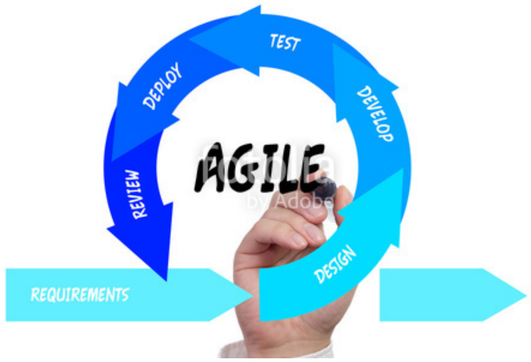



# Classe 25.10.2021: Disseny: SOLID-Solucions

Anna Puig

Enginyeria Informàtica

Facultat de Matemàtiques i Informàtica,

Universitat de Barcelona

Curs 2021/22

UNIVERSITAT DE BARCELONA

# Temari

1. Introducció al procés de desenvolupament del software
2. Anàlisi de requisits i especificació
3. Disseny
   - 3.1 Introducció
   - 3.2 Patrons arquitectònics
   - 3.3 Criteris de Disseny: G.R.A.S.P.
   - 3.4 **Principis de Disseny: S.O.L.I.D.**
   - 3.5 Patrons de Disseny
4. Del disseny a la implementació
5. Ús de frameworks de testing

# 3.4. Principis de Disseny: S.O.L.I.D.

**Principis de disseny** [Robert C. Martin 98]:

- **S**: Single Responsability Principle

  *"One class should have one and only one responsibility"*

- **O**: Open-Close Principle

  *"Software components should be open for extension, but closed for modification"*

- **L**: Liskov Substitution Principle

  *"Derived types must be completely substitutable for their base types"*

- **I**: Interface Segregation Principle

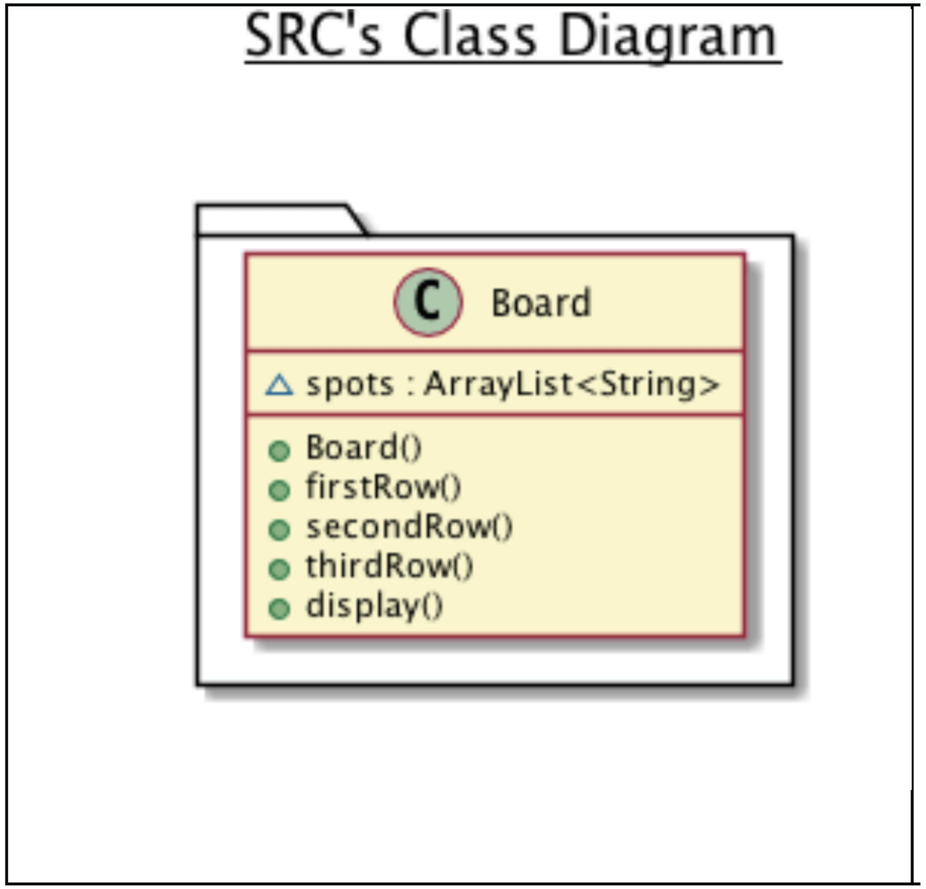
  *"Clients should not be forced to implement unnecessary methods which they will not use"*

- **D**: Dependency Inversion Principle

  *"Depend on abstractions, not on concretions"*

Articles de suport de cada principi

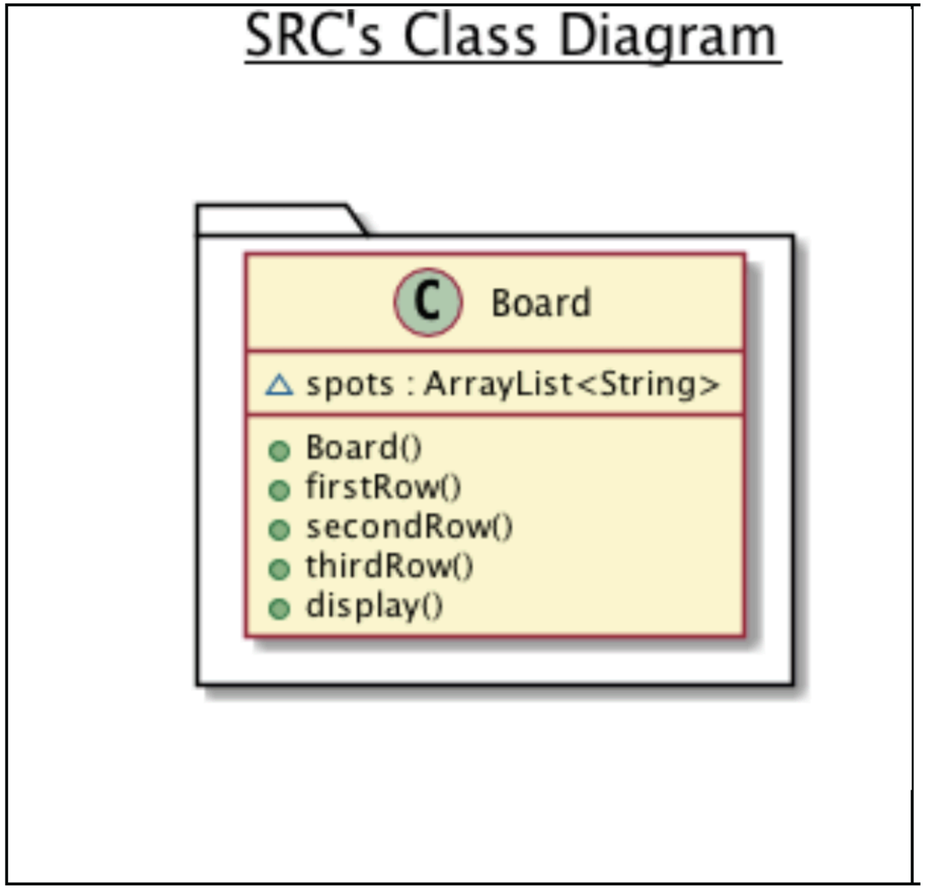
# 1. Exploració dels projectes: Projecte 1




Què fa aquesta classe?

Fa el taulell i el visualitza?

# 1. Exploració dels projectes: Projecte 1

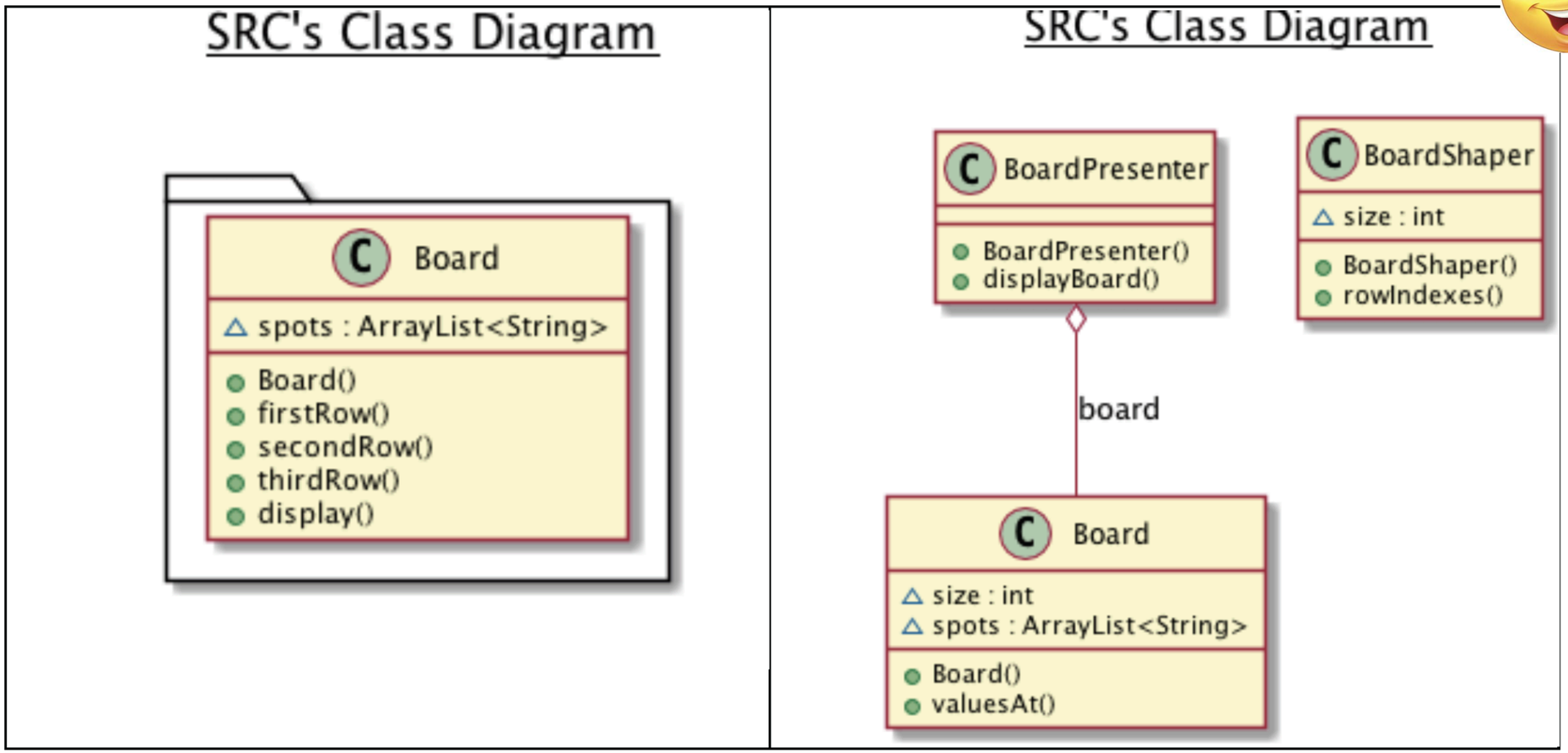



Què fa aquesta classe?

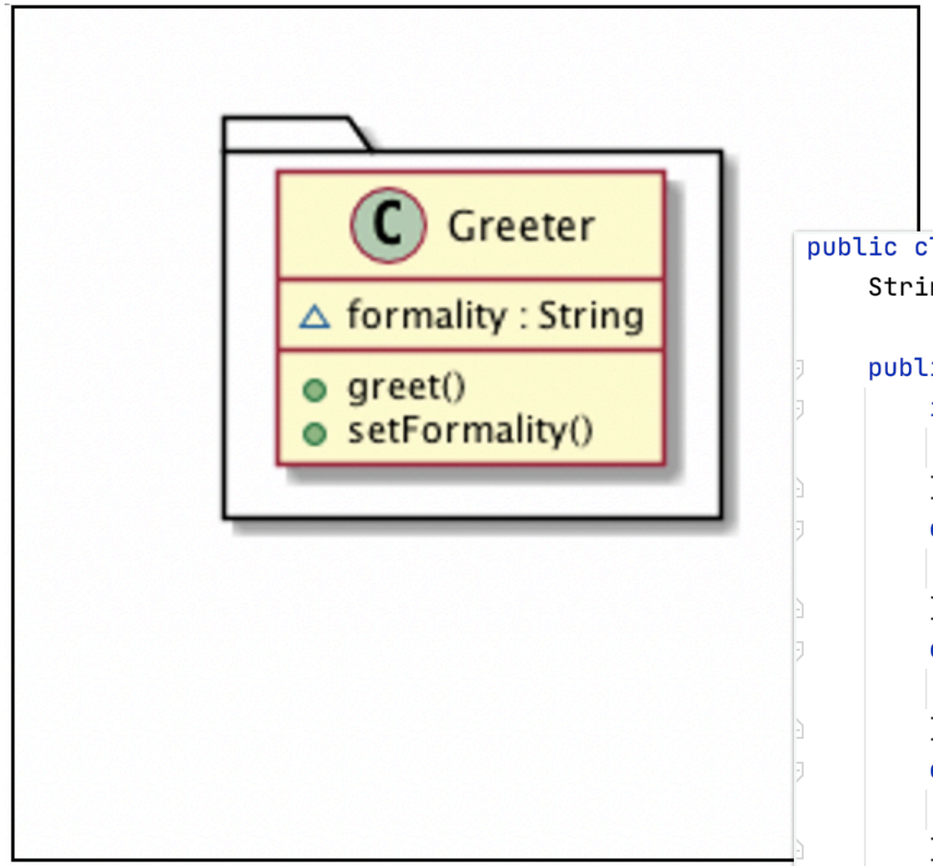
Fa el taulell i el visualitza?

**Single Responsability Principle?????**

# 1. Exploració dels projectes: Projecte 1



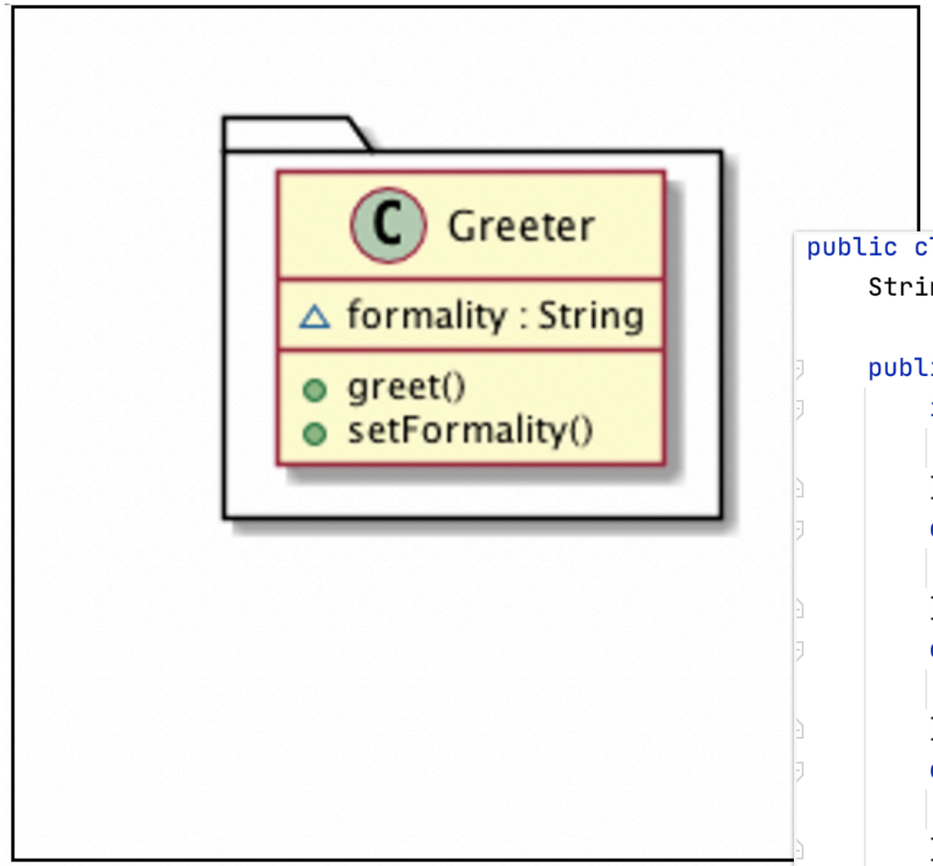
# 1. Exploració dels projectes: Projecte 2




Què fa aquesta classe?

```
public class Greeter {
    String formality;

    public String greet() {
        if (this.formality == "formal") {
            return "Good evening, sir.";
        }
        else if (this.formality == "casual") {
            return "Sup bro?";
        }
        else if (this.formality == "intimate") {
            return "Hello Darling!";
        }
        else {
            return "Hello.";
        }
    }

    public void setFormality(String formality) { this.formality = formality; }
}
```

# 1. Exploració dels projectes: Projecte 2



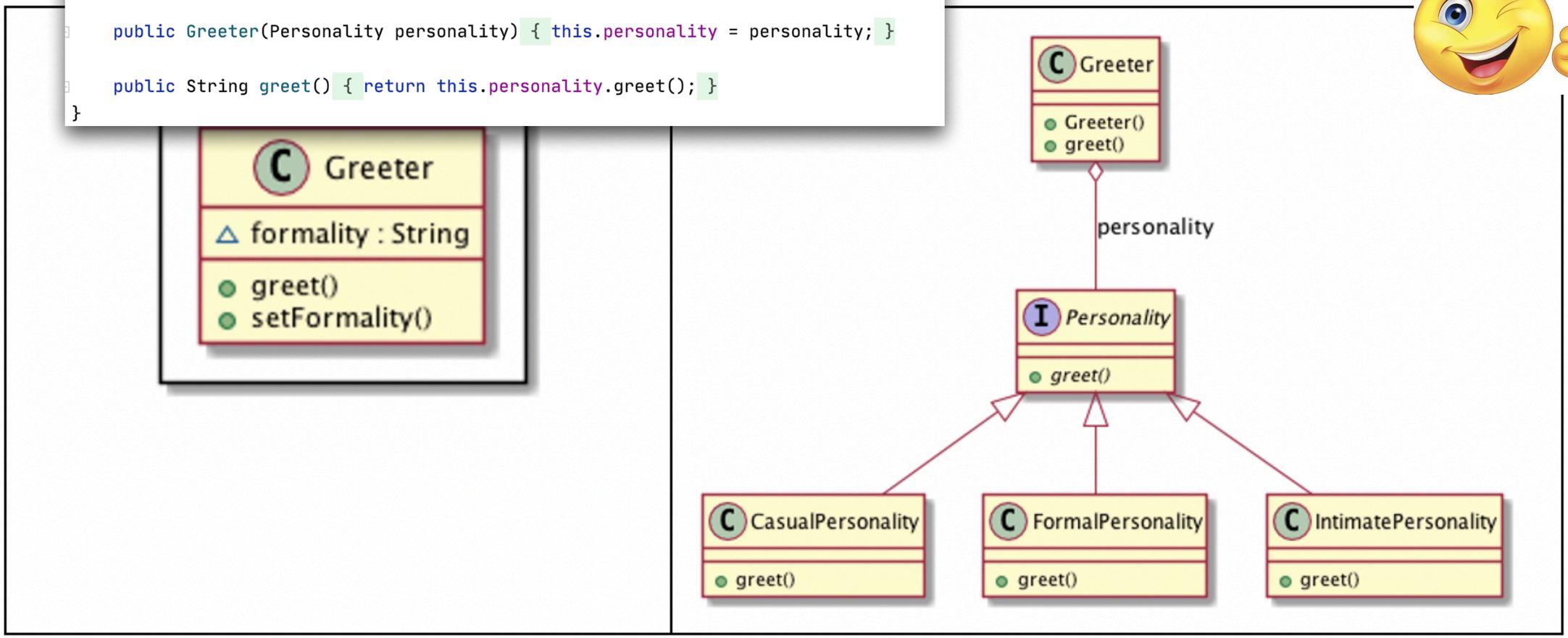

Què fa aquesta classe?

```java
public class Greeter {
    String formality;

    public String greet() {
        if (this.formality == "formal") {
            return "Good evening, sir.";
        }
        else if (this.formality == "casual") {
            return "Sup bro?";
        }
        else if (this.formality == "intimate") {
            return "Hello Darling!";
        }
        else {
            return "Hello.";
        }
    }

    public void setFormality(String formality) { this.formality = formality; }
}
```

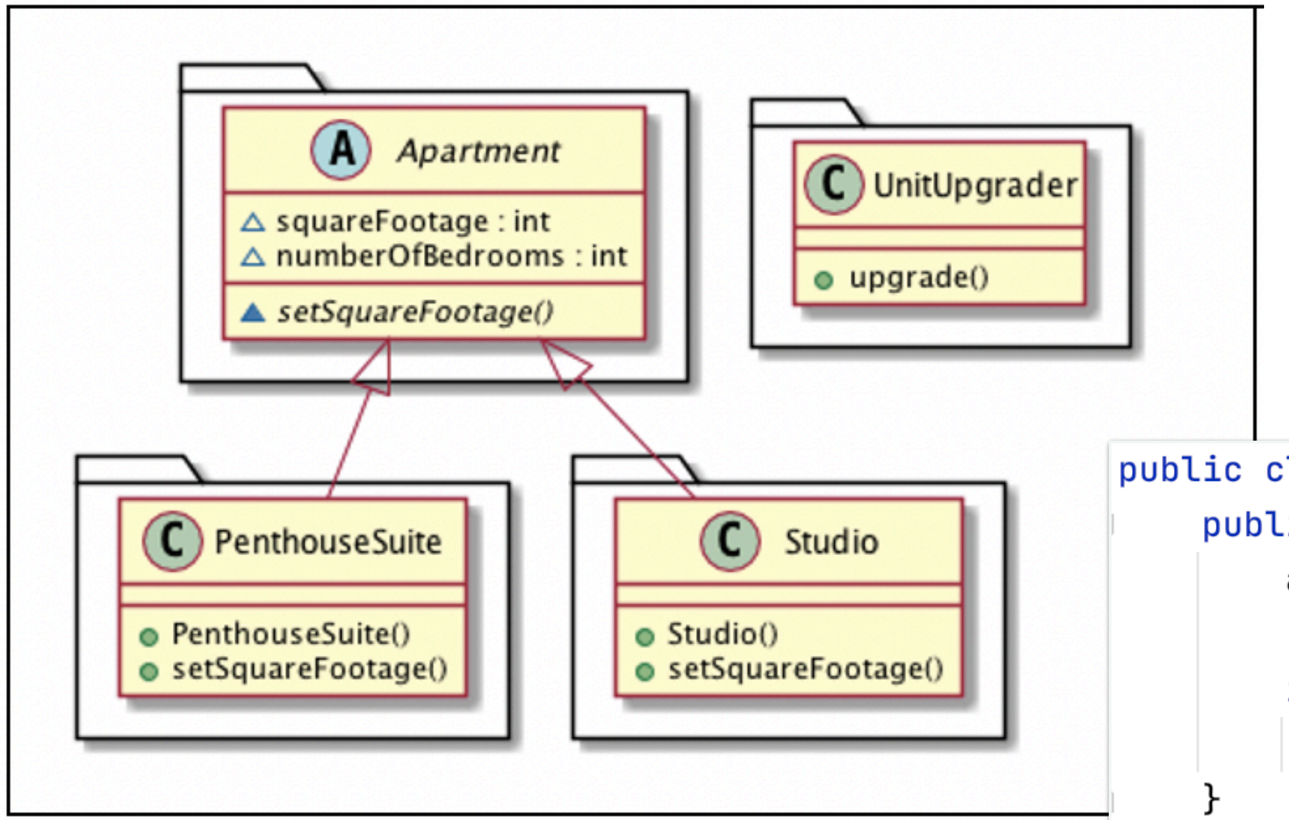
Open-Closed Principle???

# 1. Exploració dels projectes: Projecte 2

```java
public class Greeter {
    private Personality personality;

    public Greeter(Personality personality) { this.personality = personality; }

    public String greet() { return this.personality.greet(); }
}
```



# 1. Exploració dels projectes: Projecte 3

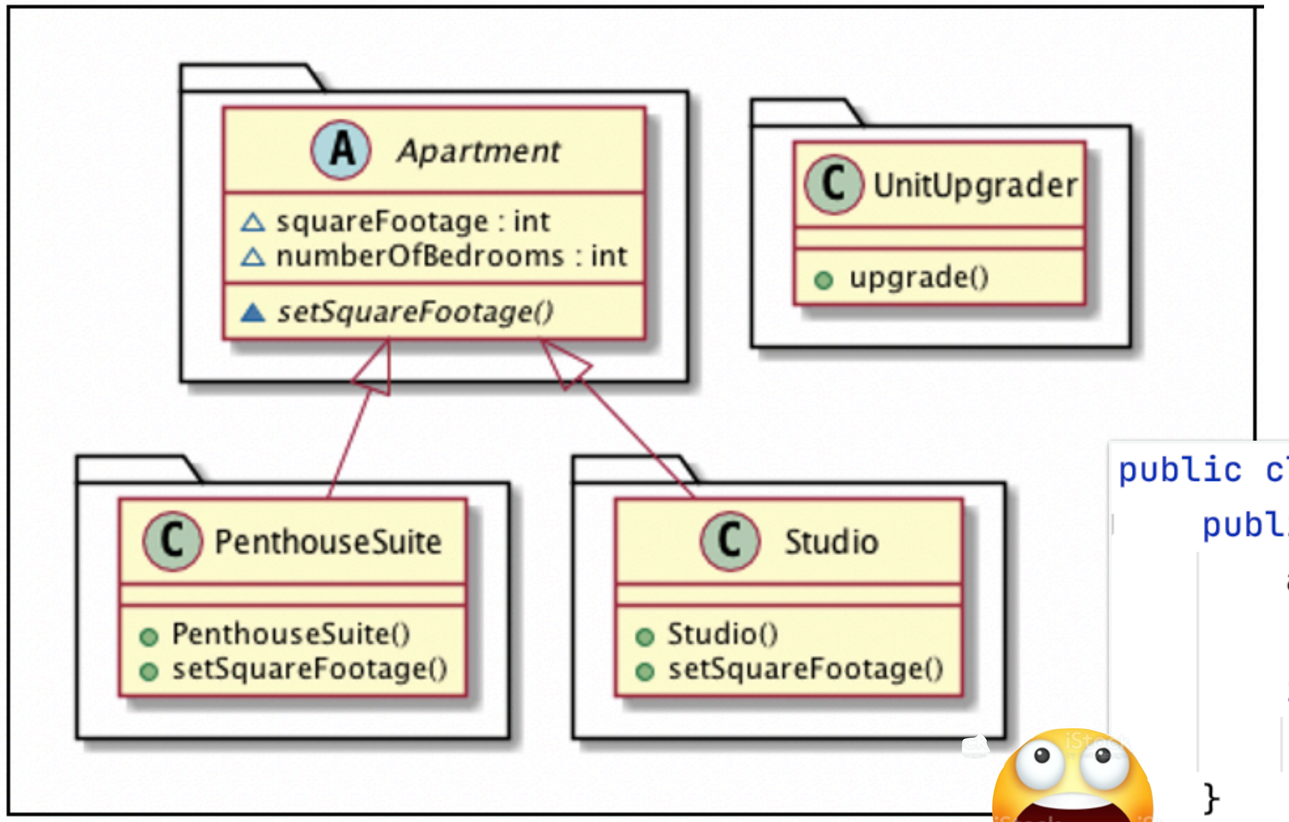



Una herència....

Tots els fills fan el que toca? I si vaig a fer un upgrade() del numero d'habitacions??

```
public class UnitUpgrader {
    public void upgrade(Apartment apartment) {
        apartment.squareFootage += 40;

        if (apartment.getClass() != Studio.class)
            apartment.numberOfBedrooms += 1;
    }
}
```

# 1. Exploració dels projectes: Projecte 3




Una herència....

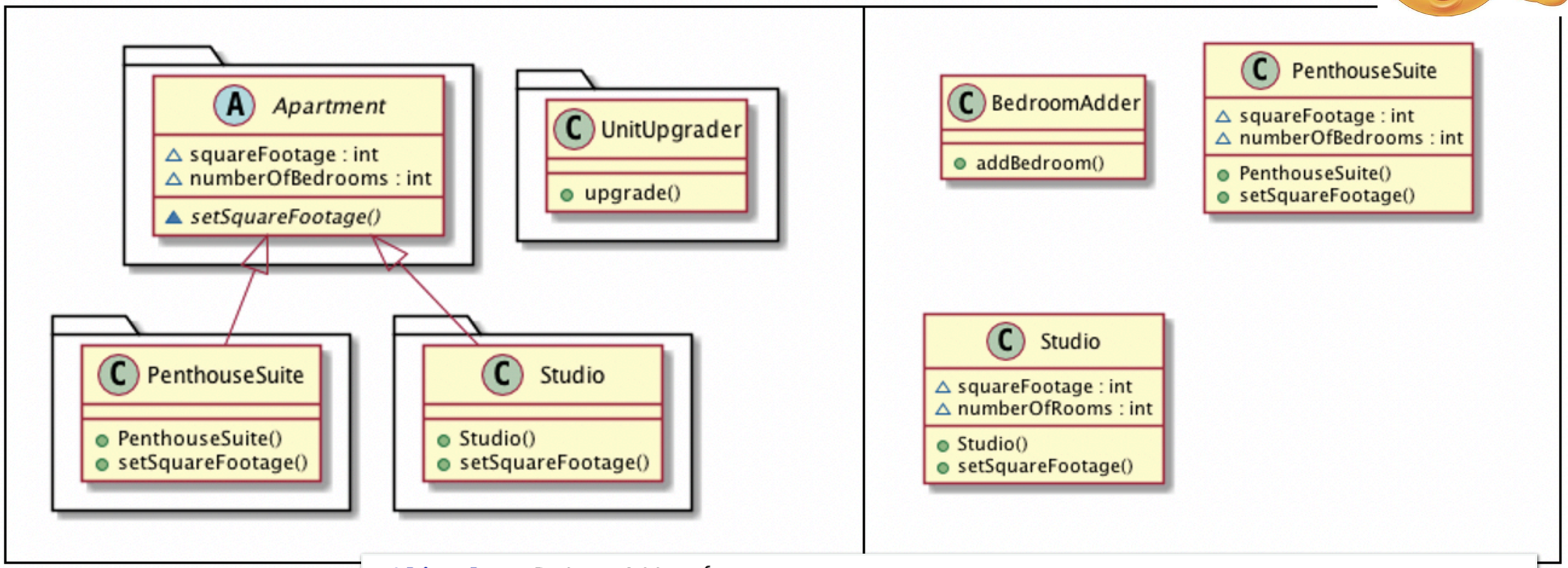
Tots els fills fan el que toca? I si vaig a fer un upgrade() del numero d'habitacions??

```java
public class UnitUpgrader {
    public void upgrade(Apartment apartment) {
        apartment.squareFootage += 40;

        if (apartment.getClass() != Studio.class)
            apartment.numberOfBedrooms += 1;
    }
```



**Liskov Substitution Principle???**

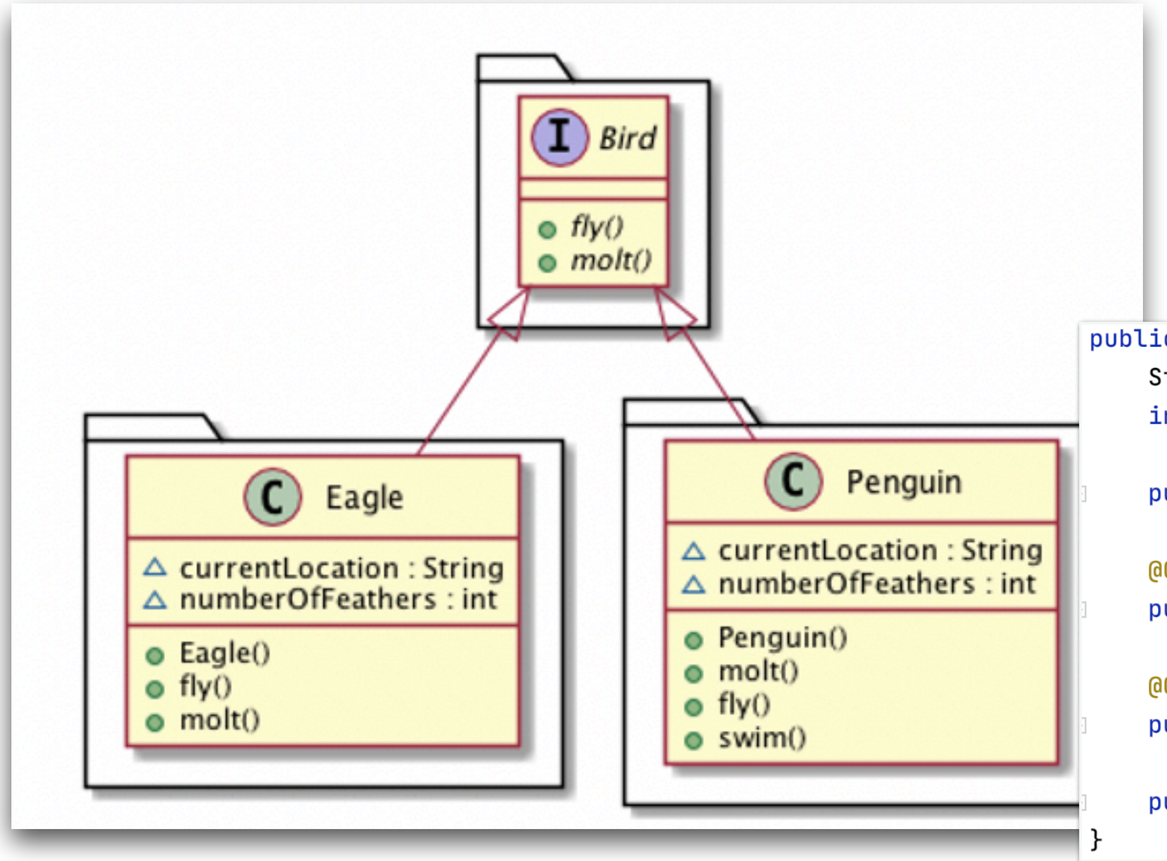
# 1. Exploració dels projectes: Projecte 3




```java
public class BedroomAdder {
    public void addBedroom(PenthouseSuite penthouse) { penthouse.numberOfBedrooms += 1; }
}
```

# 1. Exploració dels projectes: Projecte 4

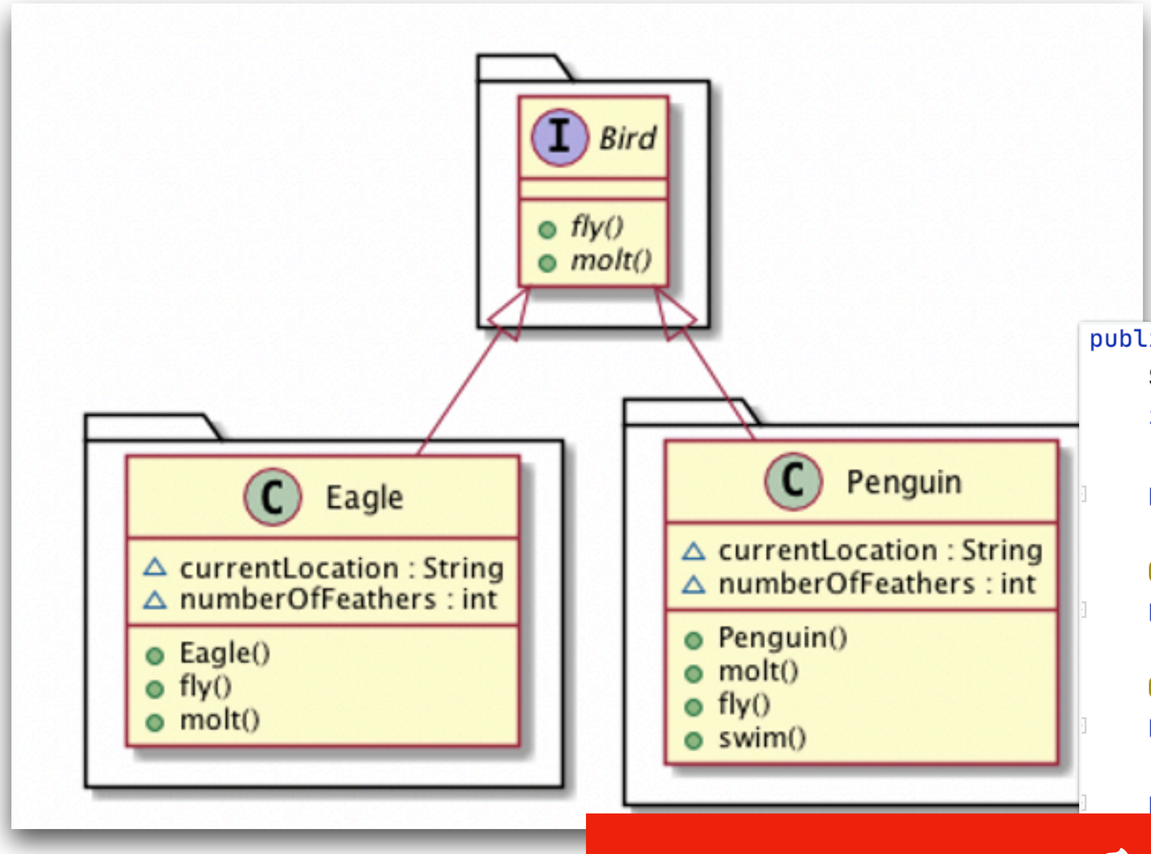
Interfície ?
Està sobrecarregada?
Com són les seves implementacions?




```java
public class Penguin implements Bird {
    String currentLocation;
    int numberOfFeathers;

    public Penguin(int initialFeatherCount) { this.numberOfFeathers = initialFeatherCount; }

    @Override
    public void molt() { this.numberOfFeathers -= 1; }

    @Override
    public void fly() { throw new UnsupportedOperationException(); }

    public void swim() { this.currentLocation = "in the water"; }
}
```

# 1. Exploració dels projectes: Projecte 4

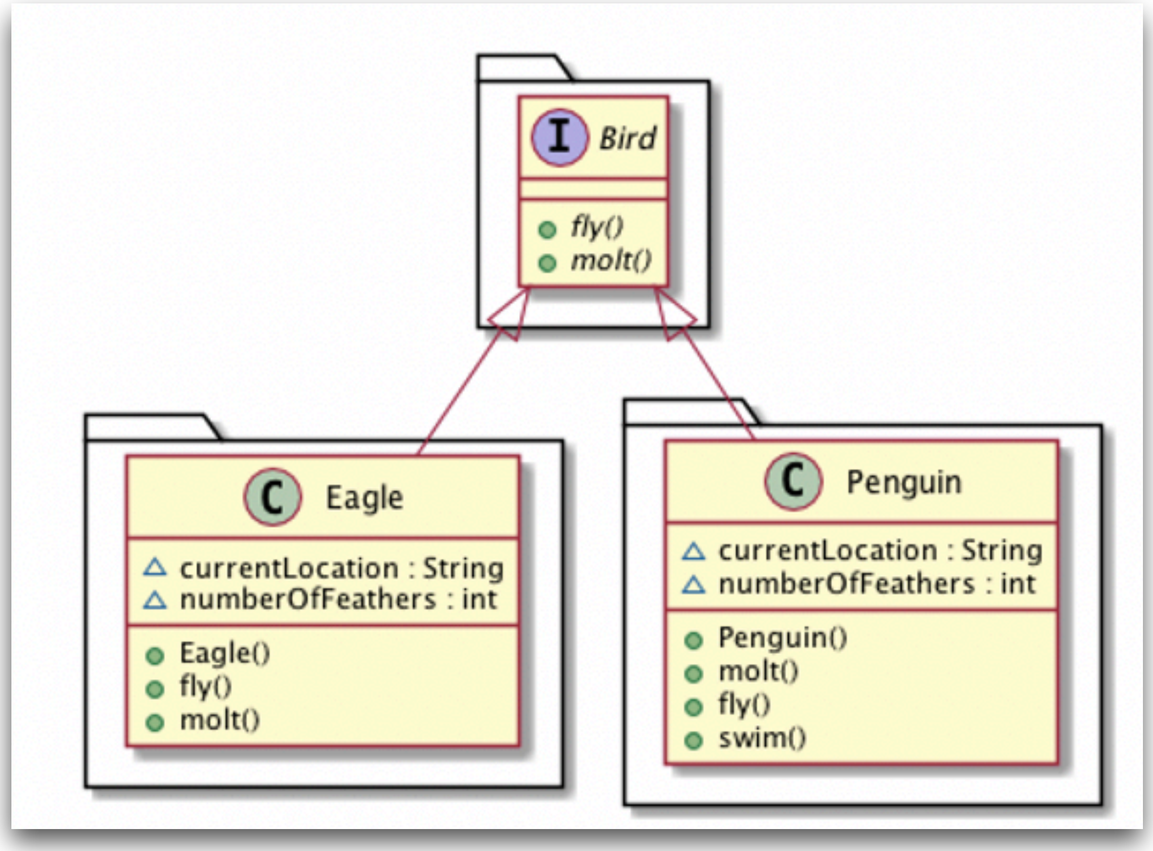
Interfície ?
Està sobrecarregada?
Com són les seves implementacions?

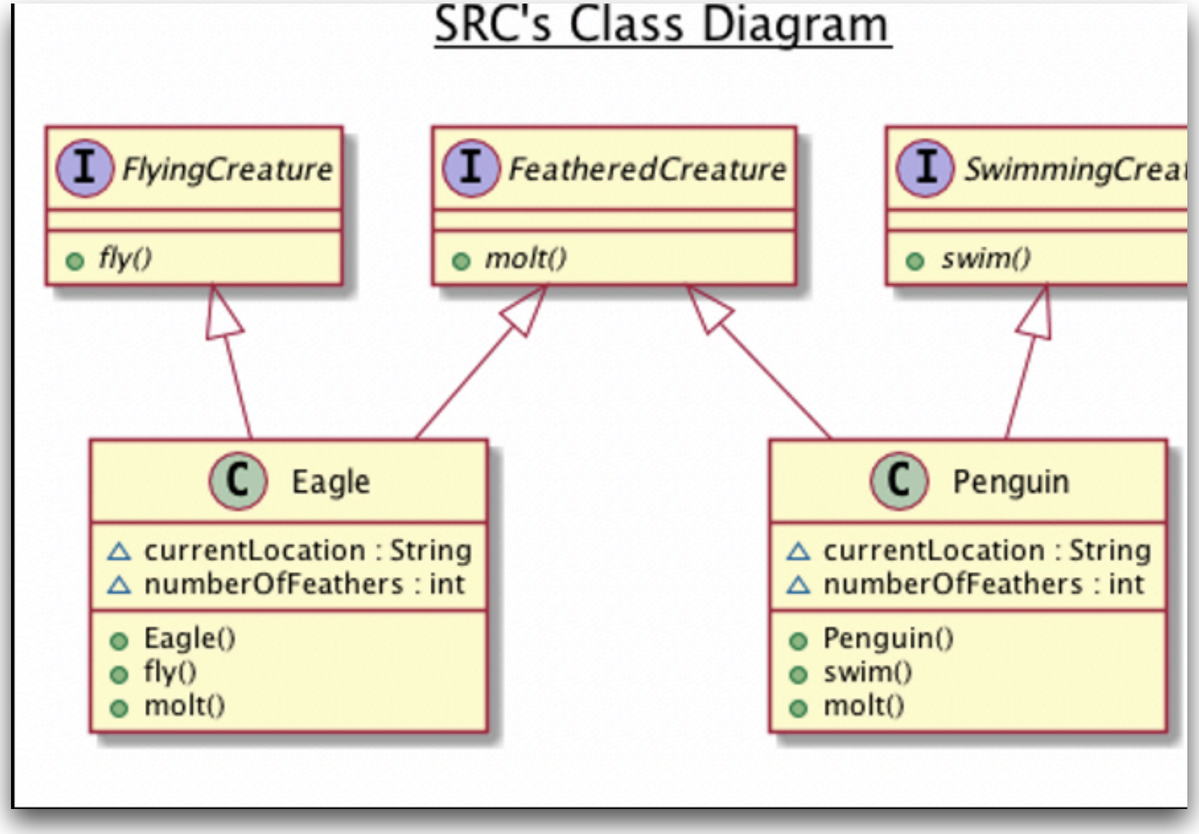



```java
public class Penguin implements Bird {
    String currentLocation;
    int numberOfFeathers;

    public Penguin(int initialFeatherCount) { this.numberOfFeathers = initialFeatherCount; }

    @Override
    public void molt() { this.numberOfFeathers -= 1; }

    @Override
    public void fly() { throw new UnsupportedOperationException(); }

    public void swim() { this.currentLocation = "in the water"; }
```

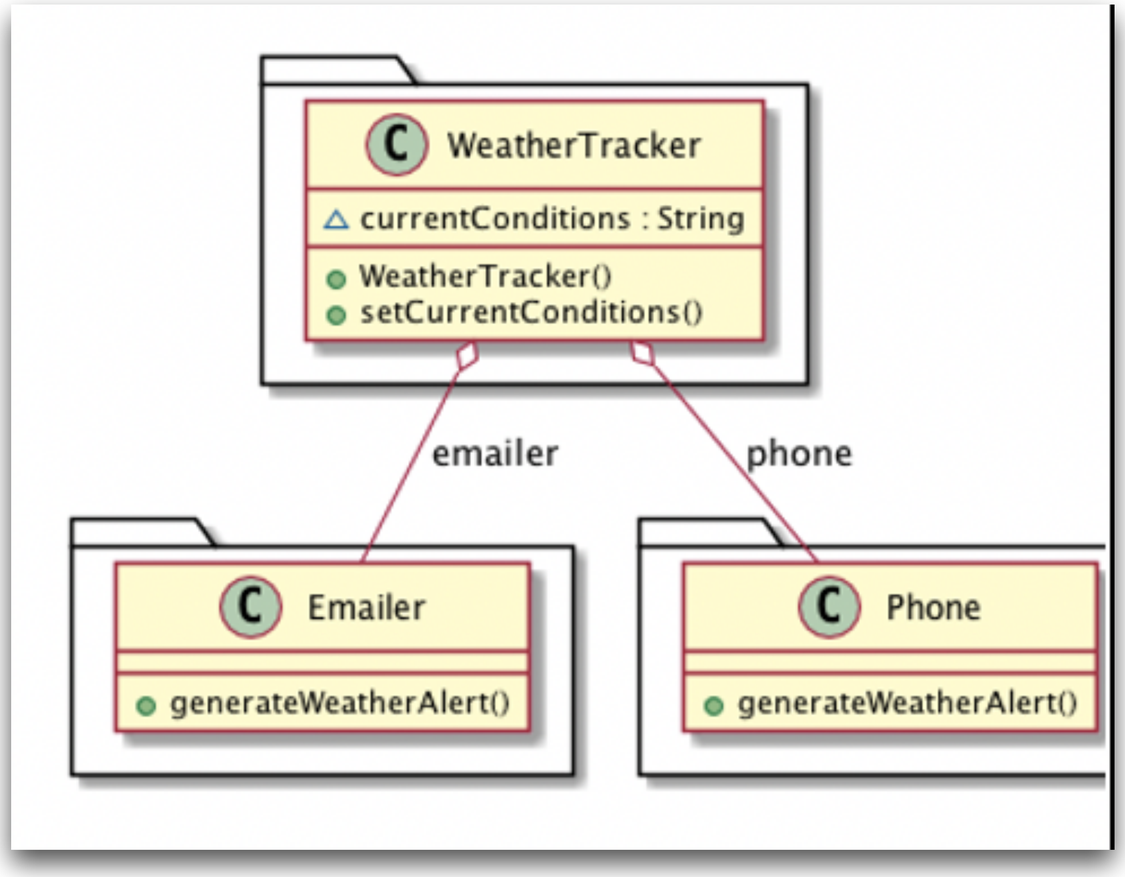
Interface Substitution Principle??

# 1. Exploració dels projectes: Projecte 4



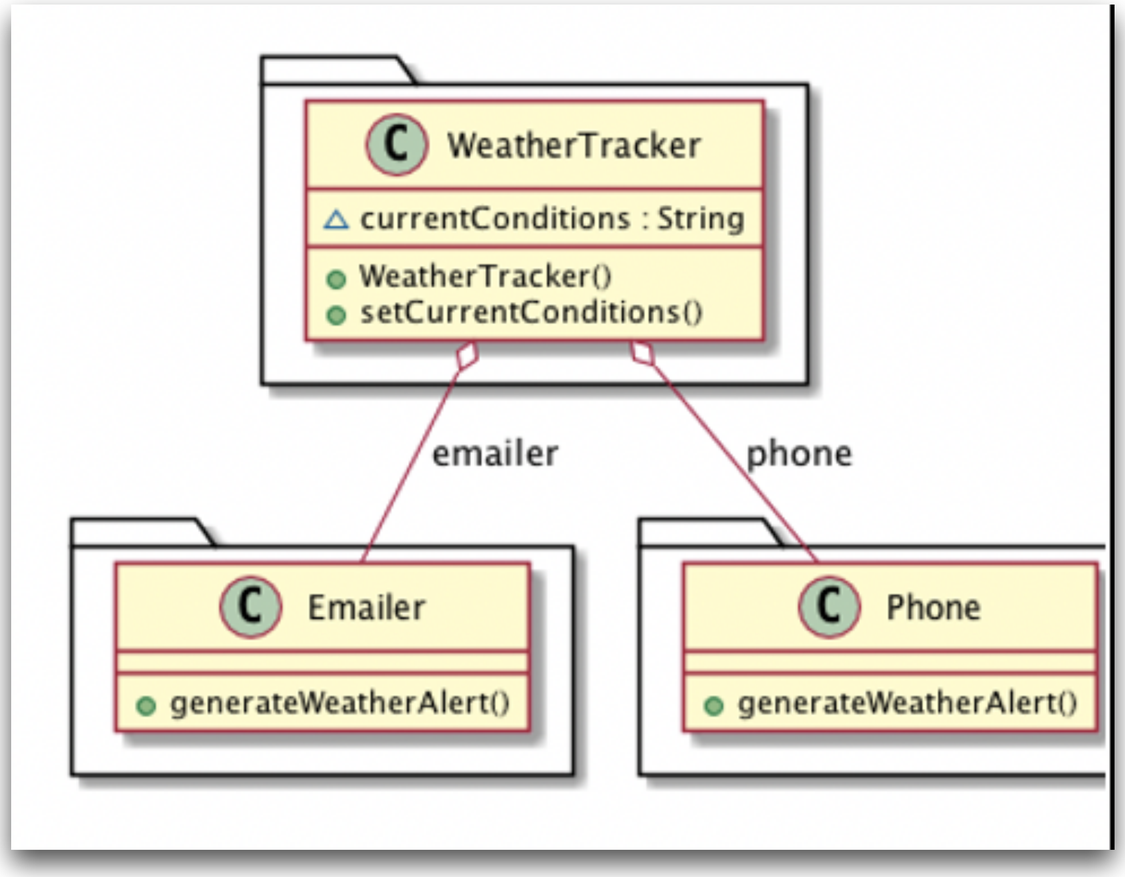
# 1. Exploració dels projectes: Projecte 5




WeatherTracker… dóna missatges per mail o per telèfon segons el temps que fa….

```
public class  WeatherTracker {
    String currentConditions;
    Phone phone;
    Emailer emailer;

    public WeatherTracker() {
        phone = new Phone();
        emailer = new Emailer();
    }

    public void setCurrentConditions(String weatherDescription) {
        this.currentConditions = weatherDescription;
        if (weatherDescription == "rainy") {
            String alert = phone.generateWeatherAlert(weatherDescription);
            System.out.print(alert);
        }
        if (weatherDescription == "sunny") {
            String alert = emailer.generateWeatherAlert(weatherDescription);
            System.out.print(alert);
        }
    }
}
```

# 1. Exploració dels projectes: Projecte 5

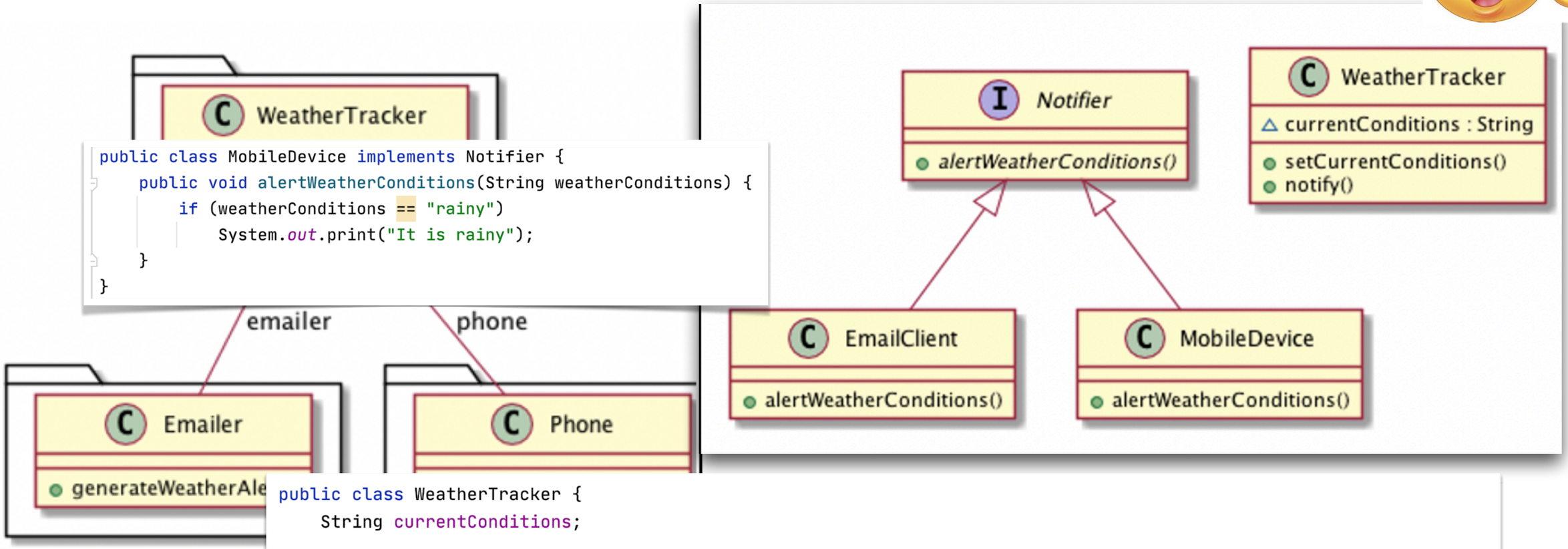



WeatherTracker… dóna missatges per mail o per telèfon segons el temps que fa….

```
public class  WeatherTracker {
    String currentConditions;
    Phone phone;
    Emailer emailer;

    public WeatherTracker() {
        phone = new Phone();
        emailer = new Emailer();
    }

    public void setCurrentConditions(String weatherDescription) {
        this.currentConditions = weatherDescription;
        if (weatherDescription == "rainy") {
            String alert = phone.generateWeatherAlert(weatherDescription);
            System.out.print(alert);
        }
        if (weatherDescription == "sunny") {
            String alert = emailer.generateWeatherAlert(weatherDescription);
            System.out.print(alert);
        }
    }
}
```

**Dependency Inversion??**

# 1. Exploració dels projectes: Projecte 5




```java
public class WeatherTracker {
    String currentConditions;

    public void setCurrentConditions(String weatherDescription) { this.currentConditions = weatherDescription; }

    public void notify(Notifier notifier) { notifier.alertWeatherConditions(currentConditions); }
}
```

# 2. Activitat exercici SOLID

1. Obre l'enunciat del problema Shape del Campus
2. Obre el projecte del Campus associat a aquest enunciat
3. Intenta seguir l'enunciat detectant les vulneracions SOLID (10 min)
4. Discutirem després  les vulneracions