Naveen Training

Index - Topics

- Data Types
- Loops, Arrays, String
- Functions
- Static & Non-static
- Constructor
- Inheritance
- Interface
- Wrapper classes
- Polymorphism
- Collections
- Abstraction
- Exception Handling ✓

Naveen
Harisankha

Naveen, Harisankha
→ TW = ① Read
② understand concept
③ do Dow que
④ practise programs

HR ⇒ 2+15+2+25+43+36+37
28+26 = (214)

- Intro
- Data types ✓
- String Handling
- Conditional Operator
- Loops
- Static Array
- Array List
- Functions
- Method overloading ✓
- Static vs Non static ✓
- Constructor ✓
- Encapsulation ✓
- Inheritance ✓
- Interface ✓
- Abstraction
- Exception Handling
- String Manipulation
- Finalize
- Wrapper Classes ✓

29-NOV ⇒ Data Types

Data types ⇒ there are two types of data types → primitive, non primitive

Primitive ⇒ defined by the programming Language
Non primitive ⇒ defined by the programmer, called as reference variable as well.

everything in Java is object.
References can not be duplicate in a class. Values can be

* primitive data types ⇒

① Boolean type -
Syntax ⇒ boolean = true/false
default value = false

② Numeric type -
Integral type ⇒ Integer :- byte, short, int, long

1) byte :- Syntax :- byte reference = number;
Size : 1byte = 8 bit
range = -128 to 127

2) Short :- Syntax :- short reference = number;
Size :- 2 byte = 16 bits
range :- -32768 to +32767

3) int :- Syntax :- int reference = number;
Size :- 4 byte = 32 bits
range :- -2147483648 to 2147483647

4) long :- Syntax :- long reference = number;
Size :- 8 bytes = 64 bits
range :- 2^{63} to $+2^{63-1}$

③ floating type : float, double

④ float :- Syntax :- float reference = number;
Size :- 4 byte = 32 bits
range :- after . can take up to 7 digits.

⑤ double :- Syntax :- double reference = number;
Size :- 8 bytes = 64 bits
range :- after . can take up to 16 digits

④ character type ⇒

Syntax :- char reference = '-';

note → char should be written in single quotes.

Char is a keyword.

Size :- 2 bytes = 16 bits

⑤ String ⇒ String is not a data type.

String is a class, which is already available in Java (Pre-defined class)

String S = "";

always written in double quotes only.

If default value of String is NULL.

30-NOV

String Concat - Incremental - Decremental

⇒ add two values or merge two values

ex.
```
String x = "Hello";
String y = "Selenium";
S.o.p(x+y);
```
O/P → HelloSelenium # execution always happen from left to right

when we are doing arithmetic operators in characters we want to follow the ASCII values.

ASCII table ⇒ a to z => 97-122
A to Z ⇒ 65-90
0 to 9 ⇒ 48-57

Increment and Decrement operators

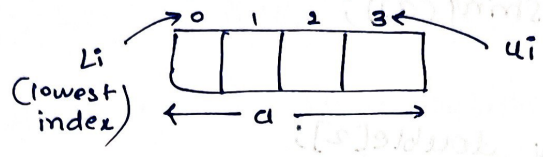
1. post increment :- i++
assigning the value, then increase the value by 1.
2. post decrement :- i--
assigning the value, then decrease the value by 1.
3. pre increment :- ++i
increase the value by 1, then assign the value
4. pre decrement : --i
decrease the value by 1, then assign the value.

(03-dec-21)-01

Static Array

Array – used to store multiple similar type of data/values.

declaration of Array → int a[] = new int[4];
↳ size/length of array



li = 0
len = 4
highest index(hi) = len - 1 = 3
len = hi + 1 = 4

→ It is continuous memory location.

→ arrays are non primitive data type.

```
a[0]=10;
a[1]=20;
a[2]=30;
a[3]=40;

SOP(a[0]); #10
SOP(a[3]); #40
SOP(a[4]); # ArrayIndexOutOfBoundsException
SOP(a[-1]); #
```

* In Java negative index are not allowed.

```
SOP(a[0]+a[2]); # 10+30=40
```

to print all the values of array : for loop:

i) for(int i=0 ; i<4 ; i++) {
SOP(a[i]); } # 10, 20, 30, 40

ii) for (int i=0; i<a.length ; i++) {
SOP(a[i]) }

limitations → ① Size is fixed/static : to overcome this problem we need to use dynamic array.

→ default value of array is zero.

② Stores similar types of data.

for each loop : enhanced loop:

```
for (int e : a){ SOP(e);}
```

```
SOP(a); # hashcode number/address
SOP (Arrays.toString(a));
```

2. double Array:

```
double d[] = new double[2];
        d[0] = 12.33;
        d[1] = 23.44;
for (double e : d){ SOP(e)}
```

3. char array:

```
char c[] = new char[3];
        c[0] = 'a';
        c[1] = 'b';
        c[2] = 'l';
        SOP((c[0]+c[1]);
```

String array:

```
String emp[] = new String[3];
        emp[0] = "Rohini";
        emp[1] = "Sai";
        emp[2] = "Hari";
for (String e (variable) : emp (array name)) {
        SOP(e); } # printing value from string array
    SOP(Arrays.toString(emp)); # ["Rohini", "Sai", "Hari"]
```

Object array: static

→ Object is super class of all class

```
Object student[] = new Object[5];
        Student[0] = "Rohini";
        Student[1] = 25;
        Student[2] = 35.55;
        Student[3] = 'F';
        Student[4] = true;
        for(Object e:Student){ SOP(e);}
```

default values:-

int = 0 , double = 0.0 , boolean = false, char = space, String = null

06/12/21 - holiday

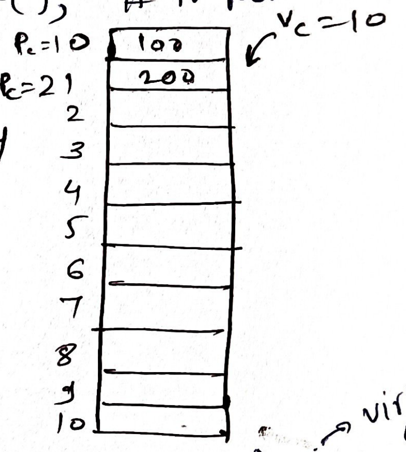
07/12/21 - TUE ARRAY - LIST

→ Arraylist is a part of collection. this is inbuilt class in Java. it is an dynamic array. continuous memory allocat

#object of arraylist :

```
ArrayList ar = new Arraylist();   # import
ar.add (100); # 0
ar.add (200); # 1
ar.add ("tom"); # 2
ar.add ('m'); # 3
```

physical capacity



li=0
hi=3
len/size = hi+1 = 4

By default it will give 10 segments (virtual). Vc = 10

Size will be calculated on the basis of physical capacity.

⊛ physical array में जो i/p दिया है वो size होगा array का

10segment full हो जाए तो आपको Vc = ? मिलेगा

* for object comparision, we are using .equals method.

09/12/21 ⇒

Methods :-

① method should be parallel to each other.
② method can not be created inside a method.
③ can not create a duplicate method - but I can overload.

① NO ip & NO return :- (void) → func^n can not return anything.

return type = void

```
public void testLoginPage() {
```

method name should be start with small letter.

```
    S.O.P("test method...");
}
```

② NO ip & some return :-

return type : int

```
public void int getNumber() {
    S.O.P("get Number method...");
    return 100;
```

return type

```
public String getTrainerName() {
    S.O.P("get trainer name");
    String name = "Naveen Automation"
    return name;
}
```

main() → starting point of execut^n.

* object can not hold static method

class obj = new class()

// one obj will be created, obj is reference variable, referring to this object.

// after creating the object, the copy of all non static methods will be given to this object.

(3) Some input & some return:-

```
public void add (int a, int b) {
        S.O.P("add member");
        int c = a+b;
        return c;
}
```

main method will never return anything

```
// Launch a browser - i/p param (String) - return: void
public void launchBrowser(String name) {
        S.O.P("browser name is :" + name);
    if (name.equals("chrome")) {
            S.O.P("Launch chrome");
    }
    else if (name.equals("firefox")) {
            S.O.P("Launch firefox");
    }
    else if (name.equals("safari")) {
            S.O.P("Launch safari");
    }
    else if (name.equals("IE")) {
            S.O.P("Launch IE");
    }
    else {
            S.O.P("Plz pass right browser");
    }
public static void main (String[] args) {
        Browser br = new Browser();
        br.launchBrowser("chrome");
    }
}
```

13 dec 21 ⇒ Static vs Non Static

→ How to call static methods and vars?

① direct calling :-

② calling by classname.

→ How to call non static methods & vars?

① by creating object

classname obj = new classname()

Static methods or static variables are not part of your object.. Seperate memory allocated to Static variable.

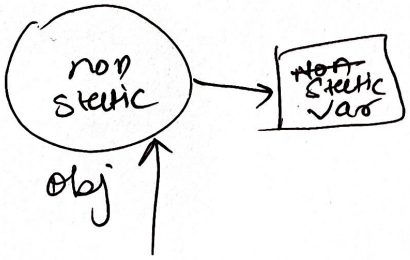
→ can I access static methods ~~& vars~~ by using object reference?

yes, but it is not good practise.

→ scope of global variable, will be available across all function, with some condn

[If it is static variable, can call directly or by using classname.

If it is non static variable, you have to create object]



→ initial or bydefault value store in static variable is 0.

→ first of all interpreter will give the memory to the static variable.

→ static method main() cannot access non static member variable.
Static static ही access होगा।
Static के अंदर only static allowed.

Constructor

⇒ duplicate constructors are not allowed.

⇒ constructor can be overloaded

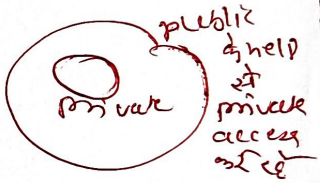
⇒ it is used to initialize class variables.

⇒ local variable of class variable can assign by using this keyword.

⇒ this keyword is used when we want to initialize class variables

⇒ Actual purpose of getter & setter is to access private variables.

extended version of encapsulation



* It means it is class entity used could define while creating object. if u want to define some class feature in the form of global variables / objects.

* constructor name = class name

* No Return type

* ① Default
  ② Parameterized

* when you create object then called constructor

16 dec 21 ⇒ encapsulation or data hiding

⇒ Hide data members → can be variable / method
define it as private.

⇒ Binding fields

⇒ declare some setter & getter method to access private variables

⇒ Some security point of view, some hacking point of view encapsulation is used.
No one from outside the class can access the private data members.

⇒ Right click → source → getter/setter

private data variables; so that these vars can not accessed directly from outside the class.

Adv ⇒ ① flexibility
② Reusability
③ to provide high level security
or to restrict data access. to improve security

⇒ we can access private with public getters

```
public class Employee {

    private String name;
    private int age;
    private double salary;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

within class can use variables

using this public layer can access variable

right click → generate getter setter → select variable → generate

```
public class EmpTest {
    public static void main(String[], args) {
        Employee e1 = new Employee();
        e1.setName("Tom");
        System.out.println(e1.getName());
    }
}
```

O/P ⇒ Tom

class customer { class cusetest {

} }

* You have private data members, giving access via public getter, setter.
* I have to create public layer to access my private member variable
* getter setter can be any method
  - ↑ get (return)
  - ↑ set the values
* Constructor help me to set the values

10 dec 21 ⇒ Method Overloading

⇒ duplicate methods are not allowed

```
public class MethodOverloading {
    public static void main (String[] args) {
        MethodOverloading obj = new MethodOverloading();
        # create object of class to call non static function
        obj.sum();
        obj.sum(10);
        obj.sum(10,20);
    }
    public void sum() {
        S.O.P("sum method.... zero param");
    }
    public void sum(int i) {
        S.O.P("sum method");
    }
}
```

Same method name with diff. ilp argu. or ilp parameter with diff. data type

method overloading ⇒ When the method name is same with diff. argument or ilp parameter within same class.

make sure they are having diff. datatypes

0 input param

1 ilp param

Same method name with same no. of argument & same data type are not allowed

// duplicate methods are not allowed.

methods are independent to each other

we can not create method inside a method.

```
# public void sum(int k) {
}
```

1 ilp param

gives you error because method is already there with one param

Same method name with same no. of arg. but diff datatype are allowed

```
public void sum(int k, int m) {
    S.O.P(k+m);
}
```

✓

* you can not create a method inside a method

x eg. public void sum() {

public void test() { } }

* duplicate methods → i.e. same method name with same number of arguments are not allowed. within same class. [same data type also not allowed]

* we can overload main().

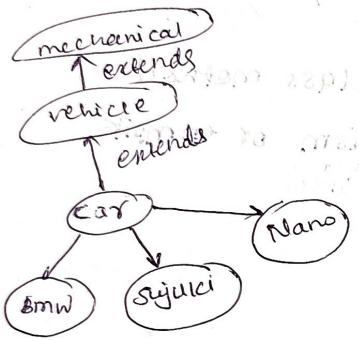
17 dec 21 = Inheritance

Inheritance → child can inherit some property from parent.

but parent cannot inherit some properties from child.

child only & only have one parent but parent can have multiple children.

If you want to make relationship betn two class then you have use "extends" keyword



Method overriding

when same methods is present in parent class as well as in child class with the same number of arguments, is called method Overriding.

→ preference will be given to overriden method.

car.java

~~OOPconcept~~

```
Public class car extends vehicle{
    public void start(){
        S.O.P("car start");}
Public void stop(){
        S.O.P("car stop");}
public void refuel(){
        S.O.P("refuel");}
```

B.mw.java

```
public class Bmw extends car { # has-a relationship
    pubic void start(){ # overriden method
        S.O.P("Bmw start");}
    public void satty(){
        S.O.P("satty");}
```

package2

TestCar.java

```
public class TestCar{
    public static void main (String[] args){
#static polymorphism --compiletime polymorphism
        Bmw b = new Bmw();
        b.start();
        b.stop();
        b.refuel();
        b.satty();
} }
```

dynamic polymorphism ⇒ Run time polymorphism

child class object can be refered by parent class reference variable also called Top casting

Car (parent) c1 = new BMW (child); — BMW को अपना object Car को देगा।

c1 can't override method v and parent class method (common method) will be called using dynamic polymorphism

If you ~~want to~~ can access ~~static~~ child class method only n only by using static polymorphism or compile time polymorphism. (independent methods in child class)

vehicle.class

```
public class Vehicle {
    public void engine(){
    }
```

Down casting ⇒ Not allowed

```
BMW b1 = new Car();
```
parent car object will not fit into child class object

if you want to use, you have to cast

```
BMW b1 = (BMW) new Car();
```
classcast Exception

At compile error Car cannot be cast

degrading parent class object to child class object then referent by child class object.

23/12/21

Abstraction ⇒

Interface can write saved logic.

// can not create object of abstract class

// but can create the object of constructor of abstractor class and it will be invoked when you create the object of child class.

Child → parent (object creatn) ⇒ Top cast

Parent → child (object creatn) ⇒ Down cast

multiple/diff system ⇒ interface (multiple classes)

One system ⇒ abstract

⇒ Abstract में abstract और कोई method हो ये जरूरी है।

Abstract class

① एक भी Abstract method class में हो तो वो Abstract class बन जायेगा।

② We can't create object of Abstract class.

③ if we extend the Abstract class then we need to implement all that methods of abstract class.

Interface

interface {

abstract methods
constants → static & final, public

}

Abstract methods को it is necessary to child class implement its body.

8th version ⇒ ① default concrete methods
② static methods

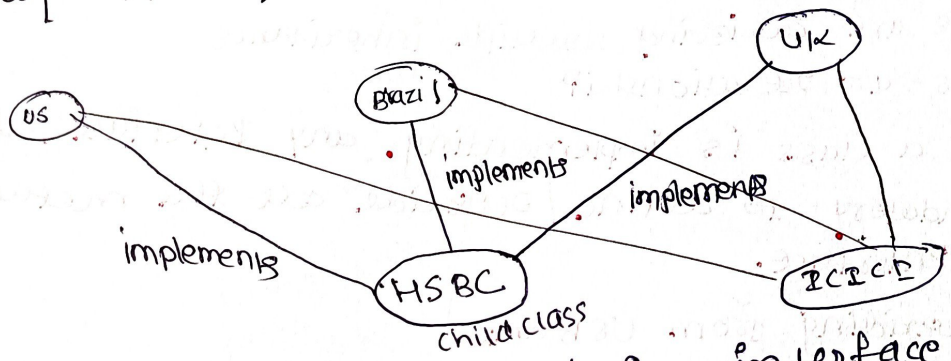
9th version में → allow ⇒ private methods

* child class में abstract की body देना must है।
* Interface का object create नहीं कर सकते
* blueprint is available in Interface but how to do that it is given in child class

20 dec 21 ⇒ Interface

Downcasting

⇒ Child class can implement multiple Inheritance. but in Inheritance child class can inherit only 1 only one parent class.



⇒ There is no main method in interface.

```
public Interface UsBank{
    public void credit();
    public void debit();
    public void transmoney();}
```

only method declaration
No method body - only method prototype.
In Interface, we can declare the variables, vars are by default static in nature.
vars value will not be change
no static method in Interface
No main method in Interface
we can not create object of Interface.
Interface is abstract in nature
cannot instantiate

public class HSBCBank implements US {
(Class) (Interface)

must give method body here of Interface here
" you have to implement all method here

HSBCBank.java

public class HSBCBank implements USBank, BrazilBank {

we are achieving multiple inheritance
Is-a- relationship
If a class is implementing any Interface, then its mandatory to define / override all the methods of Interface.

overriding from USBank

```
Public void credit(){
        S.O.P("hsbc---credit");
  }
public void debit() {
        S.O.P("hsbc---debit");
  }
public void transfermoney() {
        S.O.P("hsbc... transfermoney");
  }
```

seperate methods of HSBCBank class

```
public void educationloan() {
        S.O.P("hsbc... edu loan"); }
public void carloan() {
        S.O.P("hsbc... car loan"); }
```

brazil bank method : overriding from BrazilBank

```
Public void mutualfund() {
        S.O.P("hsbc--- mutual fund");
}
```

Exception Handling

Excepn ⇒ some unwanted conern are coming.

① Compile time Exception ⇒ checked exceptn

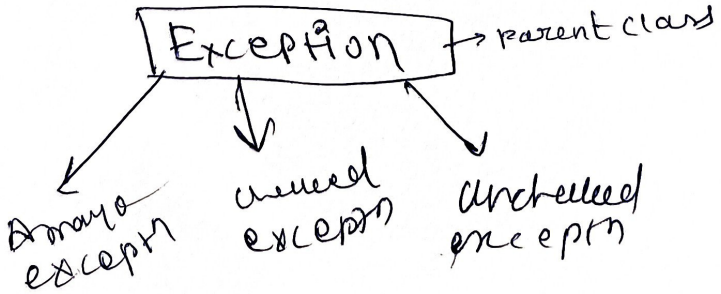
eg. down casting, Thread.sleep(5000);

② run time Exception ⇒ unchecked exception

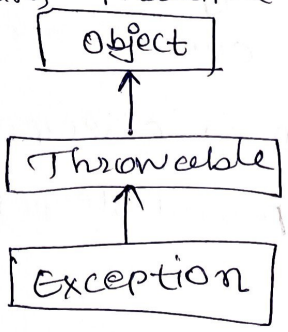
```
eg. int i=9/0;
    int K[] = new int[2];
    K[4]=20; # array index out of bound Exception
```

if excepn occur at any line then JVM will terminate program immediate. So handle exception by using try catch box.

```
try {
    int i= 9/0;
  }
catch (ArithmeticException e) {
    S.o.p("AIOB is coming...");
    e.printStackTrace();
}
```



this is not good practice to write single exception if it containing multiple exceptions



① try catch
② throws

Virtual linking with object of other classes.

with object we can't handle exception due to virtual linking

Throws → Bypass exception from one method to another

JVM calling main method
→ work after method name.

when to use ⇒

It is used to create our own exception.

Finally ⇒
- it is block
- it has to be executed finally
- exception coming or not finally block will execute
- try must be there to write finally

29/12/21 =>

finalize method => by default call by JVM.

before destroy object called finalize method.

System.gc(); => call finalize().

↳ check any method having null object if yes then called finalize();

System.gc() → is mandatory to write only then we can call finalize method.

→ clean up activity code write in finalize()

practical usecase :-

① database object
② DB connection
③ make the DB connection
④ Hit the result from DB
⑤ closing the DB connection
⑥

```
finally{
    Close DB connectn
}
finalize();
    launch browser :
    fill the form on login page
    driver = null
    finalize(){
        // clean upactivitiy
        // reinitialize the driver
        // driver.quit()
        // shut down the selenium server.
```

→ only final keyword use for local variable.

Hash map ⇒ → dynamic in nature
→ store value in key, value format

→ key always be class (wrapper)

30/12/21 ⇒

Wrapper classes ⇒ convert premetive data type into non premitive data type.

if you are getting ~~integer~~ string values & you want to perform some calculations & store as integer number then use wrapper class.

Sendkeys supports only string so wrapper classes required to convert it

funct^{n} ⇒ take msg & return order ID

Data conversion by using wrapper class ⇒

parseInt ⇒ parse into integer

* for each & every data type have wrapper class.

```
public class wrapperclass{
    public static void main (String[] args){
        String x = "120";
        S.O.P(x+20); # 12020
# data conversion : String to int
    int i = Integer.parseInt(x);
        S.O.P(x+20); # 140
# String to double conversion
    String y = "12.33";
    double j = Double.parseDouble(x)
# String to character
    String k = "A";
    Character m = Character.parseCharacter(k)
# String to boolean
    Boolean b = Boolean.parseBoolean(k);
```

int to string

```
int j = 20;
String k = String.valueOf(j); # "20"
s.o.p(k+j)
```

IoW ⇒

```
String u = "100A";
int i = Integer.parseInt(u); # numberformat exception
```

generate order uber app
bookcar, source dest

Builder pattern

⇒ this return as it is object को GIF GIF return कर रहे हैं।

⇒ every class return same class/current class object.

⇒ private & static methods can't be overridden

⇒ if same ~~hidi~~ static method is available in child class it called method hiding

method overriding ⇒

poly + morphism → dynamic (Runtime)

When you have the same method in parent class and in child class with → ① the same name
② same no. of parameters
③ same type of parameters
④ same return type

Compiler need to be decide on runtime so it is dynamic

every BMW is car

parent object = child object

Individual method of child is not there → reference type check → parent object

Top Casting ⇒ child class object can be refered by parent class reference variable

* Parent object can access child class properties due to top casting but can't access individual property of child class (there is reference check & security check)

down casting - parent class object can be referred by child class reference variable

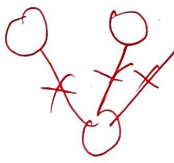
✗ child class object = parent class object # class cast Excepn

↳ Not allowed

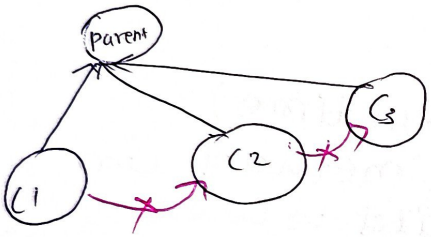
✗ every car is BMW

BMW b1 = (BMW) new car();

Reusability is achieve using Inheritance. eg. Animal ↑ Dog



Top casting ⇒ child class object ref



if decision taken in Run time it is Run time polymorphism

if decision taken in compile time it is compile time polymorphism

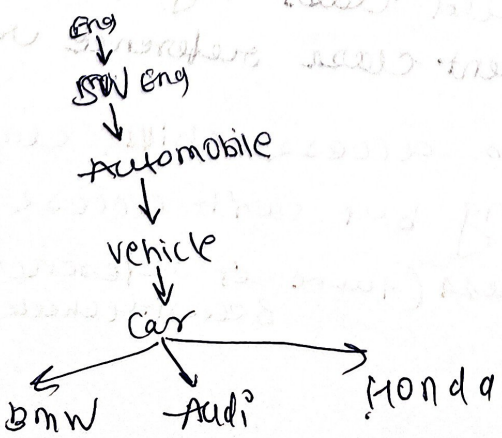
can overload constructor

Can create private constructor

can't create

If you want to restrict someone to make object of particular class then create private constructor in class then you cannot create object of that class.

only static methods are allowed.



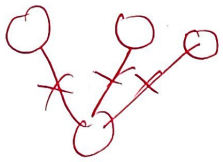
string ⇒ do all top casting with child class do all down casting with parent class

Cannot have 2 parent in Java

Can have 2 parent interfaces

Multiple Inheritance is not allowed

Multilevel is allowed.



Static variable can access with the help of ~~class~~ child class name as ~~the~~ well

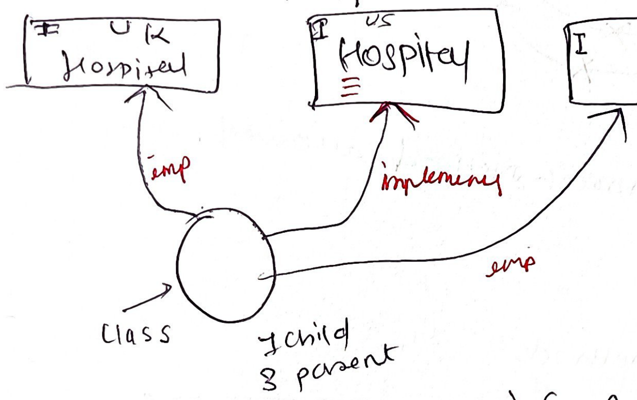
Interface

⇒ never implement the this, it just tell you that it should be ~~implemented~~ ~~elsewhere~~ should be in your class

⇒ it just define that properties should be in your class

Interface never have class body, it just help to design class

⇒ don't have body of interfaces



interface को use करने के लिए class to we must override / implement its method

Top casting on Interface

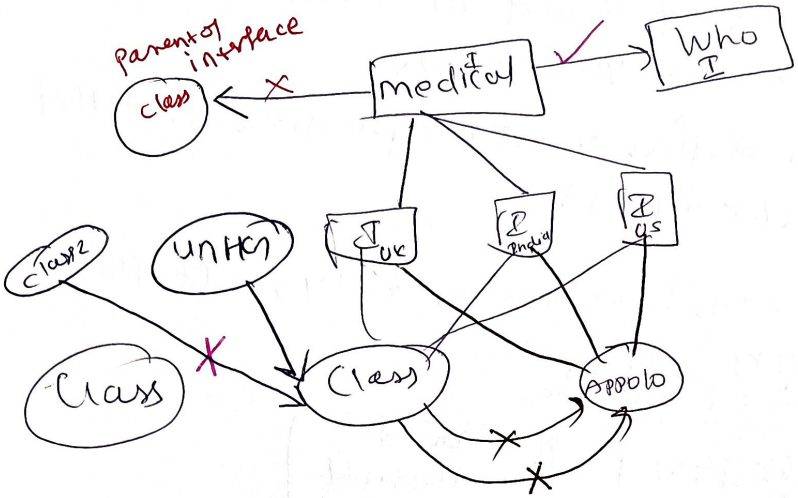
Interface = class

जिस Interface को टाइप-कास्ट किया उसी के object को Interface के methods access होंगे सिर्फ उसी के methods individual methods can't access

for local var

(imp) * if you typecast with India interface then only India methods you can access ~~...~~ except individual methods.

can't do down casting.
Can't create object of interface



⇒ can we have static abstract method allowed in interface?

No

No static abstract methods

> Abstract is always Non static method in Interface

Inerface{

No method body // only method declaration

only method prototype

only abstract method: No body

can not create the object of Interface

can not have static abstract method
non static

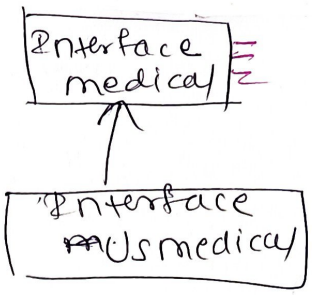
final + static

after JDK 1.8:

1. can have static method with method body:

static method can't be overridden

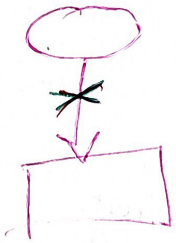
2. can have non static default method with body



~~class~~ public interface Usmedical extends medical

grant parent Interface में जो methods होगी वो भी इसी Implement करना होता है।

Super Interface must be interface



* Can override parent method in child interface

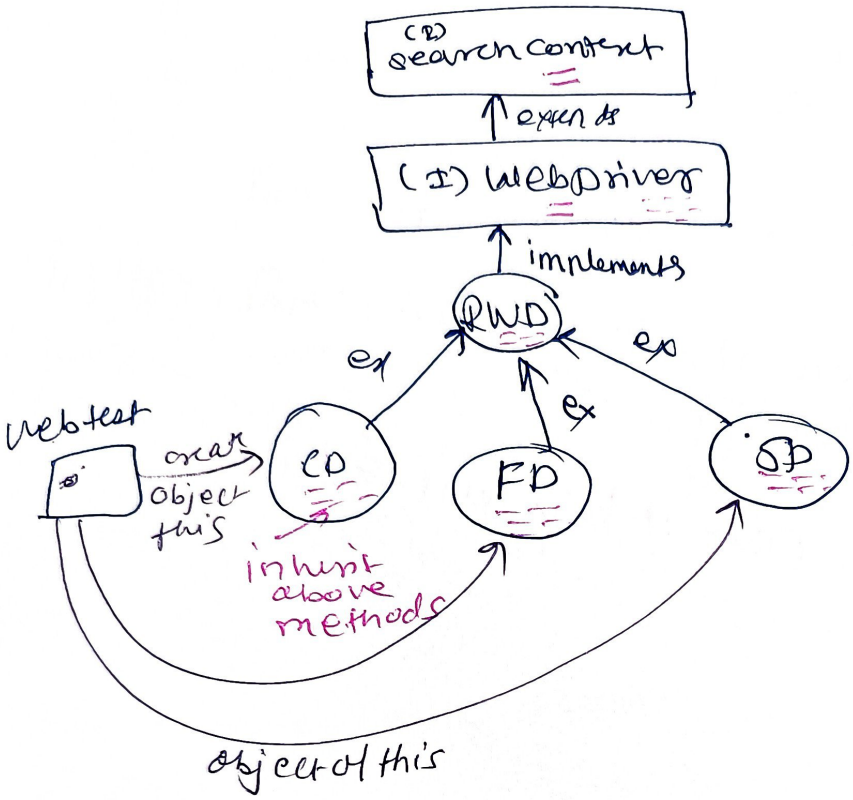
also
Final

* if a class declare with final it can not be parent.
* final method can't be overridden.

Final ⇒ ① used to provide constant variables eg. no. of days, months (for local var)
used to ② To prevent method overriding
③ To prevent Inheritance.
④ final is ~~im~~applicable for interface

ABS

Why Interface



RWD dr = new CD #top casting

parent class child class

parent Interface = child class #TC

Grandparent Interface = child class #TC

Abstraction

Public abstract class page {
0 - 100% abstraction
abstract method without body
Non abstract method with body
}

⇒ No mandatory abstract method should be in abstract class.

* 0% → No abstract method
* 100% → only abstract method (Interface) ↗ but after JDK 1.8
* partial abstract ⇒
* Can't create object of abstract class
* Can override non abstract methods
* Can't override static, final method
* ~~can't~~ Can create constructor of abstract class
  ↳ it will be invoked when you create object of child class.
  ⇒ invoke object of child class

⇒ interface constructor can't create

Execution of Constructor in Inheritance

Parent class ⟶ child class
default constructor of parent class will always be called.

⇒ can do top casting

Abstract class = child object

Before JdK 1.8, Interface was 100% abstract
but after JdK 1.8 it is partial interface.

→ super constructor will call first

with abstract classes can be achieve inheritance

Java IoW question

Que ⇒ What is method overloading?

→ When the method name is same with diff. ip arg. or ilp parameter with diff. data type within same class. is known as method overloading.