**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное образовательное бюджетное учреждение высшего образования**

**«Вологодский государственный университет»**

**(ВоГУ)**

* 1. **ОБЪЕКТНО-ОРЕНТИРОВАННОЕ программирование**
  2. **лабораторнАЯ РАБОТА №6**

**«Создание классов на языке C#»**

**Направление подготовки: 09.03.01 Информатика и вычислительная техника**

**Направленность (профиль): Программное обеспечение средств вычислительной техники и автоматизированных систем**

**Форма обучения: очная**

**Институт: Математики, естественных и компьютерных наук**

**Кафедра: Автоматики и вычислительной техники**

**Группа: 4Б09 ПО-41**

**Студент: Махонин М.Н.**

**Руководитель: Сорокин А.Н.**

Вологда

2021 г.

Цель работы: научиться разрабатывать классы на языке C#.

**Вариант 2**

1. Создать базовый класс (абстрактный класс или интерфейс) с виртуальными функциями ввода и вывода.

2. Создать производные классы: класс «дата», класс «время».

В производных классах определить: скрытые поля, конструктор по умолчанию, конструктор с параметрами (с проверкой на корректность), метод для установки значений полей (с проверкой на корректность), метод для получения значений полей, методы для ввода и вывода.

3. Разработать демонстрационный класс. В этом классе определить массив или коллекцию ссылок на объекты базового класса, методы для добавления и удаления экземпляров классов, метод для вывода всех элементов.

4. Продемонстрировать работоспособность всех методов классов.

5. Разработать программу, которая демонстрировала бы возможности сериализации. Для проверки введите данные с клавиатуры, сохраните информацию в файл (требуется сохранить несколько объектов разных классов из задания 1), затем прочитайте данные в цикле и выведите информацию на экран (в этом цикле неизвестно количество и порядок объектов в файле).

using System;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

namespace Laba

{

    interface Base

    {

        public void Print();

        public void Input(params int[] p);

    }

    [Serializable]

    public class Date : Base

    {

        int \_day;

        int \_month;

        int \_year;

        public Date()

        {

            \_day = 0;

            \_month = 0;

            \_year = 0;

        }

        public Date(int day, int month, int year)

        {

            Day = day;

            Month = month;

            Year = year;

        }

        public int Day

        {

            get { return \_day; }

            set

            {

                if(value < 0 || value > 31)

                {

                    throw new Exception("Недопустимое значение");

                }

                \_day = value;

            }

        }

        public int Month

        {

            get { return \_month; }

            set

            {

                if (value < 0 || value > 12)

                {

                    throw new Exception("Недопустимое значение");

                }

                \_month = value;

            }

        }

        public int Year

        {

            get { return \_year; }

            set

            {

                if (value < 0)

                {

                    throw new Exception("Недопустимое значение");

                }

                \_year = value;

            }

        }

        public void Input(params int[] p)

        {

            if(p.Length < 3)

            {

                throw new Exception("Количество параметров неверно");

            }

            Day = p[0];

            Month = p[1];

            Year = p[2];

        }

        public void Print()

        {

            Console.WriteLine($"День: {Day} Месяц: {Month} Год: {Year}");

        }

    }

    [Serializable]

    public class Time : Base

    {

        int \_second;

        int \_minute;

        int \_hour;

        public Time()

        {

            \_second = 0;

            \_minute = 0;

            \_hour = 0;

        }

        public Time(int second, int minute, int hour)

        {

            Second = second;

            Minute = minute;

            Hour = hour;

        }

        public int Second

        {

            get { return \_second; }

            set

            {

                if (value < 0 || value > 60)

                {

                    throw new Exception("Недопустимое значение");

                }

                \_second = value;

            }

        }

        public int Minute

        {

            get { return \_minute; }

            set

            {

                if (value < 0 || value > 60)

                {

                    throw new Exception("Недопустимое значение");

                }

                \_minute = value;

            }

        }

        public int Hour

        {

            get { return \_hour; }

            set

            {

                if (value < 0 || value > 24)

                {

                    throw new Exception("Недопустимое значение");

                }

                \_hour = value;

            }

        }

        public void Input(params int[] p)

        {

            if (p.Length < 3)

            {

                throw new Exception("Количество параметров неверно");

            }

            Second = p[0];

            Minute = p[1];

            Hour = p[2];

        }

        public void Print()

        {

            Console.WriteLine($"Секунд: {Second} Минут: {Minute} Часов: {Hour}");

        }

    }

    class Test

    {

        Base[] bases;

        Random r = new Random();

        public Test()

        {

            bases = new Base[10];

            int x = 0; ;

            for(int i = 0; i < 10; i++)

            {

                x = r.Next(0, 2);

                if(x == 0)

                {

                    bases[i] = new Date(r.Next(0, 32), r.Next(0, 13), r.Next(10001));

                }

                else

                {

                    bases[i] = new Time(r.Next(0, 61), r.Next(0, 61), r.Next(0, 24));

                }

            }

        }

        public void Print()

        {

            foreach(var e in bases)

            {

                e.Print();

            }

        }

        public void Change()

        {

            foreach (var e in bases)

            {

                if (e is Date)

                {

                    e.Input(r.Next(0, 32), r.Next(0, 13), r.Next(10001));

                }

                else

                {

                    e.Input(r.Next(0, 61), r.Next(0, 61), r.Next(0, 24));

                }

            }

        }

        public void InvokeSerialize()

        {

            BinaryFormatter binary = new BinaryFormatter();

            using (FileStream fs = new FileStream("ser.data", FileMode.OpenOrCreate))

            {

                foreach (var e in bases)

                {

                    binary.Serialize(fs, e);

                }

            }

            Console.WriteLine("Объекты сериализованы");

        }

        public void InvokeDeserialize()

        {

            BinaryFormatter formatter = new BinaryFormatter();

            try

            {

                using(FileStream fs = new FileStream("ser.data", FileMode.OpenOrCreate))

                {

                    while (true)

                    {

                        Base b = (Base)formatter.Deserialize(fs);

                        b.Print();

                    }

                }

            }

            catch (Exception e)

            {

                Console.WriteLine("Десериализация выполнена");

            }

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            Test t = new Test();

            Console.WriteLine("Проверка вывода");

            t.Print();

            Console.WriteLine("-------------------------------------------------------");

            Console.WriteLine("Проверка ввода");

            t.Change();

            t.Print();

            Console.WriteLine("-------------------------------------------------------");

            t.InvokeSerialize();

            Console.WriteLine("-------------------------------------------------------");

            t.InvokeDeserialize();

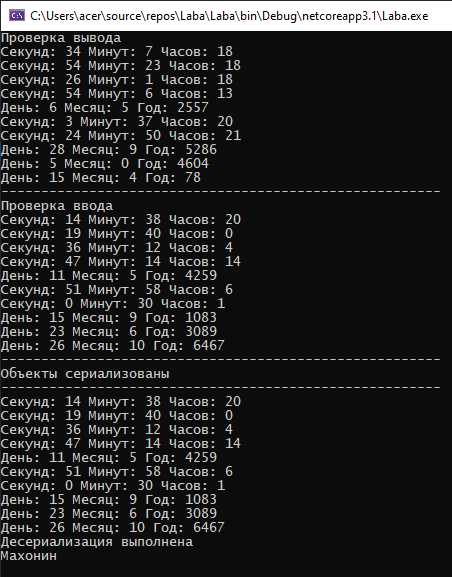
            Console.ReadLine();

        }

    }

}

Вывод:



**Контрольные вопросы**

1. **Опишите элементы классов, специфичные для языка C#.**

***Свойства***. Свойства созданы как комбинация метода и поля в одном члене типа для лаконичности и безопасности кода.

*Аксессор get*

Если в свойстве использован акссессор, то подобно методу, свойство должно возвращать значение поля, которое акссессор считывает. Тип поля должен соответствовать типу возвращаемого значения свойства.

*Мутатор set*

Мутатор, или метод set по спецификации вызова поход на метод с возвращаемым значение типа void и одним параметром. public void SetValue (object value). Среда выполнения сама его вызывает в точке указанной программистом с прописанной им реализацией.

При реализации мутатора в блоке кода свойства нужно помнить о неявном участии параметра value, который выполняет функцию буфера, промежуточной переменной, значение которой нужно предать полю типа.

***Индексаторы*** – это синтаксический инструмент языка C#, который позволяет индексировать внутренние коллекции типа.

Используются в том случае, если тип содержит в себе коллекцию.

Индексаторы – средство инкапсуляции коллекции внутри типа. С одной стороны, для программиста клиентского кода, это – удобное средство для изменения и считывания значений коллекций. С другой стороны, для программиста типа, индесаторы – средство самостоятельно реализовать любой алгоритм работы с элементами коллекции, гибкий и безопасный элемент доступа к этим элементам.

Синтаксис объявления и реализации индексаторов схож с объявлением свойств. Визуальное отличие в том, что индексаторы принимают параметры.