Java (Rikky Sir) High level language

Date - 9/12/22

C → 1971 → Denish Ricchy

Java → 1991 (invented) → 1995 (1st version developed / launched)

C → function , data ← int, char, float, string

Java → function , data , Class , object

C → Hero → main function

Java → class , main

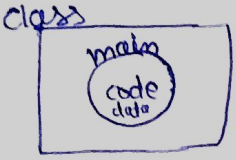
in Java one class is mandatory.

C++ is not pure object oriented because we can develop a program without writting (creating) class.

Java establise connection with oops concepts. in Java one class is mandatary that's why it is pure object oriented but not 100% object oriented.

Java identifieres rulles → back

class



primitive data type - int
char
float
string.

in C opreting system (compiler) → calls main function

in JAVA JVM calls → main function

Function

program -

1. declaration
2. input
3. logic code (formulla)
4. output

multiple line comment ⇒ /* ——
* ——
*/

```
class Add
{
public static void main (string[] args)
{
  int x, y, z;
    x = 10;
    y = 20;
    z = x+y;
  System.out.println ("The sum of x =" + x + "and y =" + y +
                      " is =" + z);
}
}
```

public - — Java notes —

static - — Java notes —

acc. to oops concept we have to creat object for calling any function. But main is called by JVM, and JVM is not a part of our class because of public keyword JVM (which is outsider) will be able to call/access main function. With using static keyword, we can run/access main function without creating an object.

//according to oops concept we have to creat object for calling each function creating object we can access main by keyword static.

2023/03/06 20:25

Now → for calling other function we have to creat object

? How to creat any object in Java.

Syntax class_name object_name = new class_name ( );

example Nidhi n1 = new Nidhi ( );

? How to call any function.

Syntax object_name . function_name ( );

example n1 . f1 ( );

Date - 13/12/22

```
public class Main
{
 public static void main (String [] args )
 {Main N = new Main ( );
    N . f2 ( );
 }
void f2 ( )
 {  int ipa , air , t ;
    Float si ;
    ipa = 1000 ;
    air = 2 ;
    t = 1 ;
    si = ( ipa * air * t ) /100 ;
    System.out. println ( "simple interest"= + si ) ;
 }
}
```

class → hottel owner

object → manager

function → working component

input - How to get value of variable from user.

functions are always part of class. (in Java)

function → System define
function → User define

Scanner is a inbuild class.

```
Scanner sc = new Scanner (System.in);
int n ;
n = sc. nextInt();
```

Here:⇒

- Scanner ⇒ class
- sc ⇒ object
- System.in ⇒ default keyword (for input device (keyboard)) // reference for method
  (System → class)

n = sc. nextInt();

- n → accepter activity (लेने वाला (LHS))
- sc. nextInt() → donar activity (देने वाला (Right hand side))
- sc → object (manager)
- nextInt() → method (working component)

---

if more than two words

inbuild class - first character of first word capital + first char. of second word capital + and so on capital

inbuild function - first character of first word small + first char. of second word capital + first character of Third word capital + and so on capital

inbuild functions

nextInt();	→	for scanning int
nextFloat();	→	float
nextChar();	→	char
next();	→	string without space
nextLine();	→	string with space

for including inbuild class (e.g. Scanner) in our program we use keyword import.

```
import java.util.Scanner;
```

14/12/2022

Local, static, instance varicibles

— explained in notes

* Student information function in pc/Javafolder

15/12/2022

1] selective statment

Desision control statment

if else, if else ladder, if else ladder

* programs in Javafolder & in notes

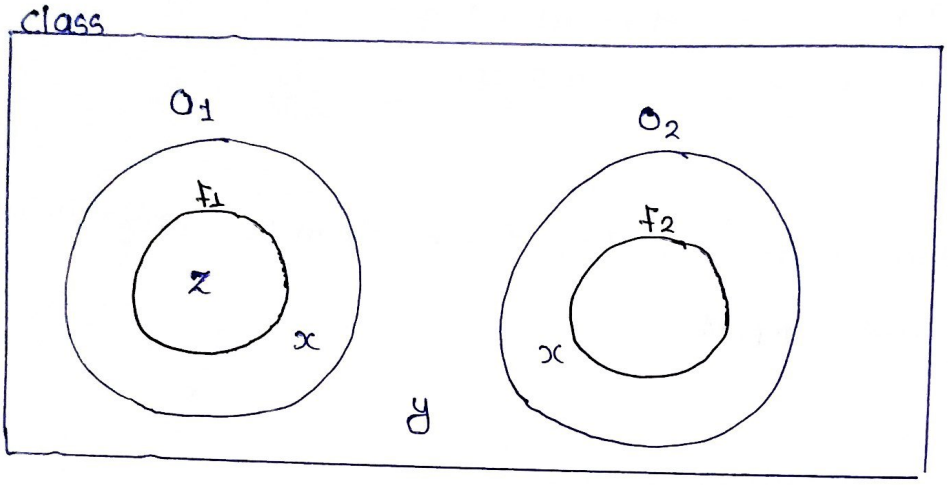
2] bytecode → explained in notes

Static variable

14/12/2022

instance variable | object variable

local variable



static variable - conected with object, only one copy and all objects can access it.

instance variable - connected with object, copies of the variable will be equal to the number of objects.

object से connected है, मतलब class के अंदर जितने भी functions है वो उस variables को access कर पायेंगे।

Example → — java notes —

18/12/2022

if else ladder

Iterative control statment

explained in notes Java

* HR points - in notes

Date - 20/12/2022

```
reverse = reverse * 10 + rem ;
```

H.W (Math logic) ⇒ ## why we multiply here 10.

in 987 the position of 9 → 100 place
8 → 10 place
7 → unit place

when we reverse this number, the positions of 9 & 7 will interchange. for negotiating/handling the gap of 10 place, 100 place, 1000 place, we have multiplied with 10.

---

for generating remender why only we use 10.

in math total available digits ⇒ 0 - 9 ⇒ 10

all available digits that can be remainder, this digits (all) (0 - 9) can be provided only by (%10)

if we use 9 than maximum remainder posible → 8
if we use 8 than maximum remainder possible → 7

★ 10 is the only on that can give maximum posible remainde → 9

10 is the only number that can provide maximum remainder as a digit.

---

if you are using any variable's value in loop for performing any task and ~~in~~ decreasing it. so firstly assign that value in another value when you need that previous variable again. and then use that another variable for decresing value

```
num1 = 10 ;
num2 = num1 ;
while() { num2--;}
```

you can again use (num1)

😊 कुरबानी हमेशा num2 की देनी है।

num2 = value; value में क्या कोई ऐसा number हो सकता है, जो (num2/10) से होता होकर zero होगा ही नहीं। in a while loop

where

```
while (num2 != 0)
{
    num2/10;
}
```

is there any number that will not become 0 or become less then 0. after this loop & opration.

Ans ⇒ this is not possible at all.

all the numbers will become 0 at the end and that's why both conditions (num2 != 0) & (num2 > 0) will give us our required output

---

Date - 21/12/2022

1. Armstrong number cheak for 3 digit number. (digit ^ number of digits)
2. Reverse any number,
3. count number of digits in any number.
4. pelendrom number or not.
5. Sum of digits of a number.

Armstrong number for any digit ⇒

for calculating power we will use pow() function.

(pow( )) power function is a buildin function of class Math.

for using this in all type of cumpiler we import

import java.lang.math;

if function belongs from the same class (means उसी class में define हुआ है) then it is not nessassary to write class name with . operator for calling that function but we can write. Text.add(); ✓ add();

here our power function (pow()) is define in Math class. and we are using it in our class so it is nessassary to write class name with . operator

```
Math.pow(    )
```

in new compiler it will run very well

if error is generated - lossy conversion, then we use typcasting

#typcasting ?

```
(int) Math.pow(   ,   );
```

we will pass two variable in parameters ( ).

```
(int) Math.pow(base, exponent)
```

base - जिसकी power निकालनी है e.g. a
exponent - कितनी power है e.g x > a^x

---

static function don't use object and . operator. if this function belongs from same class then we have no need nec to write even class name. if belongs from different class then you have to use class name..

---

in HR round if you have given program to write then use short forms - SOP
- psvm

and then simply say its meaning or full form

main target should be on code (logic)

Date 22/12/2022

Constructor - constructor is a special member function
(special → because class name = Constr. name; member → part of class)

User define static function - we can call it without creating object.
but we must have to call it if we want to use it.

> Constructor - you just have to creact object.
> there is no need of function calling.

जैसे ही object create होता है function calling implicitly होती है मतलब internal way में call हो जाएगा।

explicite way is use in normal function, by using . operator.

Constructor Rules - [1] constructor name is same as classname
[2] constructor have no return type
[3] No need of function calling.

if you write ~~written~~ return type you will get error.

Why we can not write written type ?

---

in real life → Constructor (~~function with constructor keyword~~) (Employ) is frien of (manager) object both come together there is no need of function call.

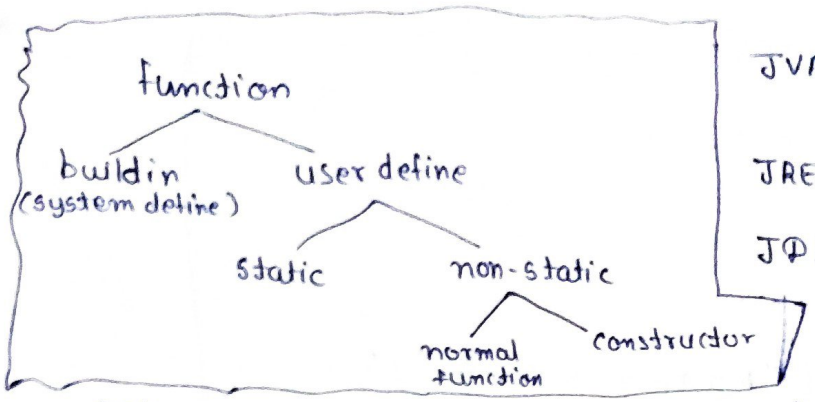
---

class - passive component
object - active component (gives order)

class
↓
object
↓
functions
&
data

Why we cannot make constructor as static.

Ans → Because of its defination.



JVM - is a component of JRE. runs byte code.

JRE - Java runtime environment

JDK -

★ If you want any application which have direct home page there ~~are~~ we can use constructor.

Constructor ♡ object because contructor name == class name

&& class ♡ objects.

Constructor → क्या हम constructor को static बना सकते हैं।

Answer - No, because for constructor object creation is must. without object constructor will never run.

कुछ function हम चाहते है कि अपने आप ही call हो जाए, तो हम constructor use करते है।

Constructor == ~~function with construct keyword~~ function which has same name as class name.

Constructor ऐसे function है जो अपने आप call होते है उन्हे user call नही करता, user call नही करता मतलब हमारे call करने में जो हमे code की lines लिखनी पडती वो नही लिखना पडेगा।

Constructor - एक से ज्यादा बना सकते है But फिर call कैसे होगा ?

```
Class Info
{
   Info ()
   {
     System.out.println ("My name is Nidhi");
   }

   Info ()
   { Sop ("village name is ABC");
   }

   Info ()
   { Sop ("I am in 2nd year");
   }

  public static void main (String[] args)
  {
     ~~Construct C1 = new Construct ();~~
     Info i1 = new Info ();
  }
```

So now the confusion is, which function (constructor) will run.

function prototype ⇨

returntype functionname (parameter / argument)

{

code ;

}

~~class~~ Constructor overloading

Constructor means the functions which have same name as class name.

We have created more fn constructor with different parameters. then this is called constructor overloading.

```
class Info
{
   Info ()
   { sop ("My name is Nidhi");
   }
  Info ( int x )
  { sop ("village name is ABC"); }
  Info ( int x , int y )
  { sop ("I am in 2nd year"); }
public static void main (string[] args)
{  int a=10 , b=20;
   Info i1 = new Info (a);
   Info i2 = new Info (a, b);
}
```

Constructor – Constructor is user define function as well as constructor is a system define function.

क्या हम शेड word [Constructor] को as a class भी use कर सकते हैं। ⇒ Answer – yes.

Parameterised constructor – parameters में value कैसे देंगे।

```
class Sourabh
{
    Sourabh ( int x )
    {  Sop ( "parameterised constructor called "+x );
    }

  public static vm ( String [] args )
  {
     Sourabh s1 = new Sourabh (10);
  }
```

or

```
psvm ( s[] a)
{ int a = 10;
  Sourabh s1 = new Sourabh (a);
}
```

or

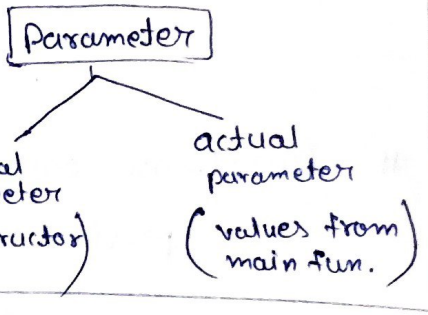
```
psvm ( s[] a )
{ int a ;
  a = sc. nextInt();
  Sourabh s1 = new Sourabh(a);
}
```

Benifits of parameterised constructor -

अगर हमने एक भी constructor बना दिया तो default constructor नहीं run होगा।

और default constructor आया ही नहीं तो object variables को value (default value) मिलेगी ही नहीं।

~~अगर default constructor~~



जब भी किसी program का object बनेगा तो default constructor implicit (अस्पष्ट, दिखाई नी देगा, means internally) way मे call हो जाएगा। कब call होगा, जब object बनेगा।

मतलब user ने constructor नहीं भी बनाया है, तो भी background में JVM हमेशा constructor को call करता है।

a default constructor always runs in background.

work of default constructor → is to provide values to object variables (instance, static variable)

int = 0
float = 0.0
string = Null
char = ?

if we do not give value to intance and static variable then the values of these variable will be that values, which were provided by default Constructor.

* default constructor does not provide value to local variables

task 1 - Just declare/create these variables and do not initialize them or do not give values to this.

and print their values by SOP.

① local
② instance
③ static

Maximum cases में variable को value function के द्वारा provide करते हैं।

और जब variable को value function से provide होती है तो हमे Data hidding का feature मिलता है।

example - ① when we give value through nextInt(); .... etc.

② default constructor provide value to object variables. constructor = special member function

---

task 2 →

instance → variable को direct value दे देना

static → direct नहीं देंगे, तो function के through दिलाना है।

→ दे पा रहे हो तो - ϕ ?
नहीं दे पा रहे हो तो - ϕ ?

when we create static variable and initialize it, here we have not provide the value by function

21/01/23

Java Primitive data types.

we can use class name for naming a function.

bit - byte - kilobyte - megabyte - gegabyte - terabyte (memory measurments)

in mobile
internal memory = 8 GB
external = 32, 64 GB..

- When we are creating any variable we have to choose data type so that we can get more memory efficiency.

1. at the time of function calling if we have call the function by sending a constant value (i.e. 10) in parameters.
   → By default compiler will consider it as int type

2. function calling के time आप parameter में जिस type का data भेजेंगे तो उसी type के data type के parameter वाले function को execute करेंगा और अगर बराबर same data type available नहीं है तो वो उससे high range के data type को parameter वाले function को execute करेंगा।

3. at the time of function calling if we have passed a variable in parameters. and previously we have declared its data type and initialize it with a constant value (i.e. 10) then this value will be consider as that data type that we declared.

अगर byte दिया है, तब और parameter में int, long, short है तो high range के data type पर जाना है पर don't be greedy. so the choice will be int.

24 jan 2023

Typecasting -

```
byte x = 10;
int y;
y = x;
```

no error

```
int x = 10;
int y;
y = x;
```

→ error :

error : incompatible types : possible lossy conversion.

for solving this problem

```
int x = 10;
int y;
y = (byte) x;
```

← compiler will read this line from right to left.

So पहले compiler को variable x मिलेगा, फिर (byte). So compiler समझ जाएगा की x की value को assign करते time या transfer करते time जब उसे (byte) लिखा हुआ मिला तब तो x की जो vaue int में थी उसे byte में convert कर देगा।

value अगर eligible है convert होने के लिए तो।

Typecasting -

for removing the error lossy convertion we use typecasting.

for this → we write that data type before the variable or expration with round brakets.

(a+b)

```
int x=10, y=20;
byte z;
z = (x+y);
```

error

```
int x=10, y=20;
byte z;
z = (byte)(x+y);
sop(z);
```

no error

(?) try to do this with other data types.

in C pointer, hacking को बढ़ावा देता है।
Java में ये pointer, C के original pointer से बहुत diff. है

This pointer - it is a pointer which points something special.

It is used at different concepts in java.

- It's first use -

```
class Main
{
  int x = 10;
  void f1()
  {
    int x = 20;
    Sop("local x = " + x);
    Sop("instance x = " + this.x);
  }
  public static void main(String[] args)
  {
    Main m = new Main();
    m.f1();
  }
```

```
class ABC
{
  int x, y, z;
  void addition(int a, int b)
  {
    x = a;
    y = b;
    z = a+b;
    Sop(z);
  }
  psvm(String[] args)
  {
    Scanner sc = new Scanner(System.in);
    int p = sc.nextInt;
    int q = sc.nextInt;
    ABC a1 = new ABC();
    a1.addition(p, q);
  }
```

main function से हम actual parameter (p, q) की help से variables की value formal parameter (int a, int b) को भेजते है addition function के अंदर। फिर इस addition function के through हम instance variable को value देंगे।

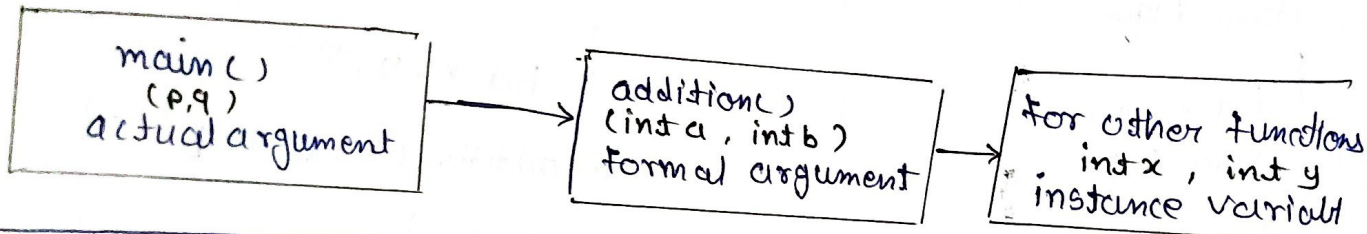
means overall value main function से जा रही, और main से value देना is a good thing. at the time of function calling.

- main ने addition को value दी और वो चाहता है कि बाकी जितने भी remaining function अगर बनाए गए है तो उन सब को भी value मिल पाए. इसलिए addition से value instance variable को दिला दी

In 2nd ~~func.~~ Program.

actual arguments (P , 9)
formal arguments (int a , int b)
instance variables int x , y , z .

we send variable values →



★ if we chang the name of formal-argument from (int a , int b) to ⟹ (int x , int y)

means formal argument and instance variable के name same कर दें।

तब क्या error आएगी ?

Answer → नहीं, Qki हम जो local variable बनाते है उसका Scop सिर्फ उस function तक ही होता है।

इसलिए हम अलग अलग functions के अंदर variables के name same रख सकते है।

और Compiler को जब local और instanse variable में से किसी एक variable को choose करना हो तो तो local को choose करेगा।

अब जो हमने addition function के अंदर x और y लिखा है, उन्हे वो (compiler) as a local variable समझेगा। और

x की value x में assign हो जाएगी।
और y की value y में assign हो जाएगी।

but अगर हम दूसरा कोई function बनाएंगे तो उसमे instance variable की value default value 0 आएगी । मतलब हम instance variable को हर जगह access नहीं कर पाएंगे ।

```
class ABC
{
  int x, y, z;

  void addition (int x, y)
  {
     x = x;
     y = y;
     z = x + y;
    S.O.P. (z);
  }

  void display ( )
  {
    S.O.p (x);
    S.O.p (y);
  }

  p.s.v.m. (~[]~)
  {
    Scanner sc = new Scan( ~)
    int p = sc. nextInt();
    int q = sc. nextInt();
   ABC a1 = new ABC();
    a1. addition();
    a1. display ();
  }
```

```
class ABC
{ ——— // ———

  void add- - - -
  {
    this. x = x;
    this. y = y;
    —//—
  }
  void display ( )
  { —//—

  } — // —

  p.s.v.m (~[]~)
  —— // ——
  {
    —//—
    —//—
    —//—
    —//—
  }
```

यहाँ हमने this keyword का use . लगाकर किया है । जिससे compiler को पता लग जाएगा की left side वाला x instance variable है ।

तो फिर instance variable मे value चली जाएगी और other function भी उसे use कर सकेंगे जैसे display() में use किया है ।

p = 10
q = 20

output ⇒ z = value

0

0

x और y की value आयी पर वो local थी तो उसने z को value तो दे दी x + y add करके । पर वो local थी ।

x, y instance मे default value ही है । (0, 0)

output 30 // value of z

10 // value of x

20 // value of y

26/01/2023

main Function can be overloaded.