**O (Open Closed Principle)**

1. **Co to za zasada?**

O - **O**pen-Closed Principle, po Polsku zasada otwarte-zamknięte, oznacza, że klasy powinny być otwarte na rozszerzenie, ale zamknięte na modyfikacje. Mówi ona tyle, że nasz kod powinien być tak skonstruowany w taki sposób, aby był zamknięty na edycję, ale był otwarty na rozszerzenia.

1. **Jaki jest jej sens istnienia?**

Zasada O istnieje, ponieważ dodaje elastyczności kodu i umożliwia edycję kodu bez konieczności ingerencji w istniejący już kod. Używanie tej zasady = unikanie konieczności modyfikowania istniejących klas = mniejsze ryzyko błędów.

1. **Przykład kodu, który nie spełnia tej zasady:**

public class ZarzadzaniePlatnosciami

{

public double ObliczCeneZamowienia(double cenaProduktu, string metodaPlatności)

{

double cenaKoncowa = cenaProduktu;

if (metodaPlatności == "karta")

{

cenaKoncowa += cenaProduktu \* 0.02;

}

else if (metodaPlatności == "przelew")

{

cenaKoncowa += cenaProduktu \* 0.01;

}

return cenaKoncowa;

}

}

(jeśli chcielibyśmy dodać nową metodę płatności, musielibyśmy zmodyfikować istniejącą klasę ZarządzaniePlatnościami)

1. **Poprawiony kod zgodnie z zasadą S:**

public abstract class MetodaPlatności

{

public abstract double ObliczOplate(double cenaProduktu);

}

public class Karta : MetodaPlatności

{

public override double ObliczOplatę(double cenaProduktu)

{

return cenaProduktu \* 0.02; //

}

}

public class Przelew : MetodaPlatności

{

public override double ObliczOplatę(double cenaProduktu)

{

return cenaProduktu \* 0.01;

}

}

1. **Przykład wzorca projektowego, który spełnia zasadę S**

Wzorzec dekoratora jest idealnym przykładem implementacji zasady O. Pozwala on na dynamiczne dodawanie nowych funkcji do istniejących obiektów, jednocześnie zachowując ich zamkniętość dla modyfikacji. Dzięki dekoratorowi możemy rozszerzać funkcjonalność klas bez dotykania pierwotnego kodu, co sprzyja utrzymaniu elastycznego i łatwego do rozbudowy systemu, zgodnie z zasadą OCP.

1. **Przykład kodu z życia**

System zarządzania parkingiem i pobierania opłat za typ pojazdu

public abstract class Pojazd

{

public abstract double Oplata();

}

public class Auto : Pojazd

{

public override double Oplata()

{

return 5.0;

}

}

public class Motocykl: Pojazd

{

public override double Oplata()

{

return 3.0;

}

}

public class ZarzadzanieOplatami

{

public double Oplata(Pojazd pojazd)

{

return pojazd.Oplata(); }

}

1. **Podsumowanie**

Zasada Open-Closed jest ważnym elementem programowania obiektowego, który promuje elastyczność, skalowalność i utrzymanie kodu. Jej przestrzeganie prowadzi do bardziej optymalnego i łatwego w utrzymaniu kodu. W dodatku zapobiega potencjalnym błędom

**Źródła:**

-<https://www.programistabyc.pl/solid-2-ocp/?ref=programistabyc.pl>

-[https://pl.wikipedia.org/wiki/Zasada\_otwarte-zamknięte](https://pl.wikipedia.org/wiki/Zasada_otwarte-zamkni%C4%99te)

-<https://www.modestprogrammer.pl/solid-open-closed-principle-ocp-wszystko-co-powinienes-wiedziec-o-zasadzie-otwarte-zamkniete>

- Lekcja