# Rozwiązanie zadania 3 z zestawu 3 z "Projektowania obiektowego oprogramowania"

Mateusz Małowiecki

20 marca 2021

## Polecenie

Dokonać analizy projektu obiektowego pod katem 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 ) );
        }
    }
}
```

# Rozwiązanie

## Dokonać analizy projektu obiektowego pod katem zgodności klasy CashRegister z zasadą OCP.

Klasa CashRegister łamie zasadę OCP, ponieważ nie jest otwarta na rozszerzenia. Korzysta ona z konkretnej implementacji TaxCalculator i nie ma możliwości, żeby mogła ona wyliczać podatek w inny sposób. Dodatkowo drukowanie paragonów jest zaimplementowane bezpośrednio w klasie CashRegister, w wyniku czego nie mamy wymienności implementacji drukowania. Jeszcze jednym problemem jest to, że klient jest uzależniony od konkretnej implementacji klasy Item, a powinien mieć możliwość dodać własną implementację tej klasy.

## 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.)

Proponowane zmiany:
1. Dodanie nowego interfejsu ITaxCalc z metodą CalculateTax (abstrakcja klasy TaxCalculator).
2. Dodanie nowego interfejsu IItem z polami Price i Name (abstrakcja klasy Item).
3. Dodanie interfejsu IItemSort odpowiedzialnego za sortowanie elementów względem zadanego kryterium.
4. W konstruktorze klasy CashRegister inicjujemy taxCalc obiektem klasy implementującej interfejs ITaxCalc.

## Narysować diagramy klas przed i po zmianach.

Diagram przed:

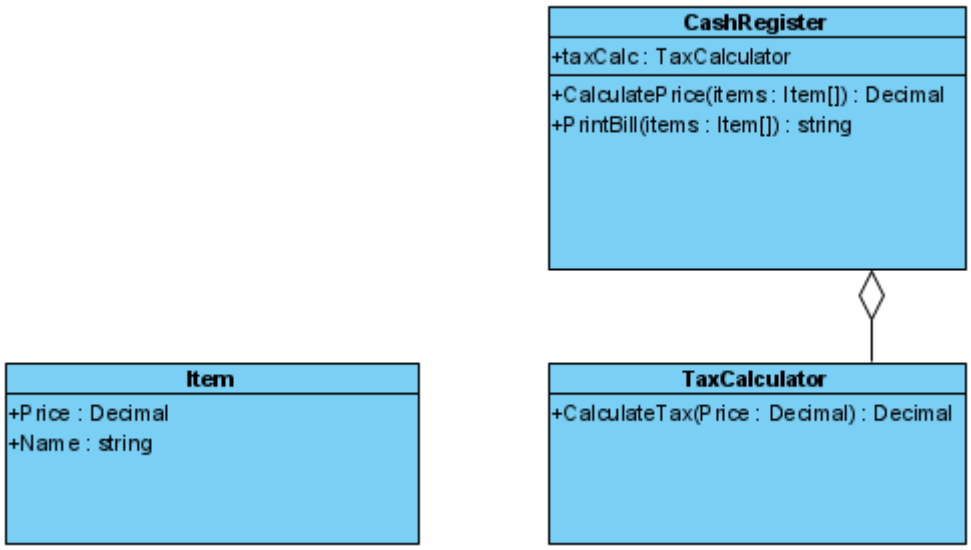
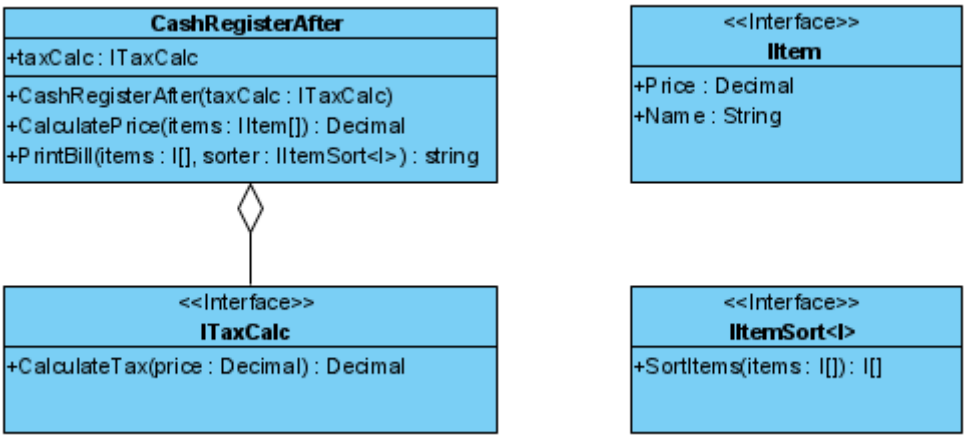



Diagram po:




## Zaimplementować działający kod dla przykładu przed i po zmianach demonstrując kilka różnych rozszerzeń.

Kod znajduje się w plikach "POO_L3_Z3_After.cs" i "POO_L3_Z3_Before.cs". W pliku "POO_L3_Z3_Test.cs" znajdują się testy.