Лабораторно упражнение № 11

Абстрактни класове и интерфейси в език C#

I. Теоретична част

1. Абстрактни класове

В програмен език C# е възможно да се декларират абстрактни класове. Те служат за наследяване и не е възможно да се дефинира (създаде) обект от абстрактен клас. Абстрактния клас съдържа абстрактни методи т.е. методи, които нямат имплементация. В производните класове тези методи задължително се предефинират. Чрез абстрактните класове и абстрактните методи се реализира т.нар. **принудителен полиморфизъм**.

В език C# за да се създаде абстрактен клас, той трябва явно да е обявен като такъв чрез ключова дума ***abstract*** в началото на декларацията. Пример:

```
abstract public class Figure
{
}
```

След като даден клас бъде определен като абстрактен, в него могат да бъдат декларирани неопределен брой абстрактни методи. Абстрактните методи в C# са аналог на чисто виртуалните функции в C++. На практика това са методи без дефиниция.

Общият вид на декларацията на абстрактен метод е:

модификатор abstract тип ИмеАбстрактен Метод(списък параметри);

Пример:

```
abstract public class Figure
{
      public abstract void Input();
      public abstract void Output();
      public abstract double Area();
}
```

Абстрактните методи са виртуални по подразбиране и не е необходимо да се обявяват като такива с ключова дума virtual. Ако това бъде направено в програмния код, ще се генерира грешка при компилация.

2. Интерфейси

Освен чрез абстрактни класове, някои от проблемите, свързани с наследяването могат да се решат и чрез използване на интерфейси. Интерфейсът е набор от семантически свързани абстрактни членове. Броят на членовете зависи от това какво поведение се моделира с помощта на интерфейса.

Интерфейсите се декларират подобно на класовете, чрез ключова дума interface:

interface ИмеИнтерфейс
{

}

В интерфейса се декларират методи без да се имплементират и без да се посочва модификатор на достъп. Интерфейсите се използват за наследяване и интерфейс не се имплементира т.е. не се дефинира обект от тип интерфейс. От това произтичат следните ограничения:

- Не е допустимо да се поставят полета в интерфейса, дори и те са статични. Полето е имплементация на атрибут на обект , а не е възможно да се дефинира обект от тип интерфейс.
- Не е разрешено да се дефинират конструктори в интерфейс, тъй като конструкторът обикновено изпълнява действия, свързани с инициализацията на полетата.
- Не може да се дефинира деструктор в интерфейс.
- Недопустимо е да се слагат модификатори на достъп пред декларациите на методите в интерфейса, тъй като модификаторът на достъп по подразбира не е public и не може да бъде променян.
- Не е разрешено да се влагат типове в интерфейс (изброявания, структури, класове, други интерфейси)
- Не е разрешено интерфейс да наследява структура или клас. Наследявайки тези типове, интерфейсът би наследил техните полета.
- Даден интерфейс може да наследи един или повече интерфейси.

Интерфейсите се създават с цел да бъдат наследявани или от класове, или от структури. Съответно, в класа или структурата методите на интерфейса имат своята имплементация.

Пример:

```
interface IExamp
{
      string Name();
}

class Examp : IExamp
{
      string IExamp.Name()
      {
      }
}
```

II. Задачи за изпълнение

1. Да се създаде абстрактен базов клас Figure, описващ фигура в равнината. Класът да съдържа абстрактни методи за въвеждане и извеждане стойностите на полетата и абстрактен метод за намиране

площта на фигурата. Да се създадат производни класове, описващи квадрат, окръжност, ромб и правоъгълник. Да се създаде демонстрационна програма за работа с тези класове.

Упътване: Да се създаде масив от различни фигури, на които да се въведат размери и да се изведе информация за площта и размерите на всяка фигура.

2. Да се модифицира програмният код като методът, намиращ площта на фигурата да се замени със свойство за четене.
3. Да се модифицира програмния код от задача 1, така че да се реализира с интерфейс, описващ функционалност на фигура.
4. Да се създаде интерфейс IPreditor, който моделира поведението на животно: ходене, бягане, хранене, почивка, издаване на звук. Чрез използване на интерфейса, да се създадат производни класове Tiger, Lion, Cat, които дават информация за съответното животно и моделират поведението му.