**Лабораторная работа №7**

* 1. ***Тема: Определение классов, статические конструкторы, конструкторы, методы-свойства.***
  2. ***Задание 1. Методы-свойства.***

Методы, называемые свойствами (Properties), представляют специальную синтаксическую конструкцию, предназначенную для обеспечения эффективной работы со свойствами. При работе со свойствами объекта (полями) часто нужно решить, какой модификатор доступа использовать, чтобы реализовать нужную стратегию доступа к полю класса.

5 наиболее употребительных стратегий:

1. чтение, запись (Read, Write);
2. чтение, запись при первом обращении (Read, Write-once);
3. только чтение (Read-only);
4. только запись (Write-only);
5. ни чтения, ни записи (Not Read, Not Write).

Открытость свойств (атрибут public) позволяет реализовать только первую стратегию. В языке C# принято, как и в других объектных языках, свойства объявлять закрытыми, а нужную стратегию доступа организовывать через методы. Для эффективности этого процесса и введены специальные методы-свойства.

1. В Visual Studio создайте проект консольного приложения. (C#)
2. Создайте класс с именем Person, у которого пять полей: fam, status, salary, age, health, характеризующих соответственно фамилию, статус, зарплату, возраст и здоровье персоны.

Для каждого из этих полей может быть разумной своя стратегия доступа. Возраст доступен для чтения и записи, фамилию можно задать только один раз, статус можно только читать, зарплата недоступна для чтения, а здоровье закрыто для доступа и только специальные методы класса могут сообщать некоторую информацию о здоровье персоны.

1. Обеспечьте доступ к закрытому полю fam класса Person следующим образом:

// стратегия: Read,Write-once (Чтение, запись при первом обращении)

public string Fam

{

set {if (fam == "") fam = value;}

get {return(fam);}

}

Метод get возвращает значение закрытого поля, метод set - устанавливает значение, используя передаваемое ему значение в момент вызова, хранящееся в служебной переменной со стандартным именем value.

1. Реализуйте стратегии доступа к остальным полям класса Person.
2. В методе Main консольного приложения напишите пример, показывающий создание и работу с полями персоны.
3. Запустите консольное приложение, чтобы убедиться, что оно работает корректно.
   1. ***Задание №2. Проектирование класса Rational, описывающего рациональные числа.***

Класс Rational определяет новый тип данных - рациональные числа и основные операции над ними - сложение, умножение, вычитание и деление. Рациональное число задается парой целых чисел (m,n) и изображается обычно в виде дроби m/n. Число m называется числителем, n - знаменателем. Для каждого рационального числа существует множество его представлений, например, 1/2, 2/4, 3/6, 6/12 - задают одно и тоже рациональное число. Среди всех представлений можно выделить то, в котором числитель и знаменатель взаимно несократимы. Такой представитель будет храниться в полях класса. Операции над рациональными числами определяются естественным для математики образом

1. В Visual Studio создайте проект консольного приложения. (C#)
2. Создайте класс с именем Rational.
3. В классе Rational определите закрытые поля:

m-числитель

n-знаменатель

1. Определите метод, вычисляющий наибольший общий делитель пары чисел. **Этот метод понадобится не только конструктору класса, но и всем операциям над рациональными числами.**
2. Определите конструктор с аргументами, которому будут передаваться два целых: числитель и знаменатель создаваемого числа

**Методические указания:**

Конструктор класса. Создает рациональное число m/n, эквивалентное a/b, но с взаимно несократимыми числителем и знаменателем. Если b=0, то результатом является рациональное число 0 -пара (0,1).

1. Переопределите метод ToString();
2. Напишите функцию PrintRational(), позволяющую выводить на печать данные о классе.
3. Для проверки корректности создания рациональных чисел в методе Main консольного приложения напишите код, который создает и печатает несколько рациональных чисел.
4. Запустите консольное приложение, чтобы убедиться, что оно работает корректно.
   1. ***Задание №3. Операции над рациональными числами***

Определим над рациональными числами стандартный набор операций - сложение и вычитание, умножение и деление. Реализуем эти операции методами с именами Plus, Minus, Mult, Divide соответственно. Поскольку рациональные числа - это, прежде всего именно числа, то для выполнения операций над ними часто удобнее пользоваться привычными знаками операций (+, -, \*, /). Язык C# допускает определение операций, заданных указанными символами. Этот процесс называется **перегрузкой операций**.

1. Откройте консольное приложение созданное в задании 1.
2. Реализуйте Метод Plus.

**Методические указания:**

По правилам сложения дробей вычисляется числитель и знаменатель результата, и эти данные становятся аргументами конструктора, создающего требуемое рациональное число, которое удовлетворяет правилам класса.

1. Перегрузите операцию «+» класса Rational.

**Методические указания:**

Именем соответствующего метода является сам знак операции, которому предшествует ключевое слово operator. Важно также помнить, что операция является статическим методом класса с атрибутом static.

1. Аналогичным образом реализуйте методы Minus, Mult, Divide и перегрузите операции «-», «\*», «/»над рациональными числами.
2. Проверьте работу всех операций, вызывая метод или операцию в зависимости от конкретной ситуации.

**Обратите внимание**: при перегрузке операций сохраняется общепринятый приоритет операций. Поэтому при вычислении выражения r3+r4\*r5 вначале будет выполняться умножение рациональных чисел, а потом уже сложение.

* 1. ***Задание №4. Константы класса Rational.***

Рассмотрим важную проблему определения констант в собственном классе. Определим две константы 0 и 1 класса Rational. Кажется, что сделать это невозможно из-за ограничений, накладываемых на объявление констант. Напомню, константы должны быть инициализированы в момент объявления, и их значения должны быть заданы константными выражениями, известными в момент компиляции. Но в момент компиляции у класса Rational нет никаких известных константных выражений. Как же быть? Справиться с проблемой поможет **статический конструктор**, созданный для решения подобных задач. Роль констант класса будут играть статические поля, объявленные с атрибутом readonly, то есть доступные только для чтения. Нам также будет полезен закрытый конструктор класса. Еще укажем, что введение констант класса требует использования экзотических средств языка C#.

1. Откройте консольное приложение созданное в 2-х предыдущих заданиях.
2. Определите закрытый конструктор:

private Rational(int a, int b, string t)

{

m = a; n = b;

}

При перегрузке методов (в данном случае конструкторов) сигнатуры должны различаться, и поэтому пришлось ввести дополнительный аргумент t для избежания конфликтов. Поскольку конструктор закрытый, то гарантируется корректное задание аргументов при его вызове.

1. Определите константы Zero и One класса Rational, которые задаются статическими полями с атрибутом readonly.
2. Задайте статический конструктор, в котором определяются значения констант: static Rational()

{

…

}

1. Добавьте в класс булевы операции над рациональными числами - равенство и неравенство, больше и меньше. При этом две последние операции сделайте перегруженными, позволяя сравнивать рациональные числа с числами типа double.
2. Создайте функцию TestRational(), демонстрирующую работу с константами, булевыми и арифметическими выражениями над рациональными числами.
3. Запустите консольное приложение, чтобы убедиться, что оно работает корректно.
   1. ***Контрольные вопросы:***
4. Как вызываются статические поля и методы?
5. Как и когда происходит создание объектов?
6. Зачем классу нужно несколько конструкторов?
7. Для чего нужен статический конструктор? Для чего нужен закрытый конструктор?
8. Что понимается под перегрузкой операции?
9. Назовите методы, которые используются для работы со свойствами?

7. Зачем необходим аргумент t при объявлении конструктора в 4 задании:

private Rational(int a, int b, string t)

{

m = a; n = b;

}

* 1. ***Выполнение лабораторной работы***

1. Лабораторная работа выполняется в аудитории вместе с преподавателем;
2. Время выполнения лабораторной работы 80 минут;
3. Отметка о выполнении должна быть поставлена на занятии. Защищать лабораторную работу и отвечать на вопросы можно на следующем занятии;

Если лабораторная работа выполнена в срок (даже если не защищена), то максимальное количество баллов – 100%, в случае не выполнения лабораторной работы на занятии (в т.ч. и в срок) максимальное количество баллов – 80%;