Карагандинский технический университет имени Абылкаса Сагинова

Кафедра: ИВС

**Лабораторная работа №1**

По дисциплине: «Технология программирования»

Тема: Разработка программного обеспечения при объектно-ориентированном подходе.

**Преподователь:**

Чангариди Ф. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*подпись*) (*дата*)

**Выполнил:**

Ст.гр. ИС23-2С Амантаев А. С.

Караганда 2024

Задание №1

Вариант-1

Постpоить класс Дата-Вpемя. Класс должен обеспечивать пpостейшие функции для pаботы с данными классами: увеличение/уменьшение на 1 день, час, минуту, секунду и т.д., изменение значений, вывод значений.

Код программы:

using System;

public class DateTimeHandler

{

private DateTime dateTime;

public DateTimeHandler()

{

dateTime = DateTime.Now;

}

public DateTimeHandler(int year, int month, int day, int hour = 0, int minute = 0, int second = 0)

{

dateTime = new DateTime(year, month, day, hour, minute, second);

}

public void AddOneDay()

{

dateTime = dateTime.AddDays(1);

}

public void SubtractOneDay()

{

dateTime = dateTime.AddDays(-1);

}

public void AddOneHour()

{

dateTime = dateTime.AddHours(1);

}

public void SubtractOneHour()

{

dateTime = dateTime.AddHours(-1);

}

public void AddOneMinute()

{

dateTime = dateTime.AddMinutes(1);

}

public void SubtractOneMinute()

{

dateTime = dateTime.AddMinutes(-1);

}

public void AddOneSecond()

{

dateTime = dateTime.AddSeconds(1);

}

public void SubtractOneSecond()

{

dateTime = dateTime.AddSeconds(-1);

}

public void SetDateTime(int year, int month, int day, int hour = 0, int minute = 0, int second = 0)

{

dateTime = new DateTime(year, month, day, hour, minute, second);

}

public DateTime GetDateTime()

{

return dateTime;

}

public void PrintDateTime()

{

Console.WriteLine(dateTime.ToString("yyyy-MM-dd HH:mm:ss"));

}

}

class Program

{

static void Main(string[] args)

{

DateTimeHandler dtHandler = new DateTimeHandler(2023, 9, 23, 12, 30, 45);

dtHandler.PrintDateTime();

dtHandler.AddOneDay();

dtHandler.PrintDateTime();

dtHandler.SubtractOneHour();

dtHandler.PrintDateTime();

dtHandler.SetDateTime(2024, 1, 1, 0, 0, 0);

dtHandler.PrintDateTime();

}

}

Результат программы:

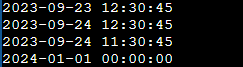


Рисунок 1

В данной лабораторной работе были использованы:

1. Классы

DateTimeHandler предоставляет основные функции для работы с датой и временем. Этот класс содержит методы для увеличения/уменьшения дня, часа, минуты, секунды, а также для изменения и вывода значений.

2. Методы

AddOneDay, SubtractOneDay — методы для увеличения или уменьшения даты на 1 день.

AddOneHour, SubtractOneHour — методы для увеличения или уменьшения времени на 1 час.

AddOneMinute, SubtractOneMinute — методы для увеличения или уменьшения времени на 1 минуту.

AddOneSecond, SubtractOneSecond — методы для увеличения или уменьшения времени на 1 секунду.

3. Структуры

SetDateTime — позволяет задать конкретную дату и время.

GetDateTime — возвращает текущую дату и время в виде объекта DateTime.

PrintDateTime — выводит дату и время в формате yyyy-MM-dd HH:mm:ss.

Контрольные вопросы:

1. Что такое класс? Для чего описываются классы?

Класс — шаблон для создания объектов. В классе описываются свойства и методы, которые будут присущи объектам этого класса. Позволяет объединить данные и функции, работающие с этими данными, в одну логическую сущность. Классы помогают организовать и структурировать код, обеспечивая повторное использование кода и инкапсуляцию данных.

2. Чем отличается класс от структуры?

Классы - это ссылочный тип, поддерживают как интерфейсное, так и классовое наследование, то есть класс может наследовать поведение другого класса.

Структуры - значимый тип, могут реализовывать интерфейсы, но не могут наследовать от других структур или классов. Также структуры не могут быть базовыми для других структур или классов.

3. Какие модификаторы используются для ограничения доступа к членам класса?

C# предоставляет следующие модификаторы доступа:

- public: доступен в любом месте программы.

- private: доступен только внутри данного класса.

- protected: доступен в классе и его наследниках.

- internal: доступен только в пределах текущей сборки.

- protected internal: доступен либо в пределах сборки, либо в наследниках.

4. Для чего в классе определяется конструктор? Сколько может быть конструкторов в классе? Когда вызывается конструктор?

Конструктор служит для инициализации объекта в момент его создания. В классе можно определить несколько конструкторов с различными наборами параметров. Конструктор автоматически вызывается при создании экземпляра класса.

5. Может ли быть конструктор описан в области доступа private?

Конструктор может быть объявлен с модификатором private. Это используется для реализации паттерна Singleton, чтобы запретить создание экземпляров класса из внешнего кода.

6. Что будет, если в классе не определен конструктор?

Если в классе отсутствует явный конструктор, компилятор автоматически сгенерирует конструктор по умолчанию без параметров, который просто вызывает конструктор родительского класса.

7. Для чего в классе определяется деструктор? Сколько может быть деструкторов в классе? Когда вызывается деструктор?

Деструктор нужен, чтобы освободить ресурсы перед тем, как объект удалится из памяти. В классе можно иметь только один деструктор, и он вызывается автоматически, когда запускается сборка мусора (garbage collection).

8. Может ли быть деструктор описан в области доступа private?

В C# деструкторы автоматически считаются private, так что их нельзя вызвать вручную — они срабатывают только, когда происходит сборка мусора.

9. Что будет, если в классе не определен деструктор?

Если деструктор не задан, сборка мусора сама освободит память, занятую объектом, но другие ресурсы, такие как файлы или подключения к базе данных, могут остаться неосвобожденными, если их не закрыть вручную.

10. Что такое конструктор, деструктор по умолчанию? Когда они вызываются?

Конструктор по умолчанию — это конструктор без параметров, который компилятор создаёт сам, если в классе нет других конструкторов. Он вызывается, когда создаётся объект.

Деструктор по умолчанию — это деструктор, который тоже генерируется компилятором, если его не написать. Он автоматически срабатывает при сборке мусора, когда объект удаляется из памяти.

11. Что такое экземпляр класса?

Экземпляр класса — это объект, созданный на основе класса, который использует его поля и методы.

12. Как можно обратиться к членам-данным и членам-функциям класса?

Чтобы обратиться к членам-данным и функциям класса, нужно сначала создать объект этого класса. Потом через этот объект можно получить доступ к его переменным и методам с помощью оператора точки (.). Например:

Класс obj = new Класс();

obj.членДанных; // доступ к переменной

obj.членФункция(); // вызов метода

Если члены класса статические, то к ним можно обратиться напрямую через имя класса, без создания объекта

Класс.статическаяПеременная;

Класс.статическаяФункция();

13. Для чего в классе задаются статические члены? В чем отличие их использования от обычных членов класса?

Статические члены в классе задаются для того, чтобы иметь доступ к ним без необходимости создавать экземпляр этого класса. Это удобно для хранения данных или функций, которые должны быть общими для всех объектов этого класса.