**ЛАБОРАТОРНА РОБОТА №3**

**Тема: ВИКОРИСТАННЯ ВЛАСТИВОСТЕЙ. АВТОМАТИЧНІ ВЛАСТИВОСТІ.**

**СКОРОЧЕНИЙ ЗАПИС ВЛАСТИВОСТЕЙ**

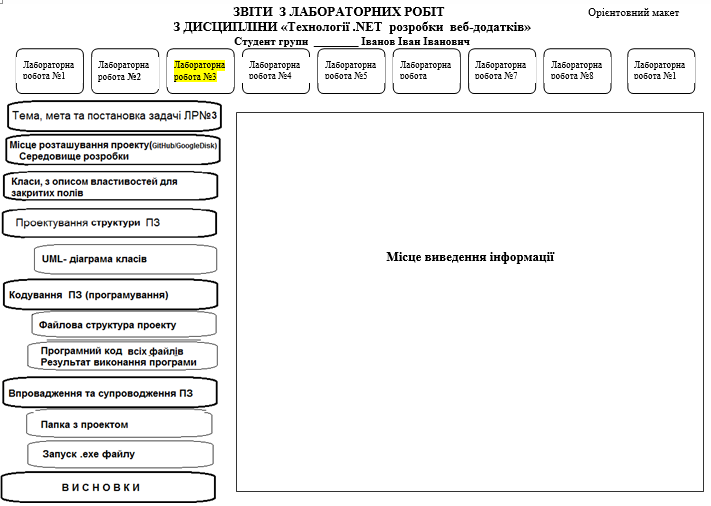
**Мета**: Здійснити введення та виведення даних закритих полів класів, застосовуючі методи-властивості.

**Постановка задачі:**

1. Доповнити програмну реалізацію класів методами-властивостями класів, введення та виведення даних закритих полів класів
2. Протестувати програму, шляхом запуску програми, введенням та виведенням значень закритих полів.

**ХІД ВИКОНАННЯ РОБОТИ**

1. У пункті меню «*Тема, мета та постановка задачі ЛР №4* » звітного HTML-документу розмістити :
   1. Тему лабораторної роботи №4
   2. Мету лабораторної роботи №4
   3. Постановку задачі лабораторної роботи №4
   4. Вказати класи та методи-властивості, які реалізуються у ціх класах.
2. У пункті меню «*Місце розташування проекту.Середовище розробки*» вказатим **місце розташування проекту** (шлях до папки з проектом). Це може бути GitHub або Googe Disk. Також написати **середовище розробки програми.**
3. У пункті меню «*Класи, з описом властивостей для закритих полів*» вказати перелік класів, у яких використовуются властивості для закритих полів. Написати призначення ціх полів
4. У пункі «*Проектування структури ПЗ*» розмістит Діаграму класів, попередньо добавивши методи-властивості у UML-діаграму класів.
5. У пункті «*Кодування ПЗ (програмування*)» показати файлову структуру проекту (скрін шот Solution explorer).. Надати програмні коди файлів проекту та програмний код класу Program та виведення результатів виконання
6. У пункті меню «*Впровадження та супроводження ПЗ*» розмістити посилання на папку з проектом, усіма файлами проекту, виконувальним файлом.ехе, з можливістю відкрити проекту зі звіту.
7. У пункті меню «*ВИСНОВКИ* »написати висновки по лабораторноій роботи №3. Зазаначити переваги використання методів-властивостей
8. Для оформлення звітного HTML-документу приведен орієнтовний макет.



**ПРИМІТКА**:

Звіт з лабораторних робіт слід підготувати у вигляді **гіпертекстового документа у форматі html.** Документ має містити меню, яке включає команди, що подані нижче. Слід реалізувати запуск програм на виконання з гіпертекстового документа.

Звіт та проекти лабораторних робіт слід записати на Гугл-Диск а. На диску має бути файл readme.tхt, який містить відомості про автора звіту та проектів.

**Титульна html- сторінка.**

Назва роботи

Автор (ПІБ, група, курс, № заліковки)

Фото

Рік навчання

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Крім звичайних методів в мові C # передбачені спеціальні методи доступу, які називають **властивості** . Вони забезпечують простий доступ до полів класів та структур, дізнатися їх значення або виконати їх установку.

Стандартне опис властивості має наступний синтаксис:

|  |  |
| --- | --- |
| 1  2  3  4 | [модификатор\_доступа] возвращаемый\_тип произвольное\_название  {      // код свойства  } |

наприклад:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | class Person  {      private string name;        public string Name      {          get          {              return name;          }            set          {              name = value;          }      }  } |

Тут у нас є закрите поле nameі є загальнодоступне властивість Name. Хоча вони мають практично однакову назву за винятком регістра, але це не більше ніж стиль, назви у них можуть бути довільні і не обов'язково повинні збігатися.

Через це властивість ми можемо управляти доступом до змінної name. Стандартне визначення властивості містить блоки **get** і **set** . У блоці get ми повертаємо значення поля, а в блоці set встановлюємо. Параметр valueпредставляє передане значення.

Ми можемо використовувати дане властивість наступним чином:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | Person p = new Person();    // Устанавливаем свойство - срабатывает блок Set  // значение "Tom" и есть передаваемое в свойство value  p.Name = "Tom";    // Получаем значение свойства и присваиваем его переменной - срабатывает блок Get  string personName = p.Name; |

Можливо, може виникнути питання, навіщо потрібні властивості, якщо ми можемо в даній ситуації обходитися звичайними полями класу? Але властивості дозволяють вкласти додаткову логіку, яка може бути необхідна, наприклад, при присвоєнні змінної класу будь-якого значення. Наприклад, нам треба встановити перевірку за віком:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | class Person  {      private int age;        public int Age      {          set          {              if (value < 18)              {                  Console.WriteLine("Возраст должен быть больше 17");              }              else              {                  age = value;              }          }          get { return age; }      }  } |

Якби змінна age була б публічною, то ми могли б передати їй ззовні будь-яке значення, в тому числі негативне. Властивість же дозволяє приховати дані об'єкти та опосредовать до них доступ.

Блоки set і get не обов'язково одночасно повинні бути присутніми в властивості. Якщо властивість визначають тільки блок get, то така властивість доступно тільки для читання - ми можемо отримати його значення, але не встановити. І, навпаки, якщо властивість має тільки блок set, тоді це властивість доступно тільки для запису - можна тільки встановити значення, але не можна отримати:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | class Person  {      private string name;      // свойство только для чтения      public string Name      {          get          {              return name;          }      }        private int age;      // свойство только для записи      public int Age      {          set          {              age = value;          }      }  } |

Хоча в прикладах вище властивості визначалися в класі, але точно також ми можемо визначати і використовувати властивості в структурах.

### **модифікатори доступу**

Ми можемо застосовувати модифікатори доступу не тільки до всього властивості, але і до окремих блоках - або get, або set:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | class Person  {      private string name;        public string Name      {          get          {              return name;          }            private set          {              name = value;          }      }      public Person(string name)      {          Name = name;      }  } |

Тепер закритий блок set ми зможемо використовувати тільки в даному класі - в його методах, властивості, конструкторі, але ніяк не в іншому класі:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Person p = new Person("Tom");    // Ошибка - set объявлен с модификатором private  //p.Name = "John";    Console.WriteLine(p.Name); |

При використанні модифікаторів у властивостях слід враховувати ряд обмежень:

* Модифікатор для блоку set або get можна встановити, якщо властивість має обидва блоки (і set, і get)
* Тільки один блок set або get може мати модифікатор доступу, але не обидва відразу
* Модифікатор доступу блоку set або get повинен бути більш обмежуючим, аніж модифікатор доступу властивості. Наприклад, якщо властивість має модифікатор public, то блок set / get може мати тільки модифікатори protected internal, internal, protected, private

### **Автоматичні властивості**

Властивості керують доступом до полів класу. Однак що, якщо у нас з десяток і більше полів, то визначати кожне поле і писати для нього однотипне властивість було б утомливо. Тому в фреймворк .NET були додані автоматичні властивості. Вони мають скорочене оголошення:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | class Person  {      public string Name { get; set; }      public int Age { get; set; }        public Person(string name, int age)      {          Name = name;          Age = age;      }  } |

Насправді тут також створюються поля для властивостей, тільки їх створює не програміст в коді, а компілятор автоматично генерує при компіляції.

У чому перевага автосвойств, якщо по суті вони просто звертаються до автоматично створюваної змінної, чому б прямо не звернутися до змінної без автосвойств? Справа в тому, що в будь-який момент часу при необхідності ми можемо розгорнути автосвойство в звичайне властивість, додати в нього якусь певну логіку.

Варто враховувати, що не можна створити автоматичне властивість тільки для запису, як у випадку зі стандартними властивостями.

Автосвойствам можна привласнити значення за замовчуванням (ініціалізація автосвойств):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | class Person  {      public string Name { get; set; } = "Tom";      public int Age { get; set; } = 23;  }    class Program  {      static void Main(string[] args)      {          Person person = new Person();          Console.WriteLine(person.Name); // Tom          Console.WriteLine(person.Age);  // 23            Console.Read();      }  } |

І якщо ми не вкажемо для об'єкта Person значення властивостей Name і Age, то будуть діяти значення за замовчуванням.

Варто відзначити, що в структурах ми не можемо використовувати ініціалізацію автосвойств.

Автосвойства також можуть мати модифікатори доступу:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | class Person  {      public string Name { private set; get;}      public Person(string n)      {          Name = n;      }  } |

Ми можемо прибрати блок set і зробити автосвойство доступним тільки для читання. В цьому випадку для зберігання значення цієї властивості для нього неявно буде створюватися поле з модифікатором readonly, тому слід враховувати, що подібні get-властивості можна встановити або з конструктора класу, як в прикладі вище, або при ініціалізації властивості:

|  |  |
| --- | --- |
| 1  2  3  4 | class Person  {      public string Name { get;} = "Tom"  } |

### **Скорочена запис властивостей**

Як і методи, ми можемо скорочувати властивості. наприклад:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class Person  {      private string name;        // эквивалентно public string Name { get { return name; } }      public string Name => name;  } |

Додаткові матеріали