

Rozwiązanie zadania 3 z zestawu 3 z ”Projektowania obiektowego oprogramowania”

Mateusz Małowiecki

March 17, 2021

Polecenie

Dokonać analizy projektu obiektowego pod kątem zgodności klasy CashRegister z zasadą OCP. Zaproponować takie zmiany, które uczynią ją niezmienną a równocześnie rozszerzalną jeśli chodzi o możliwość implementowania różnych taryf podatkowych oraz drukowania paragonów z uwzględnieniem różnego porządkowania towarów (alfabetycznie, według kategorii itp.) Narysować diagramy klas przed i po zmianach. Zaimplementować działający kod dla przykładu przed i po zmianach demonstrując kilka różnych rozszerzeń.

```
public class TaxCalculator {
    public Decimal CalculateTax( Decimal Price ){
        return Price * 0.22
    }
}

public class Item {
    public Decimal Price { get { ... } }
    public string Name { get { ... } }
}

public class CashRegister {
    public TaxCalculator taxCalc = new TaxCalculator();
    public Decimal CalculatePrice( Item[] Items ) {
        Decimal _price = 0;
        foreach ( Item item in Items ) {
            _price += item.Price + taxCalc.CalculateTax( item.Price );
        }
        return _price;
    }
    public string PrintBill( Item[] Items ) {
        foreach ( var item in Items )
            Console.WriteLine( "towar {0} : cena {1} + podatek {2}", item.Name, item.Price, taxCalc.CalculateTax( item.Price ) );
    }
}
```

Ile klas docelowo powstanie z takiej jednej klasy? Dlaczego akurat tyle? Czy refaktoryzacja klasy naruszającej SRP oznacza automatycznie, że każda metoda powinna trafić do osobnej klasy?

Rozwiązanie

Dokonać analizy projektu obiektowego pod kątem zgodności z zasadą SRP

Korzystamy z testu odpowiedzialności:

```
+ReportPrinter drukuje raport sam  
?ReportPrinter formatuje dokument sam  
?ReportPrinter pobiera dane sam
```

Widzimy teraz, że o ile drukowanie raportu jest odpowiednią odpowiedzialnością dla klasy ReportPrinter, to dwa pozostałe zadania nie są związane z drukowaniem. Jest to zatem naruszenie zasady SRP (klasa ReportPrinter ma wiele niepowiązanych ze sobą odpowiedzialności).

Zaproponować zmiany

Rozdzielamy klasę ReportPrinter na klasy z pojedynczymi odpowiedzialnościami.

Narysować diagramy klas przed i po zmianach.

Diagram przed:

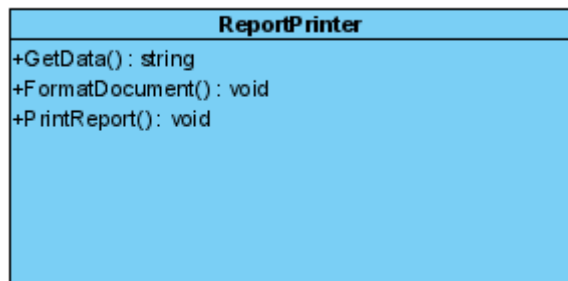
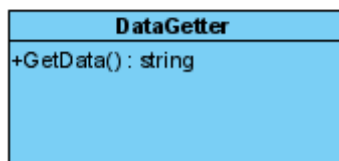
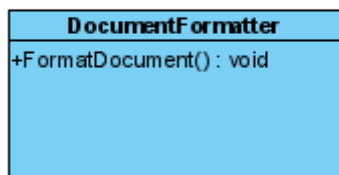


Diagram po:



Zaimplementować działający kod dla przykładu przed i po zmianach.

Kod znajduje się w plikach "POO_L3_Z2_After.cs" i "POO_L3_Z2_Before.cs". W pliku "POO_L3_Z2_Test.cs" znajdują się testy.

Ile klas docelowo powstanie z takiej jednej klasy? Dlaczego akurat tyle? Czy refaktoryzacja klasy naruszającej SRP oznacza automatycznie, że każda metoda powinna trafić do osobnej klasy?

Docelowo powstaną trzy klasy, gdyż klasa początkowa miała trzy różne odpowiedzialności. Refaktoryzacja klasy nie oznacza, że każda metoda trafi do osobnej klasy, tylko że każda odpowiedzialność powinna zostać przypisana pojedynczej klasie. Metody związane z jedną odpowiedzialnością nie powinny być rozdzielane pomiędzy wiele klas.