print 20 stars condition

```
class Hello{
    public static void main(String args[]){
        //print 20 stars
        int i=0;
        for(i=0;i<5;i++){
            System.out.println("*");
        }
    }
}
```

Dry run?

intilization

increment

```
*
*
*
*
*
```

5<5

print 20 stars condition

```
class Hello{
    public static void main(String args[]){
        //print 20 stars
        int i=0;
        for(i=0;i<5;i++){
            System.out.println("*");
        }
    }
}
```

```
*****
*****
*****
*****
*****
```

Nested looping:

```
for(.....){
    for(.....){

    }
}
```

Nested looping:

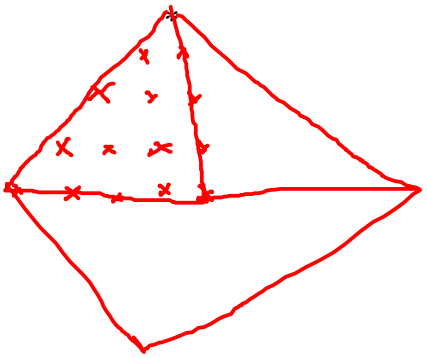
```
for(int i=0;i<5;i++){
    for(int j=0;j<5;j++){
        System.out.print("*")
    }
    System.out.println("");
}
```

i 0 < 5 1<5

*****

j 5<5 j=0,1,2,3,4,5

```
// Nested looping:
    //technally there is not limit
    //GPP: nested loop bad in peformance
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            System.out.print("*");
        }
        System.out.println("");
    }
```



```
for (int i = 0; i < 5; i++) {
    for (int j = 0; j <=i; j++) {
        System.out.print("*");
    }
    System.out.println("");
}
```

0<5

j<=i

1<=0 (F)

*

How to take input from the user?

Scanner scanner=new Scanner(System.in);

Problem statement:

i have today date:22/4/2024 31/12/2024 leap year?

23/4/2024 1/1/2025

28/2/2024

29/2/2024

1/3/2024

1. write logic to check no of days in each month
2. ..... leap year
3. ..... to increse 1 into current day

ask user to provide day month and year

31/12/2024

31+1=32

1/13/2024

```
int day, month, year;
        Scanner scanner=new Scanner(System.in);
        System.out.println("PE day month and year");
        day=scanner.nextInt();
        month=scanner.nextInt();
        year=scanner.nextInt();

        System.out.println("date is :"+ day+"/"+month+"/"+ year);
        /*
         * 1. write logic to check no of days in each month
               2. ..... leap year
               3. ..... to increse 1 into current day
         */
        //                          0     1     2.....12
        int noOfDays[]= {-1, 31,28,31,30,31,30,31,31,30,31,30,31 };

        //now i need to check if it is a leap year
        //copy paste is not a crime in programming

        if( year % 4 == 0 && year % 100 != 0 || year % 400 == 0) {
               noOfDays[2]=29;//if it is leap year then days must be 29
        }

        day=day+1;
        if(day>noOfDays[month]) {
               day=1;
               month=month+1;
               if(month>12) {
                      month=1;
                      year=year+1;
               }
        }

        System.out.println("next date is :"+ day+"/"+month+"/"+ year);
```

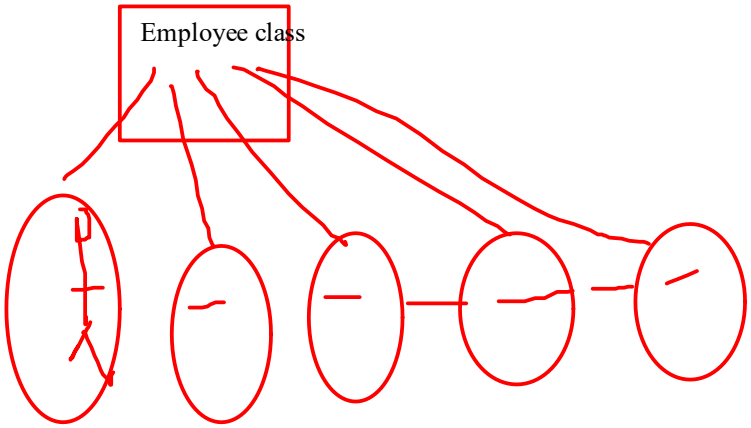
28/2/2023

day=1 month=3
if(29>28)

1

1/3/2023

clean code
modular code
i should not write everything in the main
i should multiple methods




```
class Employee{
	//instance data: per object
	private int id;
	private String name;
	private double salary;

	//class data: class variable: per class
	private static companyName="abc";

	//methods: ctr: have same name as of class, default , parameterized , copy ctr
	public Employee(){
	}
	//setter is used to set the values
	public void setSalary(double sal){
			int temp=4;// local variable , define inside a method
			salary=sal;
	}

	//getter
	public double getSalary(){
		return salary;
	}
	//Logic
	public double netSalary(){
			return salary * 0.8;
	}
}
```

How to visulized?

```
class Employee{

}
```

Employee d1=new Employee(…);

Employee d2=new Employee(…);

stack

d1

d2

heap

id=4
name=bob
salary=7000

id=84
name=raj
salary=9000

method area

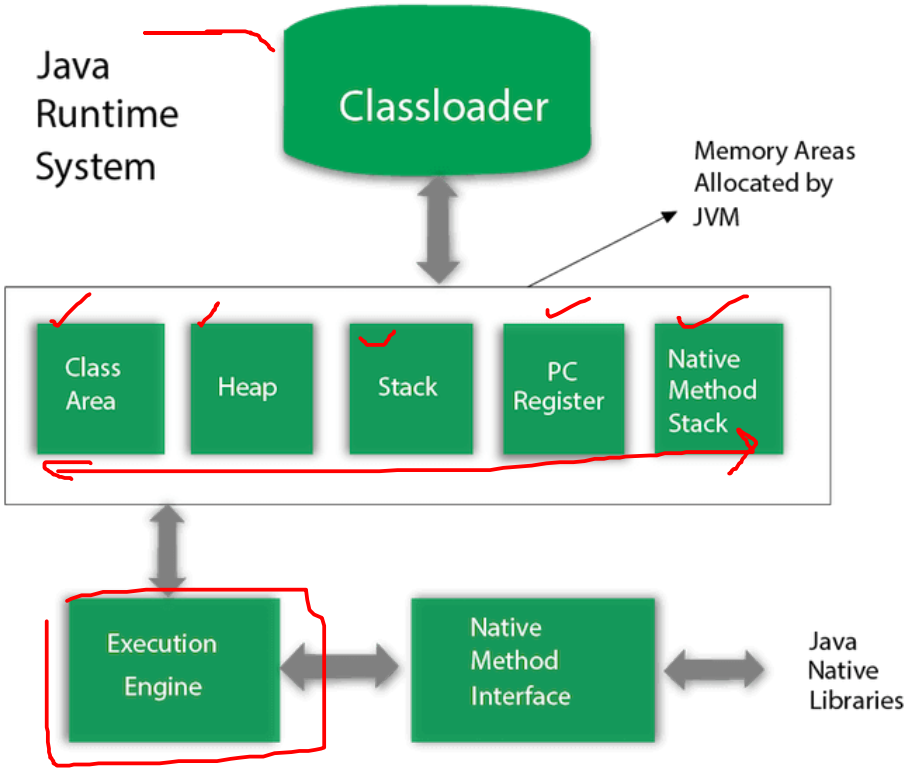
employee class is loaded into the method area

d1=null;

Java app --> JVM
(Java virtual machine)

C++: it have a concept of destuctor
java : garbage collector(inside jvm)
u dont have to do memory mgt

what is happing: object become candidate of garbage collector :)



Q: how to create classes and object

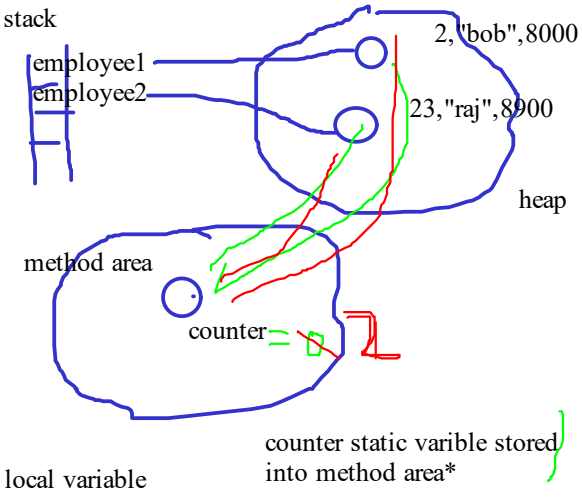
U need to count how many employee are created: counter?

```
class Employee{
    private int id;
    private String name;
    private double salary;

    //ctr: defualt ctr of the class
    public Employee() {
        System.out.println("default ctr");
    }
    //Parameterized ctr
    public Employee(int eId, String eName, double eSalary) {
        id=eId;
        name=eName;
        salary=eSalary;
    }
    public void print() {
        System.out.println("id : "+ id +" name: "+ name +" salary: "+ salary);
    }

}
public class D_EmployeeExample {
    public static void main(String[] args) {
        Employee employee1=new Employee(12,"bob",8000);
        Employee employee2=new Employee(23,"raj",8900);

        employee1.print();
        employee2.print();
    }
}
```

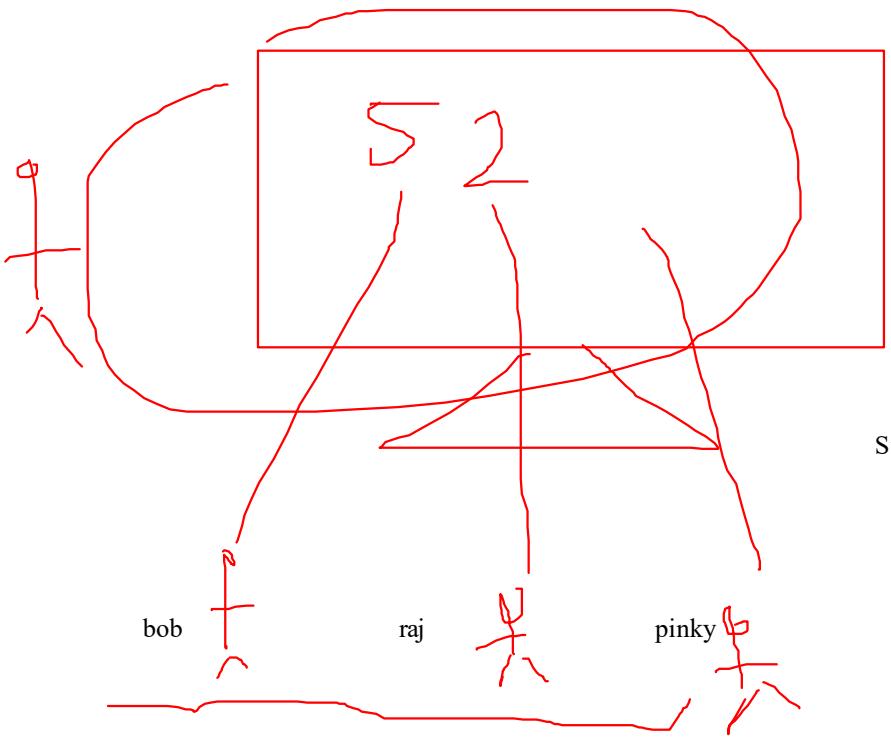
instance variable

local variable

we are copying the values form local varaible to instance variable



counter static varible stored into method area*



Static data is shared among the object

```
public Employee() {
        this(101,"foo",4000);
        counter++;
    }
    //Parameterized ctr
        public Employee(int id, String name, double salary) {
        id=id;
        name=name;
        salary=salary;
        counter++;
    }

    Employee employee4=new Employee();
```

"this"
what is use of this

1. used to call ctr of the same ctr

2. this is used to avoide confusion bw local variable and instance variable

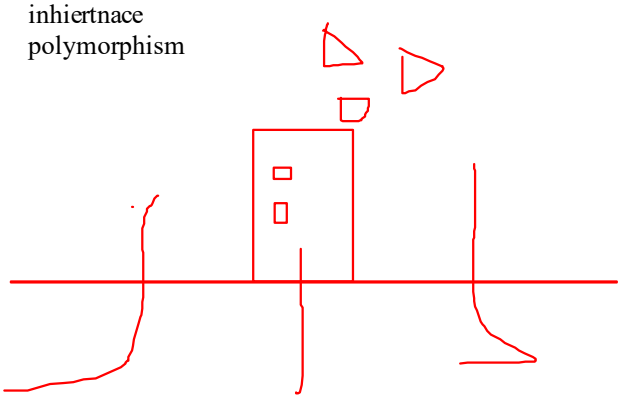
u are assigning local variable to local variable itself dont make sence

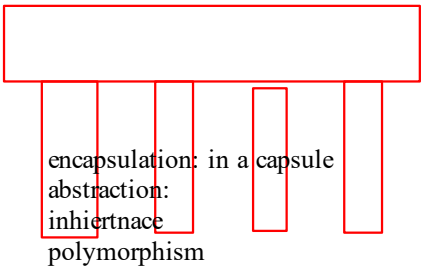
how to force java to assign local variable to the instance variable

"this"

oo
encapsulation: in a capsule data + methods : proper visiblity modifer

abstraction:
inhiertnace
polymorphism





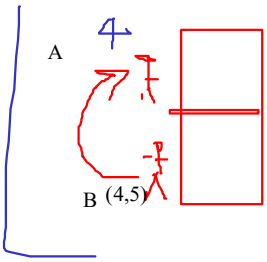
Understanding inheritance ?

```
class A{                          : private data never get inheritated ....
        private int i;
        A(){                       : ctr never ever get inhier.....
        }
        void foo(){}               : method are inheritable ...

        static int k=67;                yes

        static void fooStatic(){
        }                                yes
}
class B extends A{
        private int j;
}
```

```
class A{
        public A() {
                System.out.println("ctr of class A");
        }
}
class B extends A{
        public B() {
                super();
                System.out.println("ctr of class B");
        }
}
public class E_SimpleInheritnceEx {

        public static void main(String[] args) {

                B b=new B();

        }

}
```

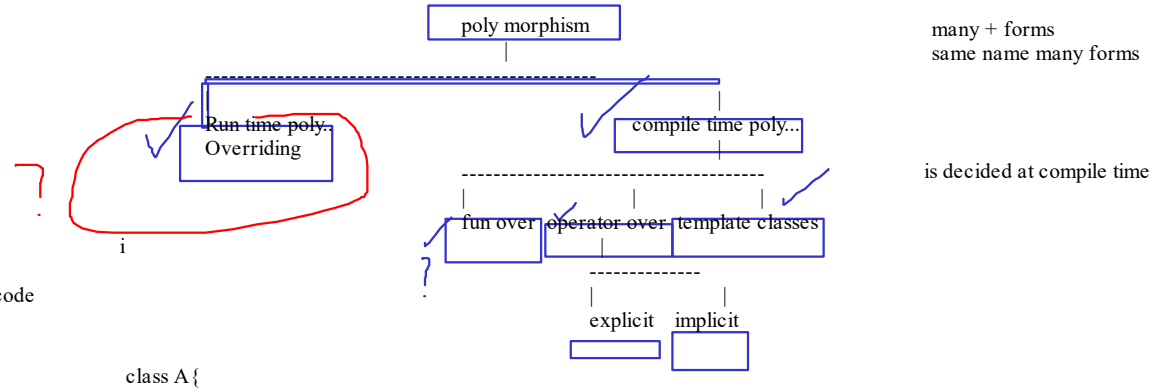
it is the call to the base class ctr
it is implicit

```
class A{
    private int i;
    public A(int i) {
        this.i=i;
    }

}
class B extends A{
    private int j;
    public B(int i, int j) {
        super(i);
        this.j=j;
    }
}
public class E_SimpleInheritnceEx {

    public static void main(String[] args) {
        B b=new B(4,5);
    }
}
```




```
class A{
    private int i;
    public A(int i) {
        this.i=i;
    }
    public void print() {
        System.out.println("value of i: "+ i);
    }
}
class B extends A{
    private int j;
    public B(int i, int j) {
        super(i);
        this.j=j;
    }
    public void print() {
        super.print();
        System.out.println("value of j: "+ j);
    }
}
public class E_SimpleInheritnceEx {

    public static void main(String[] args) {
        B b=new B(4,5);
        b.print();
    }
}
```

Polymorphism
abstract class
interface

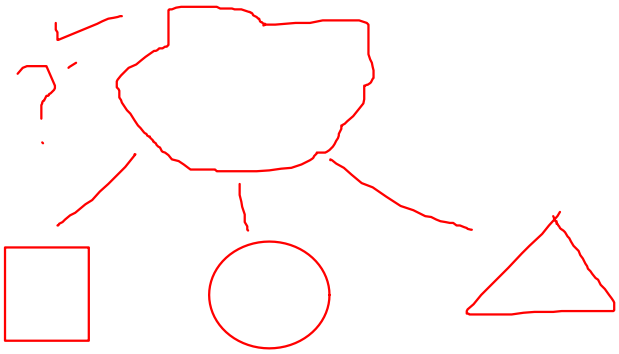


many + forms
same name many forms

is decided at compile time

provide u code
flexablity


```
class A{
  void foo(){

}
}
```

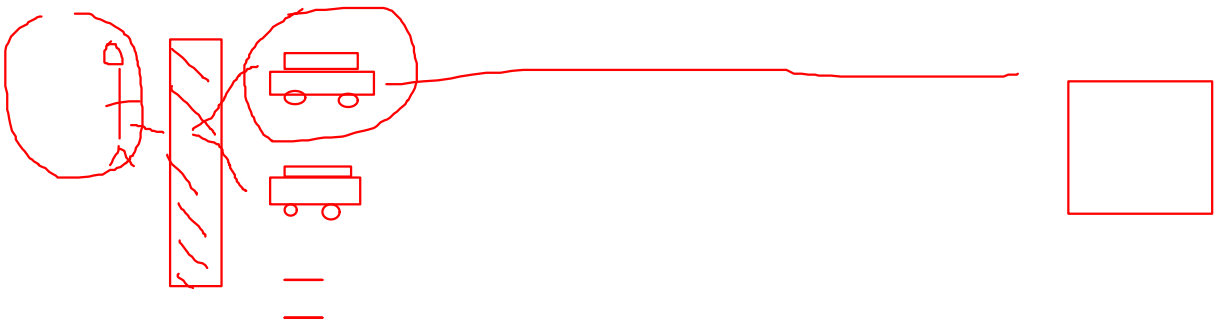


```
abstract class Employee{
        private int id;
        private String name;

        public Employee(int id, String name) {
                this.id = id;
                this.name = name;
        }

        public void print() {
                System.out.println("id : "+ id +" name: "+name);
        }
        public abstract double getSalary();
}
```

```
class PartTimeEmployee extends Employee{
        private int noOfHr;
        private int payPerHr;

        public PartTimeEmployee(int id , String name, int noOfHr,int payPerHr) {
                super(id, name);
                this.noOfHr=noOfHr;
                this.payPerHr=payPerHr;
        }
        public void print() {
                super.print();
                System.out.println("noOfHr : "+ noOfHr +" payPerHr: "+payPerHr);
        }
        //this method have overriden the base method
        public double getSalary() {
                return noOfHr*payPerHr*0.8;
        }
}
```

```
class FullTimeEmployee extends  Employee{
        private String ppfNo;
        private int baseSalary;

        public FullTimeEmployee(int id, String name, String ppfNo, int baseSalary) {
                super(id, name);
                this.ppfNo = ppfNo;
                this.baseSalary = baseSalary;
        }

        public void print() {
                super.print();
                System.out.println("ppfNo : "+ ppfNo +" baseSalary: "+baseSalary);
        }

        public double getSalary() {
                return baseSalary *10*0.8;
        }

}
```

```
Employee employee1=new FullTimeEmployee(12, "raja", "ASB555", 1000);
System.out.println("salray of employee 1 :"+ employee1.getSalary());
```



"a passanger is moving from Delhi to Taj Mahal