Abstract Modifier

what it does exactly

```
void display() {
    → S.O.P("____");
    → ___
}
```

* private, public {—}
* static
* abstract

Concrete methods
⇓
A method with body

* Abstract method

example:

Athlete → ? Sport Good at
→ ①② 100 mts run

Type | ProTyp | blueprint

① abstract class Athlete {
    abstract void whatHeDoes(); {}
}

method() {
    ___
}

? ① → class Abstract

✓ 50%   × — 50%

② abstract class Swimmer extends Athlete {

Sn phase 1

IM phase 2

}

③ class Shooter extends Athelet {
    void whatHeDoes() {
        S.O.P("He shoots");
    }
}

100% → concrete about the method definition

Note:

① Abstract is a keyword in java which can be used for methods and class

② Abstract method do not have body

③ If there is any abstract method in a class then it is compulsory to make that "class" as abstract

④ It is not compulsory to have abstract methods inside the abstract class

```
abstract class A {
    add() {
        return a+b
    }
}
```

⑤ When a subclass inherits from a abstract class it is compulsory to override the abstract method, inherited from super class

(alternate)
↓

6. We can make the subclass as "abstract"

7. We cannot create object of abstract class, if we try to create, we get an error called "Cannot instantiate the type class"

8. We can create the reference of abstract class

Athlete shooter = new Shooter() ✓

new Athlete() ✗
new Swimmer() ✗

9. We can have concrete methods in abstract class

```
abstract class Athlete {
    abstract void whatHeDoes(); {}

    void displayProfile() {
        ===
    }
}
```

10. We can have constructors for abstract class

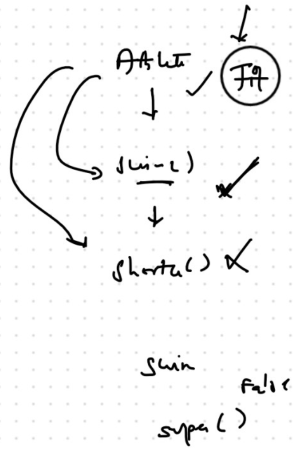
✓ properties }
✓ method

```
abstract class Athlete {
    bool isFit;
    public Athlete(bool isFit){
        this.isFit = isFit
    }
    abstract void whatHeDoes(); {}
}

class Swimmer extends Athlete {
    int weight;
    Swimmer(int weight, bool isFit){
        super(isFit)
        this.weight = weight
    }
    void whatHeDoes() {
        S.O.P("He swims")
    }
}
```

new Athlete() ✗

→ new Swimmer(45, true);

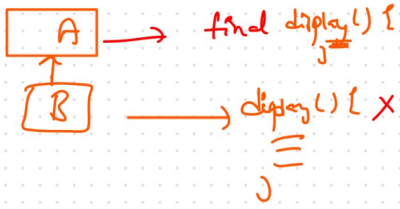


Note:

→ the need of constructor in abstract class is for "Constructor Chaining"

(11) We can have "final methods" inside the abstract class

↑ modifier



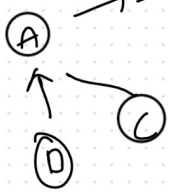

```
abstract class Athlete {

    abstract void whatteDoes();

    final void sleep(){
        s.o.p("sleeping");
    }
}
```

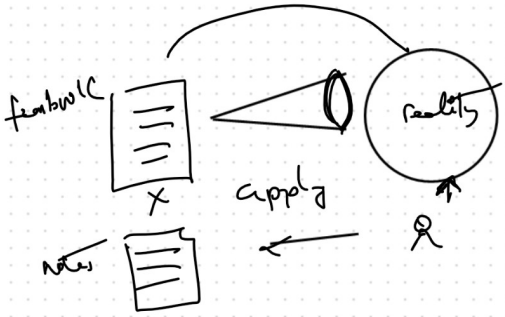
(12) We cannot use private & final static modifier along with abstract keyword.

① It does not allows you to inherit, because we need abstract method to override

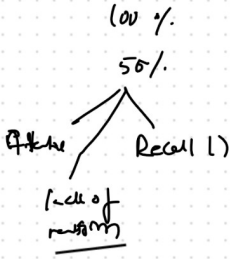


② ✓ ✗ private | abstract whatteDoes();
✗ static | abstract whatteDoes();
✗ final | abstract whatteDoes();
✗ private static abstract whatteDoes();

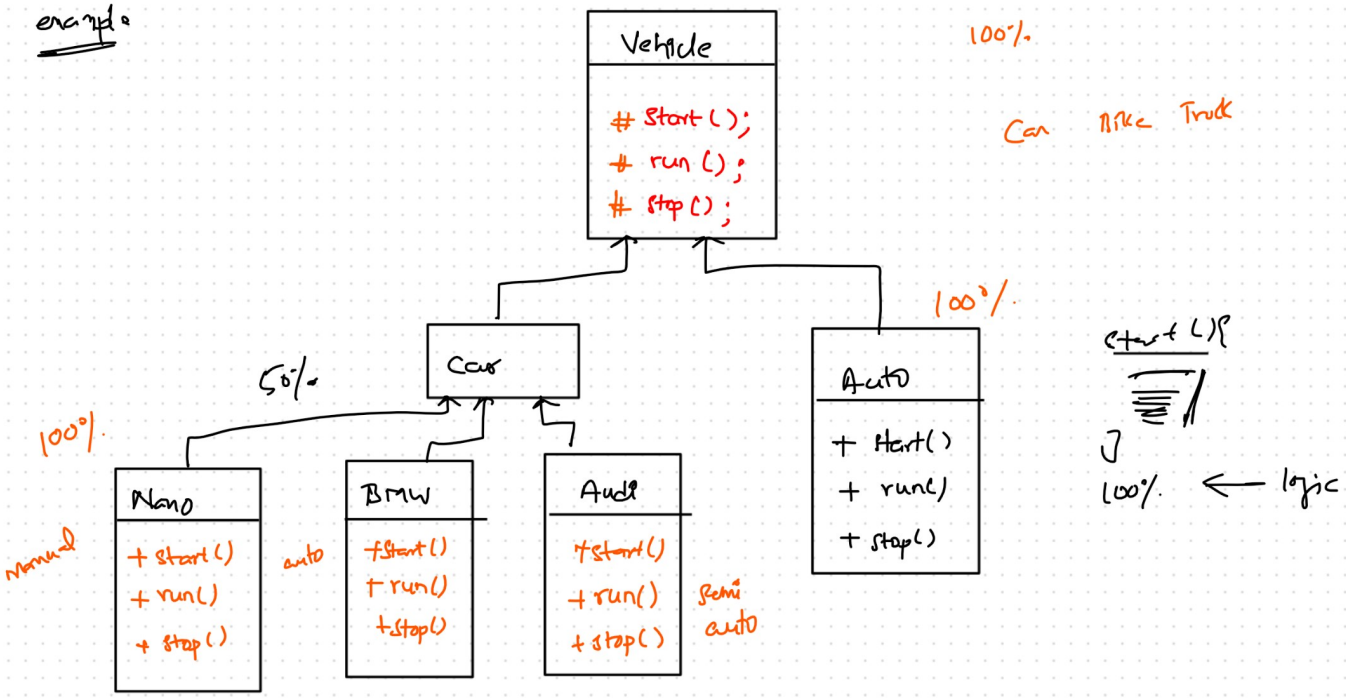
③ Since final won't allow any method to change in child class



Eg:



example


Vehicle
start();
run();
stop();
100%
Car like Truck
Car
50%
100%
Auto
100%
+ Hart()
+ run()
+ stop()
start(){
}
100% ← logic
Nano
+ start()
+ run()
+ stop()
manual
auto
BMW
+start()
+run()
+stop()
Audi
+start()
+run()
+stop()
semi auto