Chapter - 4

State affects Behaviour and Behaviour affects state.

Every class has the same methods, but they behave differently on the basis of value of instance variables.

Methods ⇒ Parameters (nothing more than a local variable)

Caller ⇒ Arguments

Java is pass-by-value, they mean pass-by-copy.

What happens when the argument is an object instead of a primitive?

If you pass a reference to an object, it means you pass the copy of the remote control.

Do I have to return the exact type I declared?

You can return anything that can be implicitly promoted to that type.

Getters and Setters ⇒ Accessors and Mutators

Encapsulation:

Mark your instance variables private and provide public getters and setters.

Instance variables get default values if you don't initialise them:

Integers ⇒ 0
Floating points ⇒ 0.0
Booleans ⇒ false
References ⇒ null

Difference between Instance and Local Variables:

1. **Instance variables are declared inside the class and not inside the method.**

```
class Horse{
        private double height = 15.2;
        private string breed;

        // more code….
}
```

2. **Local variables are declared within a method.**

```
class AddThing{
        int a;
        int b = 12;

        public int add(){
                int total = a + b;
                return total;
        }
```

3. **Local variables must be initialised before use..**

```
class Foo{
        public void go(){
                int x;
                int z = x + 3;
        }
}
```

//The compiler complains if you try to use an uninitialized local variable.

Comparing variables (Primitives or References)

1. **‘==’ Operator (used only to compare bits in 2 variables)**
   ⇒ Primitives are the same.
   ⇒ Reference variables refer to a single object on the heap.

2. **.equals() method**
   ⇒ If 2 objects are equal.