COP : Concepts of programming - Java

day-1
14 ~~19th~~ March

Java :
Editor : Notepad / Eclipse
compiler : JDK / JRE

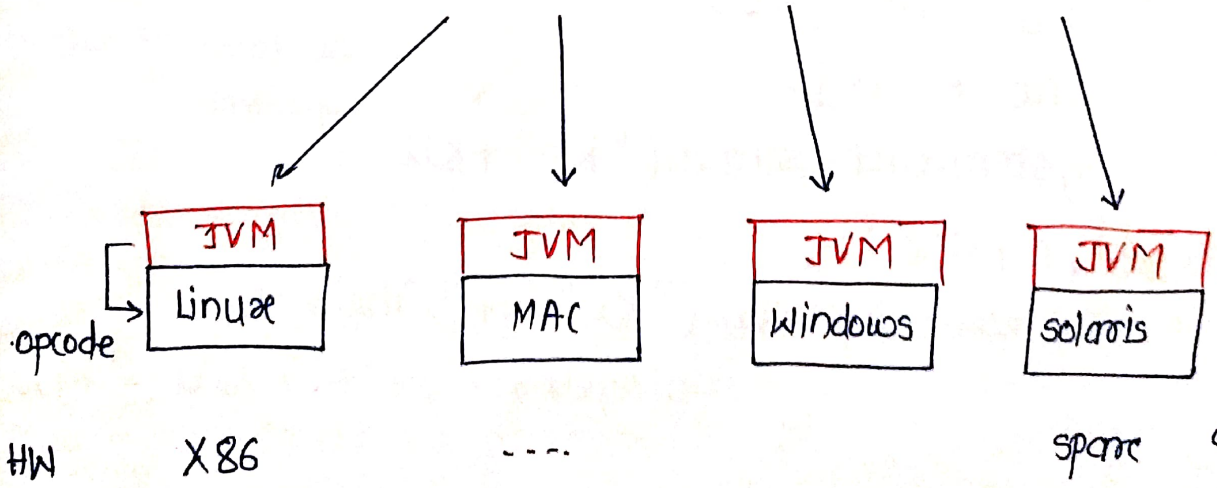
SDK : software development kit
Java : JDK : Java development kit.

Java gave biggest problem's solution i.e. Porting.
compiler used in 'c' is hardware specific i.e. machine dependant
It gives binary for specifically for that machine.
Java gave that portability to run on any machine.

Java program compilation :-

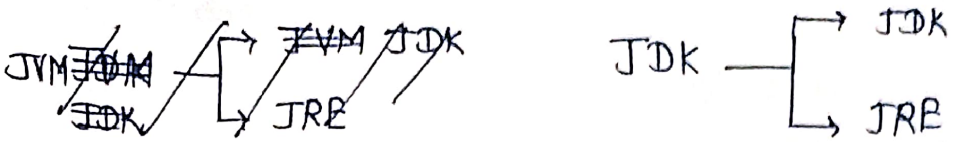
main.java
↓
[compiler]
main.class file

It converts
in bytecode for an underline system
But this ~~sys~~ system is not actual hardware on which program will run. Java compile it for a specific hardware which is made by java itself



JVM is virtual M/c, it will send instructi to actual m/c
It will convert acc to underlying system.

compilation command → Javac.exe
run command → java.exe



Basic Java program: file name :- Main.java

```
class Main
{
    public static void main (string [] args)
    {
        System.out.println ("Hello Java");
    }
}
```

class name & file name should same & it should start with capital letter

↑ In adds new line character.

To compile : → cmd → javac Main.java

To run : → cmd → java Main

Adding integers :

```
class Main
{
    public static void main (String [] args)
    {
        int i=10, j=20;
        int k=i+j;
        system.out.println("k=" +k);
    }
}
```

no need of format specifiers

if want to add in print :-

```
system.out.println("k="+k+". This is the
                answer for i="+i+"and j="+j);
```

structure in C is similar to the class in java,

C - structure

```
struct _student
{   name
    age
    ---
    ---
};
typedef struct_student student;
get_user_input( student *pte);
print_student (student pte);

int main()
{   int i;
    student S1;
    student *S2 =(student*) malloc
                (sizeof (student));
    :
    free(S2);
}
```

java - class

```
class student
{ name;
  age
  ---
  ---
  get_user_input (student*pte);
  print(student pte);
}
main()
{ int i;
  student S1;
  student S2 = new student();
}
```

- In C, we have to write Function main.
- Need to free memory
- syntax is heavy

- In java, we can write f^n in class.
- GC is there, no need to free
- syntax is easy.

15th March

structure

```
struct_circle
{
    int centre_x ;
    int center_y;
    int radius ;
}
typedef steuct_circle circle;
```

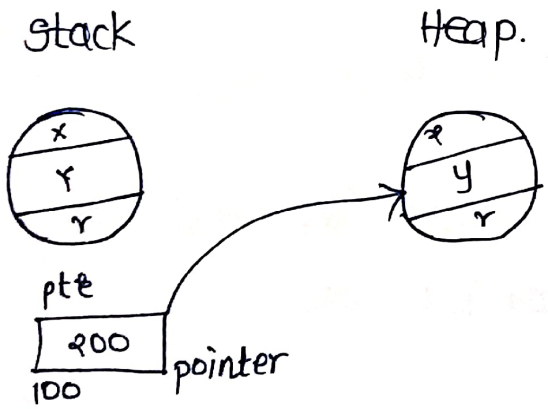
class

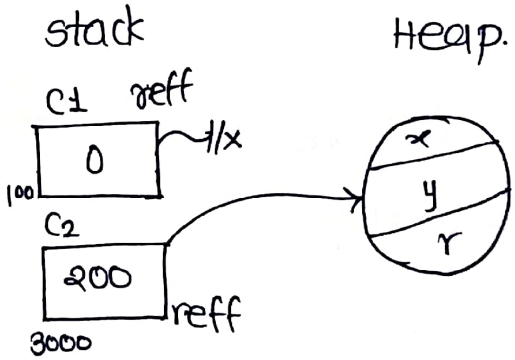
```
class Circle ← entity
{
    int centerX ← Hungerian notation
    int centerY ;   or camel case.
    int radius;
}
// All these small variables / small entities are called as attributes.
```

classes are blueprint of any real life entity (Noun)
classes are not instance.

```
int main ()
{
    circle c1 ;
    circle *pte = (circle*) malloc(sizeof (circle));
                ↑ variable
```

```
void main (string[] args)
{
    circle c1;
    // circle pte ;   { no need of *pte Java consider it as pointer/reff }
    Circle C2 = new Circle ;
                ↑ object
```






```
    free (pte);
}
```

- In java, all variables are by default zero. No Garbage value.
- In java, no need to free memory, default garbage collector is there.

In C, functions are written outside of the struct and can be called from main.

In java, f^n/behaviours/methods are written inside class and can be called from main without passing argument. coz in java behavious know who called it.

e.g.

```
void main (string [] args)
{
    Circle c1;
    Circle c2 = new Circle();
    Circle c3 = new Circle();

    c2.print_area();
    c3.print_circle();
}
```

} call to the behaviours/methods

These f^n are written inside class coz in main writting f^n is not allowed.

class → developer

main → user/client

c_1, c_2, c_3 are instance of class circle. class is just a line plan and c_1 c_2 c_3 are buildings.

~~constructors~~

```
class Circle
{
    int centerX;
    int centerY;
    int radius;

    void init()
    {
        centerX = 0;
        centerY = 0;
        radius = 1;

    void print()
    {
        system.out.println("Center: ("+centerX+","+centerY+") Radius: "+radius);
    }
}
```

Function written inside class

// Center: (0,0) Radius: 1

```
class Main
{
    public class void main (string[] args)
    {
        circle c1 = new Circle();
        circle c2 = new Circle();

        c1.init();
        c2.init();
        c2.print();
    }
}
```

Now, if, c1.init(2,3,4) ⟶ no argument found.

∴ in class,

```
void init(int x, int y, int r)
{
    ~~int~~ centerX = x;
    centerY = y;
    radius = r;
}
```

← This can be add as another f^n in our class

Function overloading.

- Two f^n having same name but differ in arguments.

f^n overloading:

We can make two or more f^n having same name but there should be diff. of arguments.

constructors:

f^n having same name as of class, are called as constructors. They are called when object is created. They dont have any return value.

e.g. instead of void init ⇒ Circle. [classname]

- Constructor is call back f^n
  - We don't need to call it in main, while creating object, it is called by java itself. without constructor, java object cannot be created.
  - constructor is used to create object, but they are not created for initialisation, we can do it as well, but not only initialis is job of constructor.
  - constructor is always called by compiler while creating an object
  - constructors can also be ~~to~~ overloaded ~~to~~ by changing the arguments.
  - If we didn't write any constructor, java automatically add a default constructors. It will not change in attributes.

Day-3 - 16th March

- Circle-point
- tostring
- User-developer
- Array
- Eclipse.

Eclipse :- Project explorer

problem console.

File → New → java project → Give your name for project

one project is created,

click on src (right) → new → class → main class.

If there present any type of constructor, then compiler will not give default constructor.

User input

To take input from user (scanf) in java,
in main, take variables first,

e.g.

```
string n = " ";
int age= 0;
char gender='M'
```

Top :- import java.util.Scanner;
like #include.

```
Scanner sc = new Scanner(System.In);
```

for keyboard i/p.

↳ we are telling scanner to scan this

age=sc.nextInt(); ← whichever int is i/p from user → get that

gender = sc.next. ← for string → but it give full string
charAt[0]; ∴ charat[0]

name = sc.next;

All the information regarding java library functions is available in java doc 8 [google]

to string :

To print something, we do. →

```
int g = 10;
system.out.println(g);
```

⇒ 10

But this will not go with class →

```
student s1 = new student(n,a,g)
system.out.println(s1).
```

↳ @ 55F96302 Hashcode.

To get proper o/p, we need to write a fn in class to print proper o/p.

↳ To use this feature ⟹ to string.

```
public String toString () ← in class
{
    return "Hello";        // It should return a string to print
}                          // whichever string is returned will get printed
```

```
System.out.println(s1); ← in main
```

↳ internally → s1.toString

// System.out.println require i/p in form of string

To print string → give string in toString & return str.

```
public static to string () {
    String str = name+ ", "+age+ ", "+gender; ← same as we used in print or display fn.
```

// To string doesn't print value, it converts class info into string and give it to sysout to print.

circle-point // Private public.

In class, if member variables are made private, they can't be accessed from outside of class. In main, we can't change attribute of class.

Now,

To change in the class variables. setters & getters are used

```
void setCenterY (int y)        int getCenterY ()
{                              {
    CenterY = y;                   return centerY;
}                              }
```

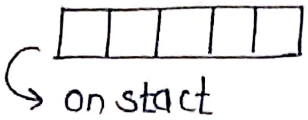
member variables must be private & fn must be public
↑ including constructors.

day- 4 - 17th March

- Array
- this
- datatypes

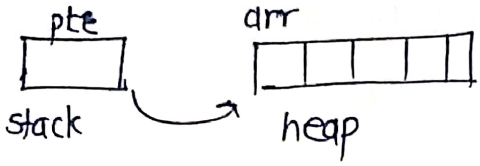
C-Array

int arr [5]



To create array on heap:

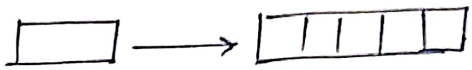
int *arr =(int *) malloc(sizeof (int)*5);



java Array

In java, Array is also an object & object is always created on heap.

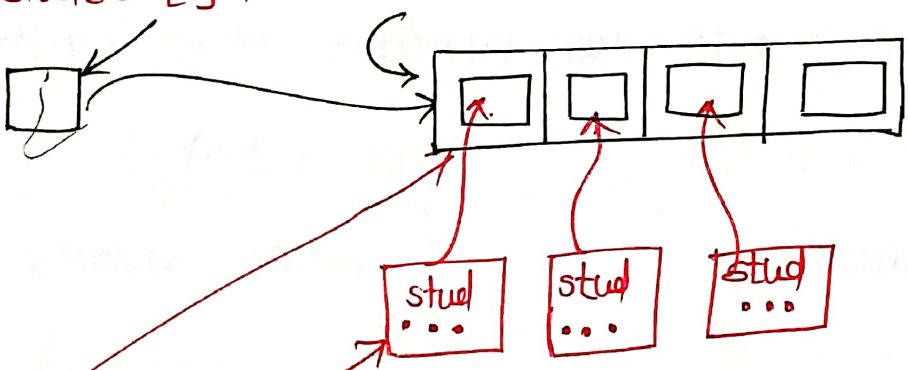
int [ ]arr = new int [5]



[ ] is used to tell that arr is not variable it is array.

```
public class Main {
    public static void main(String [] args)
    {
        int [] arr = new int [5];
        Scanner sc = new Scanner (System.in);
        System.out.println(" Enter 5 numbers: ");
        for( int i=0; i< arr.length ; i++)
        {
            arr[i] = sc.nextInt();
        }
        System.out.println(" arr[3] = "+arr[3]);
    }
}
```

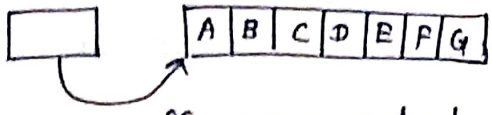
student [ ]arr = new student [4]



arr [0] = new student ("Ram", 18 ,45);

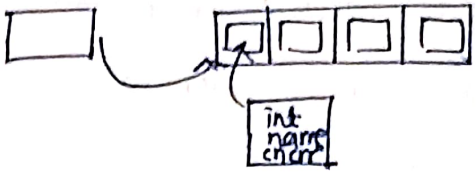
If we make array of primitive data type, on that address we can directly put value.

char [ ] arr = new char [7] ⟹



If we make array of object, onthat address, reff are created in array and on each reff we need to create an object.

student [ ] arr = newstudent [4] ⟹



this pointer :

This is the refference which points to current object (which called the fn)

attribute

```
public void setName(string name)
{
    name = name
```

but compiler will consider it as local [give preff to local.

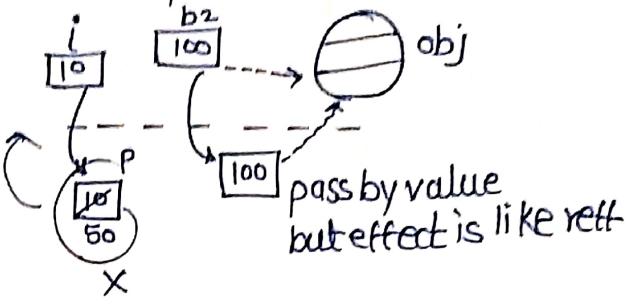
∴ this name = name

name of that object which called this fn.

- This is local variable
- This is a reff which points to current object.
- This is available in each n every fn with some exceptions

Day - 5 - 19th March

- Java supports pass by value only <u>but</u> it gives effect as per pass by reff.



Command line arguments

i/p given to the program when the program is running at command line → these are called as command line arguments.

90%–99% → command line args
10% - 1% → scanner class.

—x——x——x——x——x——x——x——x.

Whatever i/p we give at cmd, the program take it as a string.

public static void main (string[] args)

cmd line args as main is called by os.

args → array

In command line, datatype is only string

To convert string/ ~~flo~~ to int / float / char etc.

int num1 = Integer.parseInt(args[0]);
class fn string

Float ~~num1~~ f = Float.parseFloat("34.30");

boolean b = Boolean.parseBoolean("true");

int to string → tostring.

String array :-

```
string [ ] array = new String [ ] { "AA", "BB", "CC", "DD", "EE" }
  for( int i=0 ; i < array.length ; i++)
  {   sysout { array [i] } ;
  }
```

cmd program of addition.

```
public class Main {
    public static void main( string [] args)
    {
        // args [0]
        // args [1]
```

These are two string but we want addition of int.

∴ convert string to int

↳ int num1 = Integer.parseInt(args [0]);

to Float → Float f = Float.parseFloat ("34.56") ← string as hardcode.

vice-versa ⇒ to string.

Final keyword ——{ like const in c/c++

final int i = 0; ← we cant change now int i=0
we can't overwrite on value of i.

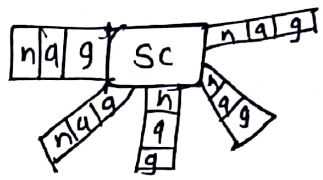
e.g.

final float pi = 3.14f ← now we can't change it.

Static variables & methods.

```
Public class student
{
    static int sc ;
    String name ;
    int age ;
    char gender;
```

← student count is static variable it is per object variable & one for class



If S_1 make change in 'sc' it will reflect in S_2 also

static will not make copy of sc in each object. It maintain it in every object

static function :

Non-static f^n called by object
static f^n called by class → if any variable in class is static
as well as object

- static variable is maintained per class not per object.

static variable can access in static as well as non static f^n
Non static variable is accessed only in non-static variable.

	static fn (class level)	Non-static fn (object level)
static variable (class)	✓	✓
Non-static variable (object)	✗	✓