- java is platform independent language
- java virtual machine (JVM)
- compiler and interpreter
- variables
- Integer class
- string class
- Function overloading
- New keyword
- constructor overloading
- this keyword
- static block
- init block
- tostring
- innerclasses
- Exception Handelling
- Exception class
- Throwable class
- Interface
- Abstract classes
- Runtime Exception class
- stack trace Element class
- stack trace Element methods
- Scanner class
- file handling
- GUI Devlopment
- Pacakages

Random methods

library wala program

methods to convert string to int

- stackTrace()
- toString()

jshell ~~by~~ b1 = b1 + 114

jshell float f = 3.14

jshell float f = 3.14f

ell compiler How to specify value for
rimitive datatype.

a instance what class represents (value)
nce type holds the objects

bits

8
16
32
4

float - 32 } → bits
double - 64

char - 0 → 16 (2 bytes)

boolean → it can hold (2 bytes)
only literals.
true / false

long int

Page No.
Date / /

Datatype
What is datatype?

Primitive datatype: Predefined datatype (Standard library)
Reserved keywords.

Primitive datatype comes with reserved keyword.

byte, short, int, char, float, double, long, boolean
these are the part of primitive datatype.

	bits	byte
1	byte - 1 bits, float - 32 } IEEE	byte - 1
2	short - 8 bits, double - 64 }	short - 2
4	int - 16 bits, boolean - 1 bit can	int - 4
8	long - 32 bits, char - 2 hold only literal	long - 8

int long = 15
it is not correct because long is reserved datatype in java.

gedit first.java (it creates file)

javac first.java { here first.java is my file
ls (ffirst.class) name & ffirst is my
java ffirst class name}
executed

jshell> Integer i = 45
i ==45
jshell> i.getclass()
==> class.lang.Integer.

Page No.
Date 17/1/24

Program - set of instruction
Process - If the program is running then it is called a process.
Object - logical entity

Why we use static, How static works

jshell byte b1 = 13

jshell ~~by~~ b1 = b1 + 114

jshell float f = 3.14

jshell float f = 3.14f

* tell compiler How to specify value for primitive datatype.

object → At a instance what class represents (value)
Reference type holds the objects
Types
Values

bits

byte	- 8	float	- 32
short	- 16	double	- 64
int	- 32	char	- 0 → 16 (2 bytes)
long	- 64	boolean	→ it can hold (2 bytes) only literals. true / false

→ bits

23-1-24

Page No.

Date / /

Java

To check size

→ Datatypename.BYTES

→ C.BYTES

jshell> char c = 'a' (2bytes)
==> 'a'

jshell> char c = 'ab' (in java it requires 4 bytes)
error

check its hexadecimal values.

Whenever we dee wing . (3.) it will treat it as a double.

jshell> Byte b = 13
==> 13

jshell> b.BYTES
==> 1

jshell> b = b + 114 × (it forces JVM to store the result in byte)
int cannot be converted into java.lang.Byte

jshell> b+114
==> 127. ✓

Page No.
Date / /

byte, short, int though they have different size your JVM will treat them as integer.

Whenever we are performing some operation on normal number JVM treat them as integer by default. you have to tell JVM whether it is long or not.

jshell long l = 2147483648l

jshell > 1/2 ✓

jshell > int k = 8l/2 it will not work

Overflow

jshell > b
==> 13

jshell > b+114
>127

jshell > (byte) b+115

jshell > (byte)(b+115) it forces the compiler to store in byte.

Page No.
Date / /

10 assignment 2022bit151

Create class name using SY Reg NO.
use Task for each & every function
return type ps.

work on character class. methods
(few which are imp)

10 function.

jshell > int i = 56_667
==> 56667.

class father {
    int balance = 23_00_000;
    public static void main(String args[]) {
        System.out.println("Hi");
        father f = New father();
        int balance = 2.00
        System.out.println(balance); → 200
        System.out.println(f.balance); → 2300000
    }
}

it is the part of class variable.

it will print 2300000

String static surname = "Gandhi"; ✗

Static String surename = "" ✓

All non-static variable/member is instance variable.

int balance = 20_00_000 ← instance variable
static Steing surname = " " ← static variable.

You can use classname also to print the surname of ~~fath~~ i.e (Father.surname)

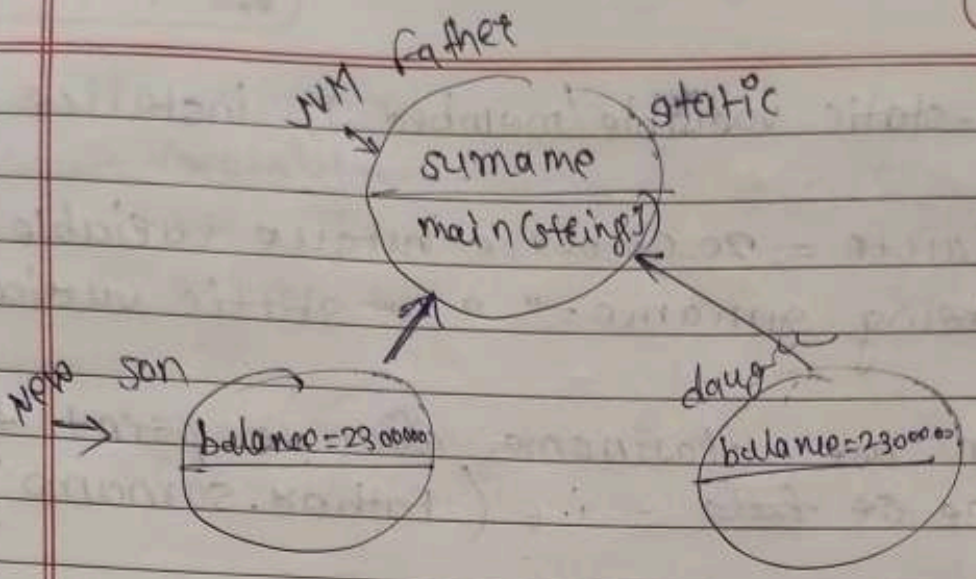
There are two types in java
primitive types
reference type

function signature :- function name (args[]).
main(Steing[])

int i = 5
float i = 5.0f } In general it is not allowed because same identifier but in func overloading it is allowed.

class Father {
int balance = 2300000
static Steing surname = Gandhi
public static void main (steing arg[]){

Father son = New Father();
system.out.println (son.balance) //2300000
- " — (son.surname) // Gandhi;
- " — (father.balance)



~~son.bal~~

```
Son.surname = "Modi";
System.out.println (son.surname);   // Modi
    - // -          (father.surname)  // Modi
    - // -          (daughter.surname) // Modi
```

```
Int balance;
static Steing surname;
static boolean statu;
            meiin {
    father son = New father();
    SOP (son.balance)
    SOP (father.surname)
    SOP (father.surname.length());
    SOP (Father.statu)
}
```

Page No.
Date / /

```
public Steing getsurname (father obj){
    return obj.surname;
}
```

can access the above instance in main ~~with~~ using son

```
Steing mysurname = son.getsurname (son);
```

Difference bet^n static method & instance method.

Instance method	Static method.
It requires an object of class	doesn't require an object of class
can access all attributes of a class.	It can access only the static attribute of a class.
The method can be accessed only using object reference.	The method is only accessed by class name.
Obj.ref. methodname()	className. methodname()
It's an exa of pass-by-value programming	pass-by-reference.

Identify if the given registration number in the array is valid or not

Firstly take the steing array.

length → 1st

Substeing

If we are not assigning any value to the variable then JVM will automatically assign value according to it's datatype.

JVM assign default values only to class variable Inside any function you need to initialize.

```
public class ByteExample {
    public static void main(Steing[]. args){
        byte myByte = 42;
        System.out.println("The value of myByte is "+myByte
    }
}
```

Array of integers

```
int[]. array = new int[5];
```

Page No.
Date / /

```
class Father {
    int balance = 2300000;
    static String surname = "Gandhi";

    public static void main(String[] args) {
        System.out.println("hii");

// Access the static variable directly
        System.out.println(surname);

    Father son = new Father();

// can access instance variable wing obj/instance
    System.out.println(son.balance);

// modify surname wing son object
    son.surname = "Modi";
    System.out.println(son.surname);    //Modi
    System.out.println(Father.surname); //Modi

// You can access surname directly wing class name
but cannot access balance because it is not static variable.
```

2024

0 1 0 1 / 0 0 0 0 (8 4 2 1)

Page No. 16
Date 18/1

```
jshell> 45/256
$==0

jshell> 278/256
==>1

jshell> 2024%256
==232

jshell> 232-256
==-24
```

Three types of variables.
1) local variable - which we define inside any function
2) instance variable
3) static variable / class variables

Two Types
primitive (comes with Reserved wd)
Reference

call by reference (value remains the same because we passes the reference)
call by value (it may changes because we directly passes the value)

same method signature

```
~~void~~ public void fun(int a, int b){
}
    public void fun (int b, int a){
    }
```

compilation error: method (int int) method already exist

Page No.
Date / /

Function Overloading.

```
class father{
    int balance = 0;

public static void main(String[] args) {

        father son = New father();
        son.fun (10, 10);
        son.fun (10l, 10);

    }
public void fun (int a, int b){
        S.o.p ("Insid fun (int, int)");
        S.o.p (a+b);
    }
public void fun (long b, int a){
        S.o.p ("Inside fun (LONG, int);
        S.o.p (a+b);
    }
```

another cond^n

```
public void fun (int a, long b){
        -//-
    }
public void fun (long b, int a){
        -//-
    }
```

it will throw error ambiguous

to solve this error

```
son.fun (10, 10l)
son.fun (10l, 10)
```

son.fun(10L, 10);

Page No.
Date / /

```
public int fun(long b, int a){
    S.op(      )
    S.o.p(a+b);
    return a+b;
}
```

error : incompatible type

because it suppose to return integer but it is returning long.

```
son.fun(Integer.MAX_VALUE, 10);

public int fun(~~int~~ long b, int a){

}
```

error : possible lossy conversion

If both the parameter are int then it will return int. No matter what the return type is.

```
S.o.p(son.fun(Integer.MAX_VALUE, 1));

public long fun(int a, int b){
        S.o.p(a+b);
        return a+b;
}
```

execute it

Page No.
Date / /

```
public static void main(string[], args){

    father son = New father()
    System.out.println(son);
    father daughter = New father()
     system.out.p~   (

{
public int ~~void~~ fun(int b){
     ~~sysout~~
     return b;
  }
```

Add one more functionality, provide the count of registration numbers whose ~~having~~ sequence number less than 50 or equal to 50.

~~count of sequence no.~~

funname ⇒ ~~approch~~ get sequenceNoless than or equ to 50.

```
                    arr1  arr2
create array of size 1000, 5000
   A1k    1000
   A5k    5000
   A20k   20000
   A50k   50000
create random enrollment year
 getrandomDigits(a):
```

explore integer class method
System class (Time)

Page No.
Date / /

e/g

Approach	A4k	A8K	A20k	A80k
App1				

Generate random number
Apply same number for different appr~

7-2-2024

In function overloading return type can be different for two diff function but the signature is same

classname is also a userdefined type.

```
class father {
    public static void main( ) {
        father son = New father();
        S.o.p ( son.fun(son));
    }
    public long fun(father b) {
        S.o.p (b)
        b.balance = 30;
        return b.balance;
```

public long fun(father b) {

}

error : incompatible type father cannot be converted into long.

~~Father get Int ( ) {~~

~~public fatt~~ public object fun(father b){
~~son~~

}

it will also work.

```
public int fun(int...b){
        String[]S = {"SGGS"};
        S.o.p (S);
}

public static void main (  ) {
        father son = New father();
        S.o.p ( son.fun (3, 10, 15));
}
public int fun (int...b){
        S.o.p (b[0]);        // 5
        return 0;
}
OR
public int fun (int a, int...b){
        S.o.p (a);
        S.o.p ( b[0] + "length is" + b.length);
        return 0;
}
```

Page No.
Date / /

```
public int fun ( int a, int...b, steing s){

}
```

error: varargs parameter must be the last parameter.

New operator assigning memory to balance
JVM assign memory to static & string sum

*for each loop

```
jshell > int i = 7.89
```
error:
incompatible type: possible lossy conversion from double to int.

```
class father{
    int balance = 100;

        public static void main ( ) {
            father son;
            father dau = new father();
            if( 3 == 3){
              return;
            } else {
                s.o.p(son);
        }
```
output: 0

8-2-2024

Page No.

Date / /

```
class father {
    int balance;

    public father (){}

    public static void main ( ) {
        father son;
        father daughter = new father();

        s.o.p (daughter);
    }
```

If constructor is not provide jvm will automatically provide it.

Constructor don't come with returntype if it is then it is function.

Do not provide any argument to the constructor

```
// father son = new father(6);
```

this represents the newly created object.

```
class father {
    int balance;

    public father () {
        this (4);
        balance = 567;
        s.o.p (" ");
```

Remember the goal

Page No.

Date / /

```
public Father (int a) {

        S.o.p (" ");
        s.o.p(a);

}
public static void main() {
    Father son;
    ~~fo~~
    Father ~~son~~ daughter = New Father (6);

}
```

execute it.

```
public Father() {
        this(5);
    // this(23);  ⟶ with or without
        balance=567;
        S.o.p (" Inside Default cons");
}
public Father (int a) {
        S.o.p (" Inside Father(int) cons");
        S.o.p(a);
}
public static void main (String[] args) {
        Father son;

        Father daughter = New Father();

}
```

error: call to this must be first statement in constructor

execute this this(23);

Here we are calling one constructor inside another constructor.

instance variable.

```
class father {
    int balance=123;

    public father () {
        this (5);
        balance = 567;
        S.O.p (" ");
    }

    public father (int a) {
        balance = balance;
    }

    public static void main(String[] args) {

        father daughter = New father ();
        S.O.p (daughter.balance);
    }
```

local variable

→ 123

```
    public father (int balance) {
        this.balance = balance;
    }
```

this is a keyword which represents current object.

We can call the parameterized constructor in main like by creating object

ex: father son = New father ();

Signature : list of modifiers.

```
}

public void Father (int balance){
        balance = balance;
}
public static void main (      ){

    Father daughter = new Father(2,55);

}
```

it will throw an error here.

execute it.

error: constructor Father in class Father cannot be applied to given types.

to solve this error we have to get the above Father parameterized constructor.

Page No.
Date / /

```
public father (int balance){
        S.O.P(balance =123456);
        S.O.P(balance);
```

rest of the code is same

output:

```
123456
123456
123
```

```
pablic father (float balance){
    if (balance+0.2==0.5){
        S.O.P(balance =123456);
    }
    S.O.P(balance)
}
public static void main(      ){
    father daughter = New father(0.3);
}
```

eeeoe: possible lossy conversion from double to float

again execute it with father daughter = New father(0.3f);

~~again execute it with father daughter = New father(0.3f);~~

```
public father (int balance){
        S.O.P(balance+0.2)
    if (balance+0.2 ==0.5){
        S.O.P(balance =123456);
    }
    S.O.P(balance);
}
```

execute it

Page No.
Date / /

```
class college {
    college () {
        S.O.P (" College Cons ");
    }
}

class SGGS extends college {
    SGGS (Int x) {
        S.O.P (" SGGS Const ");
    }
    public static void main (Steing[] args) {

        SGGS sggs = New SGGS (5);
    }
```

it will firstly call college constructor & then SGGS constr
Pehle Papa gaye fir badme mai aayi.
Inheritance & its properties.

observation why there is a time difference betn two approaches while executing two diff approaches → Practical

20-2-24

```
class SGGS {
    SGGS (float x) {
        S.O.P (          );
    }
    static public void main (Steing args[]) {
        SGGS sggs = New SGGS (0.3);
    }
```

SGGS sggs = New SGGS (0.3f) ← compiler will treat it as double.

Page No.
Date 20/2/24

Library management system
models- Issue, Renew, member-student, faculty, staff

* See the syntax of int main in C, C++

java SGGS.main ← this is the valid syntax for calling any static method.

```
class SGGS {

    static {
        S.O.P("SGGS's static block");
    }
    SGGS(float x) {
        S.O.P("constructor");
    }
    static public void main(String args[]) {

        SGGS S1 = New SGGS(0.3f);
        SGGS S2 = New SGGS(0.3f);
    }
```

output: SGGS's static block
constructor
constructor

if we are changing the sequence then also the output of static block will execute first.
irrespective of the sequence it will execute first.

Static block will be executed at the time of class loading.

When class is loaded inside the main static block will get executed first.

```
class A {
    static {
        S.O.P ("A's static block");
    }
}
class SGGS {
    SGGS (Float x) {
        S.O.P ("SGGS - constructor");
    }
    static {
        S.O.P ("SGGS's static block");
    }
    static public void main ( ) {
        A SGGSSGGS = new A();
        SGGS s1 = new SGGS(0.3f);
    }
}
```

At the time of execution we execute it with java classname here it will first execute the SGGS class means static block of SGGS & then a static block of A.

```
jshell >>
    java.lang.Integer

    Integer()
```

Page No.
Date / /

method syntax

modifier returntype identifier ();

jshell>>
java.lang.Integer
Inside jshell it will not work because it is a method.
>> Integer()
error: cannot find symbol.

-> New Integer() it will work.

and also -> (Integer) 123;

tool
-> java
lang
Integer.java <- these dee class
that's why it will not work in inside jshell

if New Integer()
error: no suitable constructor found for integer (no arg)

It will automatically invoke constructor

Page No.
Date / /

<public> java.lang. Integer Integer ( )
↑ modifier ↑ returntype ↑ Identifier

* Library management System.
methods →
Add books → Staff
Remove → Staff
Renew → ~~Re~~ student, faculty
Issue → student, faculty
Return → student, faculty.
New student Registration → staff
Time → 15 days for student, 30 days for faculty

firstly i will create a class which contains the function add books, Remove books, new student registration,

@ Methods for ~~staf~~ library staff

1) AddBooks
2) ReturnBooks
3) NewStudentRegistration
4) RemoveBooks
5) Checkcertainbook is available or not
6) Can assign particular time for renew.
7) Issue.

Page No.
Date / /

Methods for students
1) checkbookstatus
2) checkdaysremainingfor renew.
3) penaulty (Pay & check Penaulty cost)
4) checkNumberof books issued.

Faculty → checkbookstatus.
same as student.

~~faculty~~

~~class staff~~

~~fss~~ issue (id/RegistrationNO, booknome) {

}

Renew (~~bo~~ book) {

}

Return (book) {

}

Remove (book) {

}

penaulty

Page No.
Date / /

jshell > int i1 = New Integer(124)
↑
Primitive type/object

jshell> int i2 = New Integer(124)

jshell > i1 = i2
==> false

New operator will asign new addrey to i1 & i2
here we are comparing the addresses of i1 & i2.

Integer i2 = 124;

Integer class (object reference)

Integer i1 = 124;

==> i2 = i2
==> true.

Integer i1 = 786;

Integer i2 = 786;

==> false

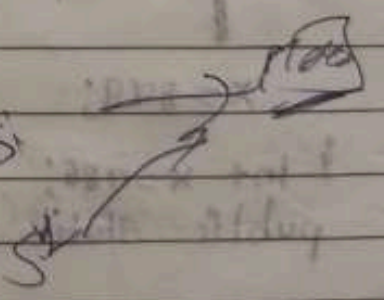
jshell> (Integer) 124 == (Integer) 124
==> true

jshell > (Integer) 127 == (Integer) 127
==> true

jshell > (Integer) 786 = (Integer) 786
==> false

Cache memory.

Cache memory.



Page No.
Date / /

Static block cannot access instance variable.
it can access static variable & static block

you cannot access non-static variable inside the static without using object.

```
class SGGS {
    int x = 786;
    static SGGS sggs = New SGGS();
    static public void main(String[] args){
        SGGS sggst = new SGGS();
        S.O.P(sggst.x);
    }
```

→ stack overflow error

To resolve this error we static SGGS sggs = New SGGS();

- Static block automatically gets invoke
- We cannot call the static block as a user.
- We can have multiple static block.
- At the time of execution sequentially the static block gets executed

```
class SGGS {
    static {
        s.o.p("At start");
    }
    x = 879;
}
int x = 786;
public static void main(@) {

}
```

Page No.
Date / /

* inti block get invoked whenever i am trying to call constructor.

```
class SGGS {

    static int x = 786;
    {
        S.o.p ("Inside SGGS init block");
    }
    SGGS(){
        S.o.p ("Inside SGGS constructor");
    }
    static public void main(String args[]){
        SGGS sggs1 = new SGGS();
        S.o.p("Inside main");
    }
    static {
        S.o.p ("Inside SGGS static block");
    }
```

- Inside SGGS static block.
- Inside SGGS inti block
  Inside SGGS constructor
  Inside main at end.

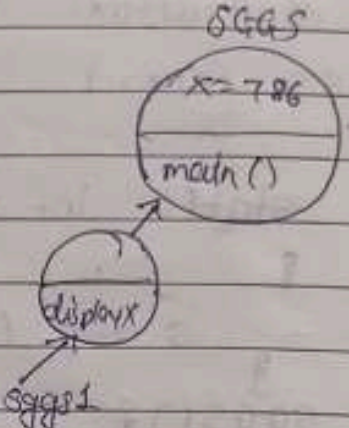
```
static int x = 786;
 {
    S.o.p (" ");
    x = 789;
    S.o.p(x);
 }
static public void main (){

    S.o.p ("Inside main " +x);
}
```

Inside the init block you can directly access the instance variable.

Page No.
Date / /

Non-static method can access the static variable directly.

```
public void displayx() {
    S.o.p(x);
}
```




```
static public void main(){
    SGGS sggs1 = NEW SGGS();
    S.o.p
    Sggst = new SGGS(); ⟶ it is valid.
```

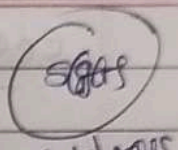
```
Static int x = 786;
int y = 2024;
{
  S.o.p("init block" +this);
} S.o.p(y);
SGGS() {
    S.o.p("Inside cons" +this);
}
SGGS(int x) {
    }
    main( ){
SGGS sggs1 = new SGGS();
S.o.p(SGGS1);
S.o.p(          )
SGGS1 = NEW SGGS(5);
S.o.p(SGGS1);
```

Here this represents the current instance of the class.

Steing is immutable means you cannot change the content.

Page No.
Date / /

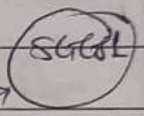
Steing s = "SGGS"



Steing s1 = "SGGS"

s == s1
true.

Steing object is immutable in java.



s = "SGGS1" it will create new object with steing Literal "SGGS1"

→ steing variable → it may vary.

== always checks the address & not the content.

Now
steing s3 = "SGGS1"

s == s3
true.

Steing s3 = New Steing ("SGGS1")

s == s3
=> false.

Page No.
Date / /

error: illegal forward Reference.

```
class college          <- parent
    static int collegeID = 456;
    {
        S.O.P ("Inside college init block:" + this);
    }
    static {
        S.O.P ("Inside clg static block" + collegeID);
    }
}
class SGGS extends college {
    static int x = 786;
    int y = 2024;
    {
        S.O.P ("Inside SGGS init block" + this);
    }
    SGGS() {
        S.O.P ("Inside SGGS constructor" + this);
    }
public static void main (String[] args) {
        S.O.P ("Inside main");
        SGGS sggs1 = new SGGS();
        S.O.P (sggs1);
}
```

Invid college static block 456
Inside SGGS static block
Inside Main
college init
SGGS init
SGGS constructor.

Page No.
Date / /

Inside college static block 456
SGGs static block
Inside Main
Inside cbg init block
Inside SGGs init

java --enable-preview fun

javac --release 21 --enable-preview fun.java.

28-02-24

StringBuffer
StringBuilder.

StringBuffer s = new StringBuffer ("abcde");
==> abcde

It will treat StringBuffer as string similar to short treat the values as int but in smaller range.

(method overloading)

jshell> s.

jshell> s
==> abcde

jshell> s.append ("abcde ab c")
==> abcde ab c

Page No.
Date / /

jshell> StringBuffer s1 = s

s1 ==> abcde ab c

s.append("SGGS")

==> abcde ab c SGGS

> s1 == s2

==> true

it is comparing the address. If the address is same the content you get will be same.

s1 gets updated.

Now s1

abcde ab c SGGS.

jshell> s

abc

s.append

s.capacity()

==> 19

jshell> s.append("1")

==> abc1

s.capacity()

==> 19

Thread

Page No.
Date / /

jshell> StringBuffer s1 = s
s1 ==> abcde ab c

s.append("SGGS")
==> abcde ab c SGGS

> s1 == s2
==> true

it is comparing the address. If the address is same the content you get will be same.

s1 gets updated.
Now s1
abcde ab c SGGS.

jshell> s
abc
~~s.append~~
s.capacity()
==> 19

jshell> s.append("1")
==> abc1

s.capacity()
==> 19

Thread

(*) static college sggs = new college()

```
class college {
    static int collegeID = 456;
    {
      s.o.p ("college init block" +this);
    }
    static {
        s.o.p ("college static block" +collegeID);
    }
    college(){
        s.o.p ("college constructor : college()::" +this)
} }
class SGGS extends College{
        static int x=786;
        static college sggs = new College();
        int y = 2024;
      {
        s.o.p ("SGGS init block" +this);
      }
    SGGS(){
          s.o.p ("SGGS constructor : SGGS()::" +this)
      }
}
  public static void main( ){
}
static{
} s.o.p (SGGS static block)
```

Simple implementation of static & init block

Page No.
Date / /

```
class SGGS {
    static int x = 786;
    int y = 2024;
{
    S.O.P ( init this)
}
SGGS() {
    S.O.P( const this)
}
static public void main( Steing[] aoys){
        S.O.P("Inside main");
        SGGS sggs1 = new SGGS();
        S.O.P(Sggs1);
        SGGS sggs2 = New SGGS();
        S.O.P(Sggs2);
    }
public Steing toSteing(){
        return "Hi";
    }
```

Inside Main
Inside SGGS init block: Hi
SGGS constru :: Hi
Hi
' init block : Hi'
SGGS constru : Hi

toHexString()

Page No.
Date / /

```
static public void main(String args[]){
    S.O.P("Inside main");
    SGGS sggs = New SGGS();
    S.O.P("Default toString() returns:" +sggs1);
    S.O.P("class's toString() returns:" +sggs1.
                                    myToString());
}
public String myToString(){
        return("y=" +y);
}
```

```
int a[] = new int[4]
```

```
String s = "SGGS"
```

~~s[0]~~ s[4]

this is a keyword which represents current object
The object which is calling the method.

```
class Demo{
    private int age;
    private String name;

    public int getAge(){
      return age;
    }
```

13-03-2024
Page No.
Date / /

```
class SGGS {

    public static void main( ) {
        S.O.P ("Inside main");
        SGGS Sggs1 = New SGGS();
        S.O.P ("Default constructor () returns:" +sggs1)
        S.O.P ("class's toSteing () returns:"
                    + sggs1.mytoSteing ());
    }

    public Steing mytoSteing () {
        return "MyToSteing" +getClass() .getName()
        +@+Integer. toHexSteing (hashCode());
    }
}
```

sggs1.mytoSteing

changes

```
S.O.P ("Default cons () returns:" sggs1);
S.O.P ("class's tosteing() returns: sggs1. toSteing());

    public Steing toSteing () {
        return "" + getClass().getName()+@
            +Integer. toHexSteing (hashCode());
```

also check it with

```
S.O.P("Default cons() returns:" );
```

If there are two or more classes we cannot make both of them public. There will be an error that second class ex: class sggs should be declared in a file named Sggs.java.

rm * .class to remove class.

Page No.
Date / /

public class SGGS extends college {

}

here class college is in another file

error: cannot find symbol
public class SGGS extends College {
When we are changing the name of the file from college.java to C.java.

String s = "sggs"
==> sggs

String s1 = "sggs"
==> sggs

String s2 = New String("sggs");
==> sggs

s.hashcode()
==> 3528256

==> s1.hashcode()
==> -11-

s2.hashcode()
-11-
The hashcode are same

s1 == s2 (here it is checking the address)
==> false

s1.equals(s2)
==> true
In equals method it is checking the hashcode

14-03-2024
Page No.
Date / /

Supeer() in inheritance.

If the content of the object are same then hascode are equal. If the hascode are equal it doesn't always mean the content are equal.

```
class College {
    Steing CollegeName;

    college (Steing collegeName){
        this.collegeName = collegeName;
    }
}

class TechUni extends college {

    Steing uniName;

    TechUni (Steing uniName){ ← supeer();
        this.uniName = uniName;
    }
    static public void main (Steing[] args){
        TechUni Sggst = New TechUni()
        S.O.P(" ");
    }
}
```

error: constructor college in class college cannot be applied to given types.

```
TechUni (Steing UniName){
```

When subclass object is created, first of all the super class default constructor is called & then only the sub class constructor is called.

For each & every constructor by default super() this thing is there it is hidden but it is always there.
This line is implicitly there but it is hidden. you can write it explicitly also. as follows:

```
TechUni (String uniName) {
        super();
        this.uniName = uniName;
```

```
        super (clgName);
```

error: non-static variable deptName cannot be referenced from a static context.

execute it & see the result

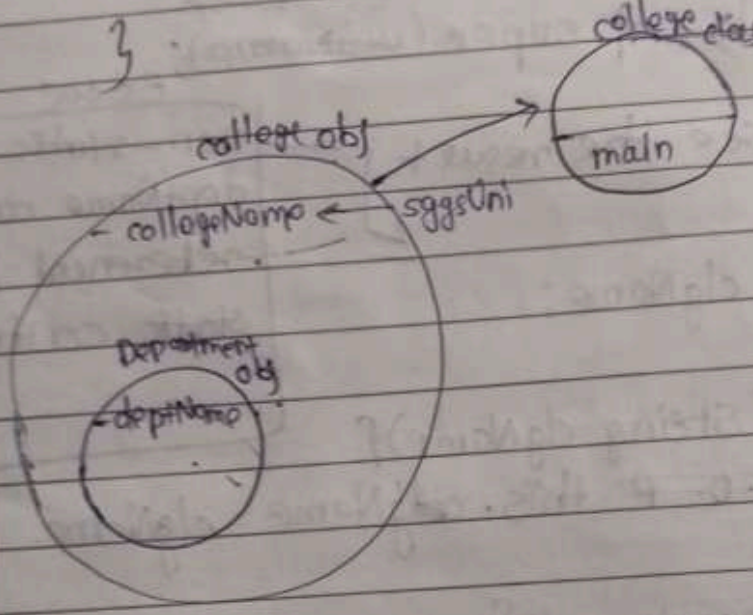
```
class College {
        String clgName;

College (String clgName){
        this.clgName = clgName.
}
class Department {
        String deptName;

        Department (String deptName){
                this.deptName = deptName;
        }
}
static public void main ( ) {
        College sgg = New College ("SGGsUni");
        S.O.P (collegeName)
        Department D1 = New Department ("IT Dept");
        S.O.P (    );
```

see the error here by executing.

error: non-static variable this cannot be referenced from a static context

```
static public void main ( ) {
    College sggst = New college ( "SGGS Uni");
    S.O.P ( sggst. collegeName);
    De      dept = sggst.new Depa  ("IT Dept");
    S.O.P (" " +sggst1);
}
```

err: cannot find symbol



Add this method to class Department

```
Steing getDeptDetails( ){
    return "college Name:" + collegeName + "Depart
                ment Name:" + deptName;

}
```

call it in main.

Page No.
Date / /

```
class college {
    Steing Name;

    college (Steing collegeName) {
        this.name = collegeName;  or
    }                              college.this.name
                                   =collegeName.
class Department {
    Steing name;

    Department(Steing deptName ) {
        this.name = deptName;
    }
    Steing getDeptDetails(Steing name){
        return  name + " In In College Name"
                + College.this.name +
                " In DeptName" + this.name;
                }
    S P V M ( ) {
        College siggit = New college ("SCET Unit");

public Steing getDeptDetails (Department dept){
```

Page No.
Date / /

```
class SGGS {
    public static void main(String[] args) {
        System.out.println("Inside main");
        SGGS sggs1 = new SGGS();
        S.O.P("Default constructor() return:" +
        S.O.P("class's toString() return:" +
                                    sggs1.myToString());
    }
    public String myToString() {
        return "MyToString" + getClass().getName() +
            "@" + Integer.toHexString(hashCode());
    }
}
```

output: Inside main
Default constructor() return: SGGS@1dbd1606
class's toString() return: MyToString SGGS@1dbd

This line prints out the result of calling the 'toString()' m... implicitly on the sggs1 object. Since you haven't overrid... the toString() method in your class, it will use the default implementation provided by object class. this default implem... includes the class name & the object hashcode.

explicitly calls the myToString() method defined in clas...

used to modify the content of mutable string

S.O.P(sggs1)

int student[] = New int[6]

eeeee:

Page No.

Date / /

int students[] = new int[6]

==>

==> int students[6]

steing s1 = "sggs";

SteingBuffer s1 = New StingBuffer("IT");

==> sggs IT.

Steing is immutable & SteingBuffer is mutable.

Subclass provides a specific implementation of a method that is already defined in its superclass.

```
class Animal {
    void soun d(){
        S.O.P ( Anl
    }
class dog extends Animal {

    void sound() {
        S.O.P ( Dog
}
```

Overloading → multiple method with same name but diff Paraame, diff method signature

Oveeriddan → method have same name, return type. & Parameter as the method in supeeclass.

19-3-2024

Page No.

Date / /

jshell> Thread.sleep(2000);

error: unreported exception InterruptedException must be caught or declared to be thrown.

```
static void disco() {
    try {
        Thread.sleep(1500);
    }
    catch (InterruptedException) {

    }
    S.O.P("\033[H\033[2J");
}
public static void main(String args[]) {
    disco();
    Pragya2024._4();
```

void modify (int x) {
this.x = x; // store local variable x into present class instance variable

}.

2-4-2024

* Inner class.

Exception → checked
Unchecked.

If we are getting exception at the time of compilation then we can check the exception.

Compile time exception.

Runtime exception → Unchecked (arrayIndexoutofbound)
→ error (Stackoverflow)

Illegal operation: Divide by zero.

In the recursive call we keep on creating objects then the memory of JVM will be full & then the stackoverflow error occurs & it is not internal (in the program).

* Command-line arguments (explore this concept)

Ex: ⇒ java ffirst java ffirst
⇒ java.

java ffirst 123.456 567
⇒ 123.456

Page No.
Date / /

Parameter & Whatever you are providing in function signature

```
public static void main (String ... args) {
    }
```

The above is the valid syntax

(String args...) ← It will give an error.

Unchecked

```
public static void main (String ... args) {
        S.O.P (args[0]);
    }
```

It will compile but.

At the time of execution

ArrayIndexOutOfBoundException

Index 0 out of bounds for length 0.

```
class FFirst {
    P.S.V.M (String ... args) {
        try {
            S.O.P (args[0]);
        }
        Catch (Exception e) {
            S.O.P ("Exception at line Number 5");
        }
    }
}
```

3-4-2024
Page No.
Date / /

getMessage()
getLocalizedMessage()
toString()
printStackTrace()

error comes at ~~the time~~ runtime.
or at the time of execution.

Parent class can we child class object

Exception class ~~ob~~ can make a call to method of Throwable class because Throwable is a parent class & exception class inherit the properties of throwable class And another point if it is private then we cannot access it.

error: 'try' without 'catch' 'finally' or resource declaration.

```
P.S.V.M (String ...args) {
    try {
        S.O.P (args[0]);
    }
    finally {
        S.O.P ("Exception at line");
    }
}
```

Page No.
Date / /

error: 'catch' without 'try'

```
P.S.V.M(String ...args){
    try {
        S.O.P(args[0]);
    }
    catch {
        S.O.P("Inside catch");
    }
    finally {
        S.O.P("Inside finally");
    }
}
```

O/P ⇒ Inside catch
Inside finally

~~java Example 0~~

java classname 0x8

O/P ⇒ 0x8
Inside finally.

checked exception → error: unreported exception InterruptException; must be caught or declared to be thrown.

```
try {
    Thread.sleep(1000)
    s.o.p(args[0]);
    Thread.sleep(1000);
  }
catch (ArrayIndexOutofBoundsE e) {
    S.O.P("Inside ArrayIndex --11-- catch");
  }
catch (InteeruptException e) {
    S.O.P("Inside Inte --11-- catch");
  }
finally {
    S.O.P("Inside finally");
  }
}
```

If i am teying to peeform some activity but i fail to peeform that time exception occue.

```
Catch (Exception e) {
    S.O.P("Inside Exception catch");
  }
```

→ Add this catch block at first & last & see the result.

```
P.S.V.M (string ..args) throws InteeruptedExc
```

4-4-2024

Page No.

Date / /

method < concrete
abstract

Interfaces contain constant variable & abstract method does not contain concrete method.

Abstract class: (don't have a definition)
Abstract class may contain concrete method
concrete method means it contains class definition.

Abstract class should contain ~~atleast~~ one abstract method

```
Interface Voter {
        void castVote();
}
interface EC {
        void checkValidityOfUser();
}
abstract class ECIndia implements Voter, EC {

        void castVote() {

        }
        void checkValidityOfVoter() {

        }
}
```

Page No.
Date / /

```
class Demo {
    p.s.v.M (Steing args[]){
        int i=0
        int j=0

        tey {
            j = 18/i;
        }
        Catch (Exception e){
            S.o.P("Something went weong" + e);
        }
        S.o.P(j);
        S.o.P("Bye");
} }
```

Catch block will be executed only if there is a exception.

```
P.S.v.M (Steing args[]){
    int i=0;
    int j=0;

    int nums[] = new int[5];
    Steing ste = null;

    tey {
        j= 18/i;
        S.o.P(ste.length());
        S.o.P(nums[1]);
        S.o.P(nums[5]);
    }
```

```
catch (ArithmeticException e)
{
    S.o.p("cannot divide by zero");
}
Catch (ArrayIndexOutOfBoundsException e)
 {
    S.o.p("Stay in your limit.");
 }
catch (Exception e){
        S.o.p("Something went wrong" +e)
 }

S.o.p(j)
s.o.p("Bye");
}
```

Exception is parent class so keep this at bottom.

```
class Demo{
    P.s.v.M(String args[]){
        int i=20;
        int j=0;
        try {
            j=18/i;
            if(j==0)
                throw new ArithmeticException("I
                    don't want");
        }
        Catch (ArithmeticException e){
            j=18/1;
            S.o.p("that is the default output");
        }
```

Page No.
Date / /

```
Catch (Exception e){
        S.O.P("Something went wrong"+e);
    }
S.O.P(j);
S.O.P("Bye");
}
```

* Custom Exception in Java

```
class NavinException extends Exception
{
    public NavinException(String string){
        super(string);
    }
}
public class Demo

try{
    j=18/i;
    if(j==0)
        throw new NavinException("I don't want to
                                  print zero");
}
Catch(NavinException e){

}
rest of the code is same.
```

You can crete your own exception with Exception class. You have to pass the msg to constructor

10-4-2024

Page No.

Date / /

~~192-II~~

error specific question
Exception handling
interface, abstract class
file handling

Throw exception

If their is a relation betn the two things like ECIndia & ECMahacashtra then you need to go with classes.

public is a weakest access specifier.
private is the weakest

gedit
vi — terminal to exit from this command press W.

abstract class Sample

error: missing method body or declare abstract.

abstract class Sample {
    void funSampleAbstract();
}

you need use abstract keyword.
abstract void funSampleAbstract();
then it will compile.

error: interface abstract method cannot have body

work with constant val

Page No.

Date / /

```
abstract class Sample {
        abstract void funSampleAbtract();
        void fun() {

        }
}
```

error: sample is not abstract and does not overeid abstract method funSampleAbstract() in sample.

```
class sample {
        -11-
}
```

@ you cannot create the object with abstract class.

Abstract class can be superclass.

Abstract class cannot hold reference ~~var~~ or instance.

Page No.
Date / /

Integer
↳ int - 4 bytes
long - 8 bytes
short - 2 bytes
byte - 1 byte → -2^7 to 2^7-1
-128 to 127

float → 4 bytes
double → 8 bytes.

char → 2 bytes.
UNICODE.
char c = 'k';

Boolean → true or false
in java (0,1) ✗
boolean b = true;

#7 Literals in java

```
public static void main(String a[]){
    // literals
    int num1 = 0b101;  (binary)
    System.out.println(num1);  //output 5
}
```

→ int num1 = 0x7E; (hexadecimal)
S.O.P(num1); // 126

→ double num1 = 56;
S.O.p(num1); // 56.0

→ char c = 'a';
c++;
S.O.p(c); // b

#8 Type Conversion and Casting.

byte b = 127;
int a = 256;
b = a;
✗

The type of a is integer which is a bigger range & byte has a small range

✓ a = b;

Widening → smaller type to larger type size
Narrowing → larger type to smaller type.

Date / /

byte b = 127;
int a = 12;

b = a; → b = (byte)a; ← explicit conversion
↓ casting.
casting.

a = b → implicit conversion

converting the integer value into a byte format.

Here we are storing a byte value into a integer.

float f = 5.6f;
int x = (int) f;

Here you will lose your point value so here when you convert it it will become 5

Complete range of byte is 256 -128 to 127

byte b = 127;
int a = 257;
b = a; → b = (byte) a; // 1

257 % 256; = 1

float f = 5.6f;
int t = (int) f;

Conversion is automatic casting.

Type promotion.

```
public static void main(String args[]){
    byte a = 10;
    byte b = 30;

    int result = a * b;
    S.O.P(result);
}
```

here byte * byte result is promoted into integer because it is going out of range.

Page No.
Date / /

#9 Assignment operators in java.

```
int num1 = 7;
int num2 = 5;

int result = num1 / num2;   // it will give quotient as a result
int result = num1 % num2;   // it will give remainder as a result.

// num1 = num1+2;
// num1 += 2;
// num1++;
// ++num1;

int result = ++num1;   // 8
int result = num1++;   // 7
```

it will first increment & then fetch the value

it will first fetch the value & then increment.

#10 Relational operators in java.

```
main
  int x = 6;
  int y = 5;
  boolean result = x < y;
  System.out.println(result);   // false
```

then check < > = < = ! = = =

Page No.
Date / /

#11 Logical Operators in java.

&&	||	!
AND	OR	NOT

#12 IF-else statement

#14 Teenary operator.

```
public class Demo
{
    public static void main(String a[]){
        int n=5;
        int result = 0;

        if (n%2==0)
            result =10;
        else
            result =20;

        result = n%2==0 ? 10:20;

        o.o.p(result);
    }
}
```

#15 Switch Statement.

Page No.
Date / /

```
public static void main(Steing a[]) {
    int n = 2;

    switch (n) {
        case 1:
            s.o.p("Monday");
            break;
        :
        case 7:
            s.o.p("Sunday");
            break;
        default:
            s.o.p("Invalid");
    }
```

New Switch statement in java

without using break.

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

    Steing day = "sunday";

    Switch(day){
        case "Saturday", "Sunday" -> s.o.p("6am");

        case "Monday" -> s.o.p("8am");

        default -> s.o.p("7am");
    }
```

Page No.
Date / /

Use switch as a expression.

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

    Steing day = "sunday";
    steing result = "";

    result = switch(day)
    {
      case "saturday", "sunday" -> "6am";
      case "Monday" -> "8am";
      default -> "7am";
    };
    System.out.println(result);
}
```

Instead of -> (arrow) we can use : yeild "6am";

#16 Need for loop in java

#17 While loop in java.

```
public static void main(steing a[]){
    int i=1
    while (i<=4){
        System.out.println("Hi" +i);
        int j=1;
        while (j<=3){
            System.out.println("Hello" +j);
            j++;
        }
        i++;
    }
    S.o.p("Bye" +i);
}
```

Hi1 Hello1
Hello1 Hello2
Hello2 Hello3
Hello3 Hi3
Hi2

Page No.
Date / /

W3 School

Variables:

Final Variable

If you don't want others (or yourself) to overwrite existing values, use the final keyword (this will declare the variable as "final" or "constant", which means unchangeable & read-only)

ex:

```
final int mynum=15;
mynum=20 // will generate an error:
```

Display variable

To combine both text & a variable, use the + charac

```
S.o.p("Hello"+n. Steing name = "John";
                S.O.P ("Hello" +name);
```

Datatypes:

Use float or double?

The precision of float is only six or seven decimal digit, while double variable have a precision of about 15 digits. It is safer to use double for most calculations.

Scientific Numbers.

A floating point number can also be a scientific number with an "e" to indicate the power of 10:

```
float f1 = 35e3f;
double d1 = 12E4d;
```

* Non-Primitive datatypes are called reference types because they refer to objects.

Date / /

- Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot.
- A primitive type has always a value, while non-primitive types can be null.
- A primitive type starts with lowercase letter, while non-primitive starts with uppercase letter.

* Java Type Casting.

Type casting is when you assign a value of one primitive datatype to another type.

1) Widening Casting (automatically)
converting smaller type to larger type

2) Narrowing Casting (manually)
converting larger type to small size type.

ex: double myDouble = 9.78d;
int myInt = (int) myDouble; // Manual casting: double to int

* Java Numbers & Strings.

Java uses the + operator for both add" & concatenation
→ If you add two numbers, the result will be a numb
→ If you add two string, the result will be a string concaten
→ If you add a number & a string, the result will be a string concatenation

String x = "10";
int y = 20;
String z = x + y; // z will be 1020 (a string

Page No.
Date / /

Steing - Special Characters

Escape character	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

ex: Steing txt = "we are the so-called \"Vikings\" from the north.";

* Java Short Hand If...Else (Ternary Operator)

There is also a short hand ifelse, which is known as the ternary operator because it consists of three operands.

```
ex: int time = 20;
    if (time < 18) {
        S.O.P("Good day");
    } else {
        S.O.P("Good evening");
    }
```

```
int time = 20;
Steing result
= (time < 18) ? "Good day" :
                "Good evening";
S.O.P(result);
```

* Java While loop

```
int i = 0;
while (i < 5) {
    S.O.P(i);
    i++;
}
```

Do while loop

```
int i = 0;
do {
    S.O.P(i)
    i++;
}
while (i < 5);
```

Page No.
Date / /

#19 for loop

```
public static void main (String a[]){
    for (int i=1; i<=5; i++)
    {
        S.O.P ("Day" + i);

        for (int j=1; j<=9; j++){
            S.O.P (" " + (j+8) + " -" + (j+9) );
        }
    }
```

When you know exactly how many times you want to loop through a block of code, use the for loop instead of while loop.

For-Each loop

There is also a "for-each" loop, which is used exclusively to loop through elements in an array.

Syntax:

```
for ( type variableName : arrayName) {

}
```

ex:

```
String[] cars = {"Volvo", "BMW", "Ford", "M
    for ( String i : cars) {
        S.O.P(i);
    }
```

Page No.
Date / /

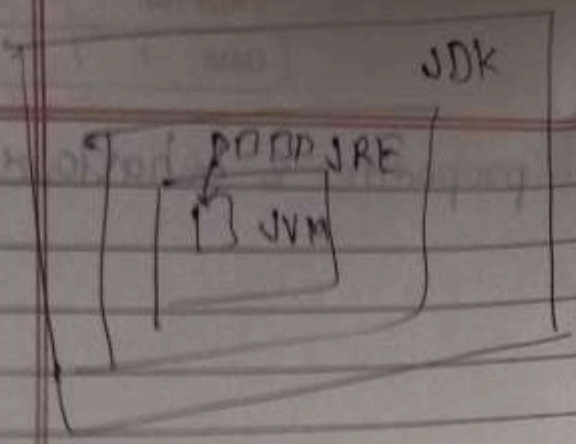
Object has two things - properties & behaviour.

#22 Class & Object.

```
class calculator {
    int a;
    public int add(){
        S.O.P ("in add");
        return 0;
    }
}
class Demo {

    Calculator ~~elae~~ calc = New Calculator();

    int result = calc.add();
    S.O.P(result);
}
// in add
 0.
```

```
class calculator{
    public int add(int n1, int n2){
        int c = n1+n2;
        return c;
    }
public class Demo {
    public static void main(String a[]){
        int num1 = 4;
        int num2 = 5;
    Calculator calc = new Calculator();
    int result = calc.add(n1:4, n2:5);
}   S.O.P(result);
```

o/p
9//



A class is like an object constructor, or a "blueprint" for creating objects.

** Methods in java

```
class Computer {
    public void playMusic() {
        s.o.p("Music playing");
    }
    public String getMeaPen(int cost) {
        if (cost >= 10)
            return "Pen";
        else
            return "nothing";
    }
class Demo {
    public static void main(String a[]) {
        Computer obj = New computer();
        obj.playMusic();
        String str = obj.getMeaPen(2);
        s.o.p(str);
    }
}
```

Page No.
Date / /

A method is a block of code which only runs when it is called.
You can pass data, known as parameters, into a method.
They are also known as functions.

Information can be passed to methods as parameters.
Parameters act as variables inside the method.

```
public class Main {
    static void myMethod (String fname) {
        S.O.P( fname + "Refsnes");
    }
public static void main (String[] args) {
    myMethod ("Liam");
    myMethod ("Jenny");
    }
}
```

O/P : Liam Refsnes
Jenny Refsnes.

When a parameter is passed to the method, it is called an argument: fname is a parameter, while Liam, Jenny are argument.

Page No.
Date / /

25) Method overloading in java.

With method overloading, multiple methods can have the same name with different parameters.

Instead of defining two methods that should do the same thing, it is better to overload one.

Multiple methods can have the same name as long as the number and/or type of parameters are different.

Different ways of method overloading
1) changing the number of parameters.
2) changing Data types of the Arguments.
3) Changing the order of the parameters of methods

26) Stack and Heap in java.

Every method will have it's own stack.

Page No.
Date / /

command line arguments.

```
class ffirst
{
    public static void main(String args...) {
        S.O.P("Inside main");
} }
```

error: ',' ';' ')' or '[' expected.

```
class ffirst{
        P.S.V.M(String ... args){
            S.O.P(args[0]);
    } }
```

n$:- javac
it will compile successfully
n$: java ffirst ~~argt~~
~~ns~~

argt
n$: java ffirst (means without any argument
Exception: ArrayIndexOutOfBoundsException:
Index 0 out of bounds for length 0.

*
```
class ffirst {
    P.S.V.M(String ...args){
        try {
            S.O.P(args[0]);
        }
        finally {
            S.O.P("Exception at line number");
} } }
```

Page No.
Date / /

You can write a try block without a catch block, but if you do so, you must include either a 'finally' block or both 'catch' and 'finally' blocks.

Error: Catch without try

This error occurs when you have a 'catch' block without a preceding 'try' block. The purpose of the 'catch' block is to handle exceptions that are thrown within the corresponding 'try' block.

ex:
```
class FFirst {
    p.s.v.m(String ...args) {
        s.o.p(args[0]);
    catch(Exception e) {
        s.o.p("Exception at line
              number");
    } } }
```

Checked exception:
Unreported exception InterruptException; must be caught or declared to be thrown occurs when you call a method that declares that it throws a checked exception, but you don't handle or declare that exception propedy in your code.

Page No.
Date / /

In Java when catching exceptions, the order of catch blocks matters. More specific exception types should be caught before more general ones.

the arrangement should be like

```
catch (ArrayIndexOutOfBon—      )
catch (InterruptedException)
Catch (Exception e).
```

```
class One{
    int x;

    One(Two t){
        this.t = t;
        x = 10;
    }
    void display(){
        S.O.P("One's x = " +x);
        t.display();
        S.O.P("Two's var = " + t.y);
    }}
class Two {
        int y;
        Two(int y){
            this.y = y;
        }
        void display(){
            S.O.P("Two's y=" +y);
    }   }
class Relate{
        P.S.V.M(String args[7]){
            Two obj2 = new Two(22);
            One obj1 = new One(obj2);
            obj1.display();
    }   }
```

An object 'obj1' of class one is created. This constructor takes an object of type 'Two' as an argument, so it initializes its instance variable 't' with 'obj2' & sets down its own 'x' to 10.

t.display(): This calls the display() method of the object 't' which is 'obj2' in this case.
Inside display S.O.P ("Two's x=" +x) is executed.
Here x belongs to the object obj2 & its value is 22.

```
class College {
    String clgName;
    College (String clgName){
        this.clgName = clgName;
    }
class Department {
    String deptName;

    Department(String deptName){
        this.deptName = deptName;
    }
    static public void main (String args[]){
        College sggsi = new College("sggs Uni");
        S.O.P (sggsi.clgName);
        Department d1 = sggsi.new Department("It depart");
        S.O.P("Hello" + sggsi);
    }
}
}
```

Inner class

o/p ⇒ Sggs Uni
~~Hello@~~
Hello College@Hadoulf3

Page No.
Date / /

```
Try {
}
Catch( ArrayIndexOutOfBoundException) {
    ....
  }
Catch( InterruptedException ex) {
...
}
Catch (Exception ex){
  ...
}
```

Output:- Work Properly.

```
try{
  ...
}
Catch(Exception ex){
...
}
Catch( ArrayIndexOutOfBoundException ex){
  ...
  }
Catch (InterruptedException ex){
  ...
}
```

Output : Error.

Page No.
Date / /

```
class Ex
{
 P.S. void m (String args[] ) throws InterruptedException {
    try {
       Thread.sleep(1000);
       S.O.P("Try 1");
    }
    Catch (Array --- ) { ... }
    catch(Exception ex){ ... }
  }
}
```

⟹ Works Properly.

Exception classes Hierarchy.

class Object
↓
class Throwable

class Exception
- class IOException
- class SQLException
- class InterruptedException
- class RuntimeException
  - ↳ class ArithmeticException
  - ↳ class NegativeArraySizeException
  - ↳ class NullPointerException
  - ↳ class IndexOutOfBoundException

class Error.
- ↳ class VirtualMachine Error
  - ↳ StackOverflow error
  - ↳ OutOfMemory error

Page No.
Date / /

Checked Exception :- The classes which are directly inherited from the class Exception represents checked Exception. For these exceptions the compiler causes unreported exception (error) at the time of compile to forcefully implement the exception handling mechanism.

Unchecked Exception: The classes which are inherited from the class RuntimeException and from class Error represent unchecked exception.
For these exceptions the compiler not cause any compile time error.

* Abstract class.

```
abstract class Sample
{
    void fun();          // will not work missing method body or
}
```

```
abstract class Sample{
    void fun(){          // will work.
    }
}
```

```
abstract class Sample
    abstract void fun();
    void funSample(){ }
}
```

We cannot create the object of abstract class. We can define abstract class reference & create subclass object.

Interface

→ It is the collection of final data members & abstract method
→ Sometimes, it only contains abstract methods
→ Methods in interfaces is by default by public and abstract, so we don't need to use prototype
→ The data members in the interface is by default public, final and static (i.e implicitly)

The interface is used to acheive multiple inheritance & polymorphism.

Syntax:-

```
access_specifier interface i-name
{
    final_member;
    ...
    abstract method;
    ..
}
```

Tuesday
30-4-2024
Page No.
Date / /

gc() → Garbage collector (detail)

many variables can point to one ~~objec~~ object.

Error: Main method not found in class PackageEx please define the main method.

```
// Import java.lang.*;
// Import java.util.Scanner;
class PackageExample{
        p.s.v.M(    ){
            S.O.P("HI:..");
        java.util.Scanner sc = new java.util.Scanner
                                (System.in);
        sc.nextInt();
}
```

```
mkdir Test
mv PackageExample.java ./Test/xyz.java.
```

$: javac -d . xyz.java

java packageExamples.PackageExample

FileReader
FileWriter.

JOptionPane showMessageDialoge
(null, "Demo", "File is created",
JOptionPane.INFORMATION_MESSAGE);

Practical file → Write about files in detail.

byteStream
CharacterStream } Explore this concepts.
BufferReader ⇐ nature of BufferReader class.

* try resource statement

* user defined exception for application.
for library Management.

explore File class. in detail
all the methods are included in it.

jshell > java.io

Given a path name you need to verify How many files are there & how many directory are there?

while o/p taking a input whether you we end of input character

whenever you use get it suppose to return something so you cannot use void while you use set you can use void

Page No.
Date / /

```
class FilesFolderExample
{   int fileCounter, folderCounter;
    public static void main(String args[])
    {
        S.O.P("HI...");
    }

Public static void main(String args[])
{
S.O.P("HI...");
FileFolderExample ffe = new FileFolderExample();
ffe.setNumberOfFilesAndFolders("./packageExample");

}
public void setNumberOfFileAndFolders(String path)
{

    S.O.P
```

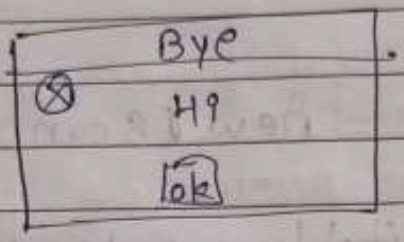
- .class
- c class
- image class
- java file

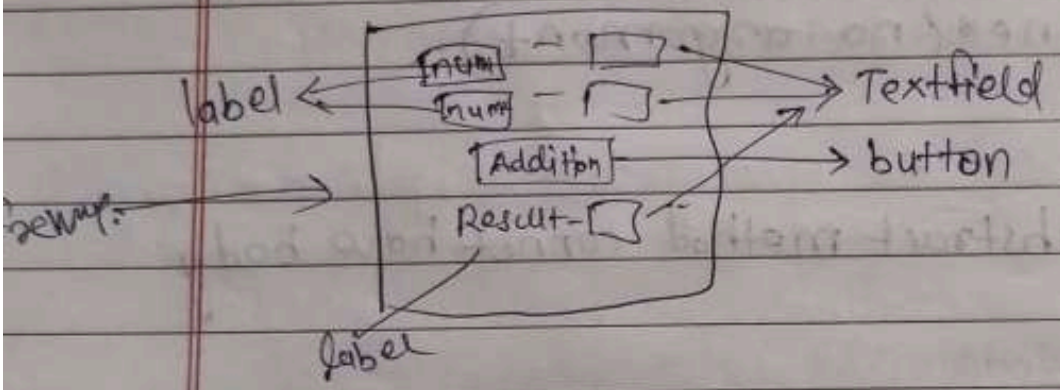
Date / /

swing, awt are used to develop GUI Application

JOptionPane.

JOptionPane.showMessageDialog(null, "Hi", "Bye",
JOptionPane.ERROR_MESSAGE);





javap. javax.swing.JLabel
.JButton.

java.awt.LayoutManager

Page No.
Date

```
import javax.swing.*;

class calsi extends JFrame{

JLabel num1Label, num2Label, reswtLabel;
JTextField num1TField, num2TField, reswtTField;
JButton addButton;

public calsi(){

num1Label = new JLabel("Num 1:");
num2Label = new JLabel("Num 2:");
reswtLabel = new JLable("Reswt:");



setLayout(new FlowLayout(FlowLayout.LEFT,20,20

public static void main(String[] args){

calsi myCalsi = new calsi();
myCalsi.setVisible(true);

}

add(num1Label); add(num1TField);
add(num2Label); add(num2TField)
```

Page No.
Date / /

java.awt.Frame

setDefaultCloseOperation (*EXIT_ON_CLOSE*);

jshell > java.awt.

jshell > java.awt.event

words starting with small letter are packages & starting with capital letter are classes.

javap java.awt.event.ActionListener.

jshell > javax.swing.JButton jb = new javax.swing.JButton();

jshell > int i = 10;

jshell > steing j = i + "";
=> "10"