Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Рязанский государственный радиотехнический
университет имени В.Ф. Уткина»

Кафедра «Электронные вычислительные машины»

Отчёт о лабораторной работе №2
«Переопределение методов и операторов»
по дисциплине
«Визуальное программирование»

Выполнили:
студенты группы 340
бригады №3
Денисов А.В.
Стежкин П.А.
Проверили:
ст. преп. каф. ЭВМ
Хизриева Н.И.
ст. преп. каф. ЭВМ
Бастрычкин А.С.

Рязань 2026

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

1) Разработать класс Person, который будет хранить информацию:

− Фамилия

− Имя

− Отчество (может отсутствовать)

− Дата рождения

− Адрес

− Номер телефона

Необходимо переопределить метод ToString (возвращает ФИО). Также нужно определить виртуальный метод GetInfo, который возвращает в виде строки полную информацию о персоне.

2) Разработать класс Student (унаследовать от Person), который будет хранить информацию:

− Номер зачетной книжки (ID)

− Гражданство

− Форма обучения (перечисление: бюджет, коммерция, целевое)

Необходимо переопределить методы Equals и GetInfo. Метод Equals должен возвращать true, если все данные у двух студентов совпадают. Метод GetInfo возвращает в виде строки всю информацию о студенте. Также нужно переопределить операторы == и != (логика как в Equals).

3) 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 (get) к студентам по номеру зачетной книжки. Предусмотреть вывод

исключений, если пользователь пытается добавить студента с уже существующим номером зачетки или удалить несуществующего студента.

Для поиска по списку студентов можно использовать методы First, Where, Any (почитать документацию).

Метод ToString должен возвращать номер группы и список студентов в алфавитном порядке.

Код программы

```
namespace lab_2;

public class Person
{
   public string FirstName { get; }

   public string LastName { get; }

   public string? Patronymic  { get; }

   public DateTime DateOfBirth { get; }

   public string Adress { get; }

   public string PhoneNumber { get; }

   public Person(string firstName,  string lastName, string patronymic, DateTime dateOfBirth, string adress, string phoneNumber)
   {
      FirstName = firstName;
      LastName = lastName;
      Patronymic = patronymic;
      DateOfBirth = dateOfBirth;
      Adress = adress;
      PhoneNumber = phoneNumber;
   }

   public override bool Equals(object? obj)
   {
      if (obj is Person p)
      {
         if (FirstName == p.FirstName && LastName == p.LastName && Patronymic == p.Patronymic && DateOfBirth == p.DateOfBirth && Adress == p.Adress && PhoneNumber == p.PhoneNumber)
         {
            return true;
         }

      }
      return false;
```

```
    }

    public override int GetHashCode()
    {
        return HashCode.Combine(FirstName, LastName, Patronymic, DateOfBirth, Adress,
PhoneNumber);
    }


    public override string ToString()
    {
        return $"{FirstName} {LastName} {Patronymic}";
    }

    public virtual string GetInfo()
    {
        return $"{FirstName} {LastName} {Patronymic} \n {DateOfBirth} \n {Adress} \n
{PhoneNumber}";
    }
}

namespace lab_2;

public enum EducationForm
{
    Budget,
    Commercial,
    Targeted
}

namespace lab_2;

public class Student : Person
{
    public int Id { get; }

    public string Nationality { get; }

    public EducationForm EducationForm { get; }

    public Student(string firstName,  string lastName, string patronymic,
        DateTime dateOfBirth, string adress, string phoneNumber, int id, string nationality, EducationForm
educationForm)
        : base(firstName,  lastName, patronymic, dateOfBirth, adress, phoneNumber)
    {
        Id = id;
        Nationality = nationality;
        EducationForm = educationForm;
```

```
    }

    public override string GetInfo()
    {
        return $"{base.GetInfo()} \n {Id} \n {Nationality} \n {EducationForm} ";
    }

    public override bool Equals(object? obj)
    {
        if (obj is Student student)
        {
            if (Id == student.Id && Nationality == student.Nationality && EducationForm ==
student.EducationForm && base.Equals(obj))
            {
                return true;
            }
        }

        return false;
    }

    public static bool operator ==(Student student1, Student student2)
    {
        if (student1.Equals(student2))
        {
            return true;
        }

        return false;
    }

    public static bool operator !=(Student student1, Student student2)
    {
        if (!student1.Equals(student2))
        {
            return true;
        }

        return false;
    }

}

namespace lab_2;
```

```
public class Group
{
  public int GroupId { get; }

  public Student[] Students { get; set; } = Array.Empty<Student>();

  public Group(int groupId)
  {
    GroupId = groupId;
  }

  public void Add(Student student)
  {
    if (Students.Any(s => s.Id == student.Id))
    {
      throw new Exception($"Студент с №{student.Id} уже существует");
    }

    Student[] newStudents = new Student[Students.Length + 1];

    for (int i = 0; i < Students.Length; i++)
    {
      newStudents[i] = Students[i];
    }

    newStudents[newStudents.Length - 1] = student;

    Students = newStudents;
  }

  public void Remove(int id)
  {
    if (!Students.Any(s => s.Id == id))
    {
      throw new Exception($"Студента по №{id} не существует");
    }

    Students = Students.Where(s => s.Id != id).ToArray();
  }

  public string this[int id]
  {
    get
```

```
        {
            var student = Students.First(s => s.Id == id);

            if (student == null)
            {
                throw new Exception($"Зачетка №{id} не найдена");
            }

            return student.GetInfo();
        }
    }

    public override string ToString()
    {
        var sortedStudents = Students.OrderBy(s => s.LastName);

        string result = $"Список группы №{GroupId}:\n";

        foreach (var st in sortedStudents)
        {
            result += "\n" + st.GetInfo() + "\n";
        }

        return result;
    }
}

using lab_2;

var person = new Person(
    "Алексей",
    "Абакумов",
    "Викторович",
    new DateTime(2005, 7, 15),
    "ул. Гагарина, д.60",
    "+79001234533"
);

var student1 = new Student(
    "Иван",
    "Иванов",
    "Иванович",
    new DateTime(2003, 5, 15),
    "ул. Пушкина, д. 10",
```

```
    "+79001234567",
    1,
    "Русский",
    EducationForm.Budget
);

var student2 = new Student(
    "Петр",
    "Сергеев",
    "Иванович",
    new DateTime(2005, 7, 15),
    "ул. Пушкина, д. 15",
    "+79001238767",
    2,
    "Серб",
    EducationForm.Commercial
);

Group group340 = new Group(340);

group340.Add(student1);
group340.Add(student2);

Console.WriteLine("Метод ToString() класса Person:");
Console.WriteLine(person.ToString());

Console.WriteLine("Метод ToString() класса Group:");
Console.WriteLine(group340.ToString());

Console.WriteLine("Переопределние индексатора для класса Group:");
Console.WriteLine(group340[1]);

Console.WriteLine("Метод GetInfo() у класса Student:");
Console.WriteLine(student1.GetInfo());

Console.WriteLine("Метод Equals() у класса Student:");
Console.WriteLine(student2.Equals(student1));

Console.WriteLine("Переопределние операций == и != :");
Console.WriteLine(student1 != student2);
```

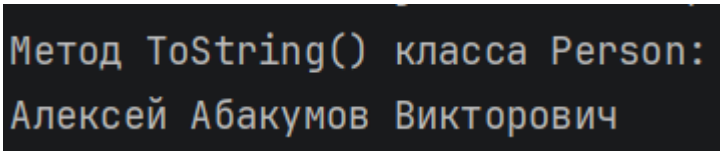

Рисунок 1 – Метод ToString класса Person

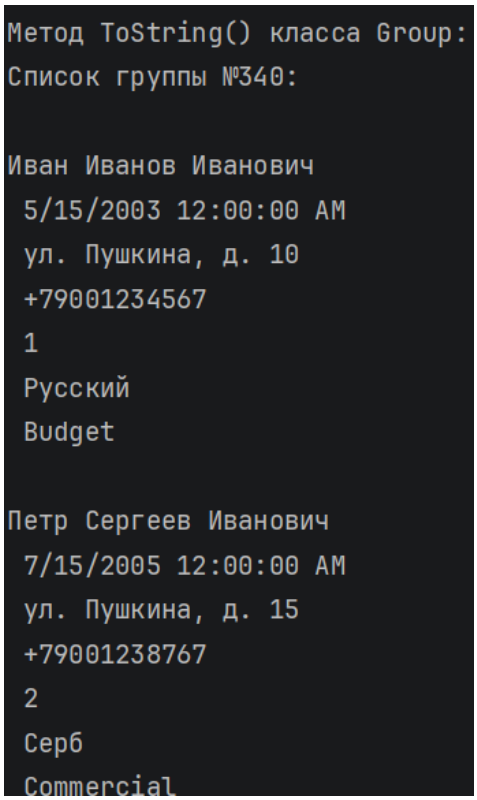

Рисунок 2 – Метод ToString класса Group

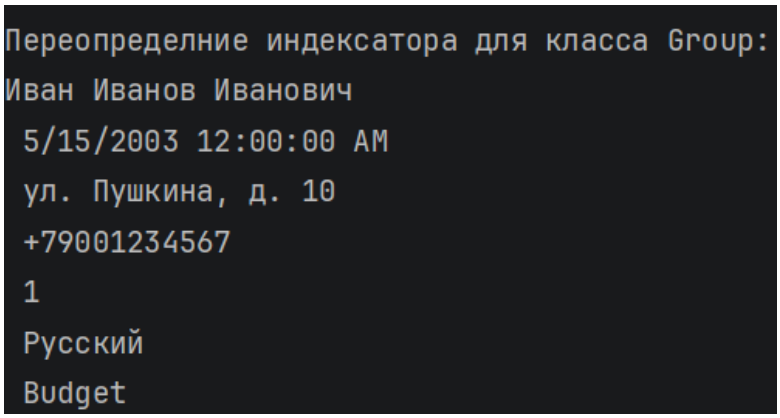

Рисунок 3 – Переопределние индексатора для класса Group

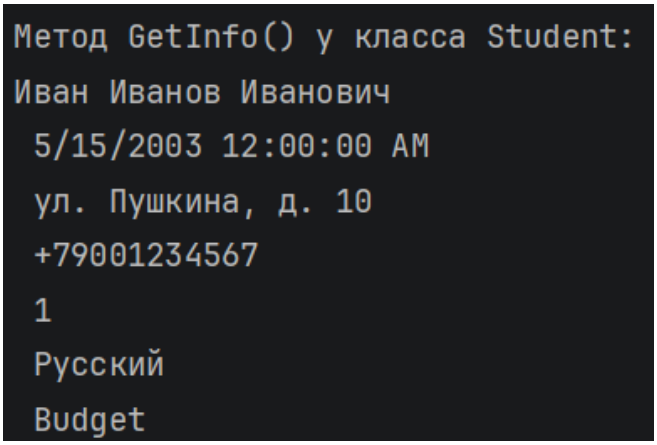

Рисунок 4 – Метод GetInfo() класса Student

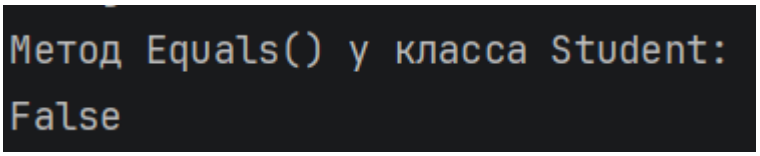

Рисунок 5 – Метод Equals() класса Student

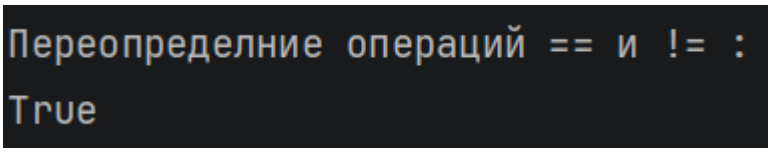

Рисунок 6 – Переопределние операций == и !=

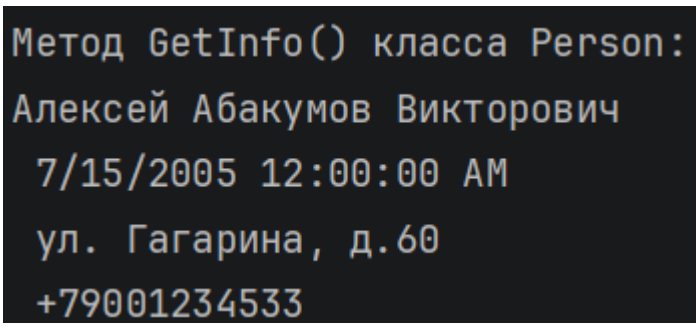

Рисунок 7 – Метод GetInfo() класса Person