10-15 %

85-90%

Easily
- Maintainable
- Understandable
- Extensible

Maintaining
- Testing
- Find bugs
- Fix bugs
- Refactor

Understandability
- Understand others code
- Review Pull requests
- KT

Extensibility
- Regression bugs
- Merge conflicts

- OOP (Inheritance, Abstraction, Polymorphism, encapsulation)
- SOLID
- Design Patterns
- UML (diagrams) [Class diagram*]

* Give an OOD of bird.

Amazon

Template / BluePrint

Bird
→ Properties
→ Actions / Behaviour.

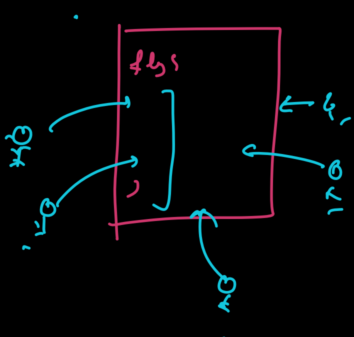
```
Bird hen = new Bird();
    hen.fly();          ⇐

Bird eagle = new Bird();
    eagle.fly();        ⇐
```

```
Class Bird {
  // Attributes
  String color;
  double ht, wt;
  String type;
    ⋮
  // Methods
  void eat() {
     // ________
  }
  void sleep() {
     // ________
  }
  void fly() {
     // ________
  }
}
```

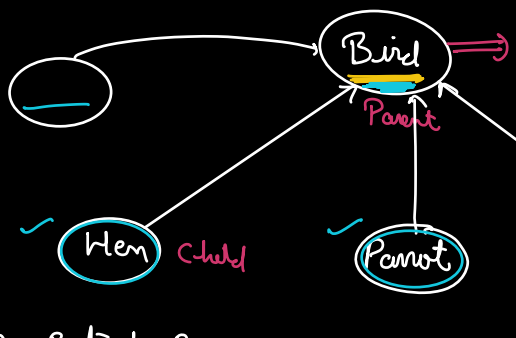
```
void fly() {
  if (this.type = "hen") {
     // fly like a hen      ⇐ [15 code]
  }
  else if (this.type = "eagle") {
     // fly like an eagle   [12-15]
  }
     ⋮
}
```

+50 if/else

+1 if/else



⇒ MergeConflicts



All common things
↳ Common attributes (wt, ht)
↳ Common methods

```
Class Hen extends Bird {
    void fly() {
        // fly like a hen
    }
}
```

```
Class Eagle extends Bird {
    void fly() {
        // fly like an eagle
    }
}
```

Single Responsibility Princ

Open/ close princ
↳ extention
↳ Modification

Overriding

Polymorphism

No one should be allowed to create object of Bird class.
↳ Abstract class (Java)

```
abstract class Bird {
    String color;
    double ht, wt;
    String type;
    ⋮
    void sleep() {
    }
    abstract void fly();
}
```

abstract class ⇒ No object can be created

abstract method ⇒ ① No implementation
② Child classes must override.

Abstract class can have normal methods (all normal also possible)

But an abstract method can not be present in a non-abstract class.

```
int add (int a, int b) {
    ret a+b;
}

main () {
    string a = "abc", b = "xyz";
    print (add (a, b));
}
```

```
double add (double a, double b)
    ret a+b;
}

String add (String a, String b)
    ret a+b;
}
```

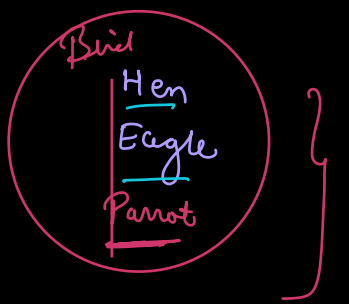
Method Overloading

Multiple forms shown by add (--)

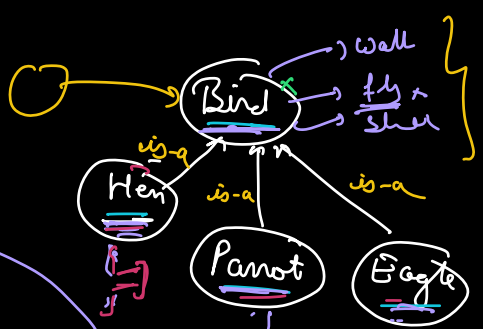
Poly morphism

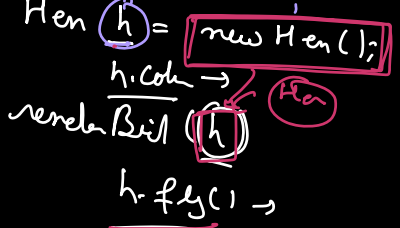
```
Class AngryBird {

    void renderBird(Hen h) {
        h.walk();
        h.fly();
        h.sleep();
    }

    void renderBird(Eagle e) {
        e.walk();
        e.fly();
        e.sleep();
    }

    void renderBird(Parrot p) {
        p.walk();
        p.fly();
        p.sleep();
    }

    ⇓

    void renderBird(Bird b) {
        b.walk();
        b.fly();
        b.sleep();
    }
```

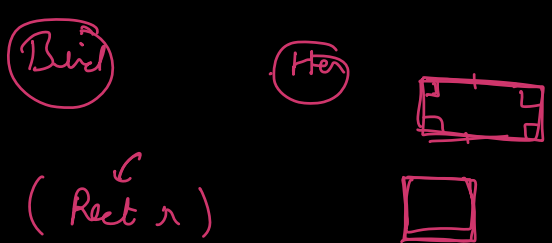
Abstraction



you wrote it







Java 7 | Java 8

HashMap — (A) — (B)

```
class FlyObject {

    void Render (Flyable f) {
        f.fly();
    }
}
```

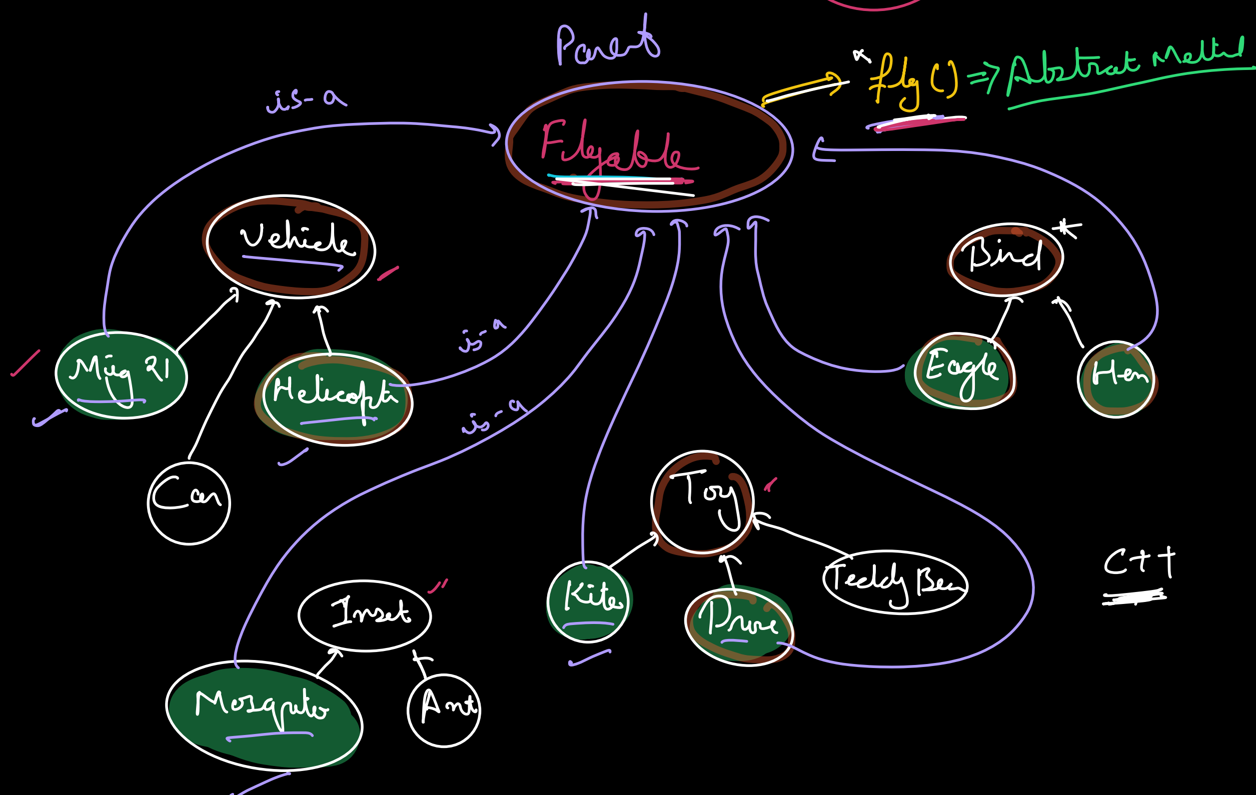
Objects that can fly

Vehicle
- Mig 21
- Helicopter

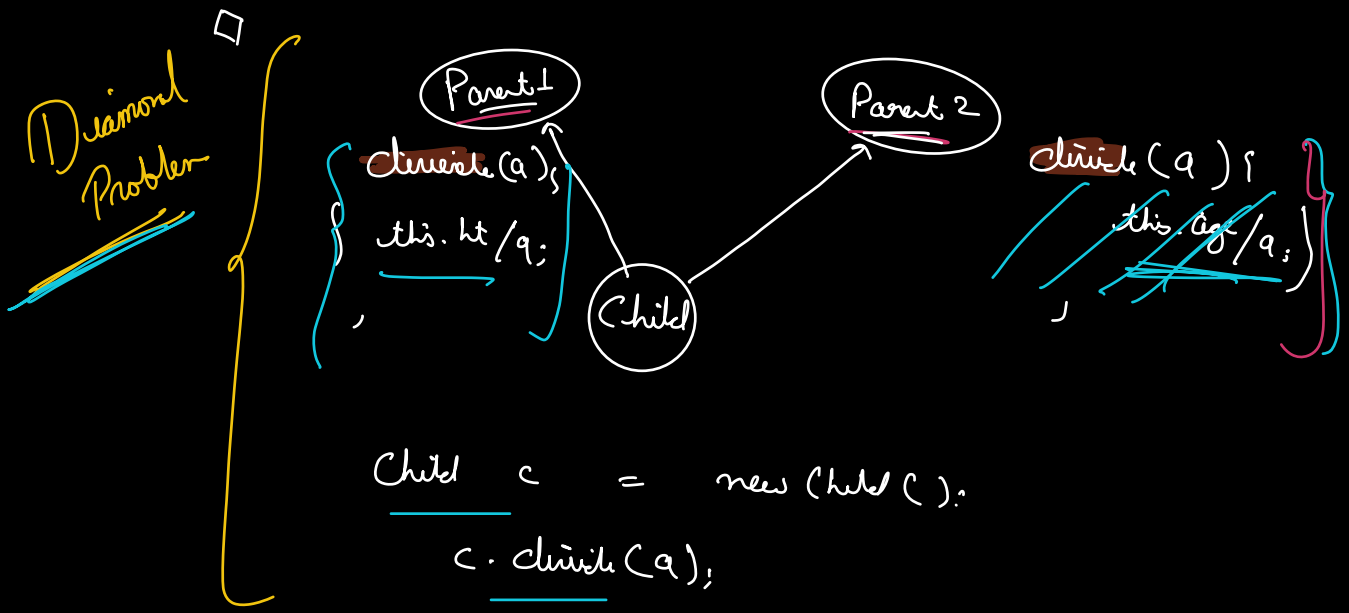
Birds
- Hen
- Eagle

Toy
- Kite
- Drone

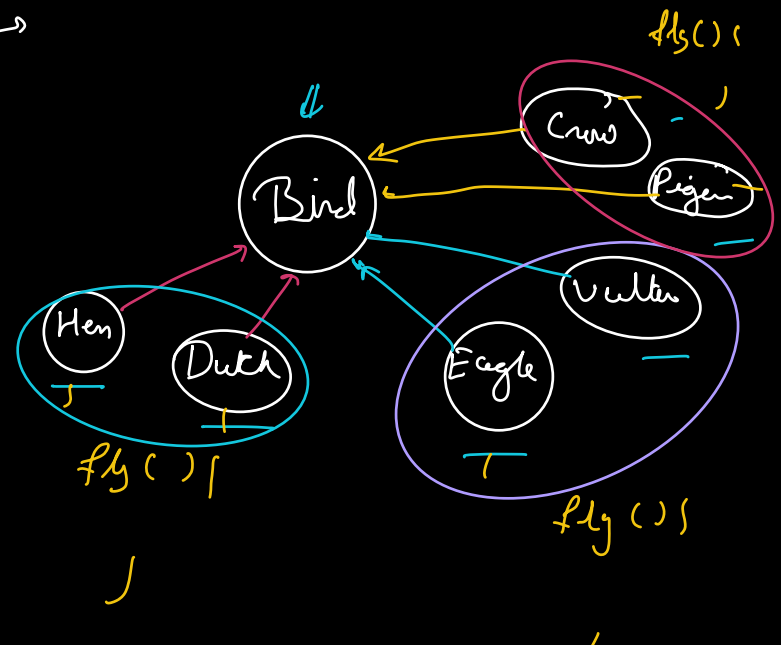
Insect
- Mosquito
- Crickets

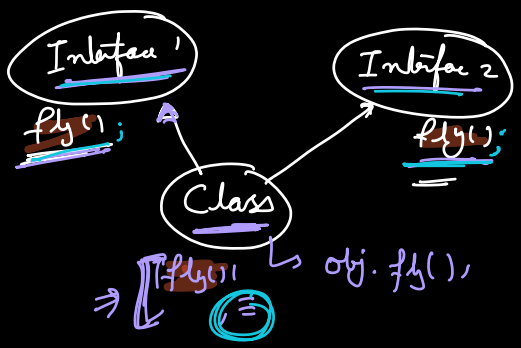


C++




```
Interface flyable {
    void fly();
}

reach ( flyable f ) {
    f.fly();
}
```






```
interface flyable1 {
    void fly();
}

interface flyable2 {
    void fly();
}

Class C impl f1, f2 {
    fly() {
}

Class A impl flyable1 {
    void fly() {
        // ———
    }
}

Class B impl flyable2 {
    void fly() {
    }
}

ClassA. fly();
ClassB. fly();
```