Interfejsy i klasy abstrakcyjne

Przegląd zagadnień

- Klasa abstrakcyjna
- Metody abstrakcyjne
- Słowo kluczowe sealed
- Pojęcie interfejsu
- Jawna implementacja interfejsu
- Konwersje

Klasa abstrakcyjna

- Nie może być utworzony obiektu tej klasy
- Słowo kluczowe abstract
- Konwencja – do nazwy dodajemy przyrostek Base

```
abstract class ABase {
}
class B : ABase {
}
int Main() {
   // ABase pA = new ABase(); // błąd: ABase
                              // jest abstrakcyjna
   ABase b = new B();         // ok
}
```

Metody abstrakcyjne

```
public abstract void f();
```

- Przez domyślność wirtualna
- Może być definiowana tylko w klasach abstrakcyjnych
- Nie może być prywatna

Słowo kluczowe sealed

- W przypadku funkcji oznacza, że jej nie można nadpisać

```
class Bazowa{
   public virtual void f(){...};
}
class Pochodna{
   public sealed override void f()
   {...};
}
```

- W przypadku klasy oznacza, że po niej nie można dziedziczyć

```
sealed class  Nazwa {...};
```

  - wszystkie metody domyślnie sealed

Pojęcie interfejsu

- Zbiór funkcji pod wspólną nazwą
- Słowo kluczowe interface
- Same deklaracje - brak implementacji
- Brak pól
- Wszystkie składowe publiczne (przez domyślność)
- Klasa (interfejs) może implementować wiele interfejsów
- Klasa musi implementować wszystkie metody swoich bazowych interfejsów

Interface: przykład użycia

```
interface INazwa{
   void f();
   String Wlasciwosc{ get; set; }
   event EventHandler  zdarzenie;
   //int i; - błąd
}
```

```
class Klasa : INazwa {
   String s;
   public virtual void f(){...}
   public virtual String Wlasciwosc {
      get {return s;}
      set {s = value;}
   }
    public virtual event
     EventHandler zdarzeznie;
}
//Słowo virtual jest opcjonalne
```

Jawna imaplementacja interfejsu

```
interface Interfejs1 {
  void f();
}
interface Interfejs2 {
  void f();
}
class Klasa : Interfejs1, Interfejs2 {
   public void f() {
      Console.WriteLine(
        "Implementacja w sposób niejawny");
  }
    void Interfejs2.f() {
      Console.WriteLine(
       "Implementacja w sposób jawny");
   }
}
```

Interfejsy: ciekawostka

```
interface IInterfejs1 {
     string f( );
}

abstract class A {
  public virtual string f( ) { ... }
}

class B: A, IInterfejs1 {
    //public override string f( ) { ... }
}
```

Konwersje

- Operator ( )
  - `pochodna = (Pochodna)bazowa;`
  - rzucany jest wyjątek
- is
  - `if(bazowa is Pochodna){ ...}`
- as
  - ```
    Pochodna pochodna =
                        bazowa as Pochodna;
    if(pochodna != null)
    {...}
    ```
    ```