Министерство науки и высшего образованияРоссийской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет

имени В.Ф.Уткина»

Кафедра «Электронные вычислительные машины»

Отчет по лабораторной работе №2

на тему

«Переопределение методов и операторов»

по дисциплине  
**«Визуальное программирование»**

Выполнили:

Студенты группы №140

Бригада 5

Сафаров Д. А.

Тимохин Е. С.

Проверили:  
ст. преп. Хизриева Н.И.

ст. преп. Бастрычкин А.С.

**Цель работы**

Изучить переопределение методов, операторов и разработку индексаторов на языке C#.

**Задание**

1) Разработать класс Student, который будет хранить информацию:

− Номер зачетной книжки (ID)

− Фамилия

− Имя

− Отчество (может отсутствовать)

− Дата рождения

− Адрес

− Номер телефона

Необходимо переопределить методы Equals и ToString. Метод Equals должен возвращать true, если все данные у двух студентов совпадают. Метод ToString возвращает в виде строки всю информацию о студенте.

2) 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 к студентам по номеру зачетной книжки.

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

**Практическая часть**

Код программы представлен в приложении А. Результат её работы представлен на рисунке 1.

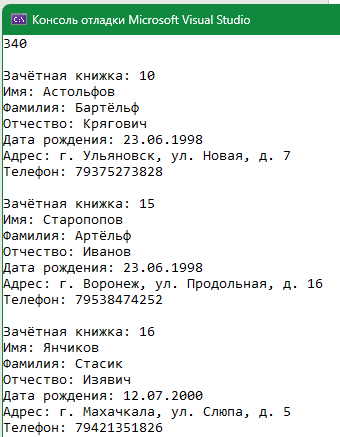


Рисунок 1 – Пример работы программы

Приложение А. Текст программы

**Student.cs**

namespace Classes

{

public class Student

{

public int Id { get; } // Номер зачётной книжки

public string Name { get; } // Имя

public string Surname { get; } // Фамилия

public string MiddleName { get; } // Отчество

public DateOnly BirthDate { get; } // Дата рождения

public string address; // Адрес

public char[] phoneNumber; // Номер телефона

public Student(int id, string name, string surname, DateOnly birthDate, string address, char[] phoneNumber, string middleName = null)

{

Id=id;

Name=name;

Surname=surname;

MiddleName=middleName;

BirthDate=birthDate;

this.address=address;

this.phoneNumber=phoneNumber;

}

public string GetFullName()

{

string fname = new string(Surname);

fname += ' ';

fname.Concat(Name);

if (MiddleName != null)

{

fname += ' ';

fname.Concat(Name);

}

return fname;

}

public override string ToString()

{

string str =

$"Зачётная книжка: {Id}\n" +

$"Имя: {Surname}\n" +

$"Фамилия: {Name}\n";

if (MiddleName != null) str +=

$"Отчество: {MiddleName}\n";

str +=

$"Дата рождения: {BirthDate}\n" +

$"Адрес: {address}\n";

string phoneStr = new string("");

foreach(var c in phoneNumber)

{

phoneStr += c ;

}

str+= $"Телефон: {phoneStr}\n";

return str;

}

public override bool Equals(object? obj)

{

if (obj == null) return false;

if (obj == this) return true;

return (this.ToString().Equals(obj.ToString()));

}

}

}

**Group.cs**

namespace Classes

{

public class Group

{

public int Number { get; }

private List<Student> \_students = new List<Student>();

public Group(int groupNumber)

{

this.Number = groupNumber;

}

private void Sort()

{

for (int viewedN = 0; viewedN < \_students.Count-1; viewedN++)

{

int max\_k = viewedN;

for (int i = viewedN+1; i < \_students.Count; i++)

{

if (\_students[i].GetFullName().CompareTo(\_students[max\_k].GetFullName()) < 0)

max\_k = i;

}

Student studref = \_students[viewedN];

\_students[viewedN] = \_students[max\_k];

\_students[max\_k] = studref;

}

}

public void Add(Student student)

{

\_students.Add(student);

Sort();

}

public void Add(int id, string name, string surname, DateOnly birthDate, string address, char[] phoneNumber, string middleName = null)

{

\_students.Add(new Student(id, name, surname, birthDate, address, phoneNumber, middleName));

Sort();

}

public bool Remove(int studentId)

{

foreach (var student in \_students)

{

if (student.Id == studentId)

{

\_students.Remove(student);

return true;

}

}

return false;

}

public bool Remove(Student student)

{

foreach (var groupStudent in \_students)

{

if (groupStudent.Equals(student))

{

\_students.Remove(student);

return true;

}

}

return false;

}

public string GetInfo()

{

string info = new string("");

info += Number.ToString();

info += "\n\n";

foreach (var student in \_students)

{

info += student.ToString();

info += "\n";

}

return info;

}

public Student? this[int id]

{

get

{

foreach (var student in \_students)

{

if (student.Id == id)

return student;

}

return null;

}

}

}

}