Code smells

- Duplicated Code

- For determining the fullName of the Customer

- In the customer.withdraw method we can see similar code

- Possible refactoring: Extract method

https://refactoring.guru/uk/extract-method

Відокремлення методу

Також відомий як:
Extract Method

Проблема

У вас є фрагмент коду, який можна згрупувати.

```
void printOwing() {
  printBanner();

  // Print details.
  System.out.println("name: " + name);
  System.out.println("amount: " + getOuts
}
```

Рішення

Виділіть цей фрагмент в новий метод (чи функцію) і викличте його замість старого коду.

```
void printOwing() {
  printBanner();
  printDetails(getOutstanding());
}

void printDetails(double outstanding) {
  System.out.println("name: " + name);
  System.out.println("amount: " + outstan
}
```

- Long Method

- Customer.withdraw method has more than 5 lines of code

- Possible refactorings: Extract method

https://refactoring.guru/uk/extract-method

- Divergent Change

- Customer class has more than one reason to change. This is a violation of the SRP (Single Responsibility Principle)

- Possible refactoring: Extract class

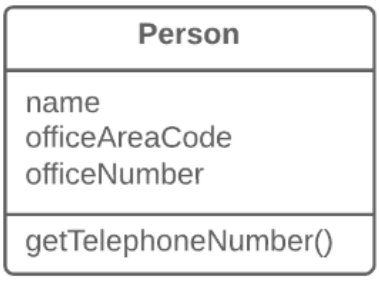
https://refactoring.guru/uk/extract-class

Відокремлення класу

Також відомий як:
Extract Class

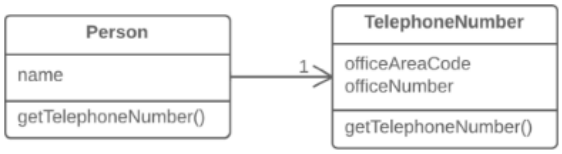
Проблема

Один клас працює за двох.



Рішення

Створіть новий клас, перемістіть в нього поля і методи, що відповідають за певну функціональність.



- Comments

- The comments in the customer.withdraw method

- Possible refactorings: make code speak for itself

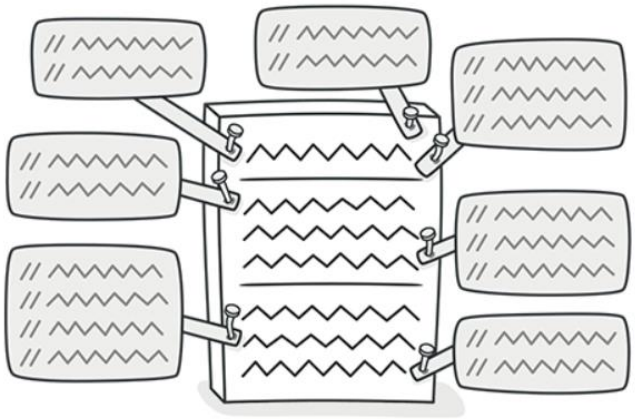
https://refactoring.guru/uk/smells/comments

Коментарі

Також відомий як:
Comments

Симптоми і ознаки

Метод містить безліч пояснювальних коментарів.



Рекомендований рефакторинг

Лікування

- Якщо коментар призначений для того, щоби пояснити складний вираз, можливо, цей вираз краще розбити на більш зрозумілі підвирази за допомогою **відокремлення змінної**.

- Якщо коментар пояснює цілий блок коду, можливо, цей блок можна витягнути в окремий метод за допомогою **відокремлення методу**. Назву нового методу вам, скоріш за все, підкаже сам коментар.

- Якщо метод вже виділений, але для пояснення його дії, як і раніше, потрібен коментар, дайте методу нову назву, що не вимагає коментаря. Використайте для цього **перейменування методу**.

- Якщо потрібно описати якість правила, що стосуються корректної роботи методи, спробуйте рефакторинг **введення твердження**.

- Switch Statements

- Customer.withdraw method has a some switches

- Does not respect OCP (Open Closed Principle)

- Every time we want to add a new customer type we have to modify the two switches

- Possible Refactoring: Replace conditional with polymorphism

https://refactoring.guru/uk/replace-conditional-with-polymorphism

Заміна умовного оператора поліморфізмом

Також відомий як:
Replace Conditional with Polymorphism

Проблема

У вас є умовний оператор, який, залежно від типу або властивостей об'єкта, виконує різні дії.

Рішення

Створіть підкласи, яким відповідають гілки умовного оператора. У них створіть спільний метод і перемістіть в нього код з відповідної гілки умовного оператора. Згодом проведіть заміну умовного оператора на виклик цього методу. Таким чином, потрібна реалізація вибиратиметься через поліморфізм залежно від класу об'єкта.

```
class Bird {
  // ...
  double getSpeed() {
    switch (type) {
      case EUROPEAN:
        return getBaseSpeed();
      case AFRICAN:
        return getBaseSpeed() - getLoadFactor() * numberOfCoco
      case NORWEGIAN_BLUE:
        return (isNailed) ? 0 : getBaseSpeed(voltage);
    }
    throw new RuntimeException("Should be unreachable");
  }
}
```

```
abstract class Bird {
  // ...
  abstract double getSpeed();
}

class European extends Bird {
  double getSpeed() {
    return getBaseSpeed();
  }
}
class African extends Bird {
  double getSpeed() {
    return getBaseSpeed() - getLoadFactor() * numberOfCoconuts
  }
}
class NorwegianBlue extends Bird {
  double getSpeed() {
    return (isNailed) ? 0 : getBaseSpeed(voltage);
  }
}

// Somewhere in client code
speed = bird.getSpeed();
```

- Feature Envy

- Customer class has a fascination over the methods of the Account class

- see the Customer.printCustomerAccount

- also Customer.withdraw

- high coupled with the Account class

- Possible refactorings: Move method

https://refactoring.guru/uk/move-method

Переміщення методу

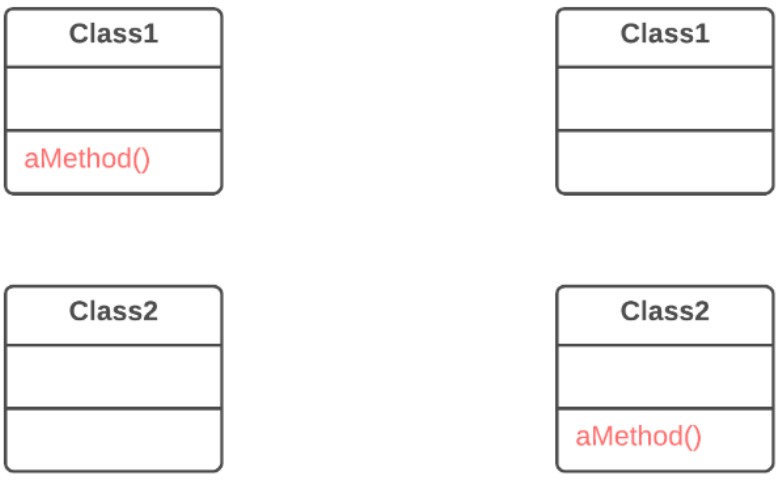
Також відомий як:
Move Method

Проблема

Метод використовується в іншому класі більше, ніж у власному.

Рішення

Створіть новий метод в класі, який використовує його більше інших, і перенесіть туди код із старого методу. Код оригінального методу перетворіть на звернення до нового методу в іншому класі або приберіть його взагалі.



- Inappropriate Intimacy

- Customer.printCustomerDaysOverdrawn only uses methods from the Account class

- Account.printCustomer only uses methods from the Customer class

- it's a bidirectional feature envy

- Possible refactorings: Move methods

https://refactoring.guru/uk/move-method

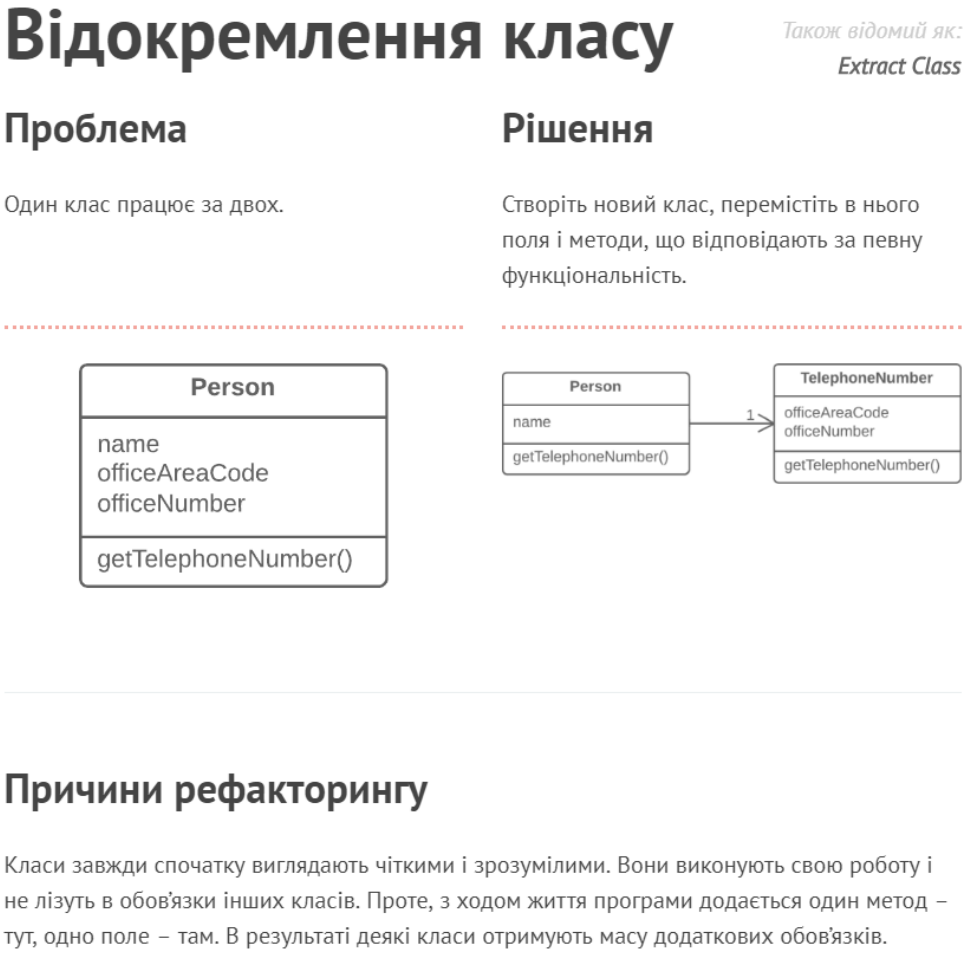
- Primitive Obsession

- Use a primitive type instead of a small object

- Account.money should be in it's own class

- Possible refactorings: Extract class, Introduce Parameter object

https://refactoring.guru/uk/extract-class



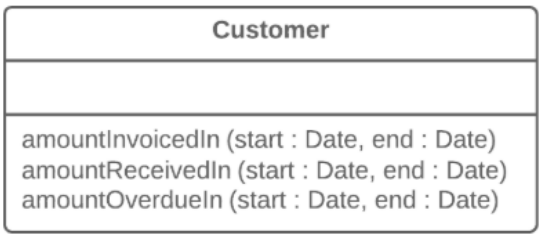
https://refactoring.guru/introduce-parameter-object

Заміна параметрів об'єктом

Також відомий як:
Introduce Parameter Object

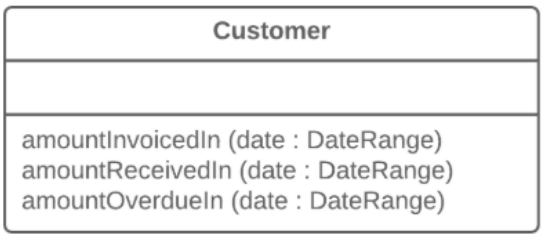
Проблема

У ваших методах зустрічається група параметрів, що повторюється.



Рішення

Замініть ці параметри об'єктом.



https://dzone.com/articles/practical-php-refactoring-35

- Data Clumps

- This is about groups of objects that are always grouped together

- Account.money and Account.currency should have a class

- Possible refactorings: Extract class, Introduce Parameter object

https://refactoring.guru/introduce-parameter-object

https://refactoring.guru/extract-class

- Lazy Class

- A class that's not doing enough

- AccountType

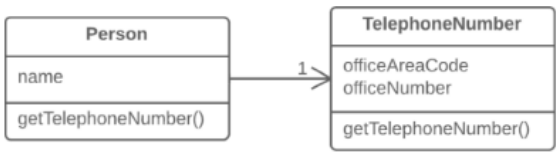
- Possible refactorings: Inline class

https://refactoring.guru/inline-class

Inline Class

Problem

A class does almost nothing and isn't responsible for anything, and no additional responsibilities are planned for it.



Solution

Move all features from the class to another one.