Ghansham

static ~~& final~~ Final keywords

⇒ Static keyword is used ~~for~~ as metadata (data about data)

→ Static members are used to give information about a class

* If you have only data related to clas
  ⇒ Create static data members
* If you want to perform some operations on ~~them~~ some static data member
  ⇒ create static methods / functions
* You have lot of info and functionality (lot of methods
  ⇒ create nested classes
* You can also able to create some static blocks also

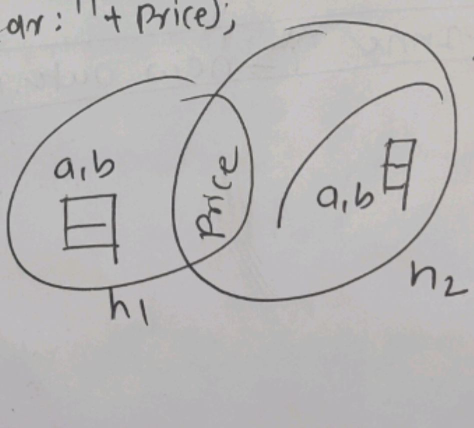
⇒ the variable inside class who don't give alot of info about class can be stated as classic

→ the static members of class are unique over all objects of that class that why [they provide info about class]

* a class for car

```
class Hondacar{
static int price = 1000;
  int z,y;

static ~~void~~ int carPrice(){
  ~~switch(city)~~{
  s.o.p("Price of car:" + price);
  }
}
```

```
main(){
Hondacar h1 = new Hondacar();
Hondacar h2 = new Hondacar();
}
```



⇒ the static value from a class are unique to the

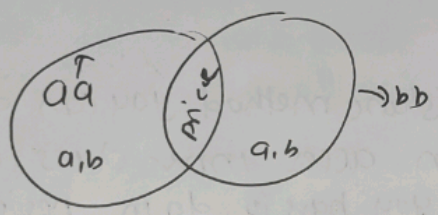
⇒ ~~others~~ ~~static~~ non-static variable are unique for each object.

~~for st that~~

⊛ Static Variable In class

```
class Hondacity {

Static int price = 100;
int x,y;

Static double onRoadprice (string city) {
    city = city.to(lowercase)
    switch(city) {

    case "mumbai":
        return price + price * 0.1;

    case "delhi":
        return price + price * 0.11;

    }
}
```

AS previously said if you have a common static variable throughout the whole objects so



so if any object change value of static so value for all object for that static variable will also change

So create object

```
Hondacity aa = new Hondacity();
Hondacity bb = new Hondacity()

sop(aa.price);//100
sop(aa.price);//100

so
aa.price = 120;
so.p(aa.price);//120
s.o.p(bb.price) //120
```

→ static data can be used as shared data

⇒ Also static function can able to access through className/ interface name as they are metadata for that class

Ex:

```
interface zz {
    int m;
    void display();

    static m(){
        s.o.p("Hell");
    }
}
```

Static class → zz.display()

zz is interface name

zz.m();

⇒ inside a interface all ~~functions~~ data members are static by default

⇒ static value (method or data members() are common through out class / interface

→ the static value can exist even the object of that class is not created.

⇒ static value are not created inside heap they are available in method area (Datasegment of code)

static methods

→ If you want to perform oprations on static members we will create static methods.

→ static methods will only able to access static members & can't access non static method

In last page class

method onRoad price cant access a & b

→ you can access static method without creating object

: Hondacity.OnRoadPrice();

can directly be called.

static inner classes

so If you have multiple static method you can create inner static class and you can access inner class without a lot of hard word which you have to do in nested inner loops

⇒ ~~Nest~~ static ~~cla~~ inner ~~also~~ will only able to access static members

⇒ Inside ~~static~~ ~~note~~ static nested class you cant be able use keyword like super or this.

⇒ you class create static classes you only have permission to create a static ~~cla~~ inner class

Static class Hell{

};

↑
not possible

Static keyword can only used to create inner classes

Class Hell{

static class Hell2{

}

}

↑
possible

static blocks

If you have some static value in your code and if you want to initialise them then we use static block

```
class hell {

    static int z;

    static {
     s.o.p ("blah blah");
     z = 100;
    }

    static {

    s.o.p ("static block");
    }
```

static block :

```
static {
    =|some-code|=
}
```

static blocks are like static functions and they only can access static variables

=> there can be multiple static blocks and they gonna execute in order at which you write them

=> the static block from class will invoked while initialisation of that class is taking place (loading of class)

=> static blocks are there to setup static values

=> static blocks are not most commonly use static variable + static method gonna usefull in more cases

=> the class which we are going to use will get loaded, if a class is there in file but used once so at that time that class will not used so static block inside them will not going to ~~not~~ work

instance block

```
final int z;
    {
     z = 100;
    }
```

=> this are static block but can used to assign value inside instance block

-> you can assign values to variables other than static