Java Language

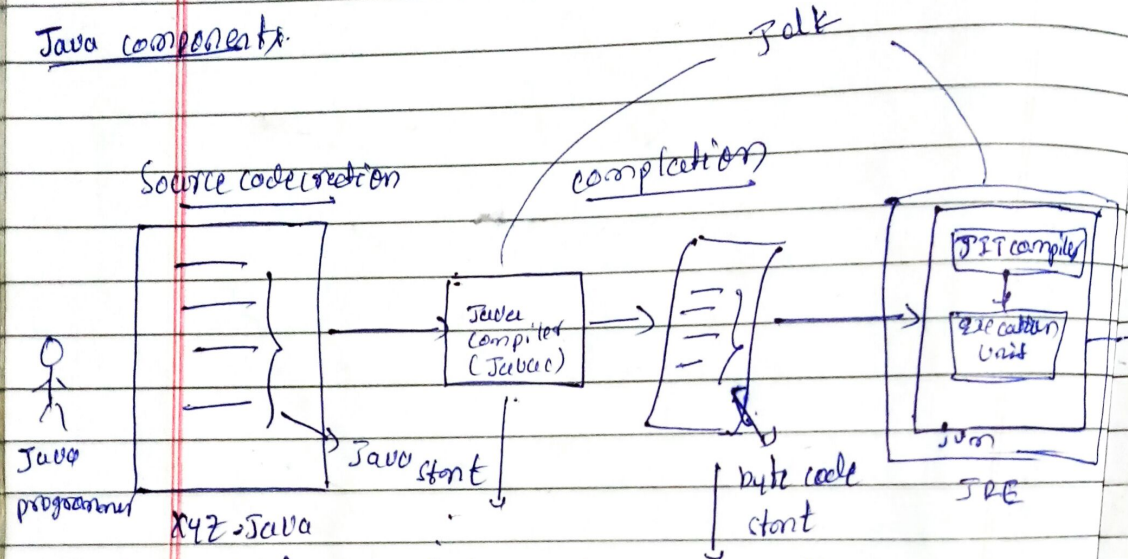
4/3/19

* Secured language
* OOPs based
* platform independent
* huge features
* multi threading

→ bcz when you install jdk software on your s/m then automatically JVM are installed on your s/m. for every operating s/m seperate JVM is available which is capable to read the .class file or byte code.

→ we can develop a multithreads program using java. it extends the idea of multitasking into application where you can subdivide specific operations within a single application into individual threads each of the threads can run in parallel

Java components



* Check syntax
* check rule
* translation
* "compile time Error"

HLL → High level language

JVM → Java virtual machine
JIT → Just-in-time
JRE → Java runtime Environment
JDK → Java Development kit

NOTE → classname should be same as filename this above are syntax

what is Java?

Java is a general-purpose programming language that is class-based, object-oriented and designed to have as few implementation dependencies as possible

what is oops? which can contain data, in the form of fields, and code, in the form of procedures. a feature of objects is an objects procedure that can access and often modify the data fields of the object with which they are associated.

Java program

```
class cl
{
public
{
}
}
```

Java intro

part 1:-

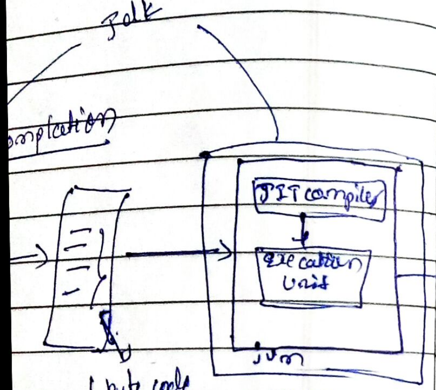
```
class
{
public
{
System.
}
}
```

part 2:- to pri

```
class
{
public
{
Syste
Syst
```

...n develop a multithreads program using ... extends the idea of 4/03/19
...tasking into application
... can subdivide specific operations
... single application into individual
...ch of the threads can run in parallel

...hen you install jdk software on
...hen automatically JVM are installed
... for every operating s/m seperately
...table which is capable to read
...e (or) byte code.



byte code
stant

xys. class
class file
(or)
Executable file

...ual machine
... time
...run time environment
...ment kit

...rename this above are Sentence

...ming language that is
...designed to have as
... possible
...e form of fields and code,
...objects is an objects procedure
...the fields of the object with
which they are associated.

Java program structure

```
class classname
{
public static void main(String [] args)
  {
     ===
  }
}
```

(String → capital)

filename.Java
↓
file name is should be give class name

| Javac filename.Java | → complition
| Java class name | → Execution

Java intro

\n → print the present line and jump to the next line
\n → line next

prgm 1:-

```
class program1
{
public static void main(string [] args)
{
System.out.println("Hi Welcome to Java programming");
}
}
```

to print different data type without format specifier

prgm 2:-

```
class program2
{
public static void main(string [] args)
{
  System.out.println(1234);
  System.out.println(12.34);
```

```
System.out.println("j");
System.out.println("spiders");
System.out.println(true);
}
}
```

System.out.println('h');	System.out.print('h');
System.out.println('i');	System.out.print('i');
o/p h	o/p : hi
i	

e.g:-

without ln → system.out.print("jm");

addition and concatination

prg 4:- → class program4

```
{
public static void main(String[] args)
{
System.out.println(20+20);
System.out.println("java"+"developer");
}
}
```

capital

if we want space between java and developer word give space it o/p :- 40 this words

concatination

Java developer

Rules for string concatination and addition of no's

prg 5:- → class program5

```
{
public static void main(String[] args)
{
System.out.println("number is"+20);
System.out.println("number is"+20+20);
System.out.println(20+" is the number");
System.out.println(20+20+" is the number");
}
}
```

o/p :-

if strings is first it decide the i should do first concatination

number is 20

number is 2020

20 is the number

40 is the number

if addition is first it will add and print

byte 1
short 2
int 4
long 8
float 4
double 8
char 2
boolean 1 bit

file name :- java variable
out classname is
java variable

prog 2 → for addi

```
import
class
{
public st
{
System.
Scanner
while
{
System
int
System.
int
System.
System.
int
it
d
```

5/03/14

byte 1
short 2
int 4
long 8
float 4
double 8
char 2
boolean 1 bit

pog 1: variable and datatype

class variable

```
{
public static void main (String [] args)
{
    int a=10;
    int b=20;
// int a,b;
    int r=a+b;
System.out.println("Result: "+r);
}
}
```

file name: javac variable1.java
but classname is
java variable

User i/p
Scanner scan = new Scanner (System.in);
int a = scan.nextInt();
↓ should be cap
int a = scan.nextFloat();
↓ should be cap

concatination to show the result.

pog 2 → for user input

```
import java.util.Scanner;
class variable
{
public static void main (String [])
{
System.out.println("welcome to my calc");
Scanner scan = new Scanner (System.in);
while (true)
{
System.out.println("enter a: ");
int a=scan.nextInt();
System.out.println("enter b: ");
int b=scan.nextInt();
// int a,b;
int r=a+b;
System.out.println("Result: "+r);
System.out.println("1 to continue\n 2 to exit\n enter choice");
int choice = scan.nextInt();
if (choice==2)
{
```

NOTE: for any data type to read using next()
↓ cap
should begin with capital letters

→ should be capital

→ should be capital when your reading i/p from screen

→ should be capital

go to last page &

Bajna Gold

i // 0

i // FFF4

i // IDE

5/03/14

Data type in java

primitive data type — Non-primitive data type.

① primitive data type

data type	size	default value.	Range
byte ←	1	0	-127 to 128
short ← → To represent integer	2	0	
int ←	4	0	
long ←	8	0L	
float } To rep float	4	0.0f	
double } float	8	0.0	
char	2	u 000 (zero)	
boolean { true, false	1bit	false	

2^8 → 0b00000000 → -2^7 to 2^7 (-2^{n-1} to 2^{n-1}) → default - signed

for sign

→ to include data types with initialization of variable in java we need to follow

3 steps in order → declaration

→ initialization

→ retrieval.

→ if data type not initialized in java, then compile error will occur, the garbage value is not initialize or assumed in java as in C. So initialization is compulsory

Bajna Gold

System.out.println("thank you. Bye bye.... :-)")
} System.exit(0);
}
}
}

for input from programm in java one have to use import java.util.Scanner

PRG 3:- import java.util.Scanner;
class ~~variable~~ cap datatype
{
public static void main(String[] args)
{
Scanner scan = new Scanner(System.in);
int a=scan.nextInt(); // nextXxx();
// string : next(); (or) nextLine();
// char : next().charAt(0);

go to last page

after two prg another one prg or ~~...~~

using variable Rules

note → next() string. string is default read

6/03/19

byte, short, int, long } integer
float, double } floating
char
boolean

In java language variable must be initialize before using many operations otherwise compiler through error

Java type definition

1) class type
2) interface type
3) enum type
4) annotation type

class type	interface type	Enum type	Annotation
class class name	interface interfacename	enum Enumname	@interface interface name
{	{	{	{
====	====	====	====
}	}	}	}

...ai. Bye bye.... :-)")

...n programs in java are have to ... java.util.scanner

...ner;

...ta type

...main(String[] args)

... scanner(system.in);

...(); // nextXxx();

...xtLine();

...t(0);

...String is default; need

6/03/19

...be initialize ... therwise

- byte
- short
- int
- long

} integer

- float
- double

} floating

- char
- boolean

Enum type:	Annotation

enum Enum name
{
=
};
	@interface interface name
	{
	=
	}

class type definition

class classname ← class declaration

{ — class defn (or) body

declare + initialize variable → member variable / data member → member of class (static variable / non static variable)

:

declare + initialize function → member function

:

}

note: we call variable & function as members of class

typical class

```
class Demo1
{
    int x=10;            -> non static member variable
    static int y=20;     -> static member variable
    void func1()         -> non static member function
    {
        =
        int a=30;        <- local variable
        =
    }
    static void func2()  -> static member function
    {
        =
        int b=40;        <- local variable
        =
    }
}
```

note: there is no concept of using global variable in java, only static, non static and local variable used

NOTE :-

* A java source file can have any number of class definition. when we compile such source file the compilier generates separate .class file for each class definition.

*the jvm can begin the execution by calling main method of the class, hence the class which contains main method should be used to begin the execution.

for execution : javac demo1.java.
→ java demo1 → error to begin with public static void main

prel :-
```
class demo1
{
  void test()
  {
    system.out.println("running test() method");
  }
}
```

→ java demo2
running demo2 class
→ java demo1
running test method

when you exe prg save classname javac demo1.java then next java demo2 which has main class

```
class demo2
{
  public static void main(String[] args)
  {
    system.out.println("running demo2 class");
  }
}
```

access specifier

note :- * the members of one class can be reffered or accessed another class.

* the access can be restricted by using access specifier

java supports 4 types of access specifier

1) private
2) default
3) protected
4) public

Reffering static method of class

Syntax :- `class name . member name`

prel :-
```
class demo1
{
  static int
  static
  {
    system.o
  }
}
class d
{
  public
  {
    System.
    // reffer
    System.o
      demo1
      Syste
  }
}
```

```
class
{
  static
  static
  {
    sy
  }
}
class
{
  stat
  stat
  {
    sys
    //a
    s
```

main method of
method should be

Demo1.java.
must to begin with
d main

method"];

es

thod

class");

ared or accessed.

g access specifier
ifier

prel :-

```
class Demo1
{
 static int x=12;
 static void test()
 {
  System.out.println("running test() method");
 }
}
class Demo2
{
 public static void main(String[] args)
 {
 System.out.println("main method started");
 // referring static members of Demo1
 System.out.println("x value: "+Demo1.x);
  Demo1.test();
  System.out.println("main method ended");
 }
}
```

class name.member name

similarly

NOTE:- we can change static variable value by referring on outer class i.e. -> or initialization

Static initialization can do

prel :-

```
class Demo1
{
 static int x=12;
 static void test()
 {
  System.out.println("running test() method");
 }
}
class Demo2
{
 static int y=45;
 static void disp()
 {
 System.out.println("running disp() method");
 //referring static member of Demo1
 System.out.println("x value: "+Demo1.x);
  Demo1.test();
```

oop's -> object oriented programming

Bafna Gold
Date: Page:

```
}
}
class Demo3
{
public static void main (String [] args)
{
System.out.println(" main method started");
// referring static members of Demo2
System.out.println("y value: "+ Demo2.y);
Demo2.disp();
System.out.println("main method ended");
}
}
```

Referring

1) static → classname.membername
Ex → Demo1.x;
Demo1.test()

2) non static
object.membername
new Demo1().x;

7/03/19

Referring non-static member of a class

→ create object of class
→ new operator to create object of class

Syntax

```
new classname();
```

e.g:- Demo1 is a classname contain non static

```
new Demo1();
```

→ to refer the non static member syntax is

```
object.membername
new Demo1().x
```

referring NON-static

Pgm:-

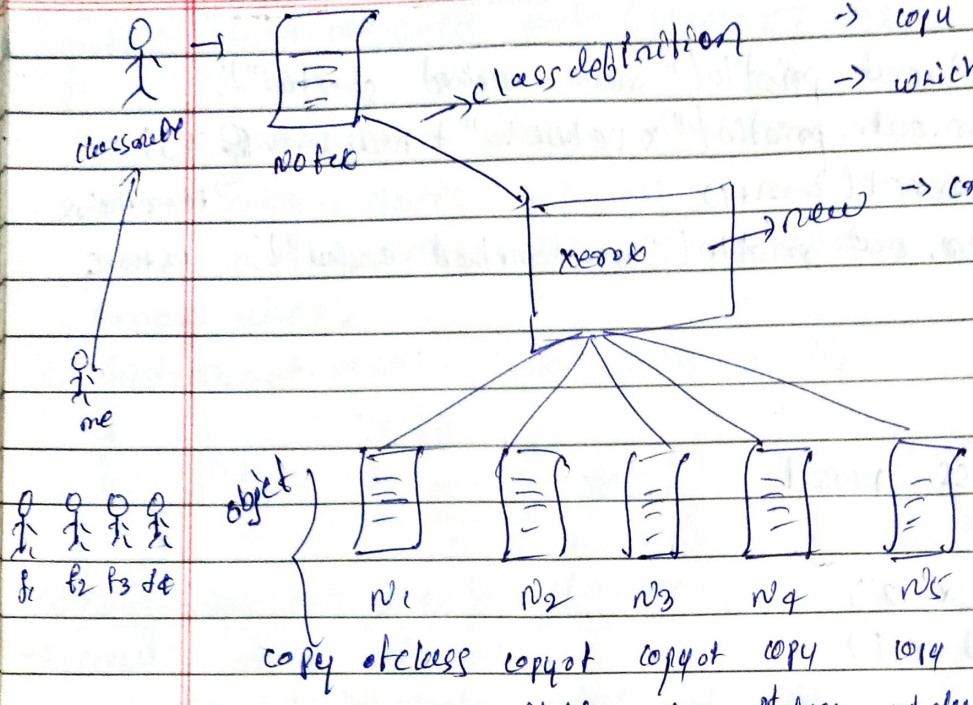
```
class Demo1
{
int x = 12;
void m1()
{
System.out.println("running m1() method...");
}
}
```

Pgm :-

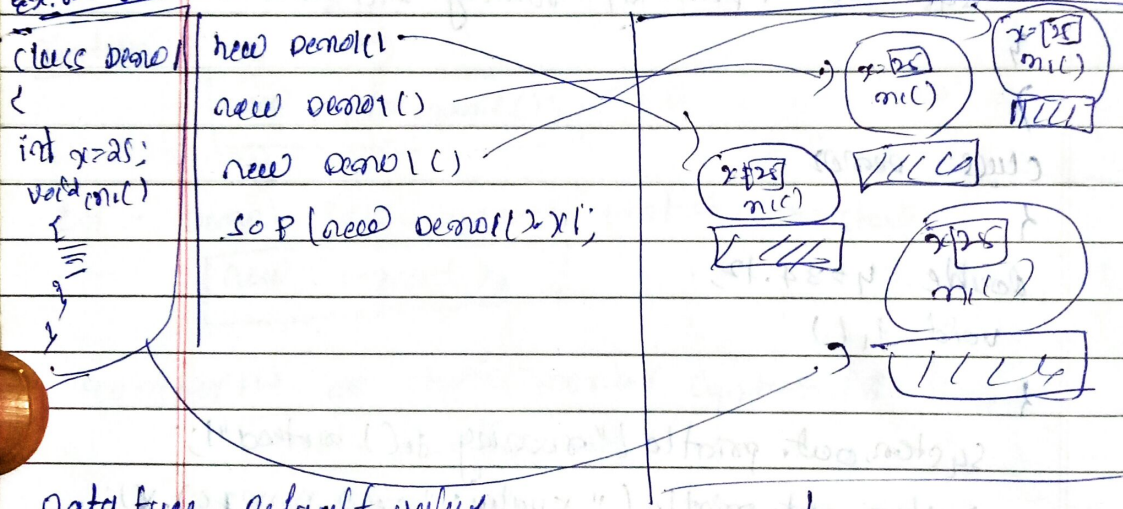
```
class mainl
{
public static void main (String [] args)
{
System.out.println(" main method started");
System.out.println(" x value : " + new Demo1().x);
new Demo1().m1();
System.out.println(" main method ended");
}
}
```

Pgm :-

```
class Demo1
{
int x=12;
void m1()
{
  System.out.println("running m1() method...");
}
}
class Demo2
{
Double y=34.12;
 void f1()
{
 System.out.println("running f1() method");
 System.out.println("x value:"+new Demo1().x);
 new Demo1().m1(); -> return non-static member.
 System.out.println("main method ended");
}
}

class mainclass
{
 public static void main( String [] args)
{
 System.out.println("main method started");
 System.out.println(" y value: " +new Demo2().y);
  new Demo2().f1();
   System.out.println("main method ended");
}
}
```



→ copy of class-object
→ which creates object
new
→ created by class definition



Data type	Default value
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
string (or any object)	null
boolean	false

to access the first object, need to give variable name to every object.

variable

classified into t

→ scope defines

can be

(a) member var

→ declared

→ scope in

access s

(b) local var

→ declared

→ scope in

Difference b

Declaration

class name

e.g.:

C1 is

per

P1 is

movie

m1 is a

Note:- cl

variable in java

classified into two types based on scope

→ scope - defination where variable can be accessed.

(a) member variable
→ declared in a class body
→ scope is decided based on access specifier

(b) local variable
→ declared inside function body
→ scope is limited to function body

classified into two type on declaration type

Type variable name;

(a) primitive variable
→ variable declared using data types → (8)
→ they are used to store values
eg:- int age;
double marks;
char ch;

(b) non-primitive (or) Reference variable
→ variable are declared using user types
→ they are used to reference the object
eg:- car c1;
pen p1;

8 | 08 | 04

Reference variable for non-primitive variable

declaration

Classname variable name;

e.g.: Car c1;
c1 is a reference of car type
pen p1;
p1 is a variable of pen type
movie m1;
m1 is a variable of movie type

Initialization

* reference variable are initialized to null or object

(a) initializing to null

```
c1 = null;
p1 = null;
m1 = null;
```

you can initialize like this to do this

(b) initializing to object

```
c1 = new car();
p1 = new pen();
m1 = new movie();
```

if you want to create object do this

Note:-
```
class demo1
{
    int p=12;
    void test()
    {
        system.out.println("running test() method.....");
    }
}
```

Note - don't use, if we use we get "null-pointer-exception" → no value. It a keyword

copy of class-object which creates object new
→ created by class definition

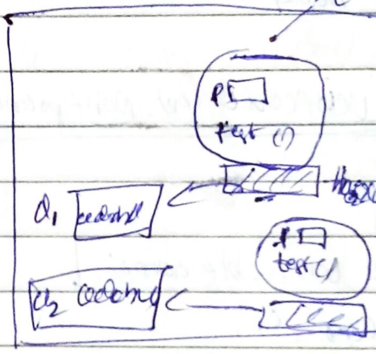
object, need to to every object

```
class mainclass3
{
 public static void main(String[] args)
 {
   System.out.println("main method started");
   // declare reference variable
   Demo1 d1;
   // initialize to object
   d1 = new Demo1();
   System.out.println("p value: " + d1.p);
   d1.test();
   System.out.println("main method ended");
 }
}
```

NOTE

PRG :-

```
Demo1 d1;
Demo1 d2;
d1 = new Demo1();
d2 = new Demo1();
sop(d1);
sop(d2);
```



d1, d2 > ref. variable used to refer to the address of object.

d1.d2 → copy of class Demo1

reference variable print → Demo1 prg

```
class mainclass4
{
 public static void main(String[] args)
 {
   System.out.println("main method started");
   // declare reference variable
   Demo1 d1;
   Demo1 d2;
   // initialize to object
   d1 = new Demo1();
   d2 = new Demo1();
   System.out.println("d1: " + d1);
   System.out.println("d2: " + d2);
   System.out.println("main method Ended");
 }
}
```

note :- without ref. variable each one when we inside object of creates new copy of class

O/P : d1 : Demo1@address
d2 : Demo1@address

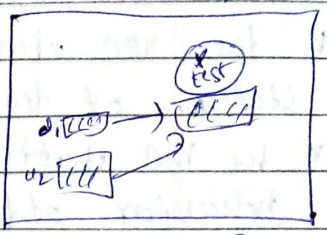
note :- one object can refer by 'n' number of reference variable

pgm:-

```
class demo1
{
int p=12;
void test()
{
System.out.println("running of test() method.....");
}
}
class mainclass
{
public static void main(String[] args)
{
System.out.println("main method started");
// declare reference variable
demo1 d1;
Demo1 d2;
// initialize to object
d1 = new demo1();
d2 = new demo1();
System.out.println("p value of first object: "+d1.p);
System.out.println("p value of second object: "+d2.p);
d1.p=34; -> reinitialization
System.out.println("p value of first object: "+d1.p);
System.out.println("p value of second object: "+d2.p);
System.out.println("main method ended");
}
}
```

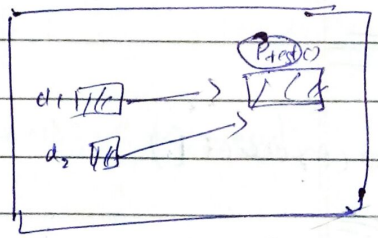
$d_1 = d_2$ → objective addresses

Bafna Gold
Date: Page:



object (created for demo1 class)

if ($d_2 = d_1$) // d_1 & d_2 holds address of same object



NOTE:-

we can write also like this

```
Demo1 d1;
Demo1 d2;
d1 = new demo1();
d2 = d1;
```

Declaration and initialization

Demo1 d1 = new Demo1(); → object constructor

classname → class name

d1 → reference variable

new → operator

objectname → object reference

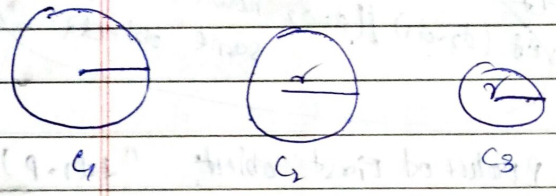
classname referencevariable = new constructor();

Class and objects

* any entity having it's own states and behaviours known as object.
* A class is a definition block which is used to define states and behaviour of the object.
* the non static member variable reference the states of the object.
* the non static member function () represents the behaviour of the object.
* the new operator creates copy of the class in the memory which contains the states and the behaviour hence it is known as object. it can construct any number of objects to recognize each object we make use of reference variable

12/3/19

different radius value for diff. circumference, memory space in all circle




```
class circle
{
double real;
static double pi = 3.14;
void area()
{
}
void circle()
{
}
void destroy()
{
}
}
```

can static

```
circle c1 = new circle();
c1.real = 2.4;
circle c2 = new circle();
c2.real = 1.8;
circle c3 = new circle();
c3.real = 1.2;
```

from a class, you can create multiple objects and call class many time

```
class circle
{
double real
static doub
void area
{
double
system.o
}
void d
{
double
system.o
}
void
{
doub
syst
}
}
cl
{
}
sy
c
```

prog → class circle

```
{
double real;
static double pi = 3.14;
void area ();
{
double a1 = pi * real * real;
System.out.println ("area :" + a1);
}
void diameter()
{
double d1 = d * real;
system.out.println ("diameter :" + d1);
}
void circum()
{
double c1 = d * pi * real;
system.out.println (" circumference: " + c1);
}
}
class main class 6
{
public static void main(string[] args)
{
system.out.println ("main method started");
circle c1 = new circle ();
c1.real = 2.1
System.out.println ("reallies: " + c2.real);
c1. diameter ();
c1. area ();
c1. circum();
circum c2 = new circle();
c2. real = 2.6;
System. out. println("reallies : " + c2. real);
c2. diameter ();
c2. area ();
c2. circum();
```

Bafna Gold
Date: Page:

System.out.println("main method ended")
}

~~prg~~ - class student
{
string name;

class student
string name;
int age;
string gender;
double marks;
long int id;

general format

program using set and get

class Employee — should be capital — Here save Employee.java

javac Employee.java
java Employee
it shows error like this
could not find or load main class
do like this
set class - " C:\> jdk are run.
C:\program files\java\jdk.co..\bin
C:\ ——— \lib

```
class Employee
{
int id;          in string
string name;
double salary;
void setId(int arg1)  -> rewriting   (I - capital letter)
{
id= arg1;
}
int getId()  -> read   (I - should be capital)
{
return id;
}
void set name(string arg1)
{
name = arg1
}
string getName()
{
return name;
}
void set salary (double arg1)
{
salary =arg1;
}
double get salary ()
{ return salary;
} }
```

```
Employee e1 = new Employee();
sop(e1.getId());
sop(e1.getName());
sop(e.getSalary());
e1.setId(1010);
e1.setName("Parneeth");
e1.setSalary(25000.00);
sop(e1.getId());
sop(e1.getName());
sop(e1.getSalary());
```

e to print → null
or
proceed to 0.00

13/03/19

memory used in jvm

4 memory areas

1) Heap area → to store objects (non-static)
2) class area → to store static members
3) method area → to store method defn/body
4) stack area → execution area

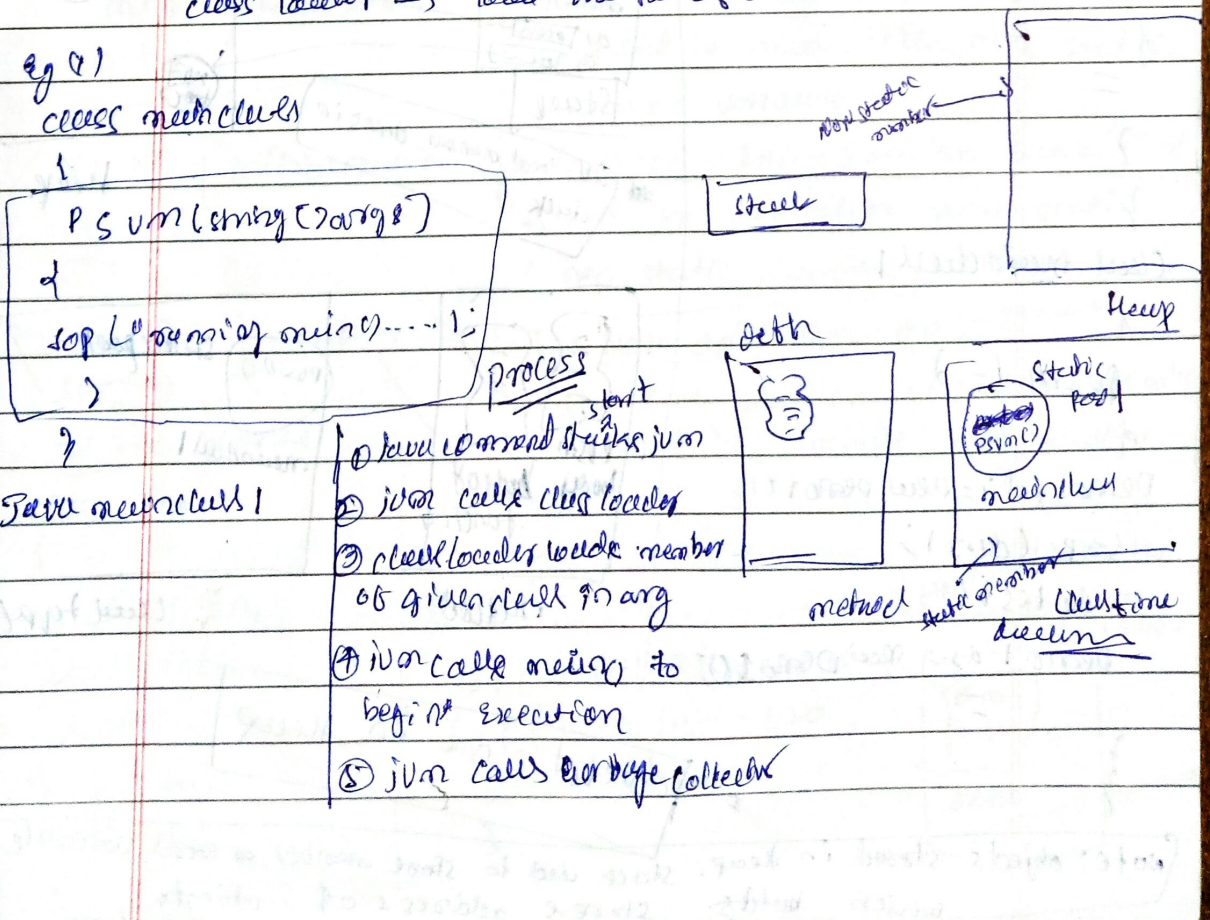
class loader → load the no of classes

eg 1)

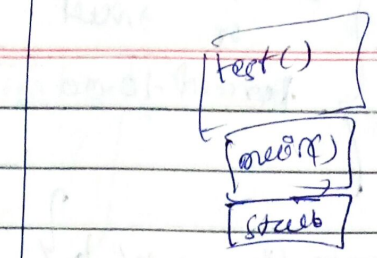
```
class mainclass
{
    P S vm(String[]args)
    {
        sop("running main.....");
    }
}
```

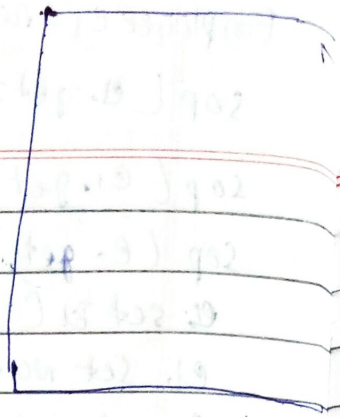
Javac mainclass.java

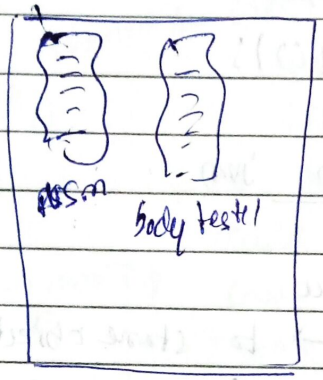
process
start
1. java command starts jvm
2. jvm calls class loader
3. class loader loads member of given class in arg
4. jvm calls main to begin execution
5. jvm calls garbage collector

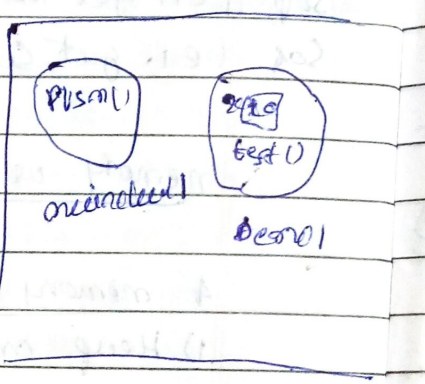


Eg :- class Demo1
{
static int x=10;
static void test()
{
=
}
}
class mainclass1
{
psvm (-)
{
sop(Demo1.x);
Demo1.test();
}
}

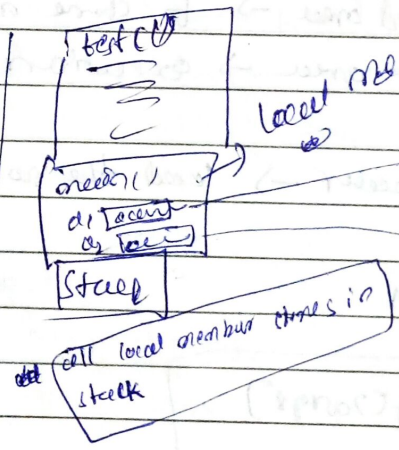


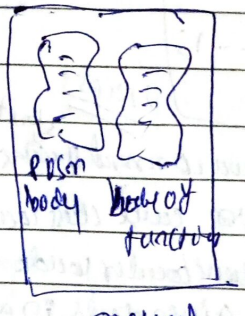


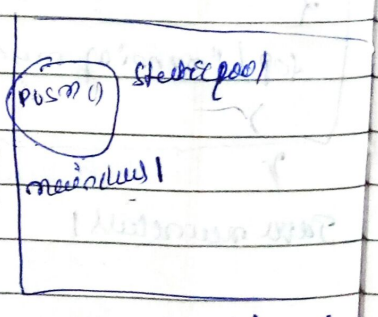


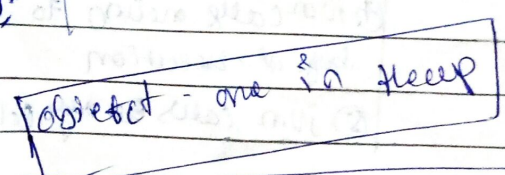


Eg :- class Demo1
{
int x=10;
void test()
{
=
}
}
class mainclass1
{
psvm (-)
{
Demo1 d1 = new Demo1();
sop (d1.x);
d1.test();
Demo1 d2 = new Demo1();
=
}
}









note: objects stored in Heap. stack used to store member or local variable which stores address of objects

14/3/19

object → class circle
{
double real = 1.2; → member or data member → Default initialize by compiler / loading time initialize
void diameter()
{
≡
}
void area()
{
}
double ar;
≡
}
void circumference()
{
≡
}
void set Radius (double arg)
{ real = arg;
}

(the functions above) → member function

read / read 1.5 / Heap

circle c1 = new circle();
object

c1.real = 3.2; → changing real value after object creation → change radius value
or
c1.set Radius (3.2); → object creation

object created when real 1.2

variable → initialize is must

initialization BLOCKS

* used to initialize member variable of class

① static initialization Block (SIB)
→ used to initialize only static member variable

② instance initialization Block (IIB)
→ used to initialize both static & non static member variable.

1st block will execute → Execution

prg → class Demo1
{
int x; } declaration
int y;

x = 10; } initialization
y = 32; }

sop(x); } printing
sop(y); }

(use block) → should be done in declaration part only
→ should be include inside function.

static blocks →

class Demo1
{
static int a; }
static int b; }
static
{ a = 10;
b = 20;
}
}

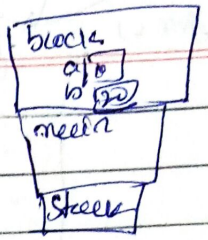
class main1
{
PVSM (-)
{
sop(Demo1.a); → 10
sop(Demo1.b); → 20
}
}

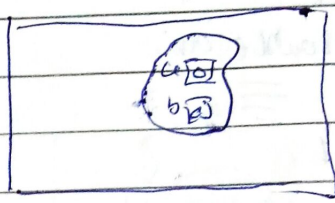
first initialize the zero then BLOCK will initialize

a = 10
b = 20
Demo

static blocks

```
class demo1
{
static int a;
static int b;
static
{
int a=10;
int b=20;
}
}
class main
{
PVSM(-)
{
sop(demo1.a);
sop(demo1.b);
}
}
```





local variables

context or body

(static acting as function and variables inside static considered as local variables when it declared inside static)

NOTE: in above example, the static member variable are referred to the main function and default value will stored.

→ if inside the static initialization block, if the variable are declared then it considered as function local variable and if static considered as function

static initialization Block

→ storing member variable in class area.

```
prog 1:- class demo1
{
static int a;
static int b;
static{
System.out.println("running 1st static block");
a=12;
b=34;
}
static{
System.out.println("running 2nd static block");
a=42;
b=54;
}
}
```

in static block if you want to initialize member variable you should use static keyword

result :- a value: 42
b value: 54

class loader → it loads new static blocks and loads sequentially for example first it loads 1st static block and then it loads 2nd static block

class maincalls

{ public static void main(String [] args)

{

System.out.println("main method started");

System.out.println("a value: "+demo1.a);

System.out.println("b value: "+demo1.b);

System.out.println("main method ended");

}

}

II. Non static block → storing member variable in heap area.

class demo2

{

int x;

int y;

}

System.out.println("running 1st non-static block");

x=12;

y=34;

}

{

System.out.println("running 2nd non-static block");

x=54;

y=67;

}

}

class main class2

{

public static void main(String[] args)

{

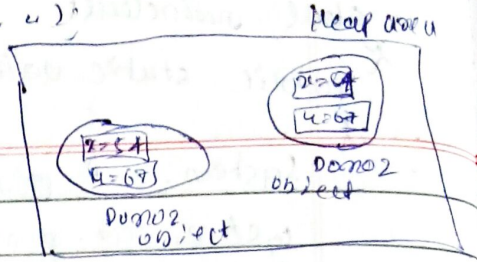
system.out.println("main method started");

Demo2 d1=new demo2();

System.out.println("x value: "+d1.x);

System.out.println("y value: "+d1.y);

System.out.println("-----");
Demo2 d2 = new demo2();
S.O.P("x value: "+d2.x);
S.O.P("y value: "+d2.y);
S.O.P("main method ended");
}
}



Initialization of both static and non static blocks

Pgm 3:- class demo3
int x; → non static block
static int y; → static block
static{
System.out.println("running static block");
y=34;
}
}
System.out.println("running instance block");
x=79;
}
}
class main3
{
P S V M (String[] args)
{
S.O.P("main method started");
demo3 d1 = new demo3();
S.O.P("x value: "+d1.x);
S.O.P("y value: "+Demo3.y);
S.O.P("-----");
Demo3 d2 = new demo3();
S.O.P("x value: "+d2.x);
S.O.P("y value: "+Demo3.y);
S.O.P("main method ended");
}
}

Result:
main method started
running static block
running non static block
x=79 y=34

→ constructor
which is
3 ways
→ during d
→ using ste
→ using

the memb
class cl
{
1) DATA
2) me-th
3) block
4) constr
}

Sy
cl
{

note: (ons
initialize

1)
2)

Constructor

Batra Gold
Date: 15/3/19 Page:

→ Constructor is one of the special member of the class which is use to initialize data members of the class

3 ways to initialize data members

→ during declaration
→ using static & non-static blocks
→ using constructors

the members that are scope inside the class are

```
class classname
{
1) DATA members
2) methods
3) blocks
4) constructor
}
```

Syntax for constructor

```
classname ( ) → arguement
{                → data
   // TASK
   //initialize data members of class
}
```

Note: constructor is a member of class, it is allocated to initialize the data members.

2 types of constructor

1) default constructor
2) programmer defined constructor

constructor program:

```
class sample
{
 int a;
boolean res;
sample ()
{
  a = 123;
res = false
}
}
class mainclass1
{
psvm(String args)
{
s.o.p("main method started");
sample s1 = new sample();
s.o.p("a value:"+s1.a);
s.o.p("res value:"+s1.res);
s.o.p("main method ended");
}
}
```

NOTE:

* constructor will have the main same as class name
* there will be no return type for constructor
* new operator implicitly calls the constructor of the class
* programmer cannot call the constructor using any reference variable

(at the time of object creation, new keyword, is used to call the constructor)

1) create an object
2) load non-static member inside object
3) call constructor implicitly.

1) Default

class pro
{
}
class ma
{
psvm(st
{
s.o.p("
progra
s.o.p(
}
}

* the c
compil
constru
* write
constru
both

② Prog

a) Zer
class
{
String
decla
stri
pen
{
s.o.
colo
pri
bra
}
}

1) Default Constructor (during compilation) class program1

class program1
{
}

→ [javac compiler] →

class program1
{
program1()
{
}
}

default constructor (written by compiler at compilation time)

class mainclass2
{
psvm(String[] args)
{
S.o.p("main method started");
program1 ref = new program1();
S.o.p("main method ended");
}
}

mainclass2() → default
{
}

* the default constructor is written by the compiler at the compilation time. if the programmer has not written any constructor within the class.
* within a class either default constructor (or) user defined constructor will be present. (initialization is default), both will not be present within a class

② Programmer defined constructor → zero argument constructor, parameterized constructor } Two types of P.D.C

a) zero argument user defined constructor

class pen
{
String colour;
double price;
String brand;
pen()
{
S.o.p("pen class constructor");
colour = "white";
price = 42.0;
brand = "cello";
}
}

```
class mainclass3
{
psvm(String[] args)
pen p1= new pen();
s.o.p("colour="+p1.colour);
s.o.p("price"+p1.price);
s.o.p("brand="+p1.brand);
s.o.p("main method ended");
}
}
```

(or)

"If object can be created for zero argument

```
pen p1= new pen();
  —
  =
  —
pen p2= new pen();
  —
  =
  —
pen p3= new pen();
  —
  =
  —
pen p4= new pen();
  =
  =
s.o.p("main method ended");
}
}
```

different

Default constructor	zero argument constructor
1) created by compiler all default const	1) created by user
2) all default const is zero argument	2) Not all zero argument is default constructor
3) default initialization is done	3) initialization if done by user
4) If data member are initialized it will not change	4) it will change as per user

parameterized

```
class movie
{ string mo
string lang
double dur
double pr
movie(stri
{
movielang
language
duration
price
}
}
class main
{
psvm(str
{
s.o.p("en
movie m1=
s.o.p(
s.o.p(
s.o.p(
s.o.p(
movie m
s.o.p(
s.o.p(
s.o.p(
s.o.p(
s.o.p(
}
}
```

* with
initialize
different

parameterized constructor

```
class movie
{
 string moviename;
 string language;
 double duration;
 double price;
movie(string uname, string lan, double dur, double pri);
{
  moviename = uname;
  language = lan;
  duration = dur;
  price = pri;
  }
 }
class mainclass4
{
psvm(string[] args)
{
 s.o.p("main method started");
movie m1 = new movie("captain America", "English", 2.2, 400);
 s.o.p(m1.moviename);
 s.o.p(m1.language);
 s.o.p(m1.duration);
 s.o.p(m1.price);
movie m2 = new movie("yaiamana", "Kannada", 25, 500);
s.o.p(m2.moviename);
s.o.p(m2.language);
s.o.p(m2.duration);
s.o.p(m2.price);
s.o.p("main method ended");
}
}
```

* with the help of parameterized constructor we can initialize different values to data member present in different object at the time of object creation.

```
class program2
{
  void program2()
  {
    S.O.P(" program 2 ");
  }
}
class needclass5
{
psvm (String [] args)
{
S.O.P ("needs method started");
program2 ref = new program2();
ref.program2();
S.O.P(" needs method ended");
```

→ referring non static function members (when return type is preceded)

* If constructor with return type, it is considered as method or function
* if constructor without return type, it is not a method
if method is returned then it shows an error.

```
void program()
{
  S.O.P ("program 2");
}
```

↓ method type

```
ref.program2();
```

return ref

```
program2()
{
S.O.P ("program2");
}
```

↓ constructor

static function members
m that it preceded)

e, it is considered as

, it is not a method
as an error.

program2()

.O.P ("program2");
}

constructor

Bafna Gold
Date: Page:

psvm (String [] args)
{
S.O.P ("main method started")
program6 P1 = new program6();
S.O.P (P1.a)
}
}

O/P
zero arg
2 - double arg
int and double arg
double arg int
2 - int arg.

constructor overloading

18/3/19

defining multiple constructor within the class which has same name but should be different either by size, type or sequence.

```
PRG :- class Demo
{
  Demo()
  {
  system.out.println("zero arg");
  }
  Demo(int a)
  {
  System.out.println("1 - int arg");
  }
  Demo(double b)
  {
  system.out.println("1 - double arg");
  }
  Demo(int a, double n)
  {
  system.out.println("double & int arg");
  }
  demo(double a, int b)
  {
  system.out.println("int & double arg");
  }
  class mainclass
  {
  public static void main(String[] args)
  {
  System.out.println("main method started");
  Demo d1 = new Demo();
  Demo d2 = new Demo(25,5.4);
```

18/3/19

os uses

arguments

```
demo d3 = new demo(54, 42);
demo d4 = new demo();
demo d5 = new demo(54);
System.out.println(" main method ended ");
}
}
```

Or

main method started S.o.p(" int arg: "+a (+) " double arg: "+b);

2 double arg

double & int arg

int & double arg

Zero arg

2 int arg

main method ended

↑ to print value of this all

↓ write this compulsory

this() statement

this() statement is used to call one constructor from another constructor within the same class.

```
prog :- class sample
    {
      sample()
    {
    System.out.println("zero arg cons");
    }
     sample(int a)
    {
        this();
    System.out.println(" int arg const = "+a);
    }
    sample(double b)
    {
        this(41);
    System.out.println(" double arg const = "+b);
    }
    }
```

O/P

Zero arg const

int arg const = 41

double arg const = 45.2

class mainclass →
{ public static void main (String [] args)
{
Sample.s1 = new Sample (4,2);
}
}

Sample(int c) { this(c); System.out.println("recursive"); }

rules for this statement
1) recursive constructor invocation not allowed.
2) this() statement must be first statement within the constructor body
3) multiple this() statements cannot be written within single constructor

this key word.

```
class mobile
{
String brand;
String colour;
double price;
String imei;

mobile (String brand, String colour, double price, String imei)
{
System.out.println("this = " + this);
this.brand = brand;
this.colour = colour;
this.price = price;
this.imei = imei;
}
}
class mainclass
{
public static void main (String [] args)
{
```

note: member variable & local variable
local variable can have same name
* constructor will always give priority to local variable
brand = brand;
both considered as local variable so we use this, keyword to distinguish b/w local and member variable

System.out.
mobile m1 =
System.o
System.o
System.o
System.o
System.o
mobile m
s.o.p(
s.o.p("
s.o.p("
s.o.p("
s.o.p("
s.o.p("
}
o/p
main method
this = mobile
m1 = mobile
brand =
colour =
price =

1) this key

class
object

mobile has
an object
battery,
use c

Note: th

)

ample() twic() S.O.P(print recursive)

) }

allowed.

ement within the

be written within

variable or data variable

able can have same name

will alway give priority

variables

= brand;

ent as local variable so we

key word to distinguish

and member variable

ble price; string iemi;

);

```
System.out.println("main method started");
mobile m1=new mobile("Apple","white",89999.9,"QBJA12");
system.out.println("m1="+m1);
system.out.println("brand="+m1.brand);
system.out.println("colour="+m1.colour);
system.out.println("price="+m1.price);
system.out.println("iemi="+m1.iemi);
mobile m2=new mobile("MI","Black",12000.3,"bgh4126");
S.O.P("m2="+m2);
S.O.P("brand="+m2.brand);
S.O.P("colour="+m2.colour);
S.O.P("price="+m2.price);
S.O.P("iemi="+m2.iemi);
S.O.P("main method ended");
}
}
```

o/p

main method started.

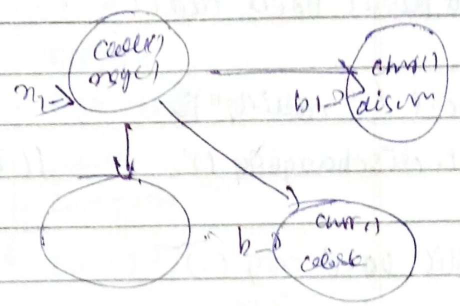
this = mobile@1db9742

m1 = mobile@1db9742

brand = apple

colour = white

price = 89999.9



rules for this key word.

1) this key word can not be used within static context

eg public static void main()
{
}

~~HAS - A relationship~~

class ~~Battery~~

{

~~public void charging~~

HAS - A relationship

mobile has an object of battery, But not calling has an object of battery or msg has all seperate object of battery.

one class having another class an object.

NOTE: this(); statement and this key word are different and rules are also change not same for both

HAS-A

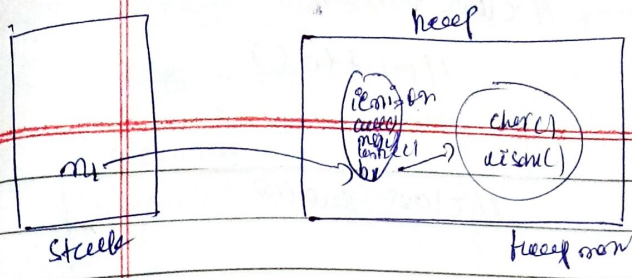
```
class Battery
{
public void charging()
{
s.o.p("charging");
}
public void discharging()
{
s.o.p("discharging");
}
}

class mobile
{
int imei = 1452;
Battery b1 = new Battery();
public void call()
{
s.o.p("calling");
b1.discharging();        // b1.discharging();
}
public void msg()
{
s.o.p("texting");
b1.discharging();        // b1.discharging
}
public void connectpower()
{
s.o.p("power");
b1.charging              //b1.charging();
}
}

class mainclass
{
Public static void main(String[] args)
{
s.o.p("main method started");
```

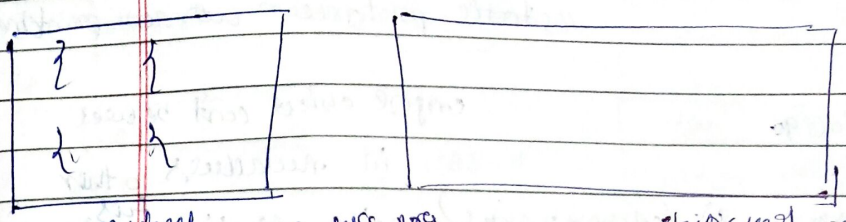
```
mobile m1 = new mobile();
m1.call();
m1.msg();
m1.connect power();
s.o.p("main method ended");
}
}
```

new mobile();

power();

method ended "/);





pkg:- → its main class pkg

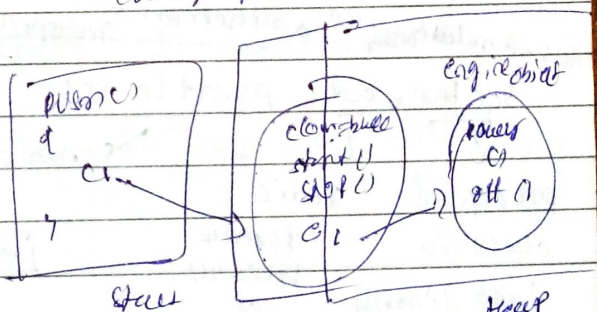
```
class mobile m1 = new mobile();
    m1.call()
    m1.b1.discharging();
    System.out.print(m1.imei);
    m1.msg();
    m1.b1.dischargeRate();
    m1.b1.charger();
    Battery b2 = new Battery();
    b2.charging();
}
```

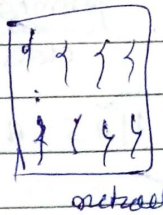
o/p

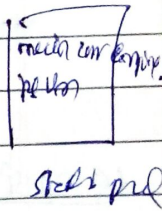
calling
discharging
1452
message
discharge
power
charger
charging

pkg:- class engine

```
{
    int power = 45;
public void turnOn()
{
    System.out.println("turn on engine");
}
public void turnOff()
{
    System.out.println("turn off engine");
}
}
class car
{ String colour = "black";
```







engine e1 = new engine();

object ref variable should be same as the one you created object in which class that class can only use otherclass can't use

```
public void start()
{
  S.O.P("start");
}
public void stop()
{
  S.O.P(" stop");
}
}
class mainclass9
{
public static main(String[] args)
{
  S.O.P("main method started");
  car c1 = new car();
  c1.e1.turnon();
  c1.stop();
  c1.e1.turnoff();
  S.O.P(c1.e1.power);
  S.O.P("main method ended");
}
}
```

// class printstream

// println()

// class system

// static printstream out=new printstream()

engine object can't be used in mainclass otherwise we like this reference.

ex c1.e1.turn()

Has-A relationship is a composition statement

System.out.println("");

- System → predefined class in java library
- out → static reference variable of printstream class
- println("") → non static method of printstream class

HAS-A relationship

composition

one class having an object of other class has its data member is known as composition.

unified modeler lang...

class diagra...

class-name
variable: da... name
method name: r...

ex:-

Engine
power: int
turnon(): void
turnoff(): voi...

battery
charging(): vo...
discharg(): vo...

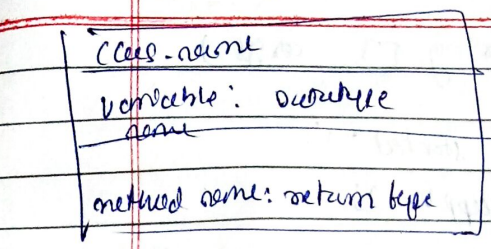
IS...

* one class acqu... known as i...
* in java inher... wor...

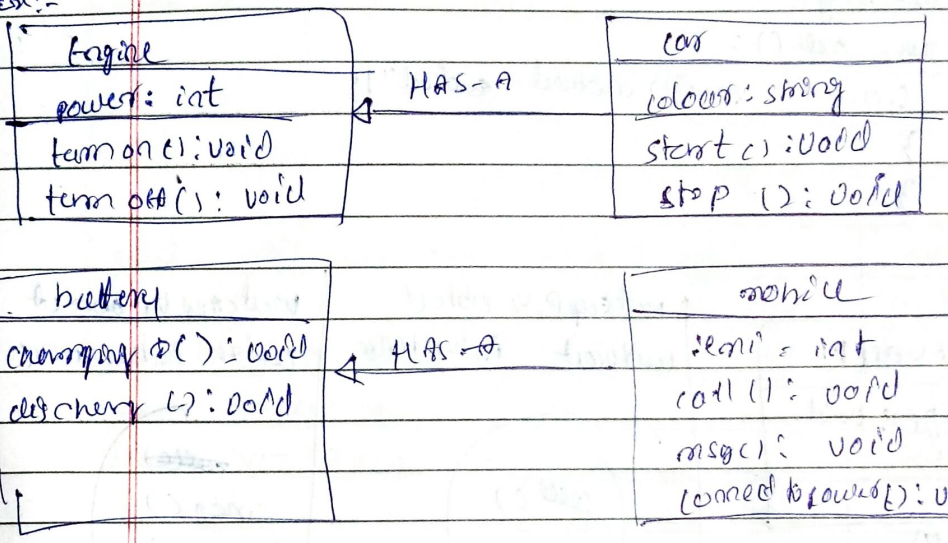
prog:- class watc...
{
public void ...
{
S.O.P(...
}
}
class watche...
{
public void ...
{
S.O.P
}}

Unified modeling language (class diagram)

Class diagrams



ex:-



IS-A relationship

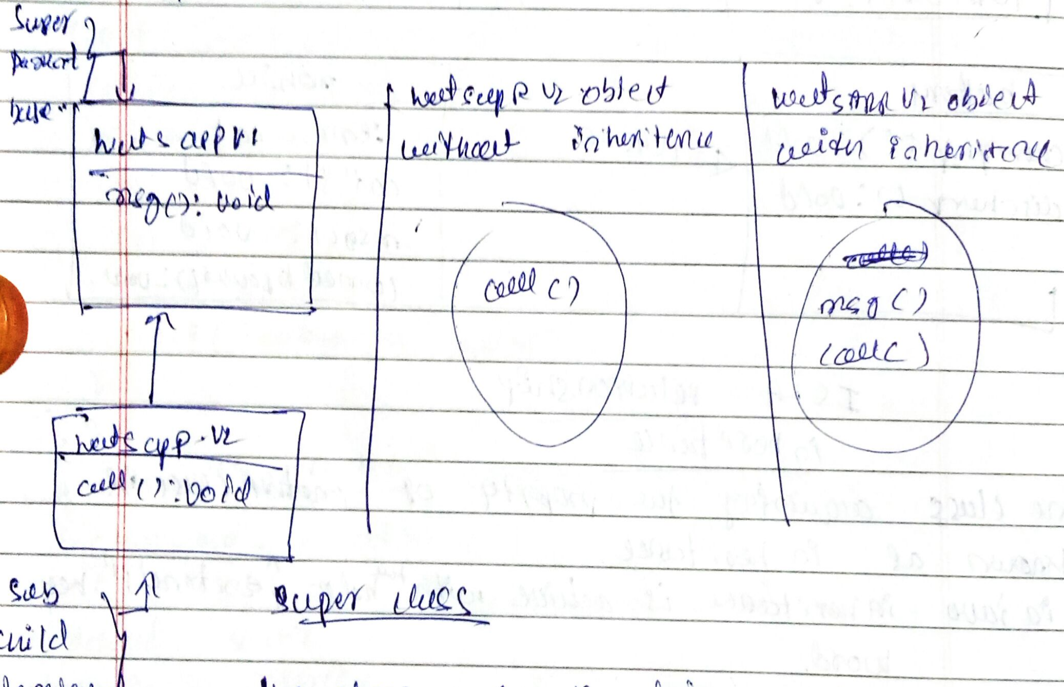
Inheritance

* one class acquiring the property of another class is known as inheritance.
* java inheritance is achieve with the help "extends" key word.

prog:- class waitsapp1

```
{
public void msg()
{
  S.O.P("msg...");
}
}
class waitsapp2 extends waitsapp1
{
public void call()
{
  S.O.P("call...");
}
```

}
class mainclass1
{
public static void main(String[] args)
{
S.O.P(" main method started ");
whatsappV2 v2 = new whatsappV2();
v2.msg();
v2.call();
S.O.P(" main method ended");
}
}



Super class

the class which is giving properties is known as super class

Sub class

the class which is aquiring the property is known as sub class

* if we create an object of subclass the sub class object will have properties of both super class & sub class
* super class will not have the properties of sub class.

Pgm :- class demo1
{ int a=21;
void test()
{
System.out.println
}
}
class demo2
{
}
class mainc
{
public static
{
S.O.P("mai
Demo2 d1=
S.O.P("a=
d1.test()
S.O.P.("
}
}

ty

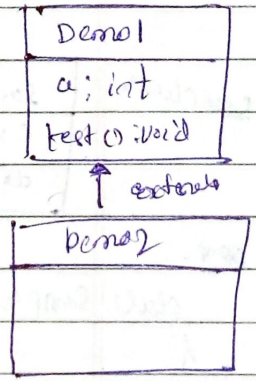
① single in
one subclass ~~one~~ of
one sub cl
super class

② multilevel
one subclass a
which in term
class is have

20/3/18

```
class demo1
{
int a=24;
void test()
{
System.out.println("test method");
}
}
class demo2 extends demo1
{
}
class mainclass2
{
public static void main(String[] args)
{
S.o.p("main method started");
demo2 d1=new demo2();
S.o.p("a="+d1.a);
d1.test();
S.o.p("main method ended");
}
}
```

o/p
main method started
a=24
test method
main method ended



types of inheritance

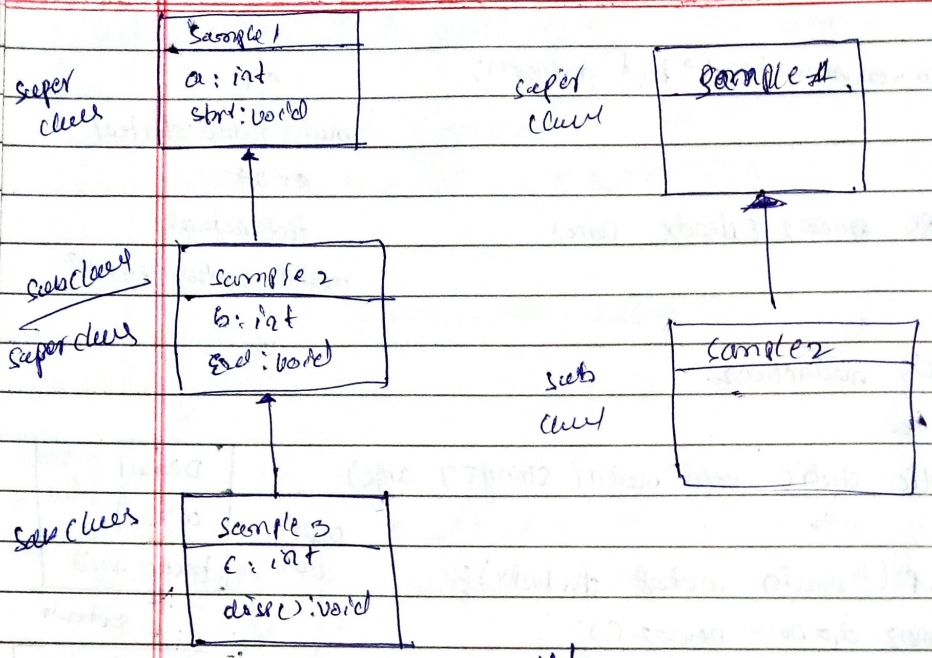
① single inheritance

~~one subclass acquiring~~ one sub class acquiring the property from one super class it known as single inheritance

② multilevel inheritance

one subclass acquiring the property from one supper class which in term acquired property from its own super class it known as multilevel inheritance

multi level



ppt.

: multi level

```
class Sample1
{
  int a = 10;
  void start()
  {
    S.O.P("Executing start method");
  }
}
class Sample2 extends Sample1
{
  int b = 40;
  void end()
  {
    S.O.P("Executing end method");
  }
}
class Sample3 extends Sample2
{
  int c = 45;
  void display()
```

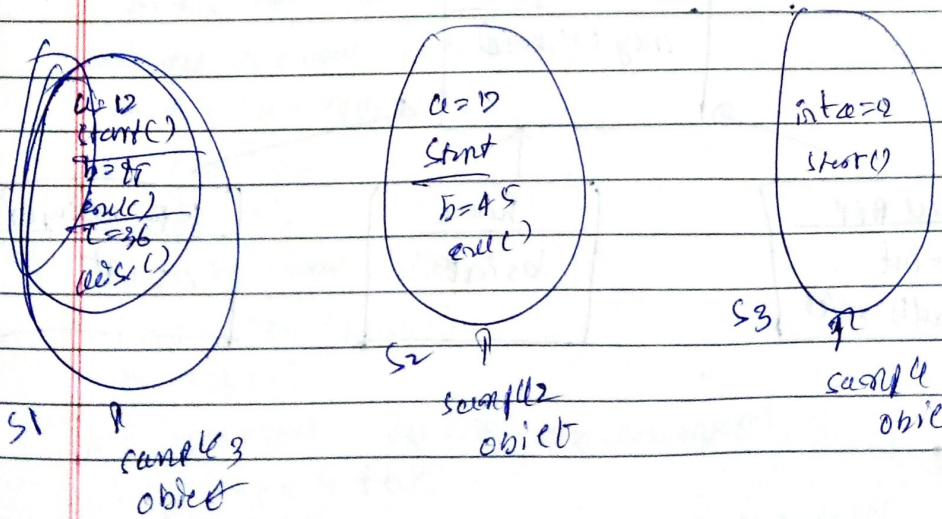
```
S.O.P("display
}
class main class
{
  P S V M(stri
  {
    S.O.P("ma
    Sample3 S
    S.O.P("a
    S.start(
    S.O.P("b
    S.end()
    S.O.P("
    S.disp()
    Sample3
    S.O.P("c
    S2.start
    S.O.P("
    S2.end
    Sample3
    S.O.P(
    S3.sta
    S.O.P(
  }
}
```

a = 10
start()
b = 40
end()
c = 45
disp()

S1

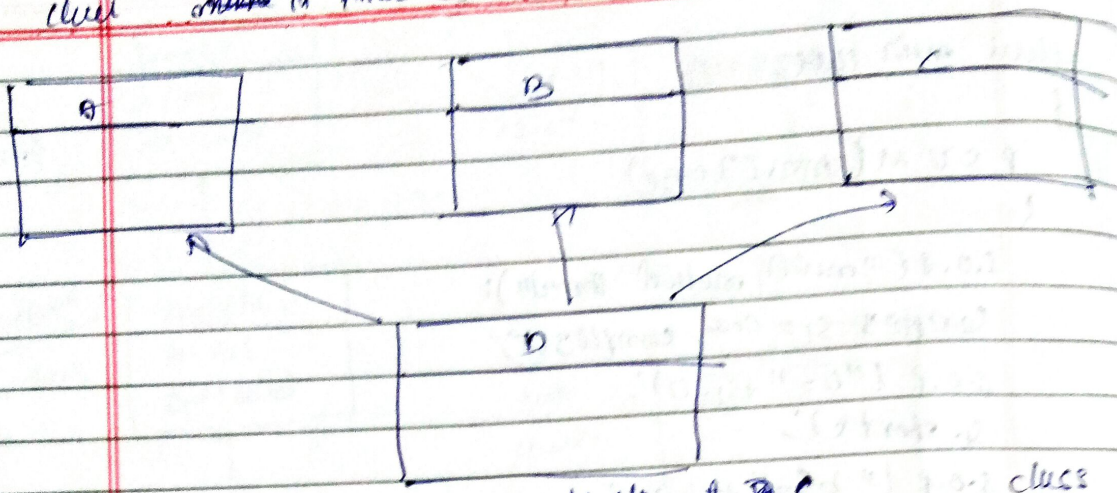
Sample
obje

```
    System.out.println("display");
  }
}
class main class3
{
  PSVM(String[] args)
  {
    S.O.P("main method started");
    Sample3 s1 = new sample3();
    S.O.P("a=" + s1.a);
    s1.start();
    S.O.P("b=" + s1.b);
    s1.end();
    S.O.P("c=" + s1.c);
    s1.disp();
    Sample2 s2 = new Sample2();
    S.O.P("a=" + s2.a);
    s2.start();
    S.O.P("b=" + s2.b);
    s2.end();
    Sample1 s3 = new Sample1();
    S.O.P("a=" + s3.a);
    s3.start();
    S.O.P("main method ended");
  }
```



3) multiple inheritance

one sub class acquiring the property from more than one super class is known as multiple inheritance.

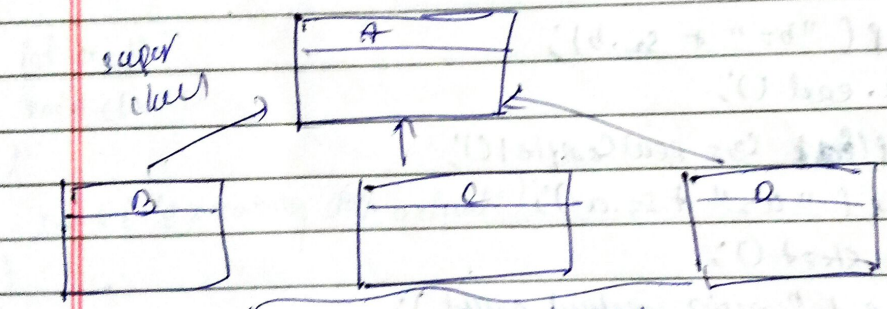


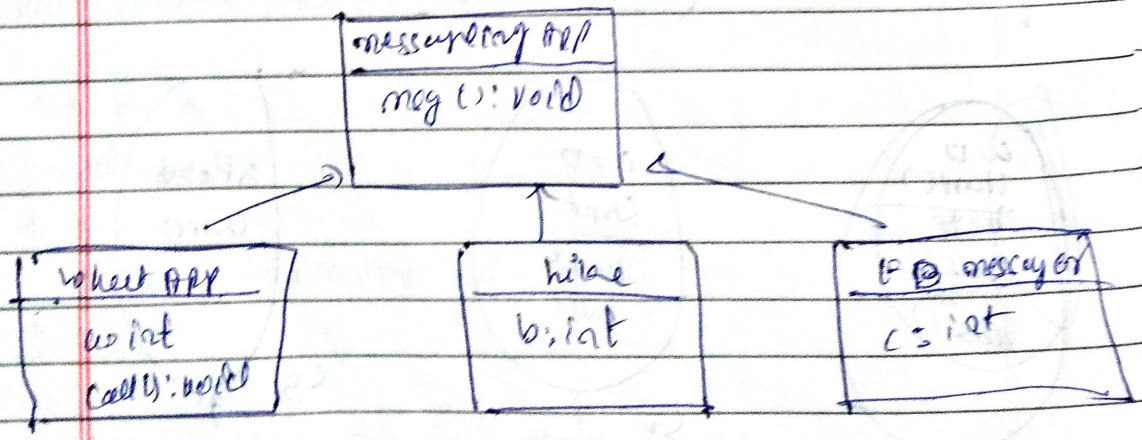
class D extends A, B, C class

multiple inheritance is not supported in java using class.

4) hierarchical inheritance

multiple sub classes acquiring the property from one super common super class is known as hierarchical





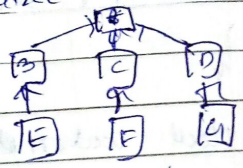
prog:-

```
class messaging app
{
  void msg()
  {
    S.O.P ("send the message");
  }
}
class waitsapp extends messageapp()
{
  int a=12;
  void call()
  {
    S.O.P ("calling...");
  }
}
class like extends messagingApp
{
   int b=35;
}
class FB messenger extends messagingApp
{
    int c=44;
}
class main class
{
p.s v m ( string[] args)
{
  S.O.P ("main method started.");
waitsapp v1 = new waitsapp();
S.O.P ("a = "+v1.a);
v1.msg();
v1.call();
like h1 = new like();
S.O.P ("b= "+h.b);
h1.msg();
FB messenger fb = new FB messenger();
S.O.P ("c = "+fb.c);
  fb.msg();   S.O.P("main method ended");
}
```

hierarchical inheritance helps us to achieve code reusability

5) hybrid inheritance

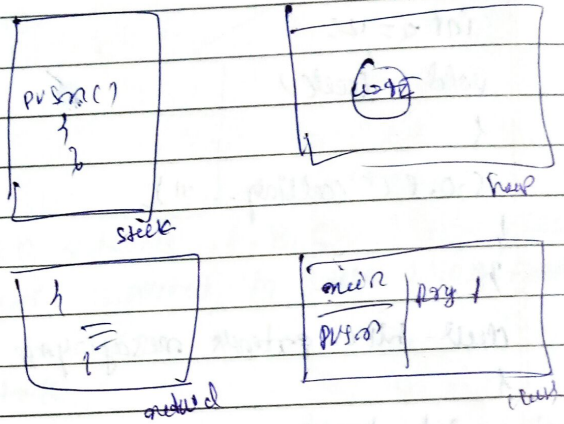
combination of more than one inheritance is known as hybrid inheritance



constructor memory location or area:

class program1
{
int a;
program1()
{
a=45;
}
}
class mainclass5
{
public static void main(String[] args)
{
S.O.P("main method started");
program1 ref=new program1();
S.O.P("a=" + ref.a);
S.O.P("main method ended");
}
}

class program1
{
int a;
program1()
{
a=45;
}
}
class program2 extends program1
{
int b;
program2()
{
b=35;



where in memory, the constructor is called by new operator and created in stack and exit (because stack use FILO-First in Last out)

class mainclass
{
public static void main(String[] args)
{
program1 ref=new program1();
S.O.P("a=" + ref.a);
S.O.P("b=" + ref.b);
S.O.P("main method ended");
}
}

Super st
class prog
{ int a
S.O.P("prog
a=45;
}
}
class program
{ int
preproc
}
Sup
S.O.P("
b=35;
}
}
class mai
{
P S V m(
{
S.O.P("
program
S.O.P
S.O.
S.O.
}
}

class main class
{ PSVM(String[] args)
{ S.O.P("main method started");
program2 ref = new program2();
S.O.P("a=" + ref.a);
S.O.P("b=" + ref.b);
S.O.P("main method ended");
}
}

Super statement

class program1
{ int a → program2
{ S.O.P("program1 = " + res);
a = 45;
}
}
class program2 extends program1
{ int b
program2()
{
super();
S.O.P("program2");
b = 35;
}
}
class main class
{
PSVM(String[] args)
{
S.O.P("main method started");
program2 ref = new program2();
S.O.P("a=" + ref.a);
S.O.P("b=" + ref.b);
S.O.P("main method ended");
}
}

op
a = 45
b = 0
because constructor will not execute.

class demo extends object
{ // properties
construct(super() statement)
}

default the properties and constructor class will inherited to demo class.

op
main method started
program1
program2
a = 45
b = 35

constructor can be inherited but class program1 it allocates

* if we write class without extends keyword by default it extends from object class → super() → default
* At every class is given by default will have the following property ① constructor ② 11 properties object class.
→ super() - default executed then above process will happen (sub class is empty no body, it will have 11 properties)

Super statement

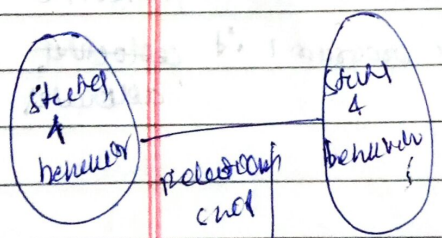
* super statement is use to call a constructor of super class from sub class constructor
* if the programmer doesn't write any super statement within the constructor by default. compiler will write zero argument super statement.

Rules for super statement

* super statement should be the first statement within the sub class constructor.
* more than one super statement can't be written within single sub class constructor
* we can't write this statement as word of super statement within single sub class constructor

note → block will execute first then constructor will execute

Q1 | 3 | 14



i) has-a
ii) is-a

has-a relationship
" object containing another object
e.g.
A type object has a reference of B type object
1) car has a Engine
2) phone has a battery
3) Boy has a book
4) pen has refill
5) Boy has a heart
6) girl has a brain

A-types
① composition
② Aggregation

IS-A rela
" B type o
" B object
eg:- 1)
2)
3)
4)

Tw
① exte
② imp

Excep
→ bindin
eg :-

Jav
collect
org
e.g.