Zaawansowane programowanie obiektowe - wykład 3

dr Piotr Jastrzębski

Polimorfizm

Definicja polimorfizmu wg wikipedii

Polimorfizm (z gr. wielopostaciowość) – mechanizmy pozwalające programiście używać wartości, zmiennych i podprogramów na kilka różnych sposobów. Inaczej mówiąc jest to możliwość wyabstrahowania wyrażeń od konkretnych typów.

Polimorfizm w C#

- statyczny
  - przeciążenie funkcji
  - przeciążenie operatorów
- dynamiczny
  - funkcje wirtualne
  - funkcje abstrakcyjne

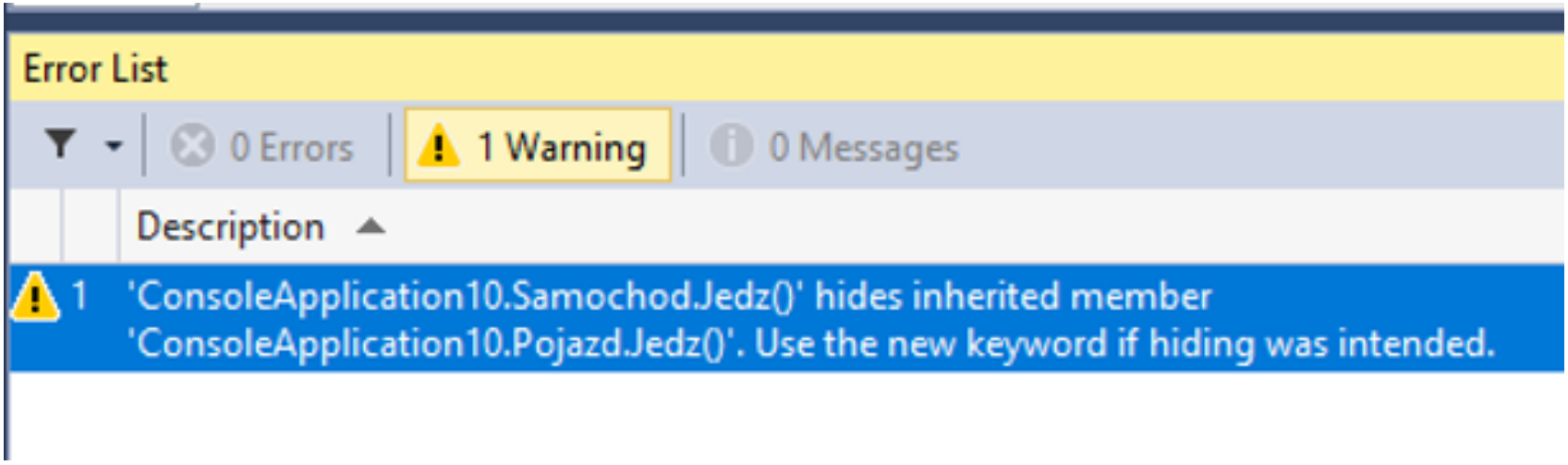
Napisanie metody - to nie jest polimorfizm

```
class Pojazd
{
    public void Jedz()
    {
        Console.WriteLine("Jadę pojazdem");
    }
}
class Samochod: Pojazd
{
    public void Jedz()
    {
        Console.WriteLine("Jadę samochodem");
    }
}
```

```
Pojazd p1 = new Pojazd();
p1.Jedz();
Samochod s1 = new Samochod();
s1.Jedz();
Pojazd sp1 = new Samochod();
sp1.Jedz();
```

Wykonanie kodu kończy się ostrzeżeniem kompilatora.

Podpowiedź sugeruje użycie `new` (czyli nadpisanie metody), jednak takie rozwiązanie nie jest najlepsze - nie daje elastyczności kodu.



Metoda wirtualna

- oznaczona słowem kluczowym `virtual`
- włącza mechanizm polimorfizmu dynamicznego
- tworzymy ją w klasie bazowej
- w klasach pochodnych przesłaniamy metody wirtualne za pomocą słowa kluczowego `override`
- przesłonięcie nie jest obowiązkowe, w razie jego braku zostanie wywołana metoda klasy bazowej
- metody statyczne ani prywatne nie mogą być wirtualne
- metody, które nie są wirtualne, nie można przesłonić za pomocą `override`

```
class Pojazd
{
    public virtual void Jedz()
    {
        Console.WriteLine("Jadę pojazdem");
    }
}

class Samochod: Pojazd
{
    public override void Jedz()
    {
        Console.WriteLine("Jadę samochodem");
    }
}
```

```
Pojazd p1 = new Pojazd();
p1.Jedz();
Samochod s1 = new Samochod();
s1.Jedz();
Pojazd sp1 = new Samochod();
sp1.Jedz();
```

Po co taka konstrukcja?

- dajemy informację dla innej osoby zajmującej się kodem
- przy tworzeniu klasy potomnych mamy elastyczność: możemy zostawić to jak jest w klasie bazowej lub przesłonić

Metody wirtualne w klasie `Object`

- w C# każda klasa dziedziczy niejawnie z klasy `Object`

```
public class Object
{
    /* składowe wirtualne */
    virtual public bool Equals(object o);
    virtual protected void Finalize();
    virtual public string ToString();
    /* itp.. */
}
```

Metody abstrakcyjne

- poprzedzone słowem kluczowym `abstract`
- zdefiniowane w klasie bazowej
- nie zawierają ciała metody (podobnie jak przy interfejsach)
- mogą być zadeklarowane tylko w klasie abstrakcyjnej (poprzedzonej słowem `abstract`)
- nie możemy stworzyć egzemplarza (obiektu) klasy abstrakcyjnej (podobnie jak przy interfejsach), ale możemy dziedziczyć po klasie abstrakcyjnej
- klasa abstrakcyjna może posiadać zwykłe metody (z implementacją)
- klasa pochodna do klasy abstrakcyjnej musi przesłonić wszystkie metody abstrakcyjne

```
abstract class Pojazd
{
    public abstract void Jedz();
}

class Samochod: Pojazd
{
    public override void Jedz()
    {
        Console.WriteLine("Jadę samochodem");
    }
}
```

Wzorce projektowe

Wzorce projektowe

Wzorzec projektowy (ang. design pattern) – 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.

Elementy wzorca projektowego

Wzorzec projektowy składa się z czterech podstawowych elementów:

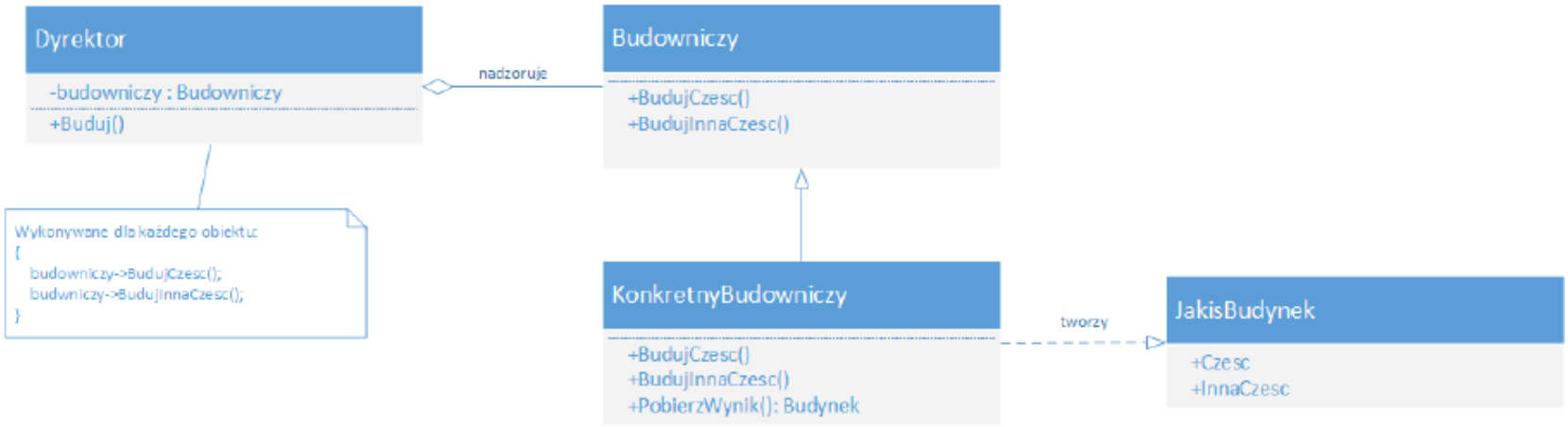
- nazwy wzorca;
- problemu – opisuje sposoby rozpoznawania sytuacji, w których możemy zastosować dany wzorzec oraz warunki jakie muszą zostać spełnione, by jego zastosowanie miało sens;
- rozwiązania – opisuje elementy rozwiązania: ich relacje, powiązania oraz obowiązki, zawiera także wskazówki implementacyjne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

Wzorce kreacyjne

Wzorce konstrukcyjne (kreacyjne) pozwalają w sposób abstrakcyjny tworzyć i konfigurować obiekty w celu ich wielokrotnego użycia i zachowania niezależności systemu od sposobu ich tworzenia.

Budowniczy

- cel: rozdzielenie sposobu tworzenia obiektów od ich reprezentacji
- proces tworzenia obiektu podzielony jest na kilka mniejszych etapów a każdy z tych etapów może być implementowany na wiele sposobów
- zastosowanie:
  - węzły XMLa
  - konwertowanie tekstu, zdjęć, filmów z jednego formatu na drugi.



Rysunek 1: Diagram UML Budowniczego

Kod do analizy: https://github.com/pjastr/ZPO_programy/tree/master/Budowniczy/Budowniczy