Дефиниране на класове

Инж. Венцеслав Кочанов

Какво е Обектно ориентитрано програмиране

Обектно-ориентираното програмиране е стратегия, която предоставя някои принципи за разработване на приложения или разработване на софтуер. Това е методология. Подобно на ООП, съществуват и други методологии като структурирано програмиране, процедурно програмиране или модулно програмиране. Но в днешно време един от добре познатите и известни стилове е обектната ориентация, т.е. обектно-ориентираното програмиране.

С помощта на обектната ориентация разработването на приложения или програмирането става все по-систематично и можем да следваме процедурите на инженеринг за разработване на софтуер. Както в друго инженерство, как се разработва продукт, по същия начин софтуерният продукт се разработва чрез възприемане на обектна ориентация. Ако говорим малко за друго инженерство, като строителен инженер, който строи сграда, тогава преди всичко той/тя ще направи план или ще направи проект. Докато правят дизайн или план, те може да имат много опции, но те ще изберат един от дизайните и ще го финализират. След това, след като бъде финализиран като план на хартия, тогава те ще започнат да конструират. По същия начин, електронен инженер, когато произвежда някакво устройство, той ще измисли някакъв дизайн, който е схемата на това устройство на хартия.

Обектно-ориентираното програмиране е наследник на процедурното (структурно) програмиране. Процедурното програмиране най-общо казано описва програмите чрез група от преизползваеми парчета код (процедури), които дефинират входни и изходни параметри. Процедурните програми представляват съвкупност от процедури, които се извикват една друга.

Проблемът при процедурното програмиране е, че преизползваемостта на кода е трудно постижима и ограничена – само процедурите могат да се преизползват, а те трудно могат да бъдат направени общи и гъвкави. Няма лесен начин да се реализират абстрактни структури от данни, които имат различни имплементации.

Обектно-ориентираният подход залага на парадигмата, че всяка програма работи с данни, описващи същности (предмети или явления) от реалния живот. Например една счетоводна програма работи с фактури, стоки, складове, наличности, продажби и т.н. Така се появяват обектите – те описват характеристиките (свойства) и поведението (методи) на тези същности от реалния живот.

Основни принципи на ООП

За да бъде един програмен език обектно-ориентиран, той трябва не само да позволява работа с класове и обекти, но и трябва да дава възможност за имплементирането и използването на принципите и концепциите на ООП: наследяване, абстракция, капсулация и полиморфизъм.
Сега ще разгледаме в детайли всеки от тези основни принципи на ООП.
- Капсулация (Encapsulation)
Ще се научим да скриваме ненужните детайли в нашите класове и да предоставяме прост и ясен интерфейс за работа с тях.

- Наследяване (Inheritance)
Ще обясним как йерархиите от класове подобряват четимостта на кода и позволяват преизползване на функционалност.
- Абстракция (Abstraction)
Ще се научим да виждаме един обект само от гледната точка, която ни интересува, и да игнорираме всички останали детайли.
- Полиморфизъм (Polymorphism)
Ще обясним как да работим по еднакъв начин с различни обекти, които дефинират

Клас и обекти от гледна точка на езика за програмиране.
Тук ще разберем класа и обектите от гледна точка на езика за програмиране C#. Но това е приложимо и за всеки обектно-ориентиран език за програмиране като java и C++.

Клас:
Класът е просто дефиниран от потребителя тип данни, който представлява както състояние, така и поведение. Състоянието представлява свойствата, а поведението е действието, което обектите могат да извършат. С други думи, можем да кажем, че класът е планът/планът/шаблонът, който описва детайлите на даден обект. Класът е план, от който се създават отделните обекти. В C# класът се състои от три неща, т.е. име, атрибути и операции.

Класът дефинира абстрактните характеристики на даден обект. Той е план или шаблон, чрез който се описва природата на нещо (някакъв обект). Класовете са градивните елементи на ООП и са неразделно свързани с обектите. Нещо повече, всеки обект е представител на точно един клас.

Класовете предоставят модулност и структурност на обектно-ориентираните програми. Техните характеристики трябва да са смислени в общ контекст, така че да могат да бъдат разбрани и от хора, които са запознати с проблемната област, без да са програмисти.

Обекти:

Това е екземпляр на клас. Клас се активира чрез създаване на обекти. Един обект може да се разглежда като нещо, което може да извършва дейности. Наборът от дейности, които обектът изпълнява, определя поведението на обекта. Всички членове на клас могат да бъдат достъпни чрез обекта. За достъп до членовете на класа трябва да използваме оператора точка (.). Операторът точка свързва името на обект с името на член на клас.

В обектите от реалния свят (също и в абстрактните обекти) могат да се отделят следните две групи техни характеристики:
- Състояния (states) – това са характеристики на обекта, които по някакъв начин го определят и описват по принцип или в конкретен момент.
- Поведения (behaviors) – това са специфични характерни действия, които обектът може да извършва.
Обектите в ООП обединяват данни и средствата за тяхната обработка в едно цяло. Те съответстват на обектите от реалния свят и съдържат в себе си данни и действия:
- Член-данни (data members) – представляват променливи, вградени в обектите, които описват състоянията им.
- Методи (methods) – вече сме ги разглеждали в детайли. Те са инструментът за изграждане на поведението на обектите.

Всеки обект е представител на точно един клас и е създаден по шаблона на този клас. Създаването на обект от вече дефиниран клас наричаме инстанциране (instantiation).
Инстанция (instance) е фактическият обект, който се създава от класа по време на изпълнение на програмата.
Всеки обект е инстанция на конкретен клас. Тази инстанция се характеризира със състояние (state) – множество от стойности, асоциирани с атрибутите на класа.

Как можем да създадем клас и обект в C#?

Класът в C# се дефинира чрез ключовата дума class, последвана от идентификатор (име) на класа и съвкупност от член-данни и методи, обособени в собствен блок код.
Класовете в C# могат да съдържат следните елементи:
- Полета (fields) – член-променливи от определен тип;
- Свойства (properties) – това са специален вид елементи, които разширяват функционалността на полетата като дават възможност за допълнителна обработка на данните при извличането и записването им в полетата от класа.

Дефиницията на клас започва с ключовата дума class, последвана от името на класа (тук името на класа е Calculator), а тялото на класа е оградено от чифт фигурни скоби. Като част от тялото на класа вие дефинирате членове на класа (свойства, методи, променливи и т.н.). Тук, като част от тялото, ние дефинираме един метод, наречен CalculateSum. Калкулаторът на класа е само шаблон. За да използвате този клас или шаблон, имате нужда от обект. Както можете да видите във втората част на изображението, ние създаваме обект от класа Calculator, използвайки ключовата дума new. И след това запазете препратката към обекта в променливата calObject, която е от тип Calculator. Сега, използвайки този

Дефиниране на прост клас

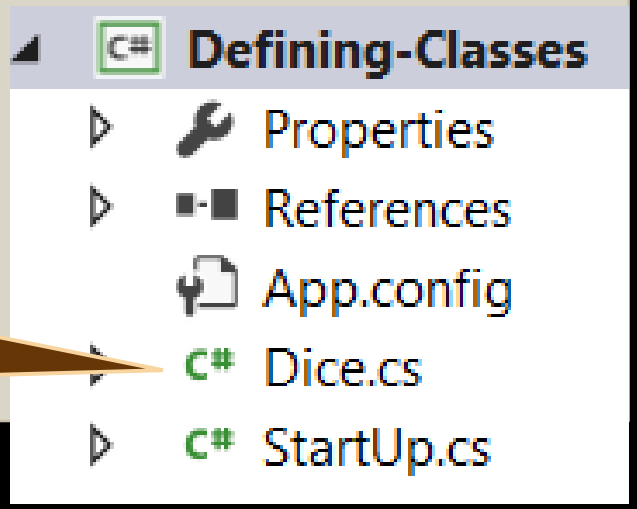
Ключова дума

```
class Calculator
{

    ...

}
```

Име на класа

Тяло на клас

Клас в отделен файл

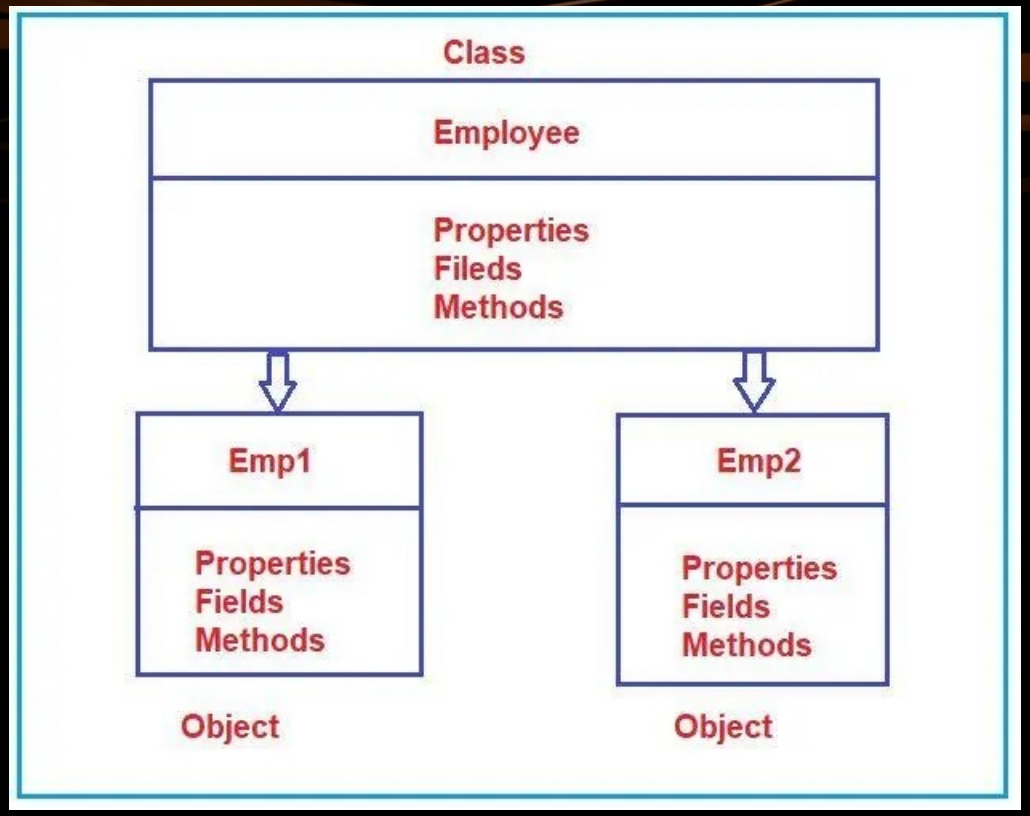


```
using System;
namespace ClassObjectsDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            //Creating object
            Calculator calObject = new Calculator();

            //Accessing Calculator class member using Calculator class object
            int result = calObject.CalculateSum(10, 20);

            Console.WriteLine(result);
            Console.ReadKey();
        }
    }

    //Defining class or blueprint or template
    public class Calculator
    {
        public int CalculateSum(int no1, int no2)
        {
            return no1 + no2;
        }
    }
```

Output: 30

Разлика между клас и обекти в C#

Много програмисти или разработчици все още се объркват от разликата между клас и обект. Както вече обсъдихме, в обектно-ориентираното програмиране, класът е шаблон или план за създаване на обекти и всеки обект в C# трябва да принадлежи към клас.



Както можете да видите на горното изображение, тук имаме един клас, наречен „Служител“. Всички служители имат някои свойства като ID на служител, име, заплата, пол, отдел и т.н. Тези свойства не са нищо друго освен атрибутите (свойства или полета) на класа Employee.

Именуване на класове

- Класовете са PascalCase
- Използвайте описателни съществителни
- Избягвайте абревиатури (освен известни: URL, HTTP, и др.)

```
class Dice { … }
class BankAccount { … }
class IntegerCalculator { … }
```

```
class TPMF { … }
class bankaccount { … }
class intcalc { … }
```


Summary