12 лабораториялық сабақ. Индексаторлармен және қасиеттермен жұмыс істеу.

Тапсырма: төменде берілген мысалдарды компьютерде орындап, түсіндіріп беріңіз.

Мысал 12.2. Қасиетті қолданатын қарапайым мысал

```
using System;
class SimpProp {
   int prop;     // MyProp қасиетімен басқарылатын өріс
   public SimpProp() { prop = 0; }
    /* Бұл қасиет жабық prop экземпляр айнымалысына қол жеткізуді қамтамасыз
    етеді. Ол тек оң мәндерді ғана меншіктеуді жүзеге асырады. */
   public int MyProp {
      get {
         return prop;
      }
      set {
         if(value >= 0) prop = value;
      }
   }
}
// Қасиетті қолдануды көрсету
class PropertyDemo {
   static void Main() {
      SimpProp ob = new SimpProp();
      Console.WriteLine("ob.MyProp qasietining  bastapqy
           mani: " + ob.MyProp);
      ob.MyProp = 100;    // мән меншіктеу
      Console.WriteLine("ob.MyProp qasietining  qazirgi mani: "
           + ob.MyProp);
      // prop айнымалысына теріс мән меншіктеуге болмайды
      Console.WriteLine("ob.MyProp qasietine -10 manin "  +
                                "menshikteuge talpynu");
      ob.MyProp = -10;
      Console.WriteLine("ob.MyProp qasietining  qazirgi
           mani: " + ob.MyProp);
   }
}
```

Бұл программаның орындалу нәтижесі:

```
ob.MyProp qasietining  bastapqy mani: 0
ob.MyProp qasietining  qazirgi mani: 100
ob.MyProp qasietine -10 manin + menshikteuge talpynu
ob.MyProp qasietining  qazirgi mani: 100
```

Мысал 12.3. Қасиеттердің жүзеге асырылуын Шеңбер класының мысалында қарастырайық.

```
using System;
class Shenber {
   public int XCoord {get; set;}
   public int YCoord {get; set;}
   private double radius;
   public double Radius {
      get {
         return radius;
      }
      set {
         if(value >0) radius = value;
      }
   }
```

```
}
class Program {
   static void Main() {
      Shenber shenber1 = new Shenber();
      shenber1.XCoord = 10; shenber1.YCoord = 5;
      shenber1.Radius = 10;
      shenber1.Radius = -5 // мән қабылданбайды,
      // себебі қасиетте анықталған шарт қанағаттандырылмады
      Console.WriteLine("Shenber centrinin coordinatalary:
          (" + shenber1.XCoord + ", " + shenber1.YCoord + "),
          shenber radius - " + shenber1.Radius);
      Console.ReadKey();
   }
}
```

Индексаторлар

```
// Использовать индексатор для создания отказоустойчивого массива.
using System;
class FailSoftArray {
int[] a; // ссылка на базовый массив
public bool ErrFlag; // обозначает результат последней операции
// Построить массив заданного размера,
public FailSoftArray(int size) { a = new irrt [size];
Length = size;
}
// Это индексатор для класса FailSoftArray.
public int this[int index] {
// Это аксессор get.
get {
if (ok(index))    {
ErrFlag = false; return a[index];
} else {
ErrFlag = true; return 0;
}
}
// Это аксессор set.
set {
if(ok(index))    {
a[index] = value;
ErrFlag = false;
}
else ErrFlag = true;
}
}
// Возвратить логическое значение true, если
// индекс находится в установленных границах,
private bool ok(int index) {
if(index >= 0 & index < Length) return true; return false;
}
}
// Продемонстрировать применение отказоустойчивого массива,
class FSDemo {
static void Main() {
```

```
FailSoftArray fs = new FailSoftArray(5); int x;
// Выявить скрытые сбои.
Console.WriteLine("Скрытый сбой."); for(int i=0; i < (fs.Length * 2); i++) fs[i] = i*10;
for(int i=0; i < (fs.Length * 2); i++)    {
x = fs[i] ;
if (x != -1) Console.Write(x + " ") ;
Console.WriteLine ();
//А теперь показать сбои.
Console.WriteLine("ХпСбой с уведомлением об ошибках."); for(int i=0; i < (fs.Length * 2);
i++)    {
fs[i] = i * 10; if(fs.ErrFlag)
Console.WriteLine("fs[" + i + "] вне границ");
}
for(int i=0; i < (fs.Length * 2); i++)    {    N
x = f s [ i ] ;
if(!fs.ErrFlag) Console.Write(x + " "); else
Console.WriteLine("fs[" + i + "] вне границ");
}
}
}
```

Вот к какому результату приводит выполнение этой программы.

```
Скрытый сбой.
0 10 20 30 40 0 0 0 0 0
Сбой с уведомлением об ошибках.
fs[5] вне границ
fs[6] вне границ
fs[7] вне границ
fs[8] вне границ
fs[9] вне границ
O 10 20 30 40 fs[5] вне границ
fs[6] вне границ
fs[7] вне границ
fs[8] вне границ
fs[9] вне границ
```

Перегрузка индексаторов

```
// Перегрузить индексатор массива класса FailSoftArray.
using System;
class FailSoftArray {
int[] a; // ссылка на базовый массив
public bool ErrFlag; // обозначает результат последней операции
// Построить массив заданного размера,
public FailSoftArray(int size) { a = new int[size];
Length = size;
}
// Это индексатор типа int для массива FailSoftArray.
public int this[int index] {
// Это аксессор get.
get {
if(ok(index))    {
ErrFlag = false; return a[index];
} else {
```

```
ErrFlag = true; return 0;
}
}
// Это аксессор set.
set {
if(ok(index))    {
a[index] = value;
ErrFlag = false;
}
else ErrFlag = true;
}
}
/* Это еще один индексатор для массива FailSoftArray.
Он округляет свой аргумент до ближайшего целого индекса. */
public int this[double idx] {
// Это аксессор get.
get {
int index;
// Округлить до ближайшего целого.
if( (idx - (int) idx) < 0.5) index = (int) idx;
else index = (int) idx + 1;
if(ok(index))    {
ErrFlag = false; return a[index];
} else {
ErrFlag = true; return 0;
}
}
// Это аксессор set.
set {
int index;
// Округлить до ближайшего целого.
if( (idx - (int) idx) < 0.5) index = (int) idx;
else index = (int) idx + 1;
if (ok (index) )    {
a[index] = value;
ErrFlag = false;
}
else ErrFlag = true;
}
}
// Возвратить логическое значение true, если
 // индекс находится в установленных границах,
private bool ok(int index) {
if(index >= 0 & index < Length) return true; return false;
}
}
// Продемонстрировать применение отказоустойчивого массива,
class FSDemo {
static void Main() {
FailSoftArray fs = new FailSoftArray(5);
// Поместить ряд значений в массив fs.
for(int i=0; i < fs.Length; i++) fs[i] = i;
```

```
// А теперь воспользоваться индексами
// типа int и double для обращения к массиву.
Console.WriteLine("fs[1]: " + fs[1]);
Console.WriteLine("fs[2]: " + fs[2]);
Console.WriteLine("fs[1.1]: " + fs[l.l]);
Console.WriteLine("fs[1.6]: " + fs[1.6]);
}
}
```

При выполнении этой программы получается следующий результат.

```
f S [ 1 ] :    1
fs [2] : 2 fs[1 -1 ] : 1 f s [ 1. 6 ] : 2
```

Индексаторы без базового массива

```
// Индексаторы совсем не обязательно должны оперировать отдельными массивами.
using System;
class PwrOfTwo {
/* Доступ к логическому массиву, содержащему степени числа 2 от 0 до 15. */
public int this[int index] {
// Вычислить и возвратить степень числа 2.
get {
if((index >= 0)    && (index < 16)) return pwr(index);
else return -1;
}
// Аксессор set отсутствует.
}
int pwr(int p) { int result = 1;
for(int i=0; i < p; i++) result *= 2;
return result;
}
}
class UsePwrOfTwo { static void Main() {
PwrOfTwo pwr = new PwrOfTwo();
Console.Write("Первые 8 степеней числа 2: "); for(int i=0; i < 8; i++)
Console.Write(pwr[i] + " ");
Console.WriteLine();
Console.Write("А это некоторые ошибки: ");
Console.Write(pwr[-1] + " " + pwr[17]);
```

Вот к какому результату приводит выполнение этой программы.

```
Первые 8 степеней числа 2: 1 2 4 8 16 32 64 128 А это некоторые ошибки: -1 -1
```

Многомерные индексаторы

```
// Двумерный отказоустойчивый массив.
using System;
class FailSoftArray2D {
int[,] a; // ссылка на базовый двумерный массив
int rows, cols; // размеры массива
public int Length; // открытая переменная длины массива
public bool ErrFlag; // обозначает результат последней операции
// Построить массив заданных размеров,
```

```
public FailSoftArray2D(int r, int c) { rows = r; cols = c;
a = new int[rows, cols];
Length = rows * cols;
}
// Это индексатор для класса FailSoftArray2D.
public int this[int indexl, int index2] {
// Это аксессор get. get {
if(ok(indexl, index2)) {
ErrFlag = false;
return a[indexl, index2];
} else {
ErrFlag = true; return 0;
}
}
// Это аксессор set. set {
if(ok(indexl, index2))    {
a[indexl, index2] = value;
ErrFlag = false;
}
else ErrFlag = true;
}
}
// Возвратить логическое значение true, если
// индексы находятся в установленных пределах,
private bool ok(int indexl, int index2) {
if (indexl >= 0 & indexl < rows & index2 >= 0 & index2 < cols) return true;
return false;
}
}
// Продемонстрировать применение двумерного индексатора,
class TwoDIndexerDemo { static void Main() {
FailSoftArray2D fs = new FailSoftArray2D(3, 5); int x;
// Выявить скрытые сбои.
Console.WriteLine("Скрытый сбой."); for (int i=0; i < 6; i++) fs[i, i]=i*10;
for(int i=0; i < 6; i++)    {
x = f s [ i, i ] ;
if(x != -1) Console.Write (x + " ");
}
Console.WriteLine ();
// А теперь показать сбои.
Console.WriteLine("\nСбой с уведомлением об ошибках.");
for(int i=0; i < 6; i++)    {
fs[i,i] = i *10; if(fs.ErrFlag)
Console.WriteLine("fs[" + i + ", " + i + "] вне границ
}
for(int i=0; i < 6; i++)    {
x = f s [ i, i ] ;
if(!fs.ErrFlag) Console.Write(x + " ");
else
Console.WriteLine("fs[" + i + ", " + i + "] вне границ");
}
}
```

}
Вот к какому результату приводит выполнение этого кода:
Скрытый сбой.
0 10 20 0 0 0
Сбой с уведомлением об ошибках.
fs[3, 3] вне границ
fs[4, 4] вне границ
fs[5, 5] вне границ
0 10 20 fs[3, 3] вне границ
fs[4, 4] вне границ
fs[5, 5] вне границ