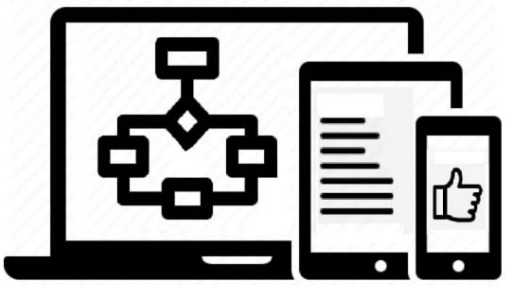
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра
«Программное обеспечение информационных технологий»

# ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

***Методические рекомендации к лабораторным работам для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» очной и заочной форм обучения***



Могилев 2023

УДК 621.01
ББК 36.4
О87

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» 26 января 2023 г., протокол № 7

Составитель канд. техн. наук, доц. Н. Н. Горбатенко

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации разработаны на основе рабочей программы по дисциплине «Объектно-ориентированное программирование» для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» очной и заочной форм обучения для использования при выполнении лабораторных работ.

Учебное издание

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

| | |
|---|---|
| Ответственный за выпуск | В. В. Кутузов |
| Корректор | И. В. Голубцова |
| Компьютерная верстка | М. М. Дударева |

Подписано в печать 07.04.2023 . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс. Печать трафаретная. Усл. печ. л. 2,33. Уч.-изд. л. 2,38 .Тираж 21 экз. Заказ № 423.

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

# Содержание

# 1 Лабораторная работа № 1. Классы. Объекты. Свойства. Инкапсуляция

**Цель работы**: получить навыки разработки простейших классов с использованием принципа инкапсуляции; научиться создавать объекты класса.

## *1.1 Теоретические сведения*

1 Основные сведения о классах [1, с. 148–152].
2 Создание объектов [1, с. 153–154].
3 Создание и использование конструкторов [1, с. 166–170].
4 Вызов перегружаемого конструктора с помощью ключевого слова `this` [1, с. 245–246].
5 Модификаторы доступа к членам класса [1, с. 210–211].
6 Свойства [1, с. 313–317].
7 Автоматически реализуемые свойства [1, с. 318–319].
8 Понятие инкапсуляции как принципа ООП [1, с. 42].

## *1.2 Задания для самостоятельного выполнения*

Спроектировать класс и продемонстрировать работу класса в консольном приложении. В программе должна выполняться проверка всех разработанных элементов класса.

1 Создать класс Point, содержащий следующие члены класса:

– поля `int x, y`;

– конструкторы с параметрами и без параметров, позволяющие создать экземпляр класса с нулевыми координатами и с заданными координатами;

– методы, позволяющие вывести координаты точки на экран, рассчитать расстояние от начала координат до точки, переместить точку на плоскости на вектор `(a, b)`;

– свойства, позволяющее получить/установить координаты точки (доступное для чтения и записи), умножить координаты точки на скаляр (доступное только для записи).

2 Создать класс `Triangle`, содержащий следующие члены класса:

– поля `int a, b, c`;

– конструкторы с параметрами и без параметров, позволяющие создать экземпляр класса с заданными длинами сторон;

– методы, позволяющие вывести длины сторон треугольника на экран, рассчитать периметр треугольника, рассчитать площадь треугольника;

– свойства, позволяющие получить/установить длины сторон треугольника (доступное для чтения и записи), установить, существует ли треугольник с данными длинами сторон (доступное только для чтения).

3 Создать класс `Rectangle`, содержащий следующие члены класса:

– поля `int a, b`;

– конструкторы с параметрами и без параметров, позволяющие создать

экземпляр класса с заданными длинами сторон;

– методы, позволяющие вывести длины сторон прямоугольника на экран, рассчитать периметр прямоугольника, рассчитать площадь прямоугольника;

– свойства, позволяющие получить/установить длины сторон прямоугольника (доступное для чтения и записи), установить, является ли данный прямоугольник квадратом (доступное только для чтения).

4 Создать класс `Money`, содержащий следующие члены класса:

– поля `int first` (номинал купюры), `int second` (количество купюр);

– конструктор с параметрами и без параметров, позволяющий создать экземпляр класса с заданными значениями полей;

– методы, позволяющие вывести номинал и количество купюр, определить, хватит ли денежных средств на покупку товара на сумму `N` р., определить, сколько штук товара стоимости `n` р. можно купить на имеющиеся денежные средства;

– свойства, позволяющее получить/установить значение полей (доступное для чтения и записи), рассчитать сумму денег (доступное только для чтения).

5 Создать класс для работы с одномерным массивом целых чисел и разработать следующие члены класса:

– поля `n`, `int [] IntArray`;

– конструктор с параметрами и без параметров, позволяющий создать массив размерности `n`;

– методы, позволяющие ввести элементы массива с клавиатуры, вывести элементы массива на экран, отсортировать элементы массива в порядке возрастания;

– свойства, возвращающие размерность массива (доступное только для чтения), и свойство, позволяющее умножить все элементы массива на скаляр (доступное только для записи).

6 Создать класс для работы с двумерным массивом целых чисел и разработать следующие члены класса:

– поля `n`, `m`, `int [,] intArray`;

– конструктор с параметрами и без параметров, позволяющий создать массив размерности `n×m`;

– методы, позволяющие ввести элементы массива с клавиатуры, вывести элементы массива на экран, вычислить сумму элементов `i`-го столбца;

– свойства, позволяющие вычислить количество нулевых элементов в массиве (доступное только для чтения), установить значение всех элементов главной диагонали массива равное скаляру (доступное только для записи).

7 Создать класс для работы с двумерным ступенчатым массивом вещественных чисел и разработать следующие функциональные члены класса:

– поля `n` (число строк массива), `double [][] doubelArray`;

– конструктор с параметрами и без параметров, позволяющий создать ступенчатый массив;

– методы, позволяющие ввести элементы массива с клавиатуры, вывести элементы массива на экран, отсортировать элементы каждой строки массива в порядке убывания;

– свойство, возвращающее общее количество элементов в массиве (доступное только для чтения), и свойство, позволяющее увеличить значение всех элементов массива на скаляр (доступное только для записи).

8 Создать класс для работы со строками и разработать следующие члены класса:

– поле `string line`;

– конструктор с параметрами и без параметров, позволяющий создать строку на основе заданного строкового литерала;

– методы, позволяющие подсчитать количество цифр в строке, выводить на экран все символы строки, встречающиеся в ней ровно один раз, вывести на экран самую длинную последовательность повторяющихся символов в строке;

– свойство, возвращающее общее количество символов в строке (доступное только для чтения).

9 Создать класс для работы со строками и разработать следующие члены класса:

– поле `StringBuilder line`;

– конструктор с параметрами и без параметров, позволяющий создать строку на основе заданного строкового литерала, и конструктор, позволяющий создавать пустую строку;

– методы, позволяющие подсчитать количество пробелов в строке, заменить в строке все прописные символы на строчные, удалить из строки все знаки препинания;

– свойство, возвращающее общее количество элементов в строке (доступное только для чтения), и свойство, позволяющее установить значение поля в соответствии с введенным значением строки с клавиатуры, а также получить значение данного поля (доступно для чтения и записи).

10 Самостоятельно изучить тип данных `DateTime`, на основе которого необходимо создать класс для работы с датой. Данный класс должен содержать следующие члены класса:

– поле `DataTime data`;

– конструкторы с параметрами и без параметров, позволяющие установить заданную дату, дату 1.01.2000;

– методы, позволяющие вычислить дату предыдущего дня, вычислить дату следующего дня, определить, сколько дней осталось до конца месяца;

– свойства, позволяющие установить или получить значение поле класса (доступно для чтения и записи), определить, является ли год високосным (доступно только для чтения).

11 Создать класс `Set` (множество). Внутренним представлением класса `Set` является одномерный массив целых чисел. Класс должен содержать следующие члены:

– поля `n` (число элементов множества), `int[] set`;

– конструктор с параметрами и без параметров, позволяющий создать множество размерностью `n`;

– методы, позволяющие ввести элементы массива с клавиатуры, вывести элементы массива на экран, добавление элементов в множество с исключением дублирования элементов, удаления элементов из множества;

– свойство, возвращающее общее количество элементов в массиве (доступное только для чтения);

– свойство, позволяющее увеличить значение всех элементов массива на скаляр (доступное только для записи).

12 Создать класс `Complex` для работы с одномерным массивом комплексных чисел. Комплексное число представляется в виде `Re + Im*i`, где `i` – мнимая единица. Разработать следующие члены класса:

– поля `n`, `int [] IntArray`;

– конструктор с параметрами и без параметров, позволяющий создать массив комплексных чисел размерности `n`;

– методы, позволяющие ввести элементы массива с клавиатуры, вывести элементы массива на экран;

– свойства, возвращающие размерность массива (доступное только для чтения), и свойство, позволяющее умножить все элементы массива на скаляр (доступное только для записи).

### *Контрольные вопросы*

1 Дайте определение терминам «класс» и «объект». Как соотносятся эти понятия между собой?

2 Приведите синтаксис создания объекта в общем виде. Проиллюстрируйте его фрагментом программы.

3 Какие члены класса содержат код?

4 Какие члены класса содержат данные?

5 В чём состоит назначение конструктора?

6 Чем конструктор отличается от обычного метода?

7 Перечислите типы конструкторов класса.

8 Сколько конструкторов может содержать класс?

9 Объясните механизм вызова перегружаемого конструктора с помощью ключевого слова `this`.

10 Что понимается под термином «деструктор»?

11 В чём состоит назначение деструктора?

12 Перечислите модификаторы доступа к членам класса.

13 В чём отличия свойств от полей?

14 Приведите формат объявления свойства.

15 Каким идентификатором представлено в `set`-аксессоре новое значение свойства?

16 Объясните назначение механизма автоматически реализуемых свойств.

# 2 Лабораторная работа № 2. Индексаторы. Перегрузка операций

**Цель работы**: получить навыки создания программ с использованием индексаторов и перегрузки операций.

## *2.1 Теоретические сведения*

1 Индексаторы [1, с. 304–307].
2 Перегрузка операций [1, с. 270–277].

## *2.2 Задания для самостоятельного выполнения*

Дополните класс, созданный в лабораторной работе № 1, индексатором и методами перегрузки операций. Составить тестирующую программу с выдачей результатов. В программе должна выполняться проверка всех разработанных элементов класса.

1 Дополнить класс `Point`:

– индексатором, позволяющим по индексу 0 обращаться к полю `x`, по индексу 1 – к полю `y`, при других значениях индекса выдается сообщение об ошибке;

– методами, осуществляющими перегрузку следующих операций: операции `++ (--)` – одновременно увеличивает (уменьшает) значение полей `x` и `y` на 1; констант `true` и `false` – обращение к экземпляру класса дает значение `true`, если значение полей `x` и `y` совпадает, иначе `false`; операции бинарный `+` – одновременно добавляет к полям `x` и `y` значение скаляра.

2 Дополнить класс `Triangle`:

– индексатором, позволяющим по индексу 0 обращаться к полю `a`, по индексу 1 – к полю `b`, по индексу 2 – к полю `c`, при других значениях индекса выдается сообщение об ошибке;

– методами, осуществляющими перегрузку следующих операций: операции `++ (--)` – одновременно увеличивает (уменьшает) значение полей `a`, `b` и `c` на 1; констант `true` и `false` – обращение к экземпляру класса дает значение true, если треугольник с заданными длинами сторон существует, иначе – `false`; операции `*` – одновременно умножает поля `a`, `b` и `c` на скаляр.

3 Дополнить класс `Rectangle`:

– индексатором, позволяющим по индексу 0 обращаться к полю `a`, по индексу 1 – к полю `b`, при других значениях индекса выдается сообщение об ошибке;

– методами, осуществляющими перегрузку следующих операций: операции `++ (--)` – одновременно увеличивает (уменьшает) значение полей `a` и `b`; констант `true` и `false` – обращение к экземпляру класса дает значение `true`, если прямоугольник с заданными длинами сторон является квадратом, иначе – `false`; операции `*` – одновременно умножает поля `a` и `b` на скаляр.

4 Дополнить класс `Money`:

– индексатором, позволяющим по индексу 0 обращаться к полю `first`, по индексу 1 – к полю `second`, при других значениях индекса выдается сообщение об ошибке;

– методами, осуществляющими перегрузку следующих операций: операции `++ (--)` – одновременно увеличивает (уменьшает) значение полей `first` и `second`; операции `!` – возвращает значение `true`, если поле `second` не нулевое, иначе `false`; операции бинарный `+ –` добавляет к значению поля `second` значение скаляра.

5 Дополнить класс для работы с одномерным массивом целых чисел:

– индексатором, позволяющим по индексу обращаться к соответствующему элементу массива;

– методами, осуществляющими перегрузку следующих операций: операции `++ (--)` – одновременно увеличивает (уменьшает) значение всех элементов массива на 1; операции `!` – возвращает значение `true`, если элементы массива не упорядочены по возрастанию, иначе – `false`; операции бинарный `*` – умножить все элементы массива на скаляр; операции преобразования класса массив в одномерный массив (и наоборот).

6 Дополнить класс для работы с двумерным массивом целых чисел:

– двумерным индексатором, позволяющим обращаться к соответствующему элементу массива;

– методами, осуществляющими перегрузку следующих операций: операции `++ (--)` – одновременно увеличивает (уменьшает) значение всех элементов массива на 1; констант `true` и `false` – обращение к экземпляру класса дает значение `true`, если двумерный массив является квадратным; операции бинарный `+ –` сложить два массива соответствующих размерностей; операции преобразования класса массив в двумерный массив (и наоборот).

7 Дополнить класс для работы с двумерным массивом вещественных чисел:

– двумерным индексатором, позволяющим обращаться к соответствующему элементу массива;

– методами, осуществляющими перегрузку следующих операций: операции `++ (--)` – одновременно увеличивает (уменьшает) значение всех элементов массива на 1; констант `true` и `false` – обращение к экземпляру класса дает значение `true`, если каждая строка массива упорядочена по возрастанию, иначе – `false`; операции преобразования класса массив в ступенчатый массив (и наоборот).

8 Дополнить класс для работы со строками:

– индексатором, позволяющим по индексу обращаться к соответствующему символу строки (доступный только для чтения);

– методами, осуществляющими перегрузку следующих операций: операции унарного `!` – возвращает значение `true`, если строка не пустая, иначе – `false`; констант `true` и `false` – обращение к экземпляру класса дает значение `true` если строка является палиндромом, `false` – в противном случае; операции

`&`: возвращает значение `true`, если строковые поля двух объектов посимвольно равны (без учета регистра), иначе – `false`; операции преобразования класса-строка в тип `string` (и наоборот).

9 Дополнить класс для работы со строками:

– индексатором, позволяющим по индексу обращаться к соответствующему символу строки;

– методами, осуществляющими перегрузку следующих операций: операции унарного `+ (-)` – преобразующей строку к строчным (прописным) символам; констант `true` и `false` – обращение к экземпляру класса дает значение true, если строка не пустая, иначе – `false`; операции & – возвращает значение true, если строковые поля двух объектов посимвольно равны (без учета регистра), иначе – `false`; операции преобразования строки в тип `StringBuilder` (и наоборот).

10 Дополнить класс для работы с датой:

– индексатором, позволяющим определить дату `i`-го по счету дня относительно установленной даты (при отрицательных значениях индекса отсчет ведется в обратном порядке);

– методами, осуществляющими перегрузку следующих операций: операции `!` – возвращает значение `true`, если установленная дата не является последним днем месяца, иначе – `false`; констант `true` и `false` – обращение к экземпляру класса дает значение `true`, если установленная дата является началом года, иначе – `false`; операции `&` – возвращает значение `true`, если поля двух объектов равны, иначе `false`.

11 Дополнить класс для работы с множеством Set:

– индексатором, позволяющим по индексу обращаться к соответствующему элементу множества;

– методами, осуществляющими перегрузку следующих операций: объединения (+), разности (–), сравнения (==) двух множеств.

12 Дополнить класс `Complex`:

– индексатором, позволяющим по индексу обращаться к соответствующему элементу массива;

– методами, осуществляющими перегрузку следующих операций: операции присваивания `(=)` значения одного комплексного числа другому; операций сложения `(+)` и умножения `(*)` комплексных чисел.

### *Контрольные вопросы*

1 Для чего предназначены индексаторы?
2 Какова роль служебного слова `this` в индексаторе?
3 Может ли в одном классе быть несколько индексаторов?
4 Какой тип допустим для параметра индексатора?
5 Объясните принцип работы индексатора.
6 Как определяется базовый тип индексатора?
7 Что записывается в качестве имени индексатора?

8 Что содержит список параметров индексатора?

9 Можно ли в классе объявить более одного индексатора? Если да, то при каком условии.

10 Может ли аргумент индексатора иметь тип, отличный от `int`?

11 Что понимается под перегрузкой операций?

12 С какой целью осуществляют перегрузку операций?

13 Как осуществляется перегрузка операций?

# 3 Лабораторная работа № 3. Наследование классов

**Цель работы**: получить представление о процессе объектно-ориентированной декомпозиции задачи; приобрести навыки программирования с использованием наследования классов.

## *3.1 Теоретические сведения*

1 Основы наследования [1, с. 329–332].

2 Доступ к членам класса при наследовании [1, с. 333–335].

3 Организация защищённого доступа [1, с. 336–337].

4 Конструкторы и наследование [1, с. 337–339].

5 Вызов конструкторов базового класса [1, с. 339–343].

6 Наследование и сокрытие имен [1, с. 343–344].

7 Применение ключевого слова `base` для доступа к скрытому имени [1, с. 344–346].

8 Порядок вызова конструкторов [1, с. 349–350].

## *3.2 Задания для самостоятельного выполнения*

Построить иерархию классов предметной области согласно варианту. В качестве основы иерархии использовать обычный класс. В классах описать конструкторы с параметрами и конструкторы по умолчанию, свойства для установки и получения значений полей классов, методы для описания поведения объектов. Каждый из создаваемых классов должен иметь не менее трёх методов, свойств, конструкторов. Для каждого созданного класса переопределить методы `Equals()`, `ToString()` класса `object`. В методе `Main()`:

– продемонстрировать всю реализованную функциональность классов;

– создать массив из объектов базового класса, заполнить его ссылками на производные классы, вывести на экран элементы массива;

– создать два объекта базового класса с совпадающими данными и проверить, что ссылки на объекты не равны, а объекты равны, вывести значения хешкодов для объектов.

1 Студент, преподаватель, заведующий кафедрой, персона.
2 Небоскрёб, дача, коттедж, жилое здание.
3 Организация, страховая компания, нефтегазовая компания, завод.
4 Журнал, книга, печатное издание, учебник.
5 Тест, экзамен, выпускной экзамен, испытание.
6 Строительное сооружение, театр, производственный корпус, гостиница.
7 Игрушка, телевизор, товар, молоко.
8 Квитанция, накладная, документ, счёт.
9 Автомобиль, поезд, самолёт, транспортное средство.
10 Республика, монархия, королевство, государство.
11 Корабль, пароход, парусник, корвет.
12 Двигатель, бензиновый двигатель, дизельный двигатель, реактивный двигатель.
13 Деталь, узел, механизм, изделие.
14 Млекопитающее, парнокопытное, птица, животное.
15 Выпускник вуза, Бакалавр, Магистр, Инженер.

### *Контрольные вопросы*

1 Для чего используется наследование классов?
2 Опишите синтаксис производного класса. Какие модификаторы доступа применяются в иерархиях классов?
3 Объясните правила доступа к членам базового класса для объектов производного класса.
4 Что такое защищённый член класса?
5 Объясните порядок вызова конструкторов базовых классов при работе конструктора производного класса.
6 Какие действия выполняются автоматически при отсутствии в конструкторе производного класса обращения к конструктору базового класса?
7 Что такое сокрытие при наследовании классов?
8 Каково назначение ключевого слова `new` в производном классе?
9 Должны ли совпадать типы возвращаемых значений при сокрытии методов?
10 Какова последовательность выполнения конструкторов при наследовании?
11 Как осуществляется доступ к элементам производных и базовых классов?
12 Назовите альтернативы наследованию классов.
13 Какие методы и операции класса `object` часто перегружают в его потомках?
14 Каковы назначение и использование операторов `is` и `as`?

# 4 Лабораторная работа № 4. Виртуальные методы. Полиморфизм

**Цель работы**: ознакомиться с понятиями полиморфизма, позднего связывания; приобрести навыки программирования с использованием виртуальных методов.

## *4.1 Теоретические сведения*

1 Полиморфизм [1, с. 43].
2 Виртуальные методы и их переопределение [1, с. 355–362].

## *4.2 Задания для самостоятельного выполнения*

Расширить иерархию классов из лабораторной работы № 3 с использованием виртуального класса в качестве основы иерархии. Показать пример использования полиморфизма методов. Для демонстрации полиморфизма в методе `Main()` создать массив из объектов базового класса. Присвоить элементам этого массива ссылки на производные классы, вывести элементы массива на экран, сравнить полученный результат с результатом выполнения лабораторной работы № 3.

### *Контрольные вопросы*

1 Что понимается под термином «полиморфизм»?
2 В чем состоит основной принцип полиморфизма?
3 Какие механизмы используются в языке C# для реализации концепции полиморфизма?
4 Что понимается под термином «виртуальный метод»?
5 Какое ключевое слово языка C# используется для определения виртуального метода?
6 В чем состоит особенность виртуальных методов в производных (дочерних) классах?
7 В какой момент трансляции программы осуществляется выбор версии виртуального метода?
8 Какие условия определяют выбор версии виртуального метода?
9 Какое ключевое слово (модификатор) языка C# используется для определения виртуального метода в производном (дочернем) классе?
10 Какие модификаторы недопустимы для определения виртуальных мето-дов?
11 Что означает термин «переопределенный метод»?
12 В какой момент работы программы осуществляется выбор вызываемого переопределенного метода?

# 5 Лабораторная работа № 5. Абстрактные классы. Интерфейсы

**Цель работы**: приобрести навыки программирования с использованием абстрактных классов и интерфейсов.

## *5.1 Теоретические сведения*

1 Применение абстрактных классов [1, с. 363–367].
2 Интерфейсы. Применение интерфейсных ссылок [1, с. 375–382].
3 Интерфейсные свойства и индексаторы [1, с. 383–387].
4 Наследование интерфейсов [1, с. 387–390].

## *5.2 Задания для самостоятельного выполнения*

**Задание 1**

Создать консольное приложение в соответствии с выданным вариантом.

1 Создать абстрактный класс `Figure` с функциями вычисления площади и периметра, а также функцией, выводящей информацию о фигуре на экран. В абстрактном классе `Figure` реализовать метод `CompareTo` так, чтобы можно было отсортировать объекты по их площадям. Создать производные классы: `Rectangle` (прямоугольник), `Circle` (круг), `Triangle` (треугольник). В методе `Main()` создать массив `n` фигур и вывести полную информацию о фигурах на экран, отсортировав объекты по их площадям, а также организовать поиск фигур, площадь которых попадает в заданный диапазон.

2 Создать абстрактный класс `Function` с функциями вычисления значения по формуле `y=f(x)` в заданной точке, а также функцией, выводящей информацию о виде функции на экран. В абстрактном классе `Function` реализовать метод `CompareTo` так, чтобы можно было отсортировать функции по коэффициенту a. Создать производные классы: `Line (y=ax+b)`, `Kub (y=ax^3+bx+c)`, `Hyperbola (y=a/x)`. В методе `Main()` создать массив `n` функций и вывести полную информацию о значении данных функций в точке `x`, отсортировав функции по коэффициенту `a`.

3 Создать абстрактный класс `Edition` с функциями, позволяющими вывести на экран информацию об издании, а также определить, является ли данное издание искомым. В абстрактном классе `Edition` реализовать метод `CompareTo` так, чтобы можно было отсортировать каталог изданий по фамилии автора. Создать производные классы: `Book` (название, фамилия автора, год издания, издательство), `Article` (название, фамилия автора, название журнала, его номер и год издания), `OnlineResource` (название, фамилия автора, ссылка, аннотация). В методе `Main()` создать массив из `n` изданий, вывести полную информацию из каталога, отсортировав каталог изданий по фамилии автора, а также организовать поиск изданий по фамилии автора.

4 Создать абстрактный класс `Transport` с функциями, позволяющими вывести на экран информацию о транспортном средстве, а также определить грузоподъёмность транспортного средства. В абстрактном классе `Transport` реализовать метод `CompareTo` так, чтобы можно было отсортировать базу данных о машинах по их грузоподъемности. Создать производные классы: `Car` (марка, номер, скорость, грузоподъемность), `Motorbike` (марка, номер, скорость, грузоподъемность, наличие коляски, при этом если коляска отсутствует, то грузоподъемность равна 0), `Truck` (марка, номер, скорость, грузоподъемность, наличие прицепа, при этом если есть прицеп, то грузоподъемность увеличивается в 2 раза). В методе `Main()` создать массив из n машин, вывести полную информацию из базы на экран, отсортировав массив данных о машинах по их грузоподъемности, а также организовать поиск машин, удовлетворяющих требованиям грузоподъемности.

5 Создать абстрактный класс `Persona` с функциями, позволяющими вывести на экран информацию о персоне, а также определить её возраст (на момент текущей даты). В абстрактном классе `Persona` реализовать метод `CompareTo` так, чтобы можно было отсортировать базу данных о персонах по дате рождения. Создать производные классы: `Enrollee` (фамилия, дата рождения, факультет), `Student` (фамилия, дата рождения, факультет, курс), `Teacher` (фамилия, дата рождения, факультет, должность, стаж). В методе `Main()` создать массив из `n` персон, вывести полную информацию из базы на экран, отсортировав массив данных о персонах по дате рождения, а также организовать поиск персон, чей возраст попадает в заданный диапазон.

6 Создать абстрактный класс `Goods` с функциями, позволяющими вывести на экран информацию о товаре, а также определить, соответствует ли он сроку годности на текущую дату. В абстрактном классе `Goods` реализовать метод `CompareTo` так, чтобы можно было отсортировать базу данных о товарах по их цене. Создать производные классы: `Product` (название, цена, дата производства, срок годности), `Party` (название, цена, количество штук, дата производства, срок годности), `Kit` (название, цена, перечень продуктов). В методе `Main()` создать массив из `n` товаров, вывести полную информацию из базы на экран, отсортировав массив данных о товарах по их цене, а также организовать поиск просроченного товара (на момент текущей даты).

7 Создать абстрактный класс `Goods` с функциями, позволяющими вывести на экран информацию о товаре, а также определить, соответствует ли она искомому типу. В абстрактном классе `Goods` реализовать метод `CompareTo` так, чтобы можно было отсортировать базу данных о товарах по возрасту детей, на которых он рассчитан. Создать производные классы: `Toy` (название, цена, производитель, материал, возраст, на который рассчитана), `Book` (название, автор, цена, издательство, возраст, на который рассчитана), `SportsEquipment` (название, цена, производитель, возраст, на который рассчитан). В методе `Main()`

создать массив из n товаров, вывести полную информацию из базы на экран, отсортировав массив данных о товарах по возрасту детей, на которых он рассчитан, а также организовать поиск товаров определенного типа.

8 Создать абстрактный класс TelephoneDirectory с функциями, позволяющими вывести на экран информацию о записях в телефонном справочнике, а также определить соответствие записи критерию поиска. В абстрактном классе TelephoneDirectory реализовать метод CompareTo так, чтобы можно было отсортировать базу данных справочника по номеру телефона. Создать производные классы: Persona (фамилия, адрес, номер телефона), Organization (название, адрес, телефон, факс, контактное лицо), Friend (фамилия, адрес, номер телефона, дата рождения). В методе Main() создать массив из n записей, вывести полную информацию из базы на экран, отсортировав массив данных справочника по номеру телефона, а также организовать поиск в базе по фамилии.

9 Создать абстрактный класс Client с функциями, позволяющими вывести на экран информацию о клиентах банка, а также определить соответствие клиента критерию поиска. В абстрактном классе Client реализовать метод CompareTo так, чтобы можно было отсортировать базу данных о клиентах банка по дате открытия их счета. Создать производные классы: Depositor (фамилия, дата открытия вклада, размер вклада, процент по вкладу), Credited (фамилия, дата выдачи кредита, размер кредита, процент по кредиту, остаток долга), Organization (название, дата открытия счета, номер счета, сумма на счету). В методе Main() создать массив из n клиентов, вывести полную информацию из базы на экран, отсортировав массив данных о клиентах банка по дате открытия их счета, а также организовать поиск клиентов, начавших сотрудничать с банком с заданной даты.

10 Создать абстрактный класс Software с методами, позволяющими вывести на экран информацию о программном обеспечении, а также определить соответствие возможности использования (на момент текущей даты). В абстрактном классе Software реализовать метод CompareTo так, чтобы можно было отсортировать массив данных по названию ПО. Создать производные классы: FreeSoftware (название, производитель), SharewareSoftware (название, производитель, дата установки, срок бесплатного использования), ProprietarySoftware (название, производитель, цена, дата установки, срок использования). В методе Main() создать массив из n видов программного обеспечения, вывести полную информацию из массива на экран, отсортировав массив данных по названию ПО, а также организовать поиск программного обеспечения, которое допустимо использовать на текущую дату.

**Задание 2**

Реализовать иерархию классов из лабораторной работы № 3 с использованием интерфейсов, при этом один из классов должен реализовывать как минимум два интерфейса. Продемонстрировать применение интерфейсных ссылок.

### *Контрольные вопросы*

1 Что такое абстрактный класс? Какое ключевое слово используется при объявлении абстрактных методов?

2 Являются ли абстрактные методы виртуальными?

3 Можно ли создавать цепочку производных классов, используя абстрактный класс?

4 Могут ли быть в программе объекты абстрактного класса?

5 Что такое интерфейс?

6 Чем отличается объявление интерфейса от объявления абстрактного класса?

7 Какие элементы языка C# могут быть членами интерфейса?

8 Сколько интерфейсов может наследовать класс?

9 Где должны быть реализованы методы интерфейса?

10 Можно ли реализовать множественный интерфейс?

11 Как проявляется принцип полиморфизма при использовании интерфейсов?

12 Можно ли объявить интерфейс с модификатором `static`?

13 Возможно ли создание ссылочной переменной интерфейсного типа?

14 В чём различия и сходства интерфейса и абстрактного класса?

15 Доступ к каким членам класса, реализующего интерфейс, обеспечивает ссылка с типом интерфейса?

## 6 Лабораторная работа № 6. Агрегация и композиция классов

**Цель работы**: приобрести навыки проектирования классов с использованием отношений агрегации и композиции.

### *6.1 Теоретические сведения*

1 Основные отношения между классами: агрегация, композиция, наследование [2, с. 237–238].

2 Программная реализация на языке C# агрегации и композиции классов [2, с. 239–241].

### *6.2 Задания для самостоятельного выполнения*

Разработайте два класса на основе существующих, используя механизмы агрегации и композиции классов. В методе `Main()` консольного приложения продемонстрировать всю функциональность созданных классов.

1 Создать объект класса Завод, используя класс Склад деталей. Методы: изготовить деталь, доставить деталь на склад, получить деталь из склада, подсчитать количество деталей на складе, использовать деталь при сборке изделия, вывести на консоль наименование детали.

Создать объект класса Здание, используя класс Отопительная система. Методы: включить отопительную систему, выключить отопительную систему, определить температуру в здании, заполнить отопительную систему водой, слить воду из отопительной системы.

2 Создать объект класса Больница, используя класс Приемное отделение. Методы: регистрация больного, выписка больного, установить диагноз болезни, назначить курс лечения, направить на обследование.

Создать объект Аэропорт, используя класс Взлётная полоса. Методы: взлёт самолёта, посадка самолёта, вывести на консоль длину взлетной полосы, количество взлетных полос.

3 Создать объект класса Ресторан, используя класс Кухня. Методы: приготовить суп, салат, кофе, составить меню, принять заказ клиента, выполнить заказ, принести приготовленную еду посетителю ресторана.

Создать объект Библиотека, используя класс Книгохранилище. Методы: задать название книги, дополнить книгохранилище книгой, вывести на консоль количество книг, выдать книгу читателю.

4 Создать объект класса Текст, используя класс Абзац. Методы: дополнить текст, вывести на консоль текст, заголовок текста.

Создать объект класса Наседка, используя классы Птица, Кукушка. Методы: летать, петь, нести яйца, высиживать птенцов.

5 Создать объект класса Автомобиль, используя класс Колесо. Методы: ехать, заправляться, менять колесо, вывести на консоль марку автомобиля.

Создать объект класса Текстовый файл, используя класс Файл. Методы: создать, переименовать, вывести на консоль содержимое, дополнить, удалить.

6 Создать объект класса Самолет, используя класс Крыло. Методы: летать, задавать маршрут, вывести на консоль маршрут.

Создать объект класса Одномерный массив, используя класс Массив. Методы: создать, вывести на консоль, выполнить операции (сложить, вычесть, перемножить).

7 Создать объект класса Беларусь, используя класс Область. Методы: вывести на консоль столицу, количество областей, площадь, областные центры.

Создать объект класса Простая дробь, используя класс Число. Методы: вывод на экран, сложение, вычитание, умножение, деление.

8 Создать объект класса Планета, используя класс Материк. Методы: вывести на консоль название материка, планеты, количество материков.

Создать объект класса Дом, используя классы Окно, Дверь. Методы: закрыть на ключ, вывести на консоль количество окон, дверей.

9 Создать объект класса Звездная система, используя классы Планета, Звезда, Луна. Методы: вывести на консоль количество планет в звездной системе, название звезды, добавление планеты в систему.

Создать объект класса Роза, используя классы Лепесток, Бутон. Методы: расцвести, увянуть, вывести на консоль цвет бутона.

10 Создать объект класса Компьютер, используя классы Винчестер, Дисковод, ОЗУ. Методы: включить, выключить, проверить на вирусы, вывести на консоль размер винчестера.

Создать объект класса Дерево, используя классы Лист. Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.

11 Создать объект класса Квадрат, используя классы Точка, Отрезок. Методы: задание размеров, растяжение, сжатие, поворот, изменение цвета.

Создать объект класса Пианино, используя класс Клавиша. Методы: настроить, играть на пианино, нажимать клавишу.

12 Создать объект класса Круг, используя классы Точка, Окружность. Методы: задание размеров, изменение радиуса, определение принадлежности точки данному кругу.

Создать объект класса Фотоальбом, используя класс Фотография. Методы: задать название фотографии, дополнить фотоальбом фотографией, вывести на консоль количество фотографий.

13 Создать объект класса Котёнок, используя классы Животное, Кошка. Методы: вывести на консоль имя, подать голос, рожать потомство (создавать себе подобных).

Создать объект класса Год, используя классы Месяц, День. Методы: задать дату, вывести на консоль день недели по заданной дате, рассчитать количество дней, месяцев в заданном временном промежутке.

14 Создать объект класса Сутки, используя классы Час, Минута. Методы: вывести на консоль текущее время, рассчитать время суток (утро, день, вечер, ночь).

Создать объект класса Птица, используя класс Крылья. Методы: летать, добывать пищу, питаться.

15 Создать объект класса Тигр, используя класс Когти. Методы: рычать, бежать, добывать пищу.

Создать объект класса Гитара, используя класс Струна. Методы: играть, натягивать струну.

### *Контрольные вопросы*

1 В каких отношениях могут находиться классы при ООП?

2 Дайте определение агрегации и композиции.

3 В чем заключается различие между агрегацией и композицией классов?

4 Объясните механизм программной реализации на языке C# агрегации и композиции.

5 Какие преимущества есть у композиции и агрегации перед наследованием?

# 7 Лабораторная работа № 7. Обобщенные коллекции

**Цель работы**: получить навыки программирования с использованием обобщенных коллекций.

## *7.1 Теоретические сведения*

1 Класс `List<T>` [1, с. 961–965].
2 Класс `Dictionary<TKey, TValue>` [1, с. 969–972].
3 Назначение, применение итераторов [1, с. 1003–1005].
4 Создание именованного итератора [1, с. 1006–1007].

## *7.2 Задания для самостоятельного выполнения*

Для заданной предметной области создать программу, состоящую из 3…5 классов. Предусмотреть использование типа данных – перечисление, коллекций `List<T>` (варианты заданий с нечётными номерами), `Dictionary<TKey, TValue>` (варианты заданий с чётными номерами). Ввод/вывод данных должен быть реализован вне классов.

1 Предметная область: АТС. На АТС хранится информация о всех клиентах станции. АТС имеет список тарифов на междугородние разговоры. Клиент АТС может совершать множество звонков в различные города. Система должна:

– позволять вводить информацию о тарифах;

– вводить информацию о клиентах и регистрировать звонки;

– по введенной фамилии о клиенте определять стоимость всех сделанных им звонков в соответствии с действующими тарифами;

– вычислять общую стоимость всех выполненных на АТС звонков;

– сортировать фамилии клиентов в порядке убывания стоимости сделанных звонков.

2 Предметная область: Вокзал. Касса вокзала имеет список тарифов на различные направления. При покупке билета регистрируются паспортные данные пассажира. Пассажир покупает билеты на различные направления. Система должна:

– позволять вводить данные о тарифах;

– позволять вводить паспортные данные пассажира и регистрировать покупку билета;

– рассчитывать стоимость купленных пассажиром билетов;

– после ввода наименования направления выводить список всех пассажиров, купивших на него билет;

– сортировать билеты в порядке возрастания их стоимости.

3 Предметная область: ЖЭС. В ЖЭС хранятся тарифы на коммунальные услуги. ЖЭС имеет информацию о всех жильцах. При потреблении жильцами коммунальных услуг информация регистрируется в системе. Система должна позволять выполнять следующие задачи:

– ввод тарифов;

– ввод информации о жильцах и потребленных услугах;

– после ввода фамилии выводить сумму всех потребленных услуг;

– выводить стоимость всех оказанных услуг;

– сортировать фамилии жильцов в порядке убывания стоимости всех потребленных услуг.

4 Предметная область: Аэропорт. Касса аэропорта имеет список тарифов на различные направления. При покупке билета регистрируются паспортные данные. Система должна:

– позволять вводить данные о тарифах;

– позволять вводить паспортные данные пассажира и регистрировать покупку билета;

– рассчитывать стоимость купленных пассажиром билетов;

– рассчитывать стоимость всех проданных билетов;

– сортировать фамилии пассажиров порядке возрастания стоимости купленных билетов.

5 Предметная область: Банк. Информационная система банка хранит описание процентов по различным вкладам. Система хранит информацию о вкладчиках и сделанных ими вкладах. Каждый клиент может поместить в банк только один вклад. Система должна позволять выполнять следующие задачи:

– хранить информацию о процентах по вкладам;

– хранить информацию о клиентах;

– пополнять клиенту величину вклада;

– вычислять общую сумму выплат по процентам для всех вкладов.

– сортировать фамилии клиентов в порядке убывания величины вклада.

6 Предметная область: Отдел расчета зарплаты. Информационная система отдела расчета зарплаты на предприятии хранит данные о величине оплаты за различные виды работ. Система хранит информацию о работниках предприятия. Система должна позволять выполнять следующие задачи:

– вводить информацию о различных видах работ;

– вводить информацию о работниках и выполненных ими работах;

– после ввода фамилии выводить для работника зарплату;

– выводить сумму выплат всем работникам.

– сортировать фамилии работников в порядке убывания заработной платы.

7 Предметная область: Фирма грузоперевозок. Фирма имеет список тарифов по перевозке грузов. Клиент регистрируется в системе, после чего может заказать перевозку определенного объема груза. Система должна позволять выполнять следующие задачи:

– ввод тарифов;

– регистрация клиента и заказ на перевозку грузов;

– вывод суммы заказа для определенного клиента;

– подсчет суммарной стоимости всех заказов;

– сортировать клиентов в порядке убывания суммарной стоимости сделанных заказов.

8 Предметная область: Гостиница. Информационная система гостиницы хранит информацию о всех номерах и их стоимости. Система регистрирует клиентов. Каждый клиент может заказать один номер. При попытке заказа номера, который занят, выводится предупреждение. Система должна позволять выполнять следующие задачи:

– ввод информации о номерах и их стоимости;

– регистрация клиента и заказ номера;

– вывод списка незанятых номеров;

– после ввода фамилии клиента вывод стоимости проживания;

– сортировать фамилии клиентов в порядке убывания стоимости проживания.

9 Предметная область: Интернет-оператор. Провайдер имеет различные тарифы доступа в Интернет за 1 Мбайт в зависимости от величины абонентской платы. Информационная система провайдера хранит данные о клиентах. Система должна позволять выполнять следующие задачи:

– ввод тарифов;

– регистрация пользователя;

– ввод данных о потребленном трафике для конкретного пользователя;

– подсчет общей стоимости реализованного трафика;

– поиск клиента, заплатившего наибольшую стоимость за услуги;

– сортировать фамилии клиентов в порядке возрастания стоимости за услуги.

10 Предметная область: Интернет-магазин. В информационной системе хранятся данные о товарах. Клиент звонит в магазин и оставляет заказ на товар. Система должна позволять выполнять следующие задачи:

– ввод информации о товарах;

– регистрация заказа клиента на покупку определенного товара;

– после ввода фамилии покупателя вывод списка заказанных им товаров;

– после ввода фамилии покупателя вывод суммы заказа;

– сортировать фамилии покупателей в убывания возрастания суммы заказа.

### *Контрольные вопросы*

1 Что такое коллекции?

2 Опишите состав пространства имен `System.Collections` и дайте характеристику основных типов-коллекций.

3 Объясните назначение классов `List<T>` и `Dictionary<TKey, TValue>`.

4 Назовите основные методы и свойства классов `List<T>` и `Dictionary<TKey, TValue>`.

5 Какие интерфейсы реализуются в классах `List<T>` и `Dictionary<TKey, TValue>`?

6 Перечислите стандартные интерфейсы .NET, которые определяют возможности сортировки и просмотра объектов с помощью оператора `foreach`.

7 Объясните, как используется конструкция `yield`.

8 Что такое итератор? Какой интерфейс описывает свойства и поведение объекта-итератора? Объясните принцип работы итератора.

# 8 Лабораторная работа № 8. Делегаты, события, лямбда-выражения

**Цель работы**: научиться разрабатывать программы с использованием делегатов, событий и лямбда-выражений.

## *8.1 Теоретические сведения*

1 Делегаты [1, с. 473–475].
2 Групповая адресация [1, с. 478–480].
3 Блочные лямбда-выражения [1, с. 492–493].
4 События [1, с. 494–495].
5 Групповая адресация событий [1, с. 496–497].
6 Применение лямбда-выражений вместе с событиями [1, с. 504–505].
7 Обобщенные делегаты [1, с. 610–611].

## *8.2 Задания для самостоятельного выполнения*

1 Разработать программу «Охота на волков». В лесу размером `N x N` хаотически движутся с некоторой скоростью четыре волка и два охотника. При совпадении координат волка и охотника волк исчезает и охотник оповещает другого охотника и лесничество о своей добыче. Если волк достигнет края леса, то он спасается. Процесс движения волков и охотников визуализировать на дисплее с помощью символов псевдографики.

2 Разработать программу «Авиаразведка». Создать условную карту местности размером `N x N`. В ячейках карты расположить некоторое количество различных целей (например, танки, пушки, склады с горюче-смазочными материалами и др.). Из произвольной точки, находящейся на границе карты, стартует самолёт-разведчик. Достигнув противоположной границы карты, он меняет курс и летит в обратном направлении. Так продолжается до тех пор, пока не будет покрыта вся карта местности. Самолёт-разведчик фиксирует цели, чьи координаты совпадают с его координатами и по достижении границ карты сообщает в штаб информацию о типе и количестве обнаруженных целей. Процесс полёта самолета и расположение целей визуализировать с помощью символов псевдографики.

3 Создать программу, в которой реализуется следующая ситуация. В лесу размером `NxN` потерялась девочка. Она хаотично ходит по лесу со скоростью `V`, и при этом кричит «ауу» в надежде, что кто-нибудь её услышит. По лесу также хаотично двигаются со скоростью `2V` – мама, папа, дедушка и бабушка девочки. Если кто-нибудь из них услышит девочку, а услышать её могут, если между девочкой и слушателем не более четырёх квадратов леса, то девочка будет спасена.

Если она самостоятельно достигнет края леса, то также будет спасена. Процесс поиска визуализировать, используя символы псевдографики.

4 Создать программу «Сапёр». В программе реализовать класс `Bomb` и класс `TimerBomb`.

Функциональность класса `Bomb`: конструктор, передаваемый параметр – время, через которое бомба взорвется; метод `Code` – принимает код обезвреживания бомбы, если принятый код совпал с кодом сгенерированным бомбой, – бомба считается обезвреженной; при создании экземпляра бомбы генерируется случайный код в заданном диапазоне и задается время срабатывания бомбы (например, код от 1 до 10, время срабатывания – 5 с).

Функциональность класса `TimerBomb` – генерирует событие `OnTick` через заданный промежуток времени. Для генерации события `OnTick` можно использовать класс `Timer` из пространства имён `System.Timers`.

### *Контрольные вопросы*

1 Что такое делегат?
2 Назовите этапы создания и применения делегатов.
3 Как осуществляется вызов методов с помощью делегата?
4 Что такое многоадресная передача делегатов?
5 Какие операторы языка `C#` используются для создания и удаления цепочки методов для многоадресных делегатов?
6 Каким должен быть тип возвращаемого значения для многоадресных делегатов?
7 Для чего используются обобщенные делегаты `Func< >` и `Action< >`?
8 Что такое лямбда-выражение?
9 Объясните синтаксис создания одиночных и блочных лямбда-выражений.
10 Как написать лямбда-выражение, соответствующее делегату?
11 Что такое событие?
12 В чём заключается механизм события?
13 Каков порядок создания пользовательского события?
14 Как используются методы класса в роли обработчика события?
15 Какой синтаксис должны иметь .NET-совместимые обработчики событий?

# 9 Лабораторная работа № 9. Использование LINQ для работы с данными

**Цель работы**: ознакомиться с языком интегрированных запросов LINQ и его использованием для работы с данными.

## *9.1 Необходимые теоретические сведения*

1 Основы LINQ: простой запрос, неоднократное выполнение запросов, связь между типами данных в запросе, общая форма запроса [1, с. 638–643].

2 Отбор запрашиваемых значений с помощью оператора `where` [1, с. 644–645].

3 Сортировка результатов запроса с помощью оператора `orderby` [1, с. 646–648].

4 Оператор `select` [1, с. 649–652].

5 Применение вложенных операторов `from` [1, с. 653–654].

6 Группирование результатов с помощью оператора `group` [1, с. 655–657].

7 Продолжение запроса с помощью оператора `into` [1, с. 655–657].

8 Применение оператора `let` для создания временной переменной в запросе [1, с. 659–660].

9 Объединение двух последовательностей с помощью оператора `join` [1, с. 660–663].

10 Анонимные типы [1, с. 663–665].

11 Методы запроса. Формирование запросов с помощью методов запроса [1, с. 670–675].

### *9.2 Задания для самостоятельного выполнения*

Разработать консольное приложение согласно варианту индивидуального задания. Запросы для поиска данных составлять с использованием технологии доступа к данным LINQ. Набор данных реализовать с помощью динамических структур данных, используя обобщенный класс `List<T>` – список (варианты заданий с чётными номерами) или обобщенный класс `Queue<T>` – очередь (варианты заданий с чётными номерами). Запись в задании реализовать в виде класса.

1 Счет в банке представляет собой структуру с полями: номер счета, код счета, фамилия владельца, сумма на счете, дата открытия счета, годовой процент начисления. Поиск по номеру счета, дате открытия и владельцу.

2 Запись о товаре на складе представляет собой структуру с полями: номер склада, код товара, наименование товара, дата поступления на склад, срок хранения в днях, количество единиц товара, цена за единицу товара. Поиск по номеру склада, коду товара, дате поступления и сроку хранения (просроченные и непросроченные товары).

3 Запись о преподаваемой дисциплине представляется следующей структурой: код дисциплины в учебном плане, наименование дисциплины, фамилия преподавателя, код группы, количество студентов в группе, количество часов лекций, количество часов практики, наличие курсовой работы, вид итогового контроля (зачет или экзамен). Зачет – 0,35 ч на одного студента, экзамен – 0,5 ч на студента. Поиск осуществлять по фамилии преподавателя, коду группы, наличию курсовой, виду итогового контроля.

4 Информационная запись о книге, выданной на руки абоненту, представляет собой структуру следующего вида: номер читательского билета, фамилия абонента, дата выдачи, срок возврата (количество дней), автор, название, год издания, издательство, цена. Поиск по полям: номер читательского билета, автор, издательство, дата возврата (просроченные).

5 Информационная запись о файле содержит поля: каталог, имя файла, расширение, дата и время создания, атрибуты «только чтение», «скрытый», «системный», признак удаления, количество выделенных секторов (размер сектора принять равным 512 байт). Поиск выполнять по каталогу, дате создания, по признаку удаления.

6 Разовый платеж за телефонный разговор является структурой с полями: фамилия плательщика, номер телефона, дата разговора, тариф за минуту разговора, скидка (в процентах), время начала разговора, время окончания разговора. Поиск по фамилии, дате разговора, номеру телефона.

7 Модель компьютера характеризуется кодом и названием марки компьютера, типом процессора, частотой работы процессора, объемом оперативной памяти, объемом жесткого диска, объемом памяти видеокарты, стоимостью компьютера в условных единицах и количеством экземпляров, имеющихся в наличии. Поиск по типу процессора, объему ОЗУ, памяти видеокарты и жесткого диска.

8 Список абонентов сети кабельного телевидения состоит из элементов следующей структуры: фамилия, район, адрес, телефон, номер договора, дата заключения договора, оплата установки, абонентская плата помесячно, дата последнего платежа. Поиск по фамилии, району, дате заключения договора, дате последнего платежа.

9 Сотрудник представлен структурой `Person` с полями: табельный номер, номер отдела, фамилия, оклад, дата поступления на работу, процент надбавки, подоходный налог, количество отработанных дней в месяце, количество рабочих дней в месяце, начислено, удержано. Поиск по номеру отдела, полу, дате поступления, фамилии.

10 Запись о багаже пассажира авиарейса содержит следующие поля: номер рейса, дата и время вылета, пункт назначения, фамилия пассажира, количество мест багажа, суммарный вес багажа. Поиск выполнять по номеру рейса, дате вылета, пункту назначения, весу багажа (превышение максимально допустимого).

11 Одна учетная запись посещения спорткомплекса имеет структуру: фамилия клиента, код и вид спортивного занятия, фамилия тренера, дата и время начала, количество минут, тариф за минуту. Поиск по фамилии клиента и тренера, по виду занятия, по дате начала, по количеству минут (больше или меньше).

12 Одна запись о медикаменте содержит следующие поля: номер аптеки, название лекарства, количество упаковок, имеющееся в наличии в данной аптеке, стоимость одной упаковки, дата поступления в аптеку, срок хранения (в днях). Поиск по номеру аптеки, наименованию препарата, дате поступления.

13 Одна запись журнала содержит поля: код игрушки, название игрушки, тип игрушки, возрастные границы (например, от 10 до 15), цена за единицу, количество в наличии, дата поступления в магазин, поставщик. Поиск по дате поступления, поставщику, возрастным границам.

14 Один элемент – автомобиль – представляет собой в базе данных структуру с полями: фамилия владельца, код марки автомобиля, марка автомобиля, требуемая марка бензина, мощность двигателя, объем бака, остаток бензина,

объем масла. Дана фиксированная цена литра бензина и заливки масла. Поиск по марке автомобиля, марке бензина, мощности двигателя, фамилии владельца.

15 Одна запись в журнале зимней экзаменационной сессии представляет собой структуру с полями: курс, код группы, фамилия студента, номер зачетной книжки, дисциплина, оценка за экзамен по дисциплине. Вычисляются средние баллы по дисциплине, по группе, по курсу. Поиск по курсу, по группе, по номеру зачетной книжки, по фамилии, по оценкам.

### *Контрольные вопросы*

1 Что такое LINQ?

2 Каковы структура и работа простого запроса LINQ?

3 Какой оператор используется для отбора данных, возвращаемых по запросу?

4 Как отсортировать результаты запроса LINQ?

5 Объясните назначение следующих операторов запроса LINQ: `group`, `into`, `let`, `join`.

6 Для чего предназначен анонимный тип? Опишите синтаксис анонимного типа.

7 Объясните назначение основных методов запроса LINQ.

## 10 Лабораторная работа № 10. Регулярные выражения

**Цель работы:** получить практические навыки по программированию регулярных выражений в языке C#.

### *10.1 Теоретические сведения*

1 Регулярные выражения [3, с. 355–359].

2 Классы библиотеки .NET для работы с регулярными выражениями [3, с. 359–365].

### *10.2 Задания для самостоятельного выполнения*

1 В файле с телефонными переговорами клиента определить по какому номеру выполнено максимальное количество звонков.

2 Выполнить анализ кода программы на наличие в нем всех циклов `for` языка C#. Напечатать результаты анализа на экране монитора.

3 Считать текст из файла и вывести на экран монитора строку, содержащую максимальное количество знаков пунктуации.

4 В файле с телефонными переговорами клиента выполнить сортировку переговоров по их стоимости.

5 Выполнить анализ кода программы на наличие в нем комментариев языка C#. Напечатать результаты анализа на экране монитора.

6 Считать текст из файла и вывести его на экран монитора, заменив цифры словами, например, «0» на слово «ноль»; «1» на слово «один» и т. д.

7 В файле с телефонными переговорами клиента определить, по какому номеру выполнено максимальное количество звонков.

8 Выполнить анализ кода программы на наличие в нем всех операторов присваивания языка `C#` (не учитывать записи типа `for (i=0; i<=10; i++)...`).

9 Напечатать результаты анализа на экране монитора. Считать текст из файла и вывести на экран монитора строку, содержащую максимальное количество чисел (не цифр). В файле с телефонными переговорами клиента определить, по какому номеру был самый продолжительный по времени разговор.

10 Выполнить анализ кода программы на наличие в нем всех операторов вывода информации на экран (консоль) языка `C#`. Напечатать результаты анализа на экране монитора.

11 Считать текст из файла и вывести его на экран монитора только цитаты текста, то есть предложения, заключенные в кавычки.

12 В файле с телефонными переговорами клиента определить, по какому номеру выполнены подряд (2 и более соединений) звонки.

13 Выполнить анализ кода программы на наличие в нем всех операторов ввода информации с клавиатуры (консоли) языка C#. Напечатать результаты анализа на экране монитора.

14 Считать текст из файла и вывести на экран монитора строку, содержащую максимальное количество повторяющихся слов.

15 В файле с телефонными переговорами клиента определить все номера, время связи с которыми было менее 5 секунд.

*Примечание 1* – В некоторых вариантах индивидуальных заданий необходимо выполнить анализ файла телефонных переговоров клиента. В этом файле, кроме прочей информации, должны содержаться записи об абонентах, включающих следующие сведения:

– номер абонента, с которым выполнялось соединение;
– дату и время соединения;
– продолжительность соединения;
– стоимость переговоров.

*Примечание 2* – Если требуется выполнить анализ кода программы на наличие в нем различных элементов языка C#, то код программы должен быть представлен текстовым файлом с расширением «.cs», в который следует записать необходимую информацию в соответствии с индивидуальным заданием.

### *Контрольные вопросы*

1 Для чего предназначены регулярные выражения?

2 Перечислите основные действия, которые можно выполнять над строками с помощью регулярных выражений.

3 Из каких элементов состоит язык описания регулярных выражений?

4 Что такое метасимволы?

5 Перечислите наиболее употребительные метасимволы.

6 Что такое мнимые метасимволы? Приведите примеры использования мнимых метасимволов.

7 Какую роль в регулярных выражениях выполняют повторители? Приведите примеры повторителей.

8 Перечислите основные методы регулярных выражений.

# 11 Лабораторная работа № 11. Многопоточное программирование

**Цель работы**: ознакомиться с организацией многопоточной обработки данных; получить основные навыки программирования с использованием потоков.

## *11.1 Теоретические сведения*

1 Основы многопоточной обработки [1, с. 834–835].
2 Создание и запуск потока [1, с. 836–838].
3 Создание нескольких потоков [1, с. 839–841].
4 Определение момента окончания потока [1, с. 841–844].
5 Передача аргумента потоку [1, с. 844–846].
6 Приоритеты потоков [1, с. 847–849].
7 Синхронизация потоков [1, с. 849–853].

## *11.2 Задания для самостоятельного выполнения*

Разработать консольное приложение согласно варианту. Требования к программе: реализовать возможность задавать приоритет каждого из порожденных потоков; использовать символы псевдографики для визуализации потоков на экране дисплея.

1 Умножение матрицы на вектор. Обработку одной строки матрицы производить в порожденном потоке.

2 Поиск всех простых чисел (простым называется число, которое является своим наибольшим делителем) в указанном интервале чисел, разделенном на несколько диапазонов. Обработка каждого диапазона производится в порожденном потоке.

Классический алгоритм Евклида определения наибольшего общего делителя двух целых чисел `(x, y)` может применяться при следующих условиях:

– оба числа `x` и `y` неотрицательные;

– оба числа `x` и `y` отличны от нуля.

На каждом шаге алгоритма выполняются сравнения:

– если `x == y`, то ответ найден;

– если `x < y`, то `y` заменяется значением `y-x`;

– если `x > y`, то `x` заменяется значением `x-y`.

3 Винни-Пух и пчелы. Заданное количество пчел добывают мед равными порциями, задерживаясь в пути на случайное время. Винни-Пух потребляет мед порциями заданной величины за заданное время и столько же времени может прожить без питания. Работа каждой пчелы реализуется в порожденном потоке.

4 Шарики. Координаты заданного количества шариков изменяются на случайную величину по вертикали и горизонтали. При выпадении шарика за нижнюю границу допустимой области шарик исчезает. Изменение координат каждого шарика в отдельном потоке.

5 Противостояние нескольких команд. Каждая команда увеличивается на случайное количество бойцов и убивает случайное количество бойцов участника. Борьба каждой команды реализуется в отдельном потоке.

6 Контрольная сумма. Для нескольких файлов (разного размера) требуется вычислить контрольную сумму (сумму кодов всех символов файла). Обработка каждого файла выполняется в отдельном потоке.

7 Бег с препятствиями. Создается условная карта трассы в виде матрицы, ширина которой соответствует количеству бегунов, а высота фиксирована, которая содержит произвольное количество единиц (препятствий) в произвольных ячейках. Стартующие бегуны (потоки) перемещаются по трассе и при встрече с препятствием задерживаются на фиксированное время. По достижении финиша бегуны сообщают свой номер.

8 Создать два потока. Первый ищет числа Фибоначчи (каждое последующее число равно сумме двух предыдущих чисел), второй – простые числа. Результат работы каждого потока сохраняется в отдельный файл. После остановки потока программа производит анализ файлов, выводит их на экран, а так же показывает количество найденных чисел Фибоначчи и простых чисел.

9 Создать два потока. Первый поток производит запись в файл случайных данных, второй производит чтение данных из этого файла и вывод их на экран.

10 Создать приложение, выполняющее вычисление значений функции `y = 23*x^2 - 33`, с шагом `x=0.01`. Первый поток выполняет расчёт функции и добавляет результаты расчёта в конец массива. Второй поток извлекает из массива значения x и y и выводит их на экран.

11 Создать приложение, которое выполняет сортировку массива данных и отображает процесс сортировки на экране. Первый поток производит сортировку по возрастанию, второй – по убыванию. После каждого перемещения элементов производится вывод на экран текущего состояния сортировки. Каждый поток работает с отдельным экземпляром массива данных.

12 Создать игру, где будут 2…3 барана и волк. При совпадении координат волка с бараном баран исчезает. При совпадении координат баранов появляется новый баран. Все движутся хаотически.

13 Создать три потока, генерирующих случайным образом целые числа от 0 до 9. При нажатии на клавишу «Enter» потоки останавливаются и результат анализируется. Цель анализа – выявить наличие следующие комбинаций цифр в потоках: три одинаковых числа, два одинаковых числа, три единицы, три семерки, две единицы.

14 Создать три потока, каждый из которых управляет перемещением псевдосимвола на экране вдоль оси `X`, и устроить «тараканьи бега» среди них.

15 Поиск указанной строки в указанном файле. Обработка одной строки в порожденном потоке.

### *Контрольные вопросы*

1 Что такое поток?
2 Как создать новый поток?
3 Чем поток отличается от процесса?
4 Как запустить поток после завершения другого потока?
5 Как присвоить потоку имя?
6 Как заблокировать данные от изменения другими потоками?
7 Как остановить выполнение потока?

## 12 Лабораторная работа № 12. Создание объектной модели предметной области

**Цель работы**: получить навыки создания объектной модели предметной области.

### *12.1 Теоретические сведения*

1 Основные элементы объектной модели [6, с. 7–11].
2 Отношения между объектами и классами [6, с. 11–14].
3 Объектная модель вузовской информационной системы [6, с. 15–20].

### *12.2 Задания для самостоятельного выполнения*

1 Создать объектную модель информационной системы «Домашняя библиотека».
2 Создать объектную модель информационной системы «Электронный магазин».
3 Создать объектную модель информационной системы «Адресная книга».
4 Создать объектную модель информационной системы «Салон видеопроката».
5 Создать объектную модель информационной системы «Бронирование номеров в гостинице».
6 Создать объектную модель информационной системы «Салон красоты».
7 Создать объектную модель информационной системы «Поликлиника».
8 Создать объектную модель информационной системы «Цветочный магазин».
9 Создать объектную модель информационной системы «Склад продовольственных товаров».
10 Создать объектную модель информационной системы «Служба такси».

### *Контрольные вопросы*

1 Что такое объектная модель предметной области? С какой целью она создается?

2 Дайте определение основным элементам объектной модели: объект, состояние объекта, поведение объекта, класс.

3 Перечислите возможные типы отношений между классами и объясните их сущность.

4 Как на диаграмме классов представляются отношения ассоциации, агрегации, композиции, обобщения, зависимости между классами?

5 Перечислите основные этапы создания объектной модели предметной области.

# 13 Лабораторная работа № 13. Проектирование программной системы

**Цель работы**: ознакомить студентов с методом проектирования системы путем CRC-карт.

## *13.1 Теоретические сведения*

Важным этапом создания программного обеспечения является проектирование. На этом шаге закладывается архитектура системы. Одним из способов проектирования является метод CRC-карточек. Этот метод проектирования является составляющей UML-проектирования.

*Шаг первый.* Для первоначального понимания структуры программы строится диаграмма вариантов использования: выявляются акторы (люди или системы, между которыми происходит взаимодействие), прецеденты системы (действия, выполняемые системой для реализации общения акторов).

Диаграмма вариантов использования для примера «Банкомат» приведена на рисунке 13.1.


Рисунок 13.1 – Диаграмма вариантов использования «Банкомат»

На самом деле прецедентов может быть очень много. Допустим: проверить пароль, контролировать транзакции передачи данных, выдать информацию на экран и т. д.

Эта диаграмма дает понять, что будет делать система, как она будет функционировать. Диаграмма использования бывает также очень полезна для общения с заказчиком – она позволяет показать наиболее значимые действия системы и проверить, правильно ли вы поняли заказчика и значимость отдельных функций для него.

*Шаг второй.* На этом этапе выявляют классы, которые необходимо будет создать в программе для реализации системы. В случае банкомата это: клиент, банк, служба безопасности банка, сам банкомат и т. д. Придумать можно много (таймер, счетчик купюр, карточка и т. д.). Далее оформляются CRC-карты. CRC-карта (Class-responsibility-collaboration card – карта «Класс-Ответственность-Кооперация») – метод, предназначенный для проектирования объектно-ориентированного программного обеспечения. CRC-карты используются в тех случаях, когда сначала в процессе проектирования ПО определяются классы и способы их взаимодействий. CRC-карты представляют собой листки бумаги, разделенные на три части (рисунок 13.2.) Примеры CRC-карт банкомата показаны на рисунке 13.3.

| Название класса | |
|---|---|
| Действия, которые он выполняет (всегда начинаются с глагола) | Классы, с которыми данный класс обменивается информацией |

Рисунок 13.2 – Оформление CRC-карты

| Клиент | |
|---|---|
| 1. Вставляет карточку в банкомат.<br>2. Вводит пароль.<br>3. Указывает тип операции (снять деньги, просмотреть остаток).<br>4. Вводит сумму.<br>5. Получает деньги.<br>6. Вынимает карточку | Банкомат |

| Банкомат | |
|---|---|
| 1. Отображает информацию для клиента.<br>2. Передает информацию в банк.<br>3. Отсчитывает купюры.<br>4. Распечатывает счет | Клиент<br>Банк<br>Служба безопасности банка |

| Служба безопасности банка | |
|---|---|
| 1. Проверяет пароль.<br>2. Проверяет подлинность карточки.<br>3. Идентифицирует клиента.<br>4. Следит за правильностью транзакций операций с деньгами | Банк<br>Банкомат |

| Банк | |
|---|---|
| 1. Проверяет возможность выдачи средств.<br>2. Сообщает о наличии денег.<br>3. Выдает информацию об остатке.<br>4. Хранит информацию о счете клиента | Банкомат<br>Служба безопасности банка |

Рисунок 13.3 – Примеры CRC-карт

*Шаг третий.* Для проверки достаточности или избыточности придуманных классов, а также корректности их взаимодействия строится диаграмма последовательности (рисунок 13.4).

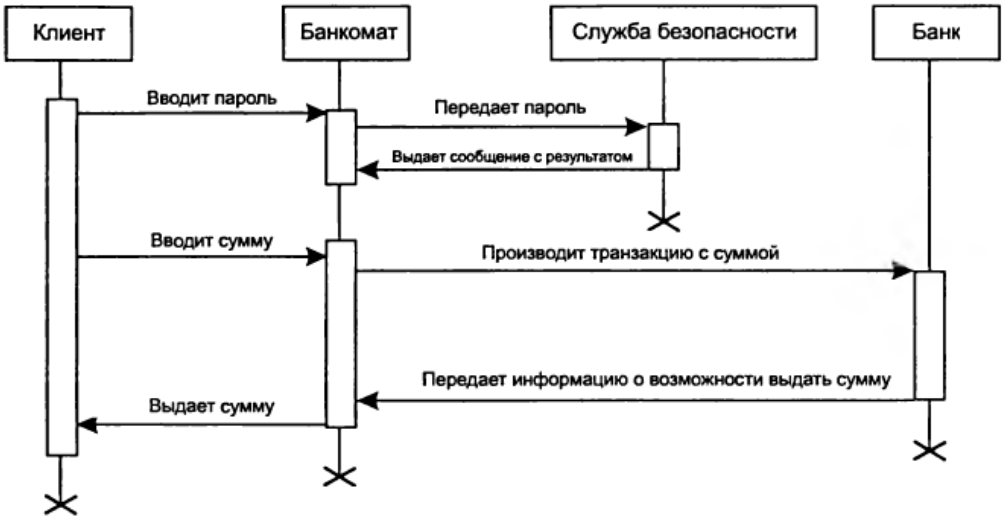



Рисунок 13.4 – Диаграмма последовательности

Метод CRC-карточек позволяет провести также инсценировку работы системы. Для этого достаточно раздать карточки с классами участникам проекта. После этого начать ролевую игру. Первый участник встает и читает действие, совершаемое его классом. Другие участники, исходя из своих карточек, сообщают об ответной реакции других классов. Если в какой-то момент реакции не последует, то это признак несовершенства проекта системы. Такая игра может подсказать и об избыточности проекта.

## *13.2 Задания для самостоятельного выполнения*

Для информационной системы из лабораторной работы № 12 необходимо:

– определить действующих лиц (акторов) системы;

– определить варианты использования системы и описать их в краткой или полной форме;

– построить диаграмму вариантов использования системы;

– определить классы проектируемой системы;

– создать CRC-карты для всех классов системы;

– построить диаграмму взаимодействия объектов системы.

## *Контрольные вопросы*

1 Охарактеризуйте проектирование прикладной программы при объектном подходе.

2 В чем заключается моделирование предметной области при проектировании прикладной программы?

3 Язык UML. Его назначение, преимущества и недостатки.
4 Перечислите диаграммы в языке UML.
5 Приведите пример диаграммы прецедентов.
6 Приведите пример диаграммы взаимодействия.
7 В чем состоит назначение и использование CRC-карт?

# 14 Лабораторная работа № 14. Разработка приложений с использованием паттерна фабричный метод

**Цель работы**: изучить паттерн проектирования фабричный метод; освоить применение паттерна при разработке прикладных программ.

## *14.1 Теоретические сведения*

Порождающий паттерн фабричный метод [7].

## *14.2 Задания для самостоятельного выполнения*

1 В базе данных системы электронного документооборота обрабатываются документы трех типов: письма, приказы и распоряжения о командировке. Каждый документ имеет номер, дату и краткую информацию о содержании. Кроме того, в письме указывается тип (входящее/исходящее) и корреспондент, в приказе – подразделение, срок выполнения и ответственный исполнитель, в распоряжении о командировке – сотрудник, период и место назначения. Необходимо вывести:

– полный перечень документов;
– содержание выбранного документа (по номеру документа).

2 В базе данных компьютерного интернет-магазина хранится номенклатура товаров трех типов: материнские платы, процессоры, жесткие диски. Каждый товар имеет номенклатурный номер, наименование и стоимость. Кроме того, для материнских плат указывается тип сокета, количество процессоров, тип оперативной памяти, частота системной шины, для процессоров – тип сокета, количество ядер, тактовая частота системной шины, техпроцесс, для жестких дисков – объем, скорость вращения, тип интерфейса. Необходимо вывести:

– полную номенклатуру комплектующих;
– детальную информацию по товару (по номенклатурному номеру).

## *Контрольные вопросы*

1 Назначение паттерна фабричный метод.
2 Когда надо использовать паттерн фабричный метод?
3 Какую проблему решает паттерн фабричный метод?
4 Поясните структуру паттерна фабричный метод, используя язык UML.
5 Как реализуется паттерн фабричный метод на языке C#?

# 15 Лабораторная работа № 15. Разработка приложений с использованием паттерна компоновщик

**Цель работы**: изучить паттерн проектирования компоновщик; освоить применение паттерна при разработке прикладных программ.

## *15.1 Теоретические сведения*

Структурный паттерн компоновщик [7].

## *15.2 Задания для самостоятельного выполнения*

1 В информационной системе хранится организационная структура предприятия (иерархия структурных подразделений). Для каждого структурного подразделения задан код, наименование и перечень должностей по штатному расписанию: наименование должности, количество ставок, оклад. Необходимо разработать набор классов для представления организационной структуры с использованием заданного шаблона проектирования. Должны быть реализованы возможности вывода штатного расписания для любого подразделения (включая подчиненные подразделения) с расчетом общего количества штатных единиц и суммарного оклада. Программа должна предусматривать добавление и удаление подразделений, должностей.

2 В САПР продукции машиностроительного предприятия хранится информация о комплектующих выпускаемой продукции, организованная в виде иерархии, узлами которой являются сборочные единицы, элементарные детали и закупаемые компоненты. Для элементарных деталей и закупаемых компонентов задается стоимость, для сборочных единиц, кроме того, время сборки и количество используемых деталей/компонентов каждого вида. САПР должна обеспечивать вывод иерархии производства продукции, расчет времени и стоимости производства изделия.

### *Контрольные вопросы*

1 Назначение паттерна компоновщик.
2 Когда надо использовать паттерн компоновщик?
3 Какую проблему решает паттерн компоновщик?
4 Поясните структуру паттерна компоновщик, используя язык UML.
5 Как реализуется паттерн компоновщик на языке C#?

# 16 Лабораторная работа № 16. Разработка приложений с использованием паттерна строитель

**Цель работы**: изучить паттерн проектирования строитель; освоить применение паттерна при разработке прикладных программ.

## *16.1 Теоретические сведения*

Структурный паттерн строитель [7].

## *16.2 Задания для самостоятельного выполнения*

1 Информационная система обеспечивает формирование отчета по лабораторным работам студентов. Рассматриваются дисциплины «Базы данных», «Компьютерные сети», «Программирование». Отчет должен включать следующие разделы: титульный лист, цель работы, задание, теоретические сведения, результаты работы, анализ результатов работы, выводы. Необходимо реализовать вывод данных и подготовку отчета в формате html с учетом особенностей дисциплины (наличие схем и других изображений или текста программы и т. п.).

2 Информационная система предназначена для сборки комплектации автомобиля. В зависимости от марки и модели комплектация может включать, например, следующие категории элементов: экстерьер, интерьер, комфорт, безопасность, мультимедиа и т. п. Реализовать сборку различных комплектаций для выбранной модели автомобиля и подготовку сравнительной таблицы опций в формате html. При выполнении задания использовать реальный каталог выбранного автопроизводителя.

## *Контрольные вопросы*

1 Назначение паттерна строитель.
2 Когда надо использовать паттерн строитель?
3 Какую проблему решает паттерн строитель?
4 Поясните структуру паттерна строитель, используя язык UML.
5 Как реализуется паттерн строитель на языке C#?

# Список литературы

1 **Шилд, Г.** C# 4.0: полное руководство: пер. с англ. / Г. Шилд. – Москва: И. Д. Вильямс, 2011. – 1056 с.

2 **Подбельский, В. В.** Язык Си#. Базовый курс: учебное пособие / В. В. Подбельский. – Москва: Финансы и статистика, 2011. – 384 с.

3 **Павловская, Т. А.** C#. Программирование на языке высокого уровня: учебник для вузов / Т. А. Павловская. – Санкт-Петербург: Питер, 2014. – 432 с.

4 C# 5.0 и платформа .NET 4.5 для профессионалов пер. с англ. / К. Нейгел [и др.] – Москва: И. Д. Вильямс, 2014. – 1440 с.

5 **Зиборов, В. В.** Visual C# 2012 на примерах / В. В. Зиборов. – Санкт-Петербург: БХВ-Петербург, 2013. – 480 с.

6 **Андрианова, А. А.** Объектно-ориентированное программирование на C# : учебное пособие / А. А. Андрианова, Л. Н. Исмагилов, Т. М. Мухтарова. – Казань: Казан. (Приволж.) федер. ун-т, 2012. – 140 с.

7 Паттерны проектирования программных систем: методические указания к проведению лабораторных работ по курсу «Архитектура программных систем» / Сост.: В. А. Алексеев. – Липецк: ЛГТУ, 2016 – 33 с.

8 Объектно-ориентированное программирование: лабораторный практикум / Сост.: А. В. Щербаков. – Минск: БНТУ, 2014. – 38 с.