage collector
{ } we can override this.

private < default < protected < public

```
class A
{
  protected/public void finalized()
  { sop("end");
  }
  psvm(String[] args)
  { A a1 = new A();
    a1 = null;
    System.gc();
  }
}
String toString()
Integer i1 = new Integer(12);
String s = i1.toString();

abstract class A
{ abstract void f();
  void g(){ }
}
class Test
{ psvm(String[] args)
  { A a;
    a = new A(); }
```

override the toString() method
override finalize from finalize print this.

(*) Java indirectly support multiple inheritance class has only default and public

Interface:

```
interface I1
{
  void g();
  void f();
}
```

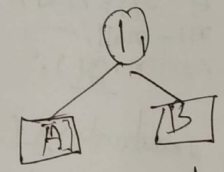
```
class A implements I1
{ void g() { S.O.P("g");
  void f()
  { S.O.P("f");
  }
```

Generalization

```
class test.
{ psvm()
  { I1 r1 = new A();
    r1.f();
    r1.g();
```

I1 ⇒ realization.

```
class B implements I1
{ void f() { } (=> error
}
```



in case of partial implementation we have to define the class as abstract.

18.02.2020

file name : Helloworld.java

```
import java.applet.Applet;
import java.awt.Graphics;
Public class Helloworld extends Applet.
{
    public void paint (Graphics g)
    {
        g.drawString("Hello world");
    }
}
```

RunHello world.html

```
<applet code = "Helloworld" width=200, height=60>
</applet>
```

How to run

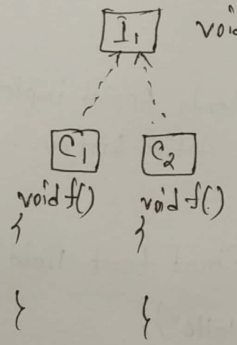
```
>javac Helloworld.java
>appletviewer RunHelloworld.html.
```

(#) Public class Applet with Textfield extends Applet implements ActionListener.

```
{
    public void init()
    {
        Label label1 = new Label("First text field");
        textfield1 = new Textfield("Hello");
        swapBtn = new Button("swap");
        swapBtn.addActionListner(this);
        Label label2 = new Label("second Text field");
        textfield2 = new Textfield("Good bye");
        textfield2.setEditable(false);
        add(label1);
        add(textfield1);      add(label2);
        add(swapBtn);         add(textfield2); }
```

P.T.O.

```
public void actionPerformed(ActionEvent e)
{ String temp = textfield1.getText();
  textfield1.setText(textfield2.getText());
  textfield2.setText(temp);
}
Textfield textfield1, textfield2;
Button swapBtn;
{
Label l1 = new Label("  ");
Textfield t1 = new Textfield();
Button B1 = new Button();
```




```
}     {
```

We cannot create object in interface

```
I1 r = new I1();
```

```
I1 r;
r = new c1();
r.f();
r = new c2();
r.f();
```

mid sem

27.02.2020

a) dynamic method dispatch

b) final (method, variable, class)

#1

```
interface I1
{
    void f();
}
```

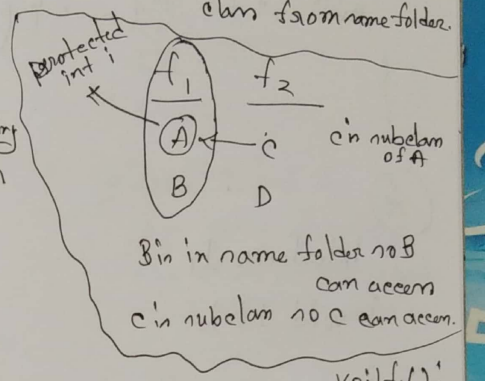
In interface all function is public

① Public ② protected
③ Private ④ default

default: any class within current folder.
Public: anywhere, anybody
Private: only in class
protected: sub-class from anywhere & non-sub class from same folder.

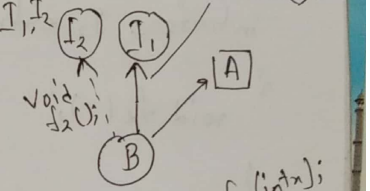
```
class c1 implements I1
{
    public void f().
    {
    }
}
```

when implementing interface function make



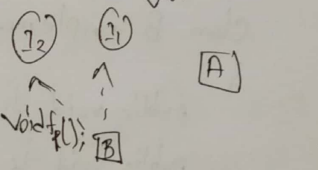
1st case

```
class B extends A implements I1,I2
{ public void f1() { ... }
  public void f2() { ... }
}
```



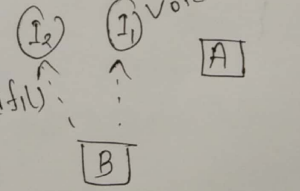
2nd case

```
class B extends A implements I1,I2
{ public void f1() { ... }
  public void f1(int x) { ... }
}
```

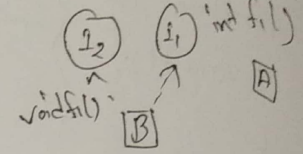


3rd case

```
class B extends A implements I1,I2
{ public void f1() { ... }
}
```



4th Case




```
class B extends A implements I1,I2
{
  public void f1( ) {---}
  public int f1( ) {---}
}
```

→ Compilation error.

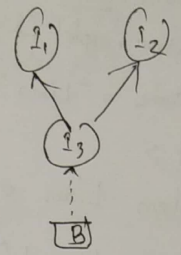
(b) advantage of interface over class

```
interface I1
{
  void f( );
}
interface I2 extends I1
{
  void f2( );
}
class B implements I2
{
  public void f1 ( ) {---}
  public void f2 ( ) {----}
}
```

(a)

```
interface I1
{ void f1();

{
interface I2
{ void f2();
 }
interface I3 extends I1, I2
{ void f3();
 }
class B implements I3
{  public void f1(){ ... }
   public void f2(){ ... }
   public void f3(){ .... }
}
```

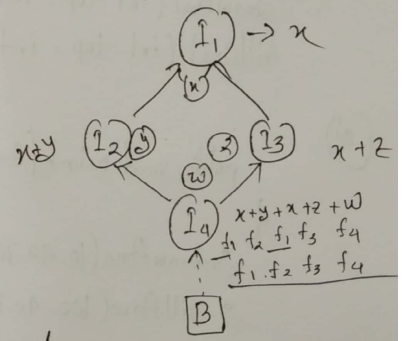


(a) When I1 and I2 has same function then in class B implements just 1 function.

(b)

```
interface I1
{ void f1( );

{
interface I2 extends I1
{ void f2( ); }
interface I3 entends I1
{ void f3( ); }
interface I4 entends I2, I3
{ void f3( ); }
```



class B implements I4
{ implement all functions
}

P.t.o.

03.03.2020

(a)

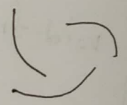
```
import java.awt.*;
import java.applet.*;
<applet code = "Lines" width = 30  height = 200>
</applet>
public class Lines extends Applet
{
    public void paint(Graphics g)
    {
        g.drawLine(0, 0, 100, 100);
        g.drawLine(0, 100, 100, 0);
        g.drawRect(10, 10, 60, 50);
        g.fillRect(100, 10, 60, 50);
        g.drawOval(190, 10, 90, 30);
        g.fillOval(10, 90, 140, 100);
```

same

drawLine(int startX, int startY, int endX, int endY)
drawRect(int top, int left, int width, int height)
fillRect(int top, int left, int width, int height)
drawOval(int top, int left, int width, int height)
fillOval(int top, int left, int width, int height)

(c)

```
public void paint(Graphics g)
{
    g.drawArc(10, 40, 70, 70, 0, 75);
    g.fillArc(100, 40, 70, 70, 0, 75);
```

-> 

void drawArc(int top, int left, int width, int height,
int startAngle, int sweepAngel)

(x) public void paint(Graphics g)

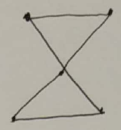
{

int x[] = {30, 200, 30, 200, 30};

int y[] = {30, 30, 200, 200, 30};

int num = 5;

g.drawPolygon(x, y, num);



void drawArc(int top, int left, int width, int height,
int startAngl, int sweepAngel)

void drawpolygon(int x[], int y[], int numberof points)

void fillPolygon (int x[], int y[], int numberofpoints)

(x)

Color c1 = new Color(255, 100, 100)

g.setColor(c1);

0-255	0-255	0-255
R	G	B
↓	↓	↓
168	100	250

(Color.RED)