~~Ob~~ Object Class

~~class~~ Every object we create has by default its own method. lets explore.

```
class Laptop {
    String model;
    int price;


}

public class Demo
{
  public static void main ( ) {

      Laptop obj = new Laptop ( );
          obj. model = "Lenovo Yoga";
          obj . price = 1000;

      Sout ( obj)      =>  what we get?
                              ↓
                          Laptop @ 7ad041f3
          ↓
```

So, actually it is Sout (obj. toString()) ~~}~~
by default

We don't have any method to string() in laptop

But it is present in Object class.

```
public String toString() {

        return getClass().getName() + @ +
                            Integer.toHexString (hashCode());
}
```

What it returns ?

In general, the toString method returns a string that textually 'represents' this object.

It returns a class name (getClass()) then it calls getName() (i.e. laptop) + @ + then it converts hashcode into hexadecimal no.

We can override that toString method():
→ we can also define it in class laptop & call that

```
class laptop {

        toString() {
                Sout("Hey");
        }
```

So, obj.toString() → Hey,
even
sout (obj) = Hey.

Now, lets make another object obj2

obj2.model = "Lenovo Yoga";
obj2.price = 1000;

~~boolean~~ Lets check if obj1 equal to obj2.

So, boolean result = ~~obj1~~ obj1.equals(obj2)

⇓

false
↓
why?

~~But~~ Our class don't have equals() method it is coming from Object class. which is

"| public boolean equal (Object obj) {
return (this == obj);
}

It compares using hexadecimal no. so its different

→ So we want to compare obj on the basis of their values not hexacode.

→ So, we will create equals ( ) method in our laptop class.

So, public boolean equals (laptop that) {
if (this.model.equals(that) && this.price == that.price)
return true

else false ;

→ Now, obj1. equals(obj 2) ⇒ true.

as their model & price are equal.

Note →

→ Please dont define our own method to compare equality. (as ~~we~~ we dont check for obj == null, & etc.

So, make IDE to create equals method as it will generate hash code for the parameters we wanna check & check for everything;

→ We can also use ide to generate

to String method();

Downcasting & Upcasting

→ **Typecasting**

```
double d = 4.5;
```

if we want d = 4.5 in integer format (ok, we will loose 0.5 data)

so,

```
int i = (int) d ;

Sout (i) = 4
```

→ In the ~~word~~ world of OOPs, can we do so?

Upcast & Downcast

```
class A
{
    public void show()
    {
            sout ( in A Show) ;
    }
}
class B extends A {

          public void show2 ( ) {   Sout ( in B show); }
}.

public class Demo
{
          main ( ) {
```

A obj = new A()
obj.show1 () → in A show

we can't use show 2 with A obj.
→ we don't have show 2 inside A, & A dont knows abt A

→ We know,

A obj = new B();
↓
But it actually is

A obj = (A) new B();
⇓
<u>UPcasting</u> (as we cast up)

obj.show1 = in A show

<u>Note</u> :- still we dont call obj.show2();
as reference of A, & it has no idea of show2();

⇒ So, we require <u>Downcasting</u>

to access ~~show2~~ show 2, we need object of type B.
so,
can we, B obj1 = obj;

So, → no, obj is of type A

B obj1 = (B) obj;
obj1.show2(); ✓