Лабораторная работа №2

Переопределение методов и операторов

Цель работы: изучить переопределение методов, операторов и разработку индексаторов на языке C#.

Теоретическая часть

Полиморфизм

Полиморфизм обозначает способность языка трактовать связанные объекты в сходной манере. Этот принцип позволяет базовому классу определять набор членов (формально называемый полиморфным интерфейсом), которые доступны всем наследникам.

Полиморфный интерфейс класса конструируется с использованием любого количества виртуальных или абстрактных членов.

«Один интерфейс — множество реализаций».

Полиморфизм помогает упростить программу, позволяя использовать один и тот же интерфейс для описания общего класса действий. Выбрать конкретное действие в каждом отдельном случае — это задача компилятора.

Один из механизмов реализации полиморфизма – переопределение. Переопределение позволяет реализовать в производном классе новую логику некоторого члена базового класса. Переопределять можно методы, свойства, события и индексаторы.

Переопределение методов

Для переопределения виртуального метода базового класса в подчиненном классе этот метод описывается с использованием ключевого слова override. Например, в каждом классе можно переопределить методы Equals, ToString и GetHashCode, так как эти методы наследуются от класса object.

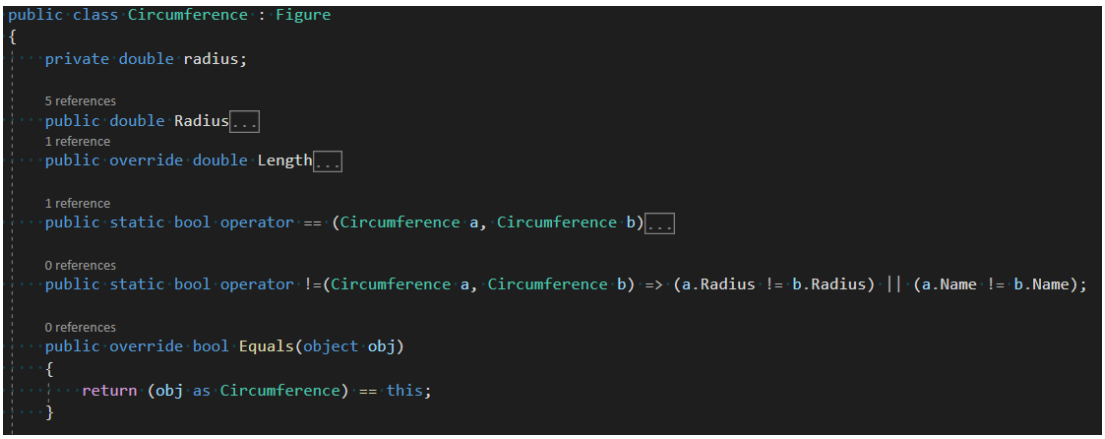

Рисунок 1 – Переопределение метода Equals

Переопределение свойств

Переопределение свойств аналогично переопределению методов. Свойство, которое может быть переопределено, в базовом классе описывается как виртуальное.

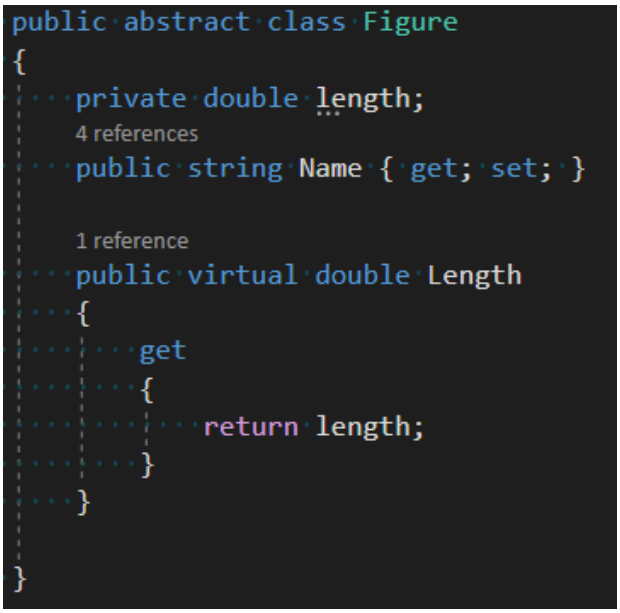

Рисунок 2 – Базовый класс Figure

```
public class Circumference : Figure
{
    private double radius;

    5 references
    public double Radius...
    1 reference
    public override double Length
    {
        get
        {
            return 2*Math.PI*Radius;
        }
    }
```

Рисунок 3 – Переопределение свойства Length в производном классе

Переопределение индексаторов

Аналогично переопределению методов. Рассмотрим пример. Есть класс User, в котором хранится Id и Name пользователя. Также есть класс UserList, в котором хранится список пользователей. Доступ к элементам списка осуществляется через индексатор.

```
public class User
{
    1 reference
    public int Id { get; private set; }
    1 reference
    public string Name { get; private set; }

    0 references
    public User(int id, string name)...
}

1 reference
public class UserList
{
    protected List<User> users;

    1 reference
    public virtual User this[int index]
    {
        set
        {
            users[index] = value;
        }
        get
        {
            return users[index];
        }
    }
}
```

Рисунок 4 – Классы User и UserList

В производном классе CheckUserList переопределим индексатор родительского класса.

```
public class CheckUserList : UserList
{
    1 reference
    public override User this[int index]
    {
        set
        {
            if (!users.Any(user => user.Id == value.Id))
            {
                users[index] = value;
            }
        }
        get
        {
            return users[index];
        }
    }
}
```

Рисунок 5 – Переопределение индексатора

Практическая часть

1) Разработать класс Student, который будет хранить информацию:
   - Номер зачетной книжки (ID)
   - Фамилия
   - Имя
   - Отчество (может отсутствовать)
   - Дата рождения
   - Адрес
   - Номер телефона

Необходимо переопределить методы Equals и ToString. Метод Equals должен возвращать true, если все данные у двух студентов совпадают. Метод ToString выводит в консоль всю информацию о студенте.

2) Разработать класс Group, который будет хранить следующие данные:

- Номер группы
- Список студентов

Необходимо разработать методы Add и Remove для добавления нового студента и удаления существующего, а также разработать индексатор для доступа к студентам по номеру зачетной книжки.

Метод GetInfo должен выводить номер группы и список студентов в алфавитном порядке.

Содержание отчета

1. Титульный лист
2. Цель работы
3. Задание
4. Код программы
5. Результат выполнения