Лабораторная работа № 5

Тема: Работа с классами и XML-документированием

Цель работы:
Научиться структурировать код с помощью классов, обеспечить инкапсуляцию данных и разделение логики программ, а также применять XML-документирование для генерации описаний методов и классов.

Задание:
Создать консольное приложение на языке C#, включающее в себя функции лабораторной работы №4 с дополнительными классами и XML-документированием.

При создании классов следует придерживаться принципа разделения логики работы программы и визуализации данных. Поля классов должны иметь модификатор доступа private. Для полей, к которым необходимо получение доступа из других классов, необходимо написать свойства.

В программе должны быть созданы следующие классы:

1. Статический класс, содержащий методы для игры по отгадыванию ответа значения функции. В классе должен быть метод, вычисляющий значение функции.

2. Класс, содержащий методы для работы с массивами. Класс должен содержать поле для количества элементов в массиве и поле для массива, а также методы для работы с массивом в соответствии с лабораторной работой 3. В классе должен быть конструктор без параметров, задающий по умолчанию число элементов в массиве равным 10, и конструктор с параметрами, в который передается число элементов, вводимых пользователем. **В основном классе программы необходимо продемонстрировать создание экземпляра класса с помощью конструктора с параметрами и с помощью конструктора без параметров.**

3. Статический класс, содержащий методы для проверки вводимых данных. Класс должен содержать методы, обеспечивающие контроль ввода.

4. Класс, содержащий реализацию игры. Класс должен содержать константу, определяющую размер поля, константы, определяющие обозначения элементов полей и методы в соответствии с лабораторной работой 4.

Требования к XML-документированию:
Каждый класс и его публичные члены должны быть документированы с помощью XML-комментариев, поясняющих их назначение и параметры.

Теоретические вопросы:

1. Назовите основные принципы ООП и расскажите о них подробно.
2. Чем класс отличается от объекта?
3. Какие модификаторы доступа используются в C#? В чём разница между ними?
4. Какие члены класса может содержать класс?
5. Поля. Синтаксис. Каким уровнем доступа следует наделять поля класса?
6. Конструктор класса. Для чего предназначен конструктор класса? Может ли конструктор возвращать значение?
7. Свойства. Синтаксис. Когда в свойствах выполняется блок с методом get? Когда в свойствах выполняется блок с методом set? Всегда ли в свойствах определены оба метода get и set?
8. В чём отличие конструктора от метода?
9. Как создать экземпляр класса? Опишите и создайте экземпляр класса Circle.
10. Что такое статический класс? Чем статический класс отличается от нестатического класса? Для чего нужен статический класс?

Практические задачи:

1. **Класс «Прямоугольник».** Создайте класс Rectangle с приватными полями для длины и ширины. Реализуйте свойства для доступа к этим полям, методы для вычисления площади и периметра, а также метод для проверки, является ли прямоугольник квадратом. Добавьте несколько конструкторов: один без параметров (устанавливающий значения по умолчанию), другой с параметрами. Продемонстрируйте работу класса в Main, создав несколько экземпляров с разными параметрами.
2. **Класс «Книга»**. Создайте класс Book с полями для названия книги, автора, цены и года выпуска. Реализуйте свойства для доступа к полям и метод для получения информации о книге в формате строки. Добавьте конструкторы: один по умолчанию, другой — для всех полей, и третий — с основными данными. В Main создайте несколько объектов Book с разными параметрами.
3. **Класс «Студент».** Создайте класс Student с полями для имени, возраста и списка оценок. Реализуйте методы для добавления и удаления оценок и для вычисления среднего балла. Создайте несколько конструкторов: один для инициализации только имени, другой для имени и возраста, третий для имени, возраста и начальных оценок. В Main создайте несколько объектов Student с разными параметрами и продемонстрируйте их методы.
4. **Класс «Банк»**. Реализуйте класс BankAccount с полями для номера счета, имени владельца и текущего баланса. Добавьте конструкторы: один по умолчанию, второй для инициализации счета и владельца, третий для всех полей. В Main создайте несколько объектов BankAccount и продемонстрируйте методы пополнения, снятия и проверки баланса.
5. **Класс «Калькулятор».** Создайте статический класс Calculator с методами для выполнения арифметических операций и методами для вычисления корня и факториала. В Main вызовите методы Calculator для демонстрации расчетов с разными данными.
6. **Класс «Точка и Линия».** Создайте класс Point для представления точки на координатной плоскости и несколько конструкторов: один без параметров (для инициализации точки в начале координат), другой с параметрами для установки

координат. Создайте класс Line, который использует Point и имеет методы для расчета длины линии. В Main продемонстрируйте создание и использование Line с различными точками.

7. **Класс «Автомобиль».** Создайте класс Car с полями для марки, модели, пробега и топлива. Реализуйте несколько конструкторов: по умолчанию, с двумя параметрами, и с четырьмя параметрами. В Main создайте несколько объектов Car, вызовите методы заправки и поездки, а также выведите их информацию.
8. **Класс «Инвентарь магазина».** Создайте класс Product с полями для названия, количества и цены. Реализуйте несколько конструкторов: по умолчанию, с названием и ценой, а также для всех полей. Создайте класс Inventory, хранящий список продуктов, и методы для добавления, удаления и отображения продуктов. В Main добавьте и удалите несколько продуктов, демонстрируя работу методов.
9. **Класс «Круг и Цилиндр».** Создайте класс Circle с полем радиуса и методами для вычисления площади и длины окружности. Добавьте конструкторы: один по умолчанию и один с параметром радиуса. Создайте класс Cylinder, наследующий Circle, с методом для вычисления объема. В Main создайте несколько объектов Circle и Cylinder с разными параметрами и продемонстрируйте работу.
10. **Класс «Игрок».** Создайте класс Player с полями для имени, уровня и опыта, с несколькими конструкторами: по умолчанию, с именем и с именем и уровнем. В Main создайте несколько объектов Player, увеличьте опыт и уровень, и выведите статистику каждого игрока.
11. **Класс «Стек».** Реализуйте класс Stack, представляющий стек. Класс должен иметь конструктор без параметров и конструктор, инициализирующий стек начальным набором элементов. В Main создайте несколько стеков с разными начальными значениями и продемонстрируйте работу методов Push, Pop, Peek.
12. **Класс «Фигура и Полиморфизм»**. Создайте абстрактный класс Shape с методом CalculateArea. Создайте Rectangle и Circle, которые переопределяют CalculateArea. Добавьте конструкторы для всех классов. В Main создайте список фигур, вычислите и выведите их площади.
13. **Класс «Часы».** Создайте класс Clock с полями для часов, минут и секунд и несколькими конструкторами: по умолчанию (начальное время — полночь), с часами и минутами и для всех полей. В Main продемонстрируйте создание нескольких объектов Clock и их метод инкремента времени.
14. **Класс «Зоопарк» и Наследование.** Создайте класс Animal с полем для имени и методом MakeSound. Наследники Dog, Cat, Bird переопределяют метод MakeSound. Добавьте конструкты для Animal и наследников. В Main создайте Zoo (список животных) и вызовите звуки каждого животного.
15. **Класс «Умный дом».** Создайте класс SmartHome с полями для температуры, уровня влажности, статуса сигнализации и освещения. Добавьте несколько конструкторов: без параметров, с температурой и влажностью, и с полными настройками. В Main создайте несколько объектов SmartHome, управляйте параметрами и продемонстрируйте работу класса.

Дополнительные материалы для подготовки:

- **https://metanit.com/sharp/tutorial/3.1.php**
- **https://metanit.com/sharp/tutorial/3.35.php**

- https://metanit.com/sharp/tutorial/1.4.php
- https://metanit.com/sharp/tutorial/3.2.php
- https://metanit.com/sharp/tutorial/3.4.php
- https://metanit.com/sharp/tutorial/3.6.php
- https://tproger.ru/translations/diving-in-oop-p1
- **https://devpractice.ru/c-sharp-lesson-9-oop-classes/**