INSTYTUT TECHNICZNY

Państwowa Wyższa Szkoła Zawodowa W Nowym Sączu

Kierunek : Informatyka

Forma studiów : stacjonarne

**Sprawozdanie z przedmiotu :**

Programowanie zaawansowane

**Temat :**

Wzorce projektowe: dekorator

**Prowadzący:**

mgr inż. Nikodem Bulanda

**Wykonał :**

Damian Basiaga

Rok akademicki 2019/2020

Spis treści

[1. Wzorzec projektowy 4](#_Toc30632178)

[2. Wybrany wzorzec projektowy: Dekorator 5](#_Toc30632179)

[2.1. Kod programu 5](#_Toc30632180)

[2.2. Działanie programu 7](#_Toc30632181)

[Bibliografia 8](#_Toc30632182)

[Spis Rysunków 8](#_Toc30632183)

# Wzorzec projektowy

Wzorzec projektowy (ang. design pattern) – w inżynierii oprogramowania, uniwersalne, sprawdzone w praktyce rozwiązanie często pojawiających się, powtarzalnych problemów projektowych. Pokazuje powiązania i zależności pomiędzy klasami oraz obiektami i ułatwia tworzenie, modyfikację oraz pielęgnację kodu źródłowego. Jest opisem rozwiązania, a nie jego implementacją. Wzorce projektowe stosowane są w projektach wykorzystujących programowanie obiektowe. Stosowanie wzorców projektowych pozwala na pisanie lepszych, bardziej efektywnych, skalowalnych, łatwiej modyfikowalnych, mniej narażonych na błędy programów.

Wszystkie wzorce można podzielić na następujące rodziny:

* Kreacyjne (konstrukcyjne) – opisujące proces tworzenia nowych obiektów; ich zadaniem jest tworzenie, inicjalizacja oraz konfiguracja obiektów, klas oraz innych typów danych.
* Strukturalne – opisujące struktury powiązanych ze sobą obiektów.
* Czynnościowe – opisujące zachowanie i odpowiedzialność współpracujących ze sobą obiektów.

Wzorzec projektowy składa się z czterech podstawowych elementów:

* nazwy wzorca;
* problemu – opisuje sposoby rozpoznawania sytuacji, w których możemy zastosować dany wzorzec oraz warunki jakie muszą zostać spełnione, by jego zastosowanie miało sens;
* rozwiązania – opisuje elementy rozwiązania: ich relacje, powiązania oraz obowiązki, zawiera także wskazówki implementacyjne dla różnych technologii;
* konsekwencji – zestawienie wad i zalet stosowania wzorca, uwzględniające informacje o jego brakach oraz kosztach rozwoju i utrzymania systemu wykorzystującego dany wzorzec.

# Wybrany wzorzec projektowy: Dekorator

Dekorator to jeden ze strukturalnych wzorców projektowych, dzięki któremu można wykorzystać kompozycję w alternatywie do dziedziczenia w celu rozszerzenia zachowania klasy. W przypadku dziedziczenia klasa rozszerzana jest w trakcie kompilacji w przeciwieństwie do dekoratorów, które rozszerzają klasy w trakcie działania programu. Dekorator dynamicznie dołącza dodatkowe obowiązki do obiektu. Wzorzec ten udostępnia alternatywny elastyczny sposób tworzenia podklas o wzbogaconych funkcjach (Gamma, 2010).

Zaletą wykorzystania tego wzorca jest bardzo duża elastyczność poprzez rozbicie funkcjonalności na wiele mniejszych klas, które mogą dynamicznie zmieniać działanie klasy bazowej. Wadą natomiast jest rozbicie projektu na małe klasy, które często są do siebie bardzo podobne.

## Kod programu

#include <iostream>

#include <string>

using namespace std;

class Laptop { /\* wersja podstawowa laptopa \*/

protected:

string laptop;

double wartosc;

public:

Laptop(){

laptop = "Cena laptopa w wersji podstawowej";

wartosc=0;

}

virtual string about(){

return laptop;

}

virtual double cena(){

return wartosc;

}

};

class Dekorator : public Laptop {

public:

virtual string about(){

return "dodatek";

}

};

class Acer : public Laptop { /\* przykładowe marki laptopów \*/

public: Acer():Laptop(){

laptop = "Acer";

wartosc=2000;

}

};

class Dell : public Laptop {

public: Dell():Laptop(){

laptop = "Dell";

wartosc=2300;

}

};

class Dysk : public Dekorator {

Laptop \*com;

public: Dysk(Laptop \*laptop):Dekorator(){

com = laptop;

}

string about(){

return com->about() + " z dyskiem SSD 1Tb";

}

double cena(){

return com->cena()+700;

}

};

class RAM : public Dekorator {

Laptop \*com;

public: RAM(Laptop \*laptop):Dekorator(){

com = laptop;

}

string about()

{

return com->about() + " z RAM 8Gb";

}

double cena()

{

return com->cena() + 300;

}

};

int main(){

Laptop \*s1 = new Acer();

Laptop \*s2 = new Dell();

cout<<"\nCena laptopa w wersji podstawowej"<<endl;

cout<<s1->about()<<" "<<s1->cena()<<endl;

cout<<s2->about()<<" "<<s2->cena()<<endl;

s1 = new Dysk(s1);

s2 = new Dysk(s2);

cout<<"\nZ Dyskiem SSD 1Tb"<<endl;

cout<<s1->about()<<" "<<s1->cena()<<endl;

cout<<s2->about()<<" "<<s2->cena()<<endl;

s1 = new RAM(s1);

s2 = new RAM(s2);

cout<<"\nZ RAM 8Gb"<<endl;

cout<<s1->about()<<" "<<s1->cena()<<endl;

cout<<s2->about()<<" "<<s2->cena()<<endl;

cout<<"\nCena pelnego wyposazenia laptopa"<<endl;

Laptop \*s3 = new RAM( new Dysk( new Acer()));

cout<<s3->about()<<" "<<s3->cena()<<endl;

Laptop \*s4 = new RAM( new Dysk( new Dell()));

cout<<s4->about()<<" "<<s4->cena()<<endl;

system("PAUSE");

return 0;

}

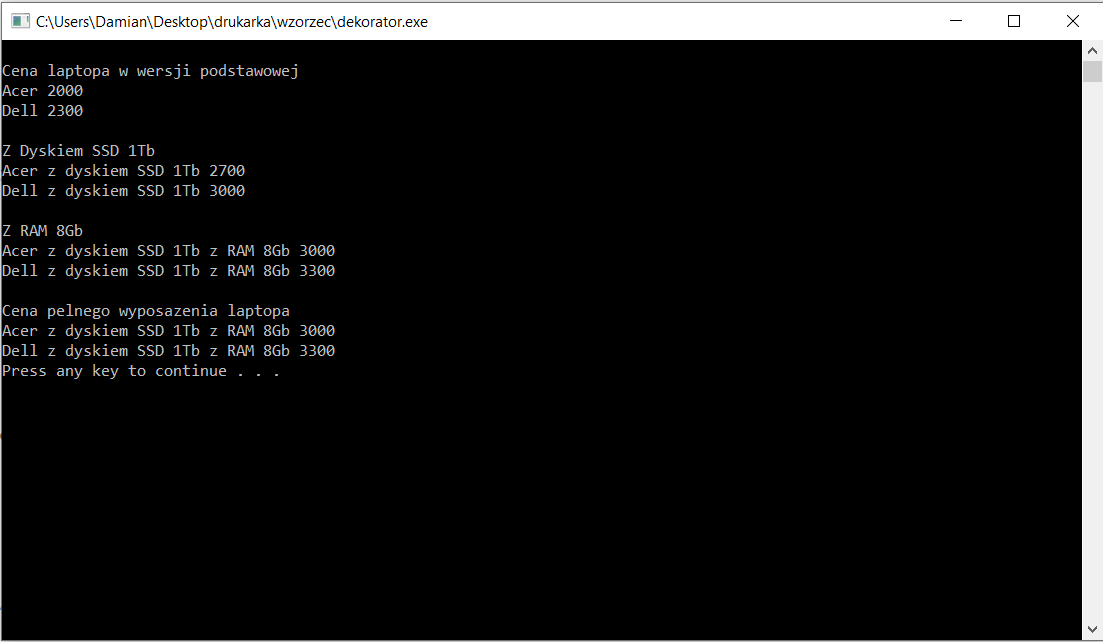
## Działanie programu

Zostały utworzone dwa obiekty, którymi są laptopy: Dell i Acer. Następnie został pokazany ich opis z wartością początkową ceny.

Obiekty zostały rozszerzone o dysk, dzięki konstruktorowi klasy Dysk. Do klasy tej został przekazany obiekt, który będziemy dekorowany. Przeciążona metoda about wywołuje w sobie metodę oraz dodaje do niej informację, o co laptop ma być wzbogacony.

Cena podstawowej wersji laptopa uległa zwiększeniu, o koszt dysku. Obiekty s1 i s2 zostały wzbogacone.

W sposób podobny realizowane jest dekorowanie obiektów o koszt pamięci RAM. Na koniec zostają utworzone obiekty s3 i s4, które są już udekorowane. Na rysunku 1 został pokazany rezultat kompilacji kodu.



*Rysunek 1.* Efekt kompilacji omawianego kodu

*Źródło: opracowanie własne na podstawie www.algorytm.org, 2020*

# Bibliografia

**Książki**

Gamma, E. (2010). *Wzorce projektowe.* Gliwice: Helion S.A.

**Strony internetowe**

www.lukasz-socha.pl/php/wzorce-projektowe-spis-tresci/

www.kobietydokodu.pl/21-wzorce-projektowe/

www.pl.wikipedia.org/wiki/Wzorzec\_projektowy\_(informatyka)

www.krzysztofjelonek.net/wzorce-projektowe-dekoratory/

www.algorytm.org/wzorce-projektowe/dekorator-decorator.html

# Spis Rysunków

[*Rysunek 1.* Efekt kompilacji omawianego kodu 8](#_Toc30632087)