Chapter 8

↳ syntax :

* name of file = class name

```
public class Car {
    instance variable;
    constructors;
    getter & setter;
    other methods;
}
```

can be private too... but mostly use public

mostly always private :
eg. private String brand;

Accessor = getters
mutator = setters

eg. (Accessors)
```
public String getBrand() {
    return brand;
}
```

eg. (mutator)
```
public void setBrand(String newB){
    brand = newB;
}
```

** mutator sets value so, returns nothing, so, it's void

2 type : ① default constructor
② alternative constructor

① non-parameterized constructor
↳ syntax :
```
public Car() {}
```
↳ don't take in any parameter
↳ default value not provided by user

② parameterized constructor
↳ syntax :
```
public Car (parameter) {}
```
↳ at least 1 parameter
↳ parameter name must be different from instance variables
↳ or else logic error
↳ values provided by user

purpose :
↳ initialised instance variables
↳ create class obj

eg. ① default constructor
```
public Car() {
    brand = "Lamborghini";
}
```

② alternative constructor
```
public Car (String new_brand) {
    brand = new_brand;
}
```

* other methods :
↳ other actions included in a class
↳ eg.
public String carPrice ...
↳ usually accepts parameters...

** ① methods arranged alphabetically
② 1 blank line between methods
③ top : default constructor

Class variables:

↳ syntax:

```
public static String carBrand = "Ferrari";
public static int numOfStudents = 0;
```

↳ belong to class only

↳ declared & initialised in a single statement

* Identify class variable by keyword: static

↳ handled like object variable

↳ x accessed out of class = private

Class method:

↳ syntax:

```
public static String carBrand (optional parameter) {}
```

↳ eg. (getter for class variable carBrand)

```
public static String getCarBrand() {
    return carBrand;
}
```

Invoke a class through dot notation: → objName.methodName.methodName

① class itself

② object of class itself

** Obj represents a single instance of a class

eg.

```
public class car {
    ⋮
    private static int carNum = 0;
    ⋮
    public car() {
        carNum++;
        carBrand = "Undefined";
        yearMade = 0;
    }
    ⋮
    public static void main(String[] args) {
        ⋮
        car.showCar();
    }
    ⋮
}
```

Instantiating objects :

↳ instantiated object have own set of instance variable

↳ each instance variable can use & modified

↳ using default constructor : (lets say it's car class)

① default

eg. Car carOne = new carDefaultConstructor

② non-default

eg. Car carTwo = new Car NonDefaultConstructor

Scope : ① Class scope

② Block scope

- visible throughout the class
- objects & methods within class can access
  ↳ can use within any method

- defined anywhere in code
- must : matching {}
- common place :
  - class body
  - method body
  - selection constructs
  - loops

Lifetime :

↳ how long variables exist until destroyed

Variable: Fields

- **Definition:** Defined outside the constructors and methods.
- **Lifetime:** Exist while their object exists.

Variable: Parameters

- **Definition:** Defined in the header of the method or constructor.
- **Lifetime:** Exist while the method or constructor is executing.

Variable: Local variables

- **Definition:** Defined within the body of a method or constructor.
- **Lifetime:** Exist while the method or constructor is executing.

Declare object :

eg. Car objCar = new car();

objCar : identity

car() : insert variable value

main method

```
49      public static void main(String[] args)
50      {
51          Car objCar = new Car();
52          objCar.display();
53          objCar.setColor("Yellow");
54          objCar.setMake("Lamborghini");
55          objCar.setModel("Diablo");
56          objCar.setYear(2000);
57          objCar.display();
58      }
```

declare object

Chapter 9

Eg. (Class + Class Driver)

Class :

```
class Person {
    private int age;

    public void setAge( int newAge) {
        if (newAge >0 && newAge <=120) {
            age = newAge;
        }
    }

    public int getAge() {
        return age;
    }
}
```

Class Driver :

```
public class PersonDriver {
    public static void main (String[] args) {
        Person Anna = new Person();
        Anna.setAge(17);
        System.out.println ("Anna is " + Anna.getAge() + "years old.");
    }
}
```

★ Other methods include tostring method :

↳ tostring method return value given in string format

↳ used in Drivers class with print statements :

(using the previous class car)

Class : (lowest part, after all methods)

```
public String toString (){
        String returnStringValue = " ";
        returnStringvalue t = " Anna's age : " + getAge () + "years old";
        return returnStringvalue ;
}
```

Class Driver :

```
public class PersonDriver {
        public static void main (String[] args) {
                Person Anna = new Person ();
                Anna.setAge(17);
                System.out.println ( Anna.toString());
        }
}
```