Programowanie obiektowe w C# - 2 skrypt:

Zajęcia laboratoryjne przeprowadzane są na komputerach z systemem operacyjnym Windows 7 z wykorzystaniem oprogramowania Visual Studio 2010 w wersji Ultimate. Poniżej omówiono kilka często wykorzystywanych skrótów:
VS – Visual Studio
LPM – Lewy przycisk myszy
PPM – Prawy przycisk myszy

Zagadnienie 1: Klasy abstrakcyjne
Klasy abstrakcyjne to klasy, które definiuje się z wykorzystaniem słowa kluczowego `abstract`. Służą one do tworzenia klas bazowych, czyli takich z których później inne klasy będą dziedziczyć. Istotną własnością klas abstrakcyjnych jest brak możliwości tworzenia ich instancji. Służą one głównie do tworzenia grup klas o podobnej aczkolwiek nie identycznej funkcjonalności. Wówczas klasa abstrakcyjna służy do definiowania rzeczy abstrakcyjnych, bardzo ogólnych natomiast klasy po niej dziedziczące dodają do tej definicję szczegółów. Korzystanie z klas abstrakcyjnych nie jest konieczne, natomiast zwiększa czytelność kodu. W praktyce klasy abstrakcyjne implementują jedynie część funkcjonalności lub nie implementują jej wcale.

Przykład:
```
abstract class Animal
{
	public virtual string Describe()
	{
		return "Not much is known about this animal!";
	}
}

class Dog : Animal
{

}
```

1. Stwórz klasę abstrakcyjną i zobacz jaki jest wynik próby stworzenia jej instancji.
2. Stwórz obiekt typu `Dog` i wywołaj metodę `Describe`.
3. Wykorzystując zdobytą dotychczas wiedzę, rozszerz funkcjonalność klasy `Animal` w klasie `Dog` o wypisanie dodatkowych informacji.

Głównym powodem tworzenia klas abstrakcyjnych jest stworzenie typu „częściowego”, który udostępnia jedynie fragment funkcjonalności, natomiast reszta jej musi zostać zaimplementowana dopiero w klasie uszczegóławiającej definicję.

Własności klas abstrakcyjnych:
- Nie można tworzyć obiektów tego typu
- może zawierać metody abstrakcyjnej
- nie można tworzyć klas zapieczętowanych (`sealed`) będących jednocześnie klasami abstrakcyjnymi
- dziedziczące klasy muszą zawierać implementację wszystkich zdefiniowanych abstrakcyjnych metod oraz własności.

Zagadnienie 2: Metody i właściwości abstrakcyjne
Metody abstrakcyjne mogą być definiowane jedynie w klasach abstrakcyjnych. Definicja ma postać identyczną jak w przypadku zwykłej metody. Jedyna różnica polega na braku definicji ciała metody. Metody abstrakcyjne służą „wymuszeniu" na klasie podrzędnej, dziedziczącej po klasie abstrakcyjnej, zdefiniowanie każdej występującej metody abstrakcyjnej.

Przykład:
```
abstract class Animal
{
	public abstract string Describe();
}
```

1. Sprawdź czy uda się zdefiniować klasę pochodną klasy `Animal` nie implementującą metody `Describe`. Jeśli nie to zapamiętaj kiedy i jaki błąd zostanie zgłoszony.

Własności metod abstrakcyjnych:
- jest domyślnie metodą wirtualną
- jej definicja jest możliwa jedynie w klasie abstrakcyjnej
- nie ma sekcji kodu implementującego jej funkcjonalność
- funkcjonalność określana jest w klasie dziedziczącej po klasie zawierającej metodą abstrakcyjną
- nie wolno używać `static` oraz `virtual` wraz z słowem kluczowym `abstract`

Własności właściwości abstrakcyjnych:
- nie mogą być statyczne

Przykład:
```
public abstract double Color
{
	get;
}
```

Zagadnienie 3: Interfejsy
Podobnie jak w przypadku wcześniej omówionych klas abstrakcyjnych Interfejsy służą definiowaniu pewnych koncepcyjnych typów pozbawionych szczegółowych definicji. Główną różnicą pomiędzy klasami abstrakcyjnymi a interfejsami jest to, że te drugie nie mogą w ogóle definiować nawet częściowej funkcjonalności. Interfejs można rozumieć jako twór będący klasą abstrakcyjną posiadającą jedynie abstrakcyjne metody oraz właściwości, przez co pozwalają one na określanie czegoś w rodzaju kontraktu. Kontrakt ten polega na tym, że wszystkie metody oraz pola interfejsu po którym następuje dziedziczenie muszą być zaimplementowane w klasie dziedziczącej. Różnicą względem klas abstrakcyjnych jest możliwość dziedziczenia wielu interfejsach jednocześnie (w przypadku klas abstrakcyjnych pozostaje ograniczenie tylko jednej klasy po której wolno dziedziczyć).

Przykład:
```
interface IPerson
```

```
{
        string Describe();

        string Name
        {
                get;
                set;
        }
}
```

Wszystkie składowe interfejsu mają zakres widoczności publiczny i nie można go zmienić.

Zagadnienie 4: Enumerations
Wyliczenia (numerations) to specjalny zbiór nazwanych wartości mapowanych na zbiór liczb (zwykle całkowitych). Definiuje się je niezależnie od klas w aktualnie używanej przestrzeni nazw co umożliwia dostęp do nich klasom z tej samej przestrzeni nazw.

Przykład:
```
public enum Days { Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday }
```

Powyższy przykład przypisuje poszczególnym nazwom kolejne liczby całkowite zaczynając od 0. Aby przypisać wileczeniom wartości zaczynając od 1 należy posłużyć następującą składnią:
```
public enum Days { Monday = 1, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday }
```

Można również uzyskać nazwę wartości podając liczbę całkowitą:
```
Days day2 = (Days)5;
Console.WriteLine(day2);
Console.ReadLine();
```

Linki:
[1] MSDN: http://msdn.microsoft.com/
[2] C# Practical Learning: http://www.functionx.com/csharp/
[3] C# for beginners: http://www.csharphelp.com/2006/12/c-tutorial-for-beginners/
[4] C# tutorial: http://csharpcomputing.com/Tutorials/
[4] C# tutorials: http://csharp.net-tutorials.com/