Agenda :

- → Polymorphism (contd.)
- → Interfaces
- → Abstract classes
- → Association
  - → Aggregation
  - → composition

[Effective Java

Java - The complete Reference]

Types of polymorphism

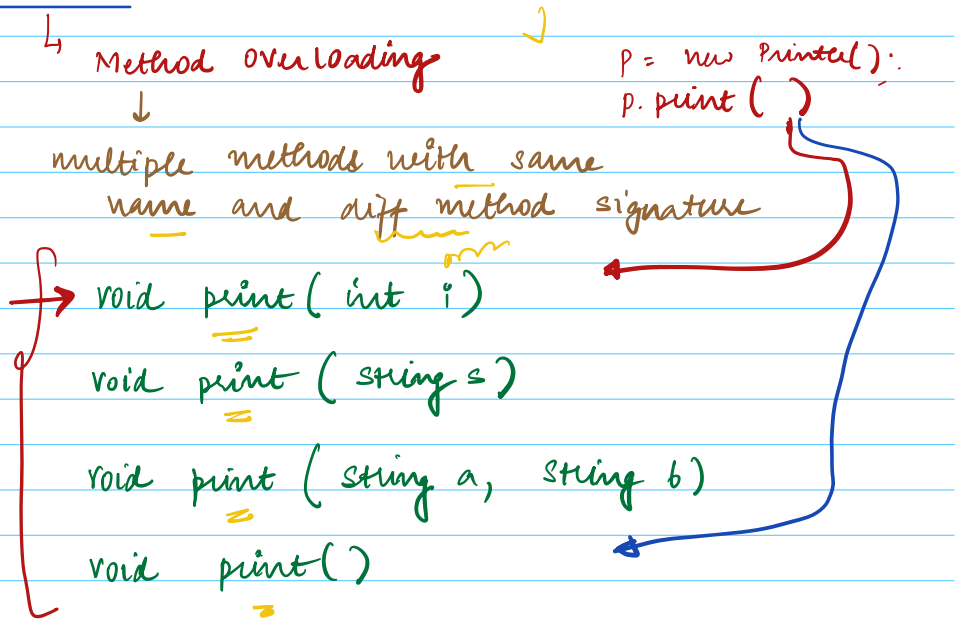
1) Compile - Time
2) Runtime

1) Compile Time

↳ Method overloading

↓

multiple methods with same name and diff method signature



We are able to decide at compile-time which function will be called. based on method signature, or no of arguments.

2) Runtime polymorphism

↓

method overriding

child class overrides parent's behaviour

class A

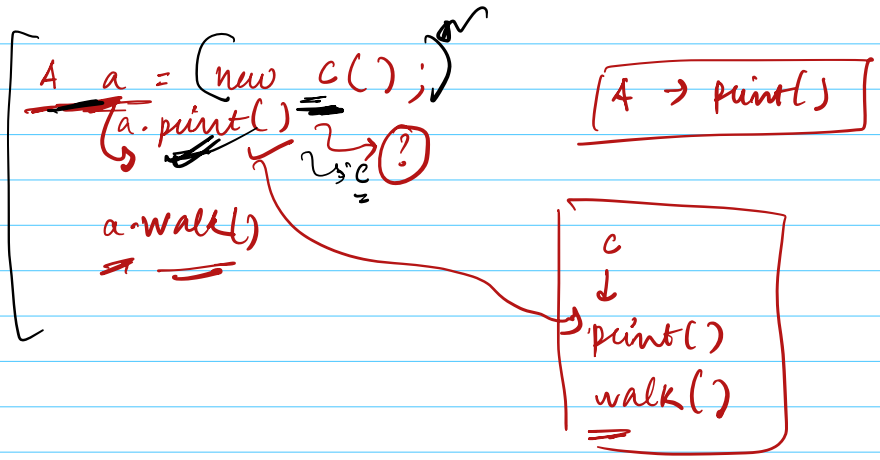
print()

class B extends A

print()

class C extends B

print()



INTERFACES

Class => Blueprint of an entity
↳ Attributes
↳ Methods ⇝ declared ✓ and defined ✓

Blueprint of a class, better behaviours
↳ contains static constants and abstract methods

Specify what a class must do, but no how it does it.

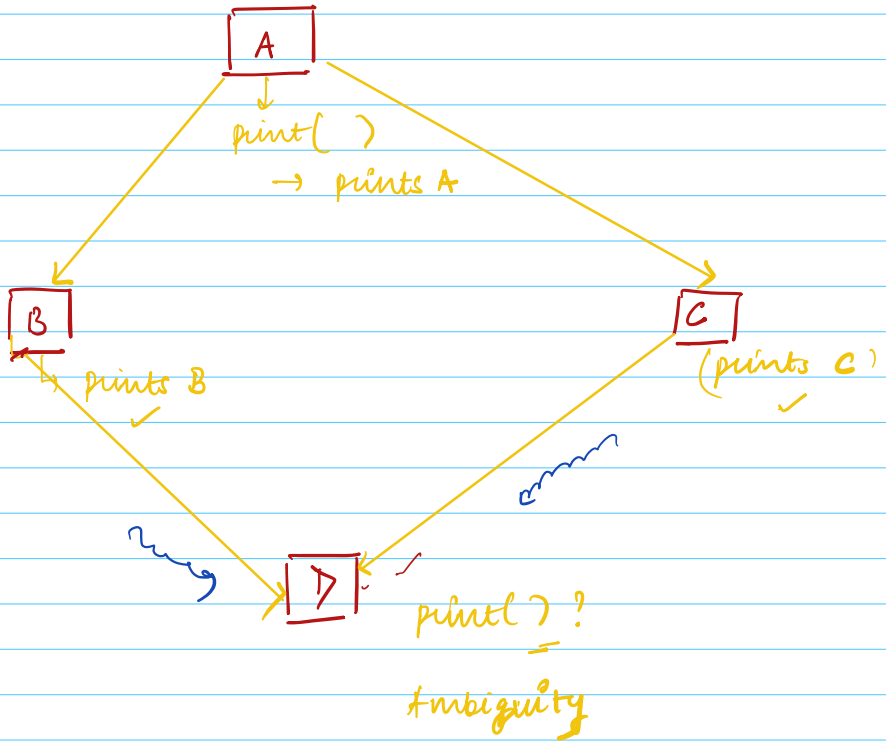
→ No instance variables
→ No method body
→ Basically no assumptions on how methods are implemented.

100% Abstraction

```
interface Flyable {

    void fly();

}
```

Diamond Problem

↳ Arises because of multiple inheritance



Java → no multiple inheritance

↓

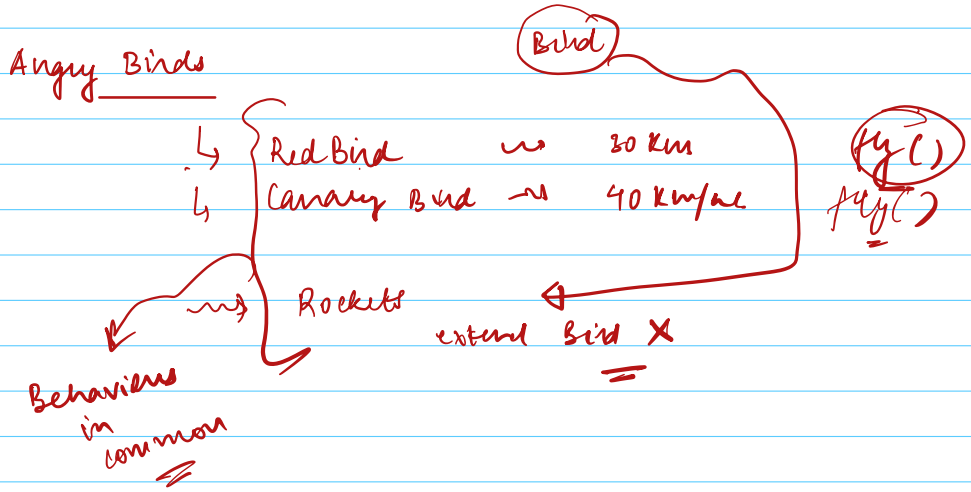
✓ No diamond problem

{1 less Feature}

```
class A {

    void print() {

    }
}
```

```
interface X {

    void print();    // declared a method

}
```

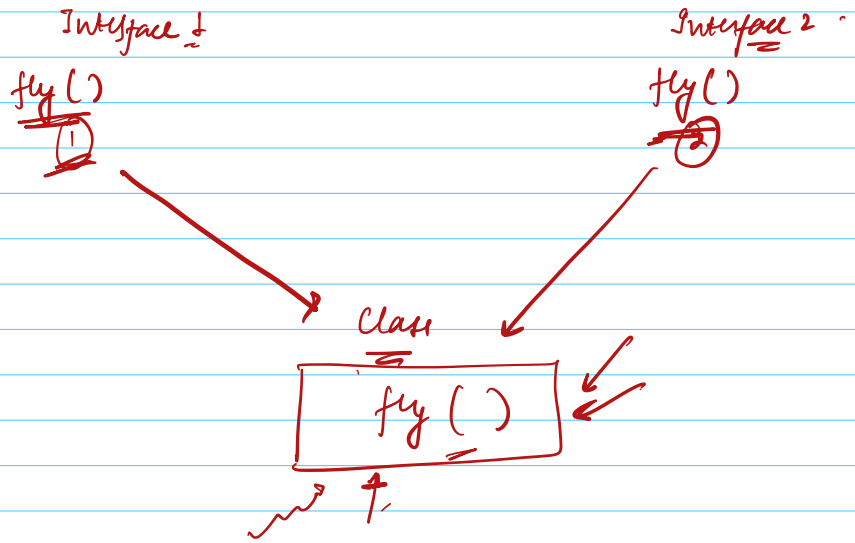
(By default all methods of interfaces are public)



```
interface Flyable {
    void fly();
}

RedBird extends Bird implements Flyable {
    void fly() {
    }
}

Rocket implements Flyable {
    void fly() {}
}
```

You can implement many interfaces

→ Abstraction
→ Blueprint of behaviour
→ Loose coupling



Abstract class

↳ You can't create object of this class

↓

Because this class might miss on some methods / behaviours.

↓

meant to be inherited

```
abstract class User {
    private String name;
    private int age;
    public void login();
}
```

abstract