

Лабораторно упражнение № 11

Виртуални методи. Динамично свързване. Полиморфизъм

I. Теоретична част

1. Полиморфизъм и динамично свързване

Терминът **полиморфизъм** в обектно-ориентираното програмиране определя, че едни и същи действия се реализират по различен начин в зависимост от обектите, върху които се прилагат. Такива действия се наричат **полиморфични**. За да се реализира полиморфно действие, класовете на обектите, върху които се прилага това действие, трябва да имат общ корен т.е. да бъдат производни на един и същи клас. Реализацията на полиморфизма е чрез методи, които се наричат **виртуални**. Виртуалните методи позволяват да бъдат извикани различни версии (реализации) на един и същ метод, според това какъв е типът на обекта. По този начин се позволява да се работи с група взаимосвързани обекти по идентичен начин.

Извикването на виртуалните методи е по механизма на динамичното свързване (late binding), чрез който се позволява определянето на коя от версиите на виртуалния метод ще се извика да е едва по време на изпълнение, а не по време на компилация.

2. Дефиниране на виртуални методи

Реализацията на полиморфизъм в език C# е чрез виртуалните методи. За да се дефинират методи като виртуални, в базовия и в производния клас тези методи трябва да са с еднакви имена, еднакъв тип на върнат резултат, еднакъв брой и тип параметри т.е. сигнатурата на метода трябва да е една и съща.

Виртуалните методи се дефинират в базовия клас чрез ключова дума **virtual**, а в наследените се предефинират с ключова дума **override**. Ключовите думи **virtual** и **override** стоят след модификатора за достъп (**public**).

Пример:

```
class Base
{
    ...

    public virtual void Info()
    {
        Console.WriteLine("Base class");
    }
}
```

Виртуалният метод задължително трябва да има реализация

Предефинирането на виртуалния метод в производния клас е по следния начин:

```
class Inherit: Base
{
    ...

    public override void Info()
    {
        Console.WriteLine("Derived class");
    }
}
```

Предефинираният виртуален метод също трябва да има реализация. В клас, който наследява производния на следващо ниво в йерархията също може да има вариант на вече предефинирания виртуален метод.

Извикването на виртуален метод по механизма на динамичното свързване е чрез референция на обект от базов клас. Причината е, че референция към обект от базов клас може да сочи както обект от базов клас, така и обект от производен. Пример:

```
Base obj1 = new Base();
Base obj2=new Inherit();

obj1.Info();
obj2.Info();
```

И двете референции са от тип Base, но едната сочи обект от базовия клас, а втората референция сочи обект от производния клас. Затова двете референции извикват различни версии на виртуалния метод Info().

II. Задачи за изпълнение

1. Да се създаде базов клас Pet, описващ домашен любимец с полета name, age, описващи име и възраст на животното. Класът да съдържа свойства за четене и запис за връзка с полетата. В класа да се дефинират два виртуални метода GetSound и Info. Чрез първият да се връща низ, съответстващ на звука, който издава животното, а чрез втория да се дава информация за животното. Да се дефинират производни класове Cat и Dog, съответстващи на коте и куче, в които да се предефинират виртуалните методи. Да се създаде конзолно приложение, което работи с 3 обекта от клас Dog и два обекта от клас Cat.

2. Да се създаде базов клас Dot2D, съответстващ на точка в равнината и производен клас Dot3D, съответстващ на точка в пространството. Класовете да съдържат конструктори по подразбиране и конструктори с параметри, които да дават стойности на полетата. Класовете да съдържат виртуални методи за извеждани стойностите на координатите. Да се създаде демонстрационна програма за работа с тези класове.