Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

КРОССПЛАТФОРМЕННОЕ ПРОГРАММИРОВАНИЕ

Лабораторный практикум
для студентов специальности 1 – 40 02 01
«Вычислительные машины, системы и сети»
дневной формы обучения

Минск 2017

Общие требования к оформлению всех лабораторных работ:

- единый стиль кода: http://www.oracle.com/technetwork/java/codeconvtoc-136057.html
- наличие документации к коду https://www.tutorialspoint.com/java/java_documentation.htm

Лабораторная работа №1

Построение кроссплатформенного графического интерфейса

Задание к лабораторной работе:

– необходимо разработать графический интерфейс с использованием менеджера компоновки.

– графический интерфейс должен выглядеть так, как изображено в варианте задания.

– функциональная часть приложения, представленная на диаграмме последовательности, должна быть реализована.

– графический интерфейс должен быть создан посредством одной из следующих библиотек: Swing, SWT, JavaFX.

Вариант 1

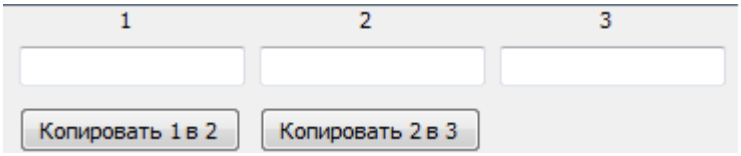

Рисунок 1 – Внешний интерфейс приложения

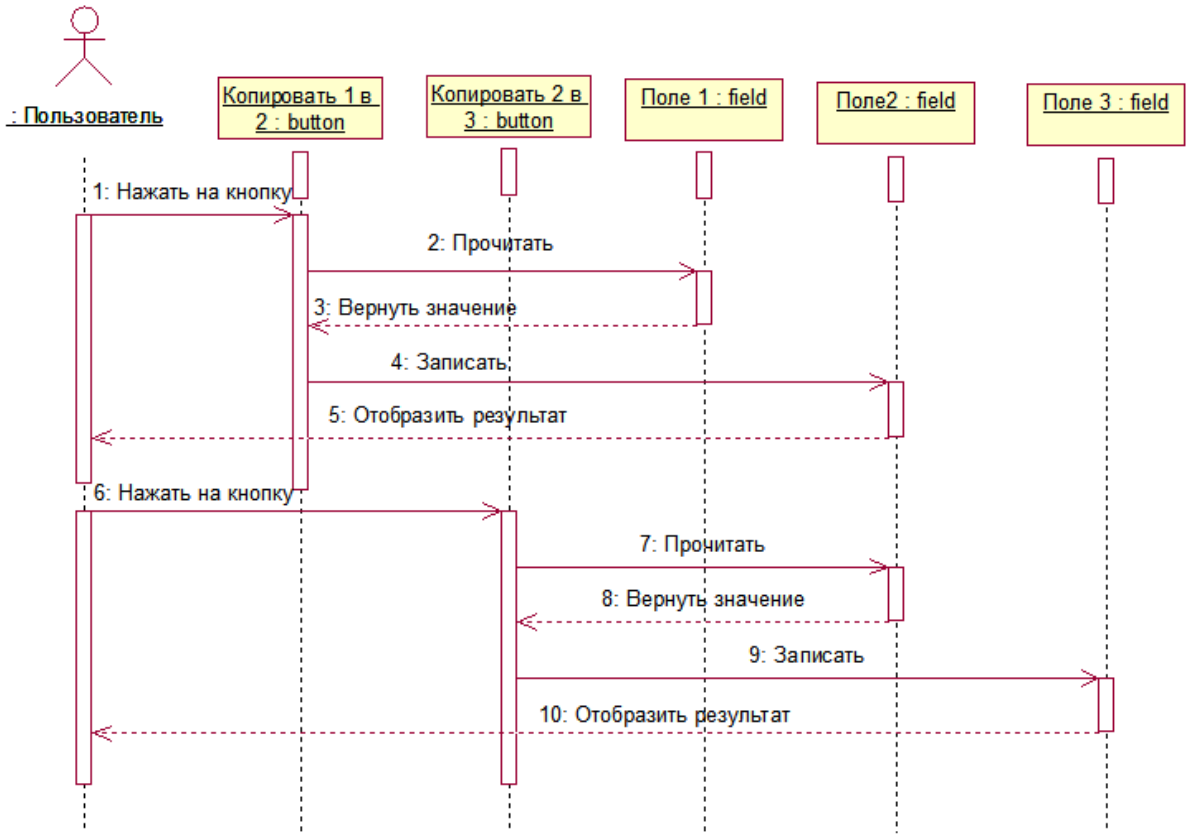

Рисунок 2 – Диаграмма последовательности для приложения

Вариант 2

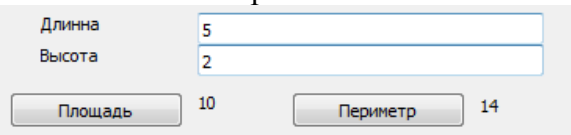

Рисунок 3 – Внешний интерфейс приложения

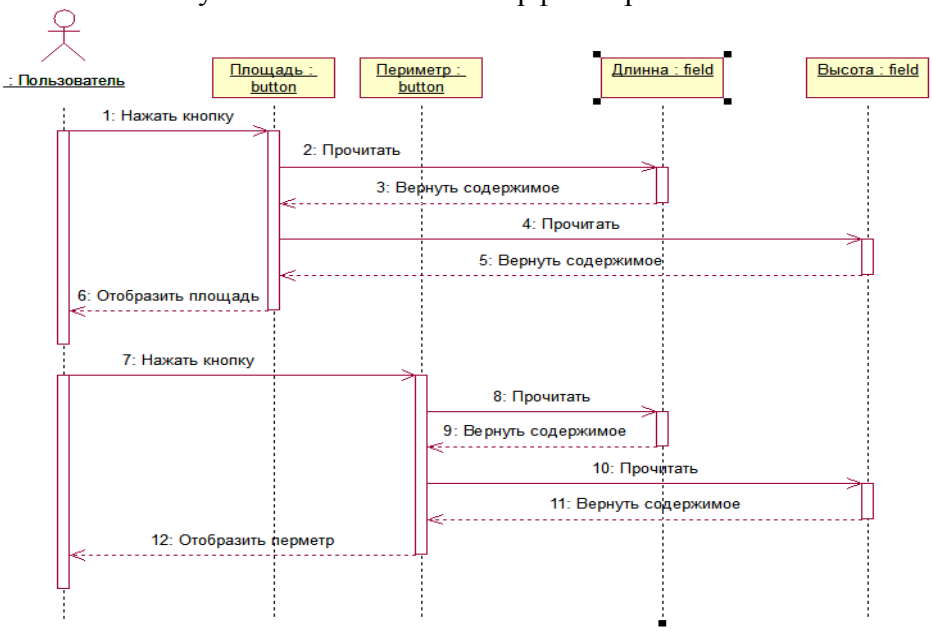

Рисунок 4 – Диаграмма последовательности для приложения

Вариант 3

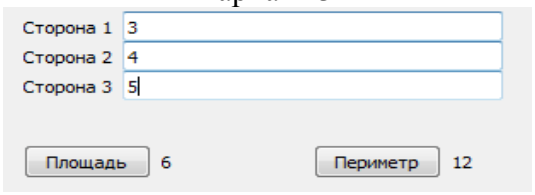

Рисунок 5 – Внешний интерфейс приложения

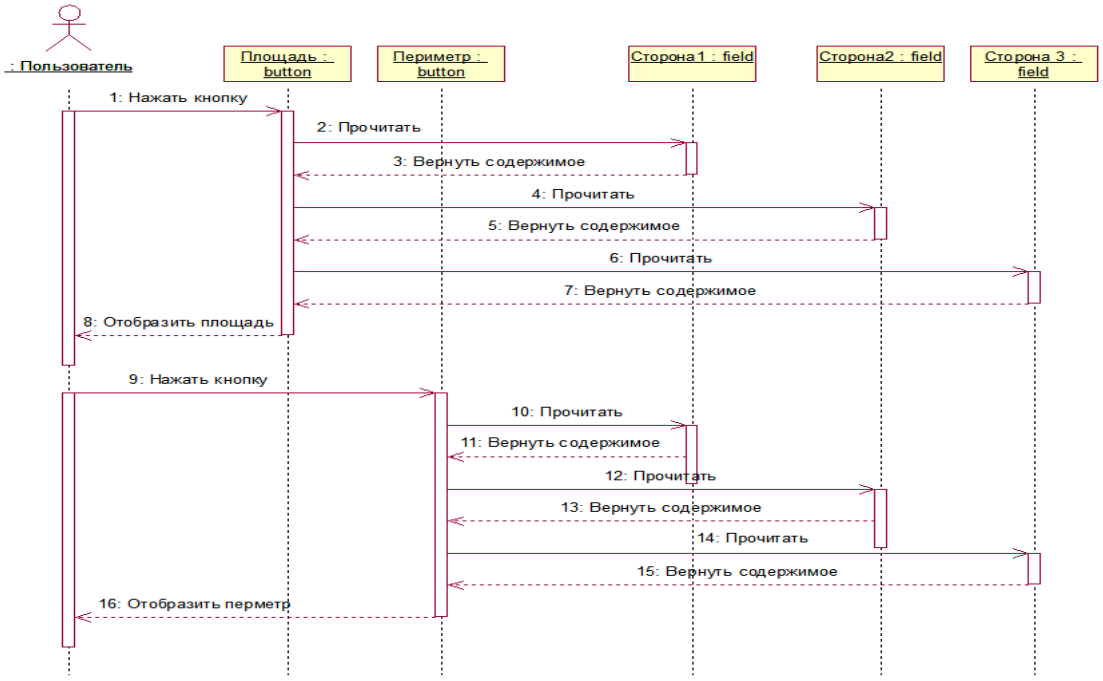

Рисунок 6 – Диаграмма последовательности для приложения

Вариант 4

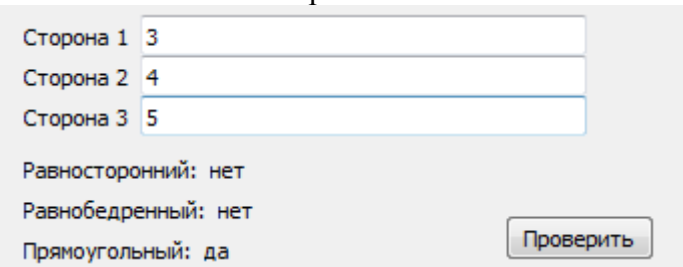

Рисунок 7 – Внешний интерфейс приложения

Рисунок 8 – Диаграмма последовательности для приложения

Вариант 5

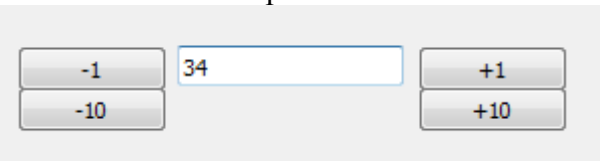

Рисунок 9 – Внешний интерфейс приложения

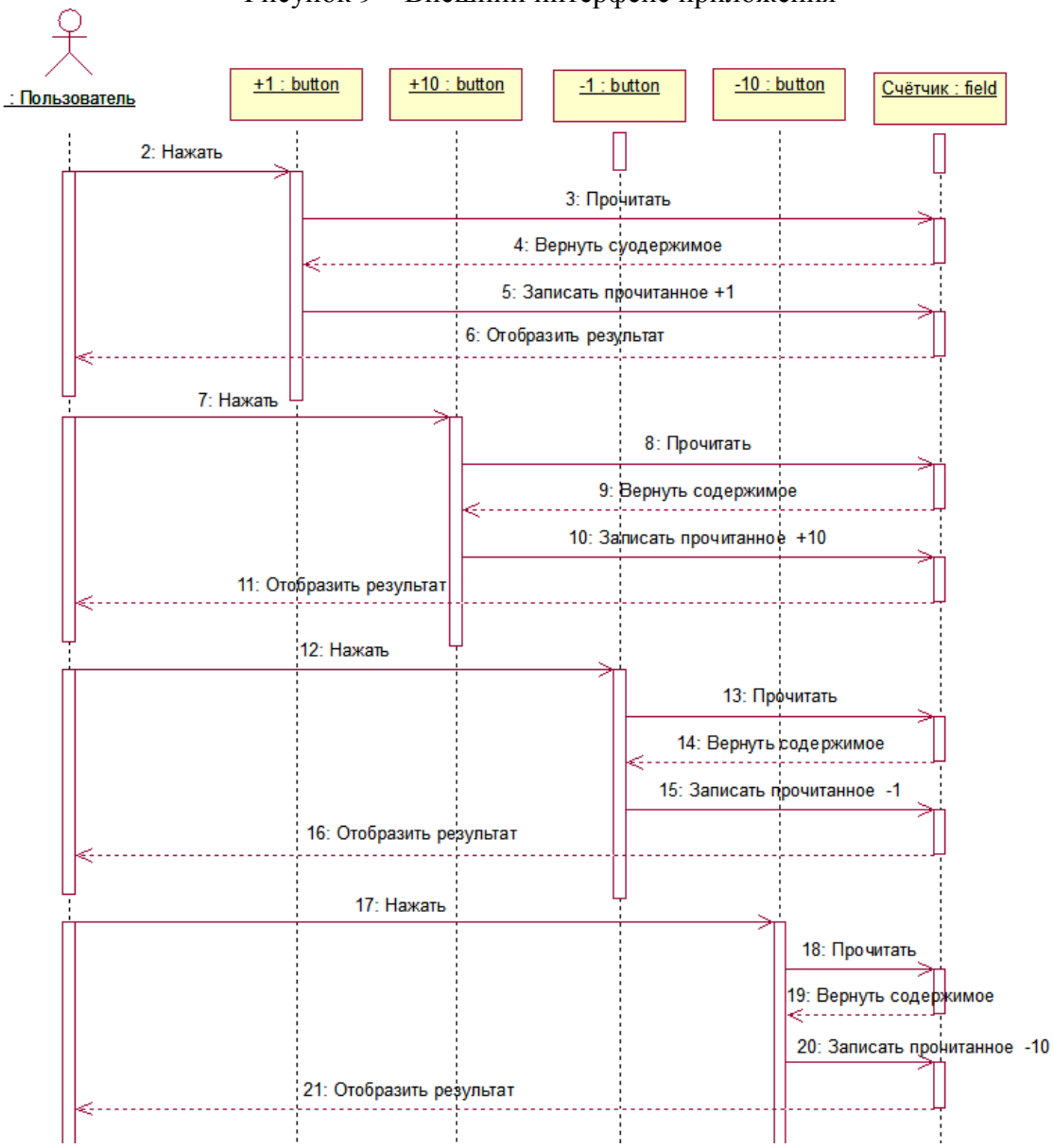

Рисунок 10 – Диаграмма последовательности для приложения

Вариант 6

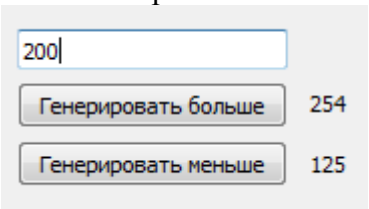

Рисунок 11 – Внешний интерфейс приложения

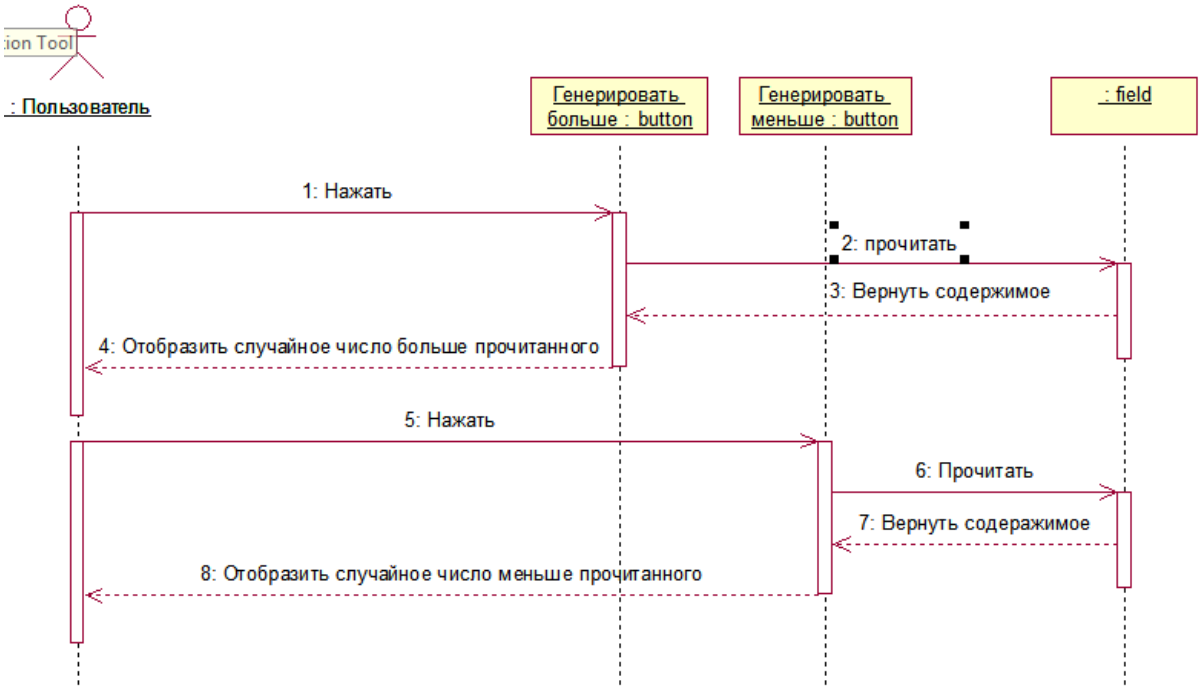

Рисунок 12 – Диаграмма последовательности для приложения

Вариант 7

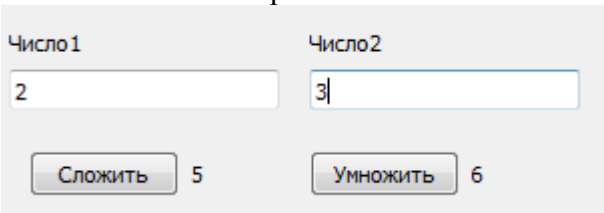

Рисунок 13 – Внешний интерфейс приложения

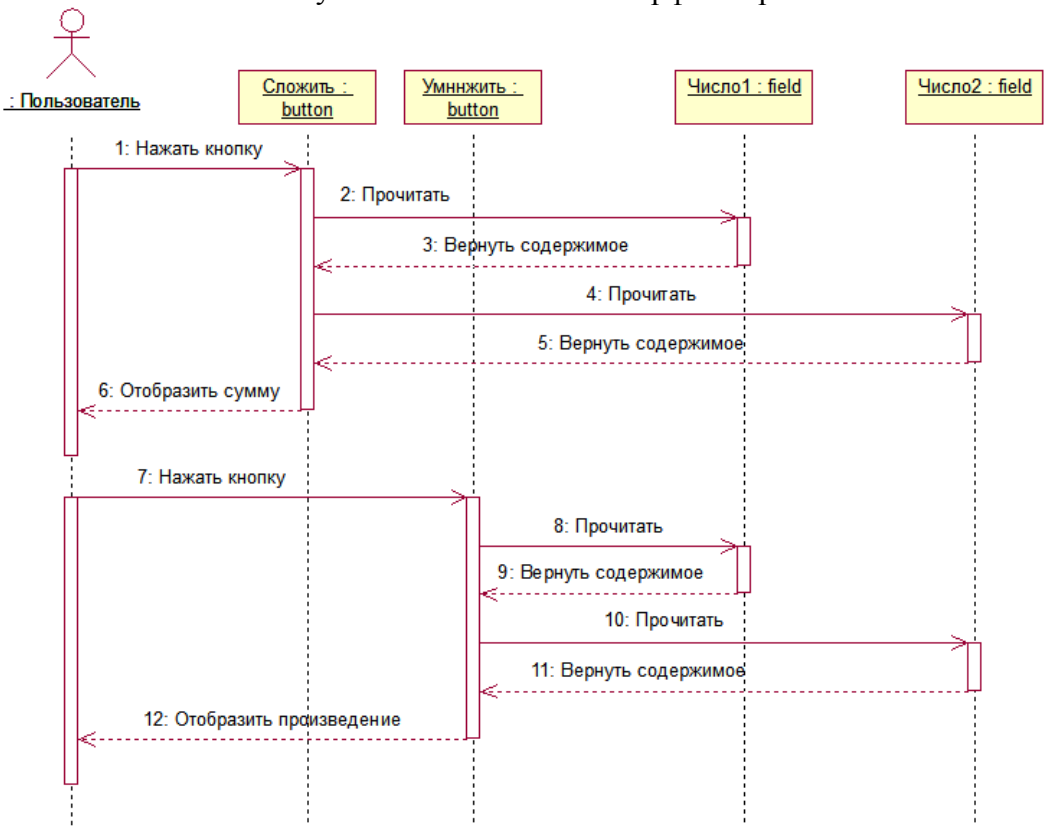

Рисунок 14 – Диаграмма последовательности для приложения

Вариант 8

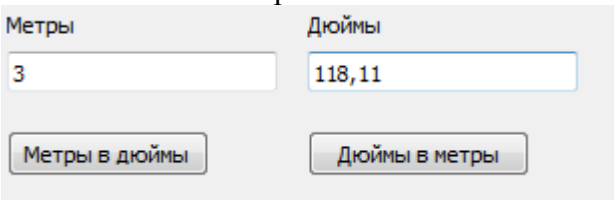

Рисунок 15 – Внешний интерфейс приложения

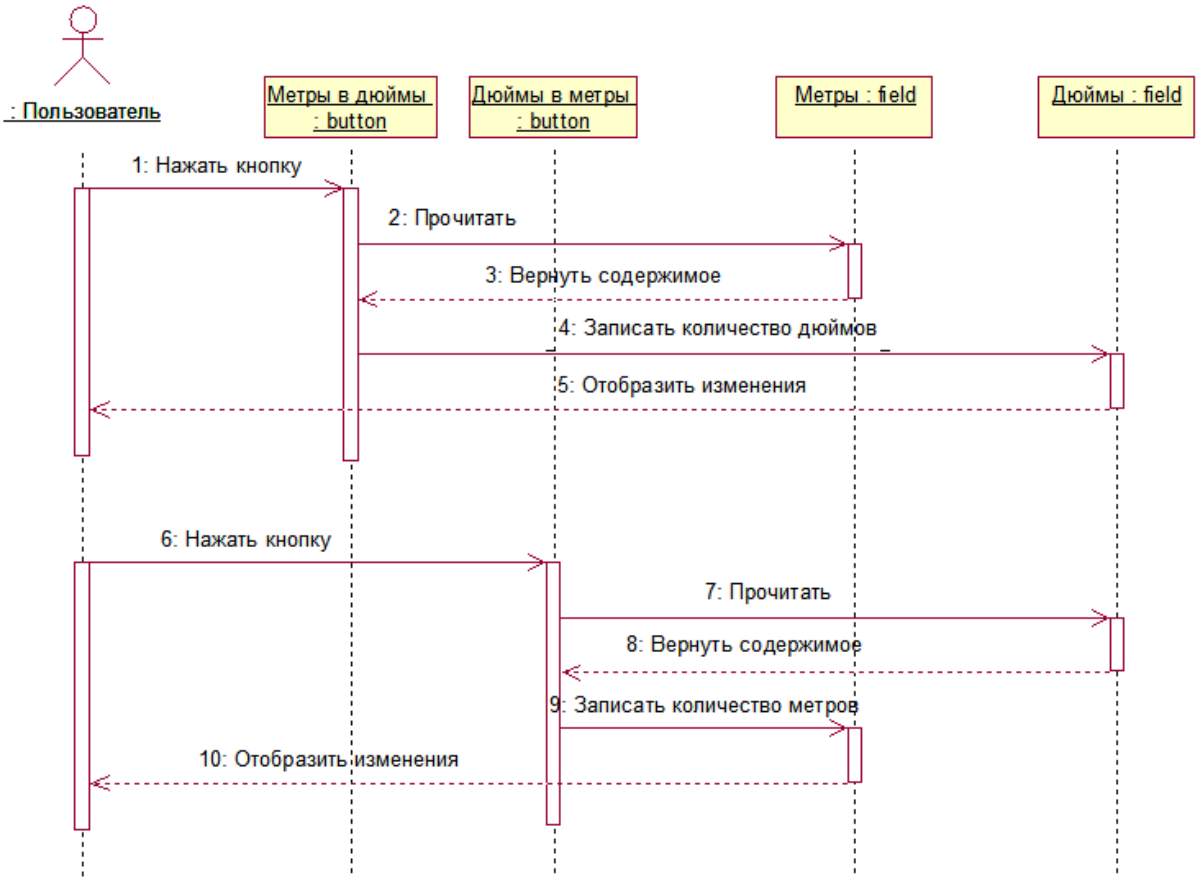

Рисунок 16 – Диаграмма последовательности для приложения

Вариант 9

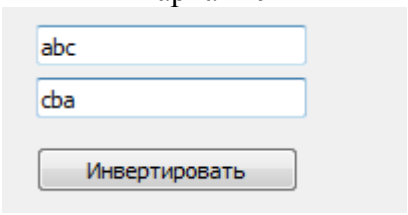

Рисунок 17 – Внешний интерфейс приложения

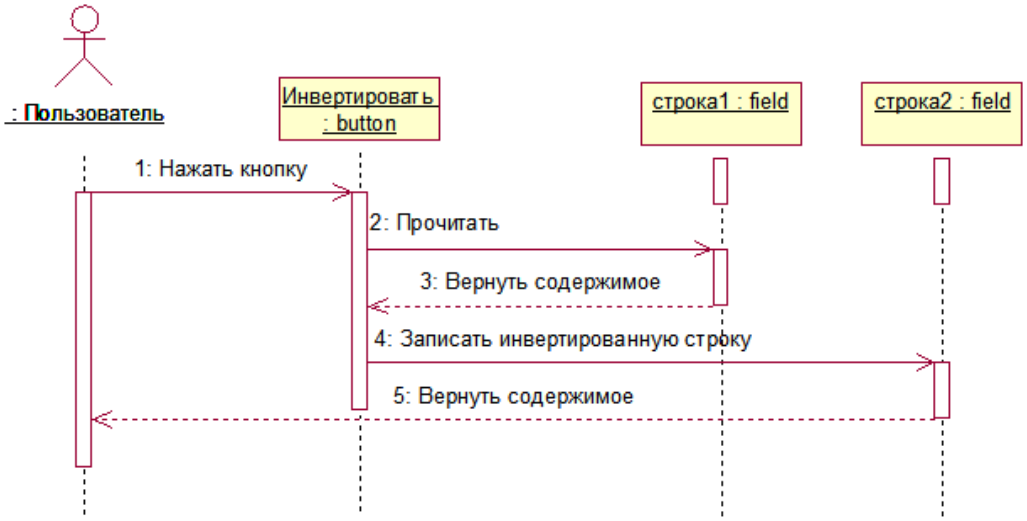

Рисунок 18 – Диаграмма последовательности для приложения

Вариант 10

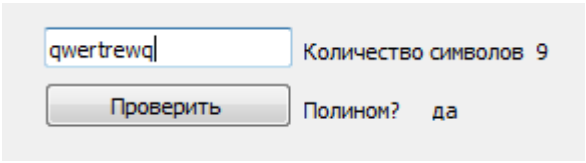

Рисунок 19 – Внешний интерфейс приложения

Рисунок 20 – Диаграмма последовательности для приложения

Вариант 11

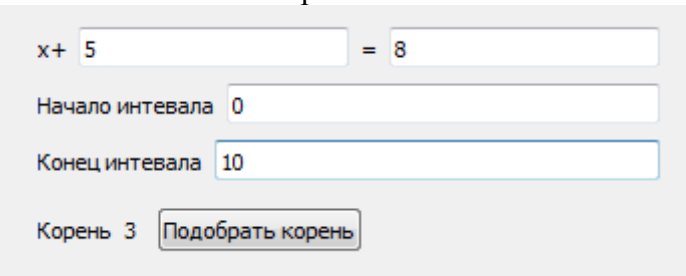

Рисунок 21 – Внешний интерфейс приложения

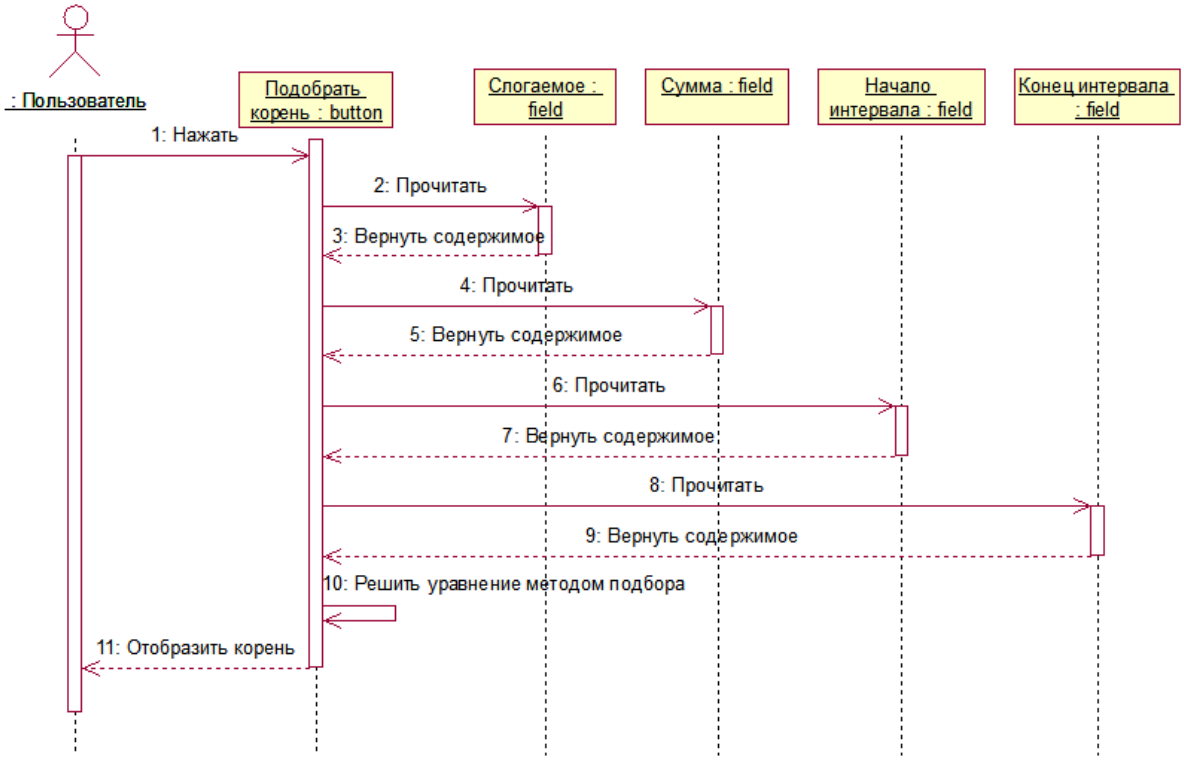

Рисунок 22 – Диаграмма последовательности для приложения

Вариант 12

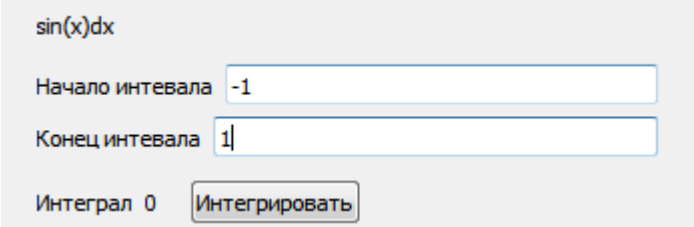

Рисунок 23 – Внешний интерфейс приложения

Рисунок 24 – Диаграмма последовательности для приложения

Вариант 13

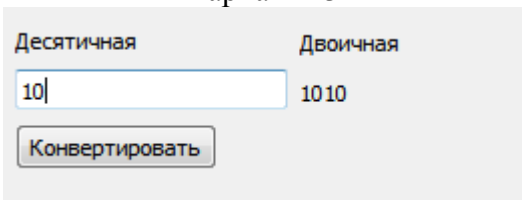

Рисунок 25 – Внешний интерфейс приложения

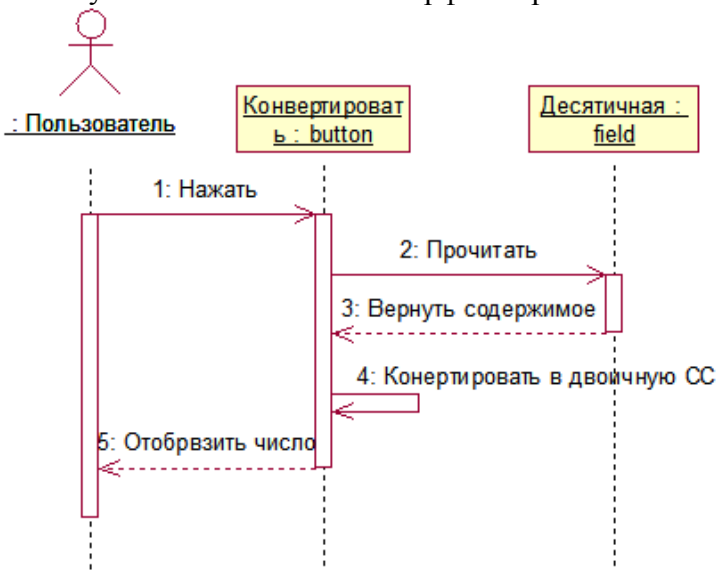

Рисунок 26 – Диаграмма последовательности для приложения

Вариант 14

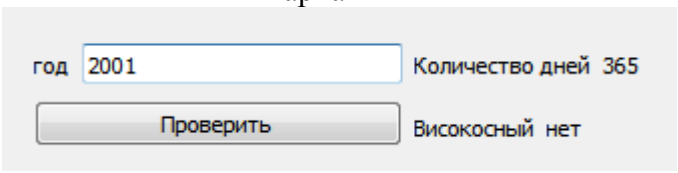

Рисунок 27 – Внешний интерфейс приложения

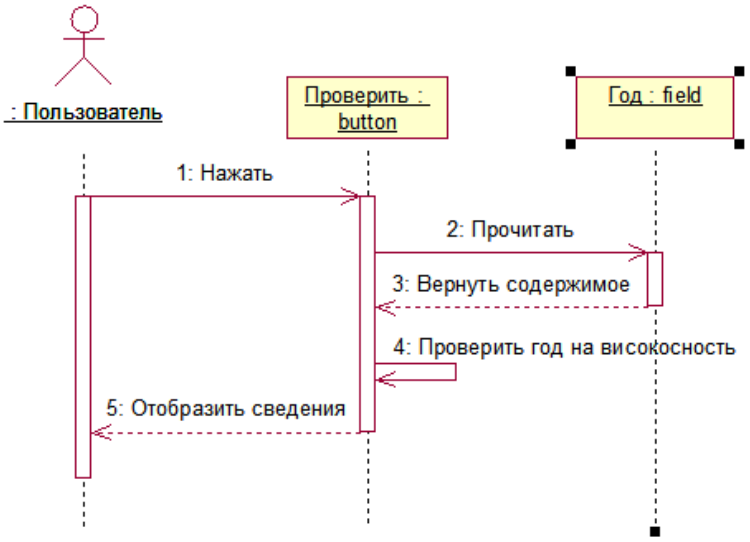

Рисунок 28 – Диаграмма последовательности для приложения

Вариант 15

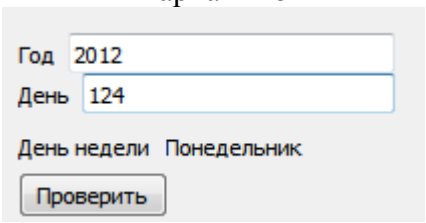

Рисунок 29 – Внешний интерфейс приложения

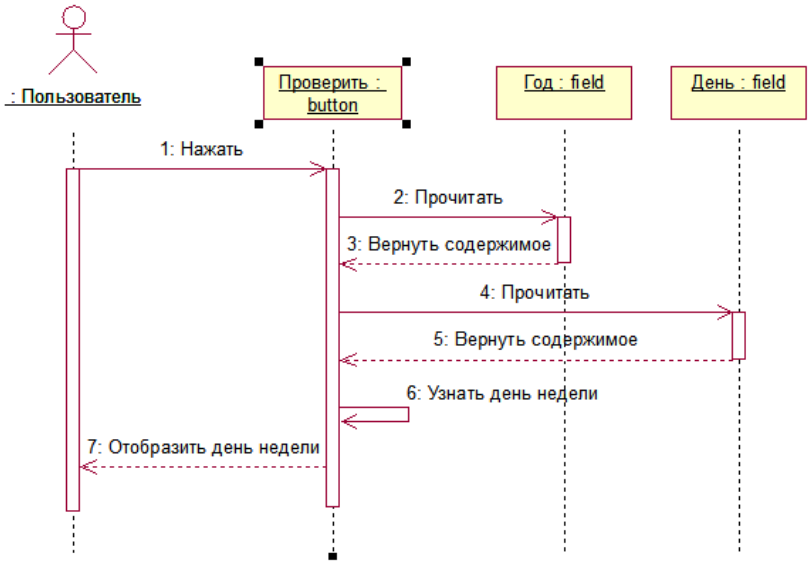

Рисунок 30 – Диаграмма последовательности для приложения

Вариант 16

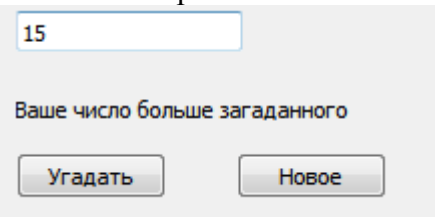

Рисунок 31 – Внешний интерфейс приложения

Рисунок 32 – Диаграмма последовательности для приложения

Вариант 17

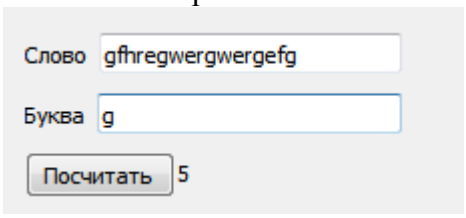

Рисунок 33 – Внешний интерфейс приложения

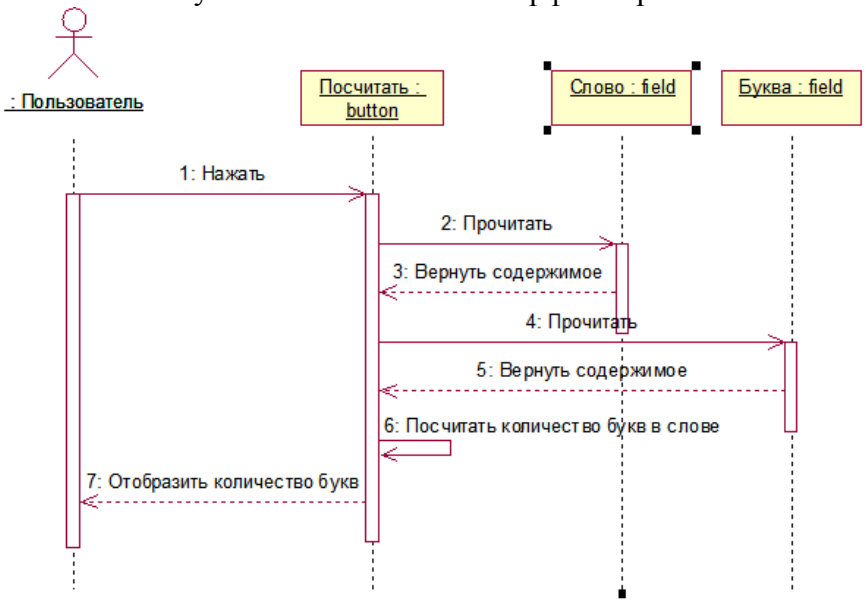

Рисунок 34 – Диаграмма последовательности для приложения

Вариант 18

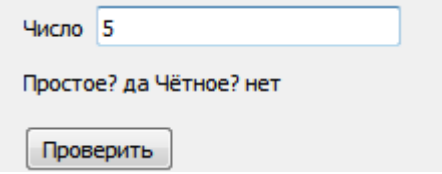

Рисунок 35 – Внешний интерфейс приложения

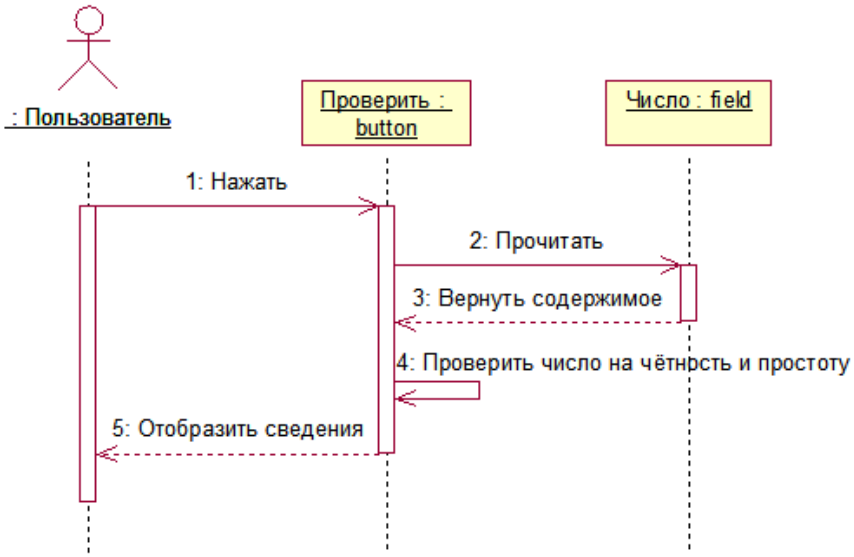

Рисунок 36 – Диаграмма последовательности для приложения

Вариант 19

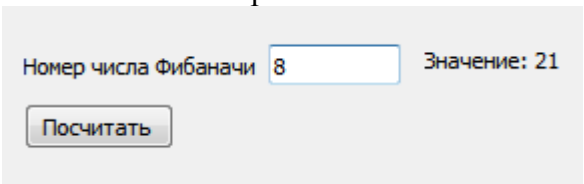

Рисунок 37 – Внешний интерфейс приложения

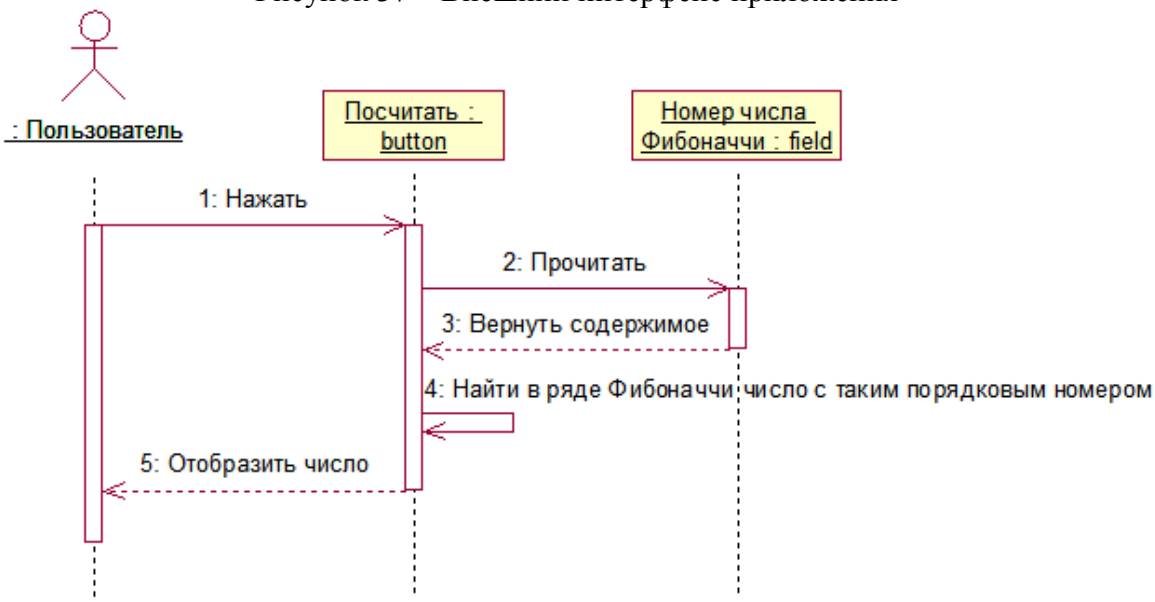

Рисунок 38 – Диаграмма последовательности для приложения

Вариант 20

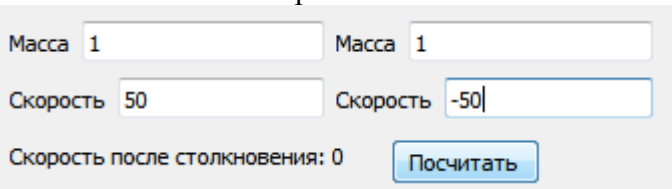

Рисунок 39 – Внешний интерфейс приложения

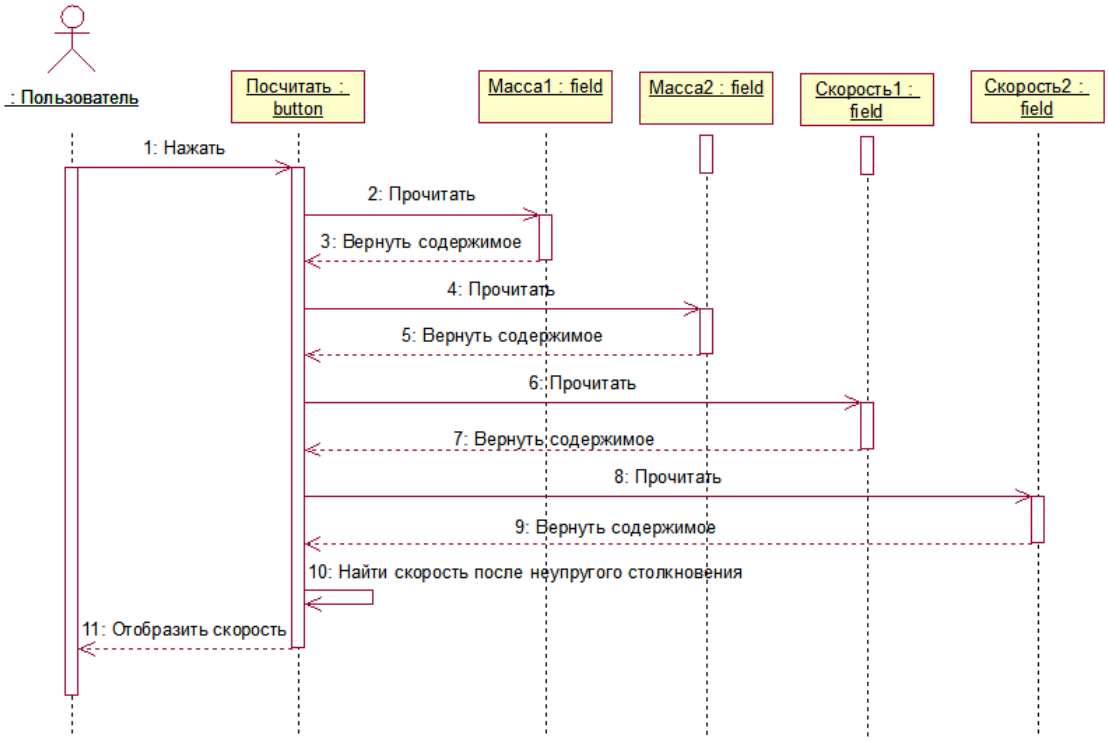

Рисунок 40 – Диаграмма последовательности для приложения

Вариант 21

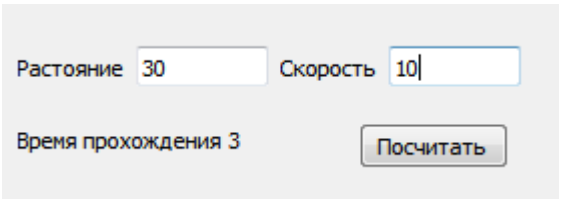

Рисунок 41 – Внешний интерфейс приложения

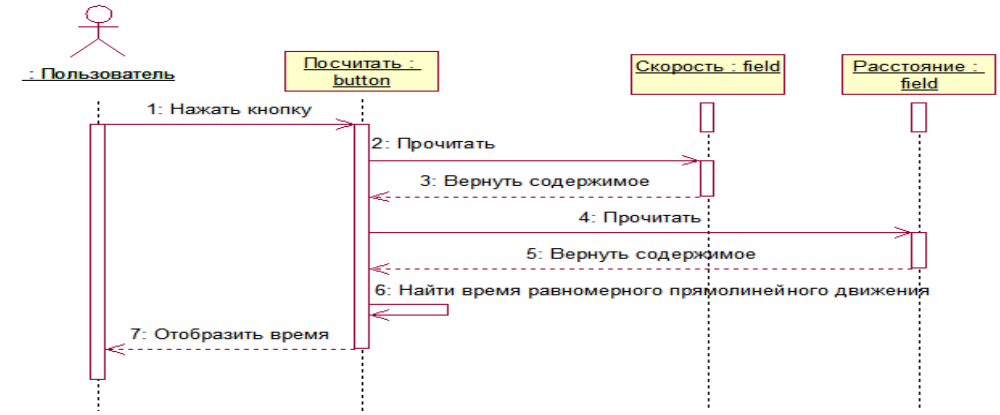

Рисунок 42 – Диаграмма последовательности для приложения

Вариант 22

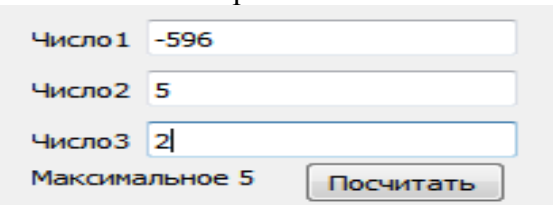

Рисунок 43 – Внешний интерфейс приложения

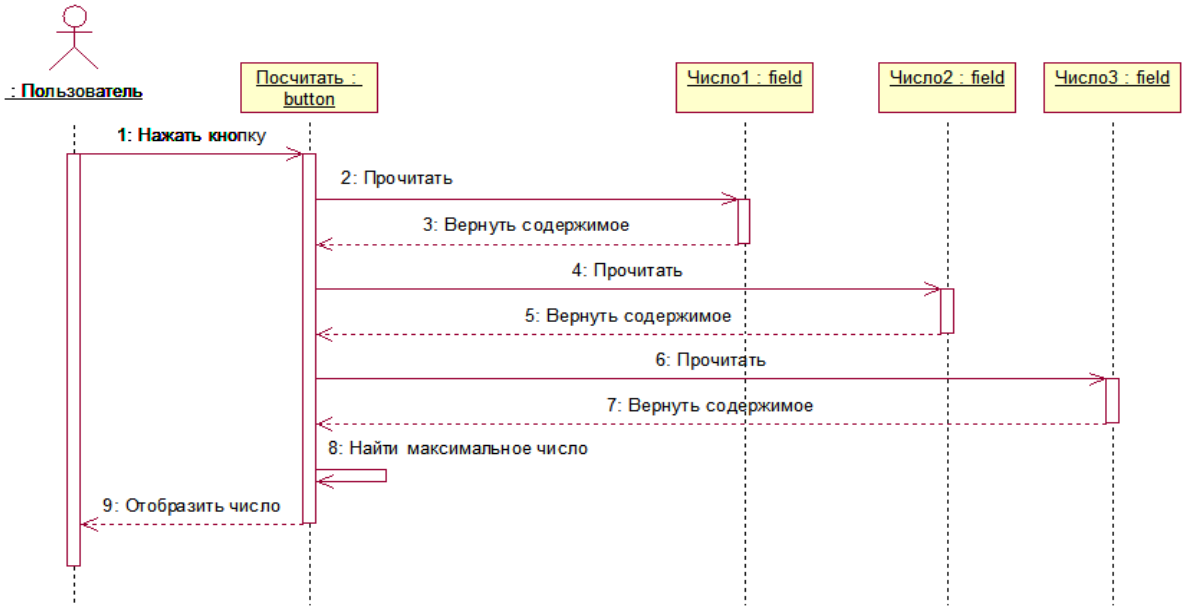

Рисунок 44 – Диаграмма последовательности для приложения

Вариант 23

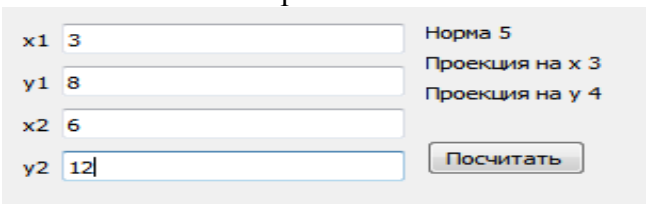

Рисунок 45 – Внешний интерфейс приложения

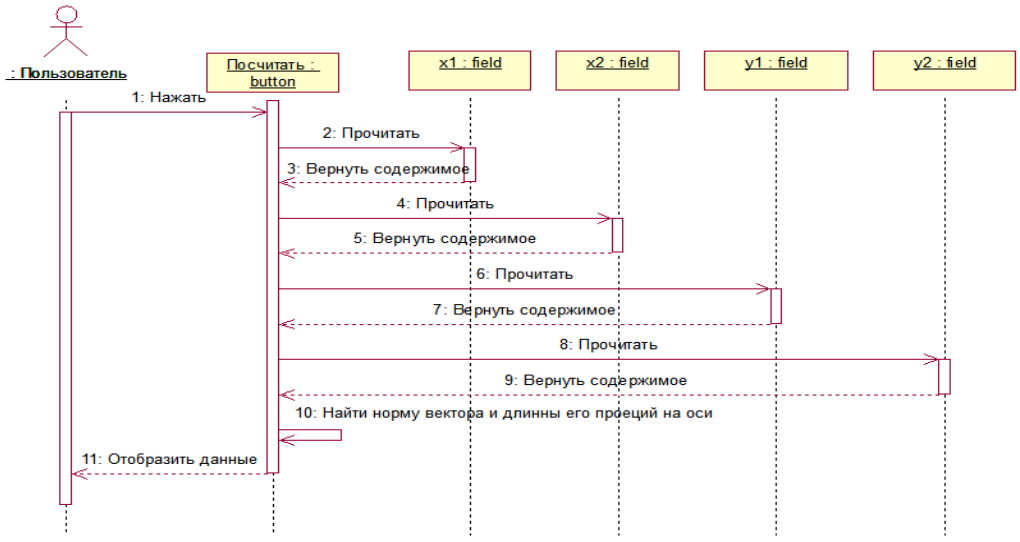

Рисунок 46 – Диаграмма последовательности для приложения

Вариант 24

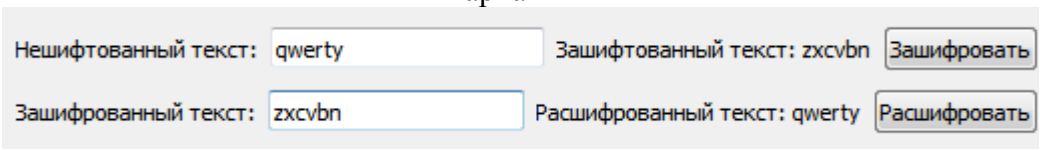

Рисунок 47 – Внешний интерфейс приложения

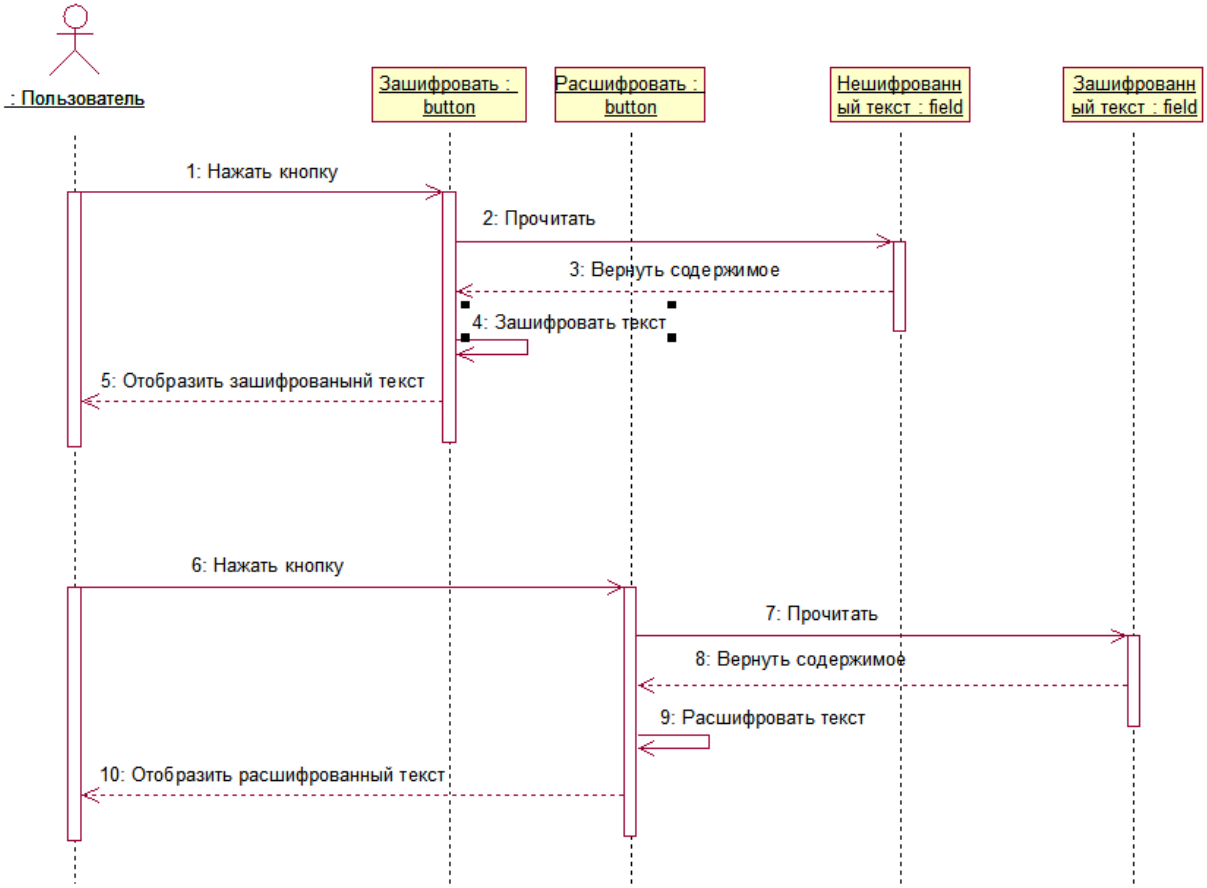

Рисунок 48 – Диаграмма последовательности для приложения

Вариант 25

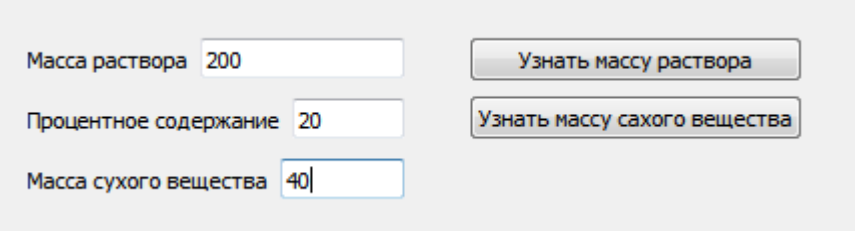

Рисунок 49 – Внешний интерфейс приложения

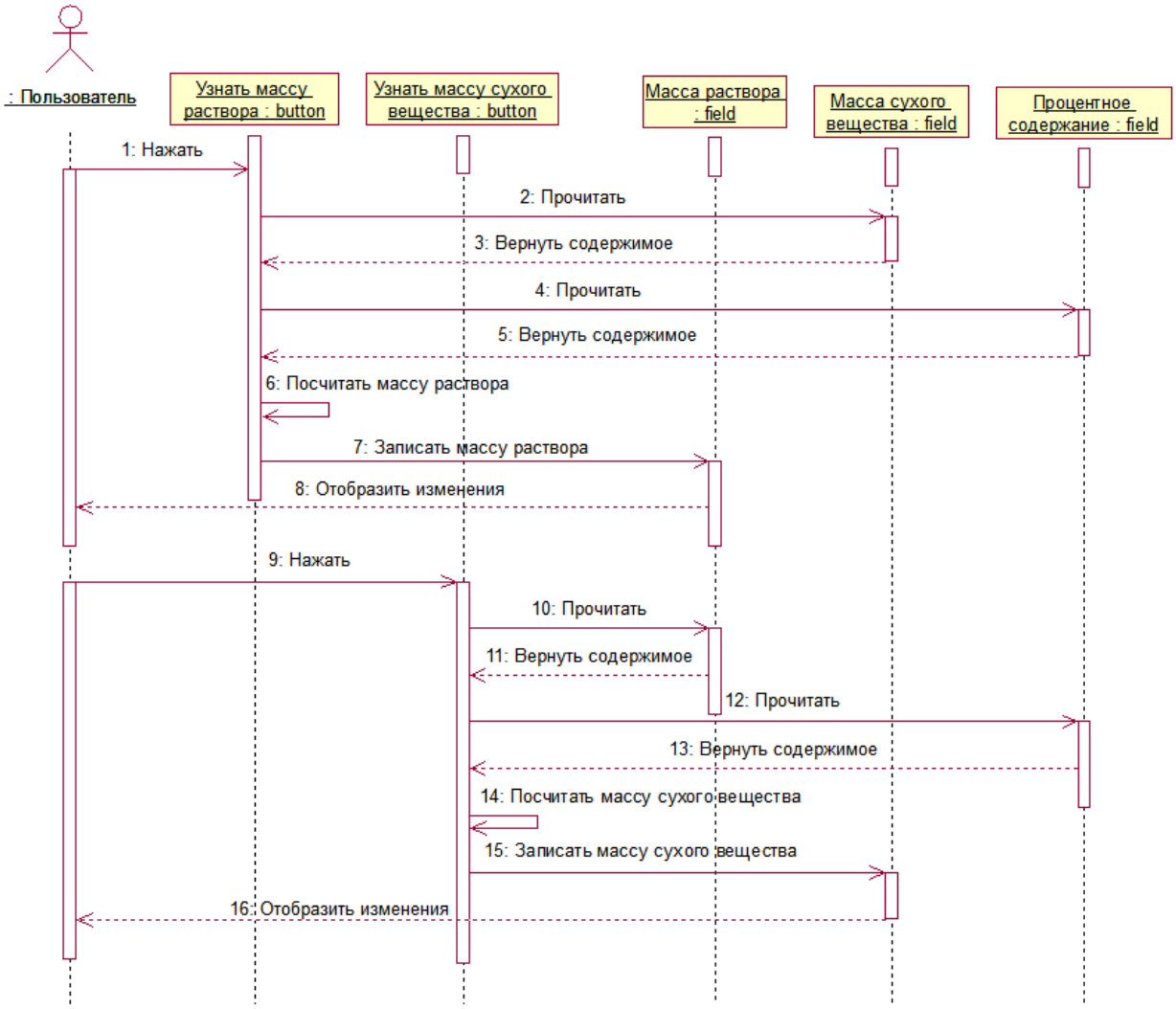

Рисунок 50 – Диаграмма последовательности для приложения

Вариант 26

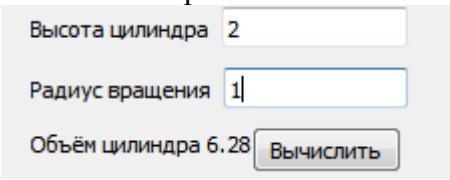

Рисунок 51 – Внешний интерфейс приложения

Рисунок 52 – Диаграмма последовательности для приложения

Вариант 27

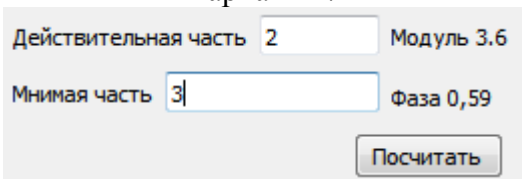

Рисунок 53 – Внешний интерфейс приложения

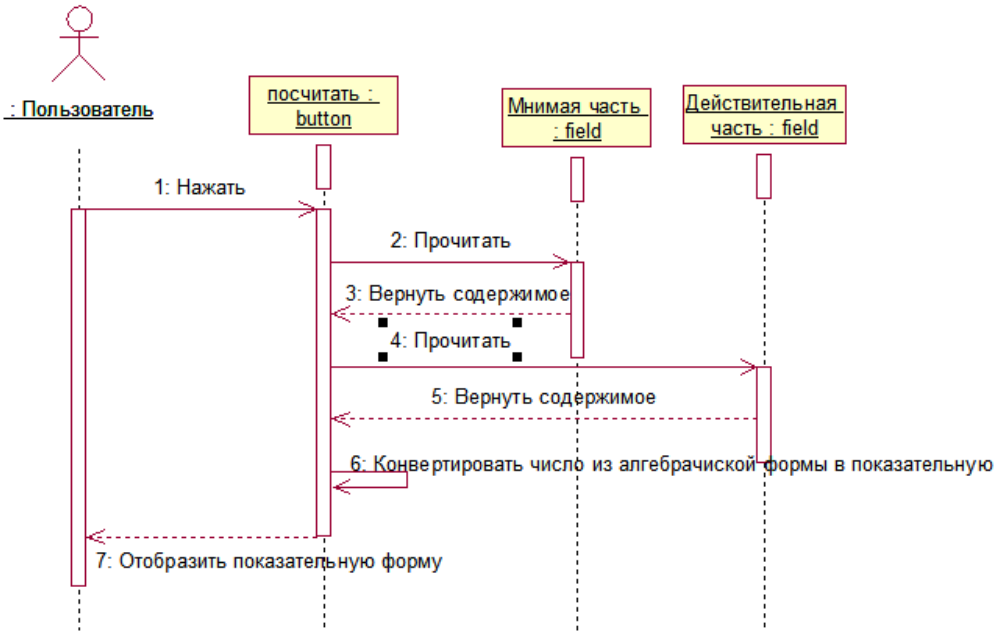

Рисунок 54 – Диаграмма последовательности для приложения

Вариант 28

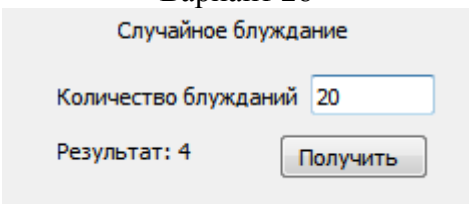

Рисунок 55 – Внешний интерфейс приложения

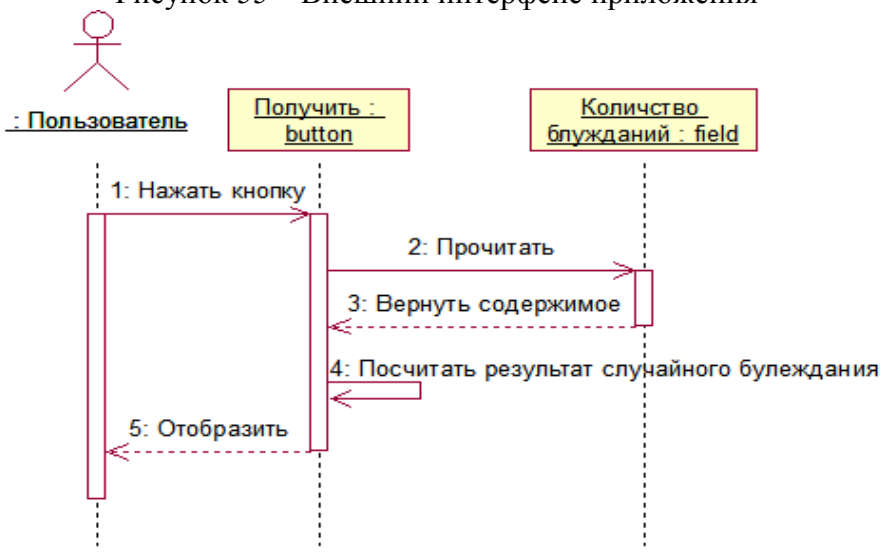

Рисунок 56 – Диаграмма последовательности для приложения

Вариант 29

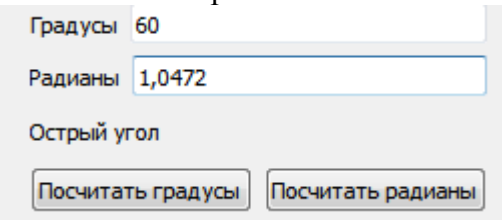

Рисунок 57 – Внешний интерфейс приложения

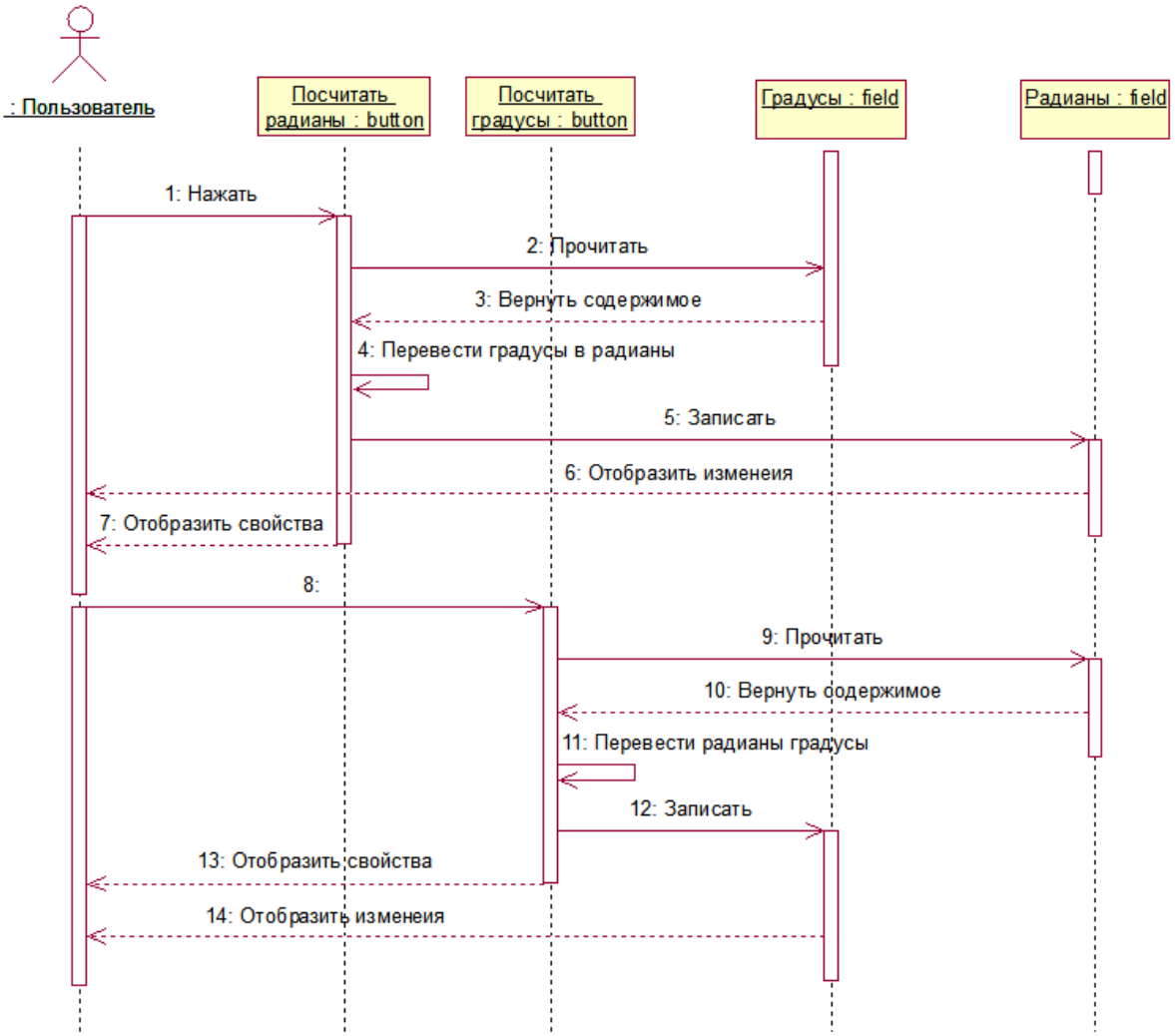

Рисунок 58 – Диаграмма последовательности для приложения

Вариант 30

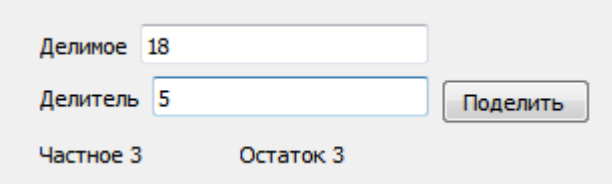

Рисунок 59 – Внешний интерфейс приложения

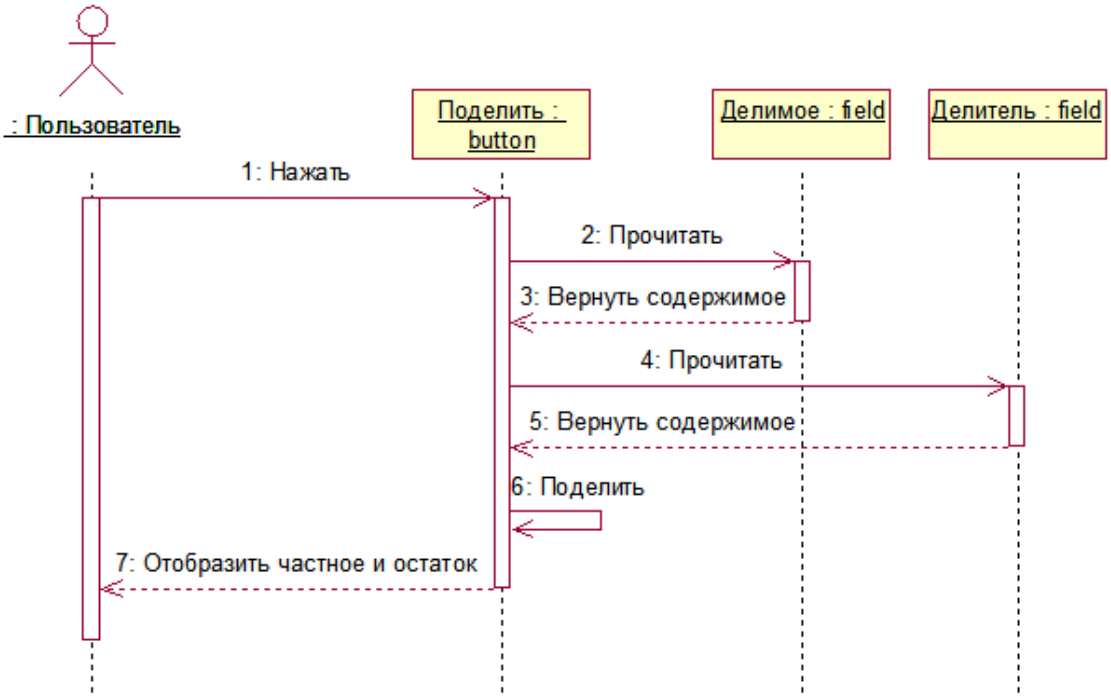

Рисунок 60 – Диаграмма последовательности для приложения

Лабораторная работа №2

Программирование алгоритмов с использованием механизмов объектно-ориентированного программирования

Задание к лабораторной работе:

– необходимо разработать графический интерфейс с использованием менеджера компоновки;

– представление графического интерфейса разрабатывается с учётом предоставления всей функциональности предложенного варианта;

– объекты и их взаимоотношения, имеющиеся в варианте задания, должны быть реализованы;

– функциональная часть приложения, представленная диаграммой последовательности, должна быть реализована;

– графический интерфейс должен быть создан посредством одной из следующих библиотек: Swing, SWT, JavaFX.

Вариант 1

Реализовать классы Университет, Студент и Преподаватель, Лекция. Университет может зачислить Студента, составить план Лекций и нанять Преподавателя, создав экземпляр соответствующего класса. Студент может посетить Лекцию, узнав её тему, Преподаватель может прочитать лекцию. Староста, потомок класса Студент, имеет возможность отметить присутствующих на лекции, занеся их в список. Преподаватель также может отметить присутствующих на лекции, занеся их в список.

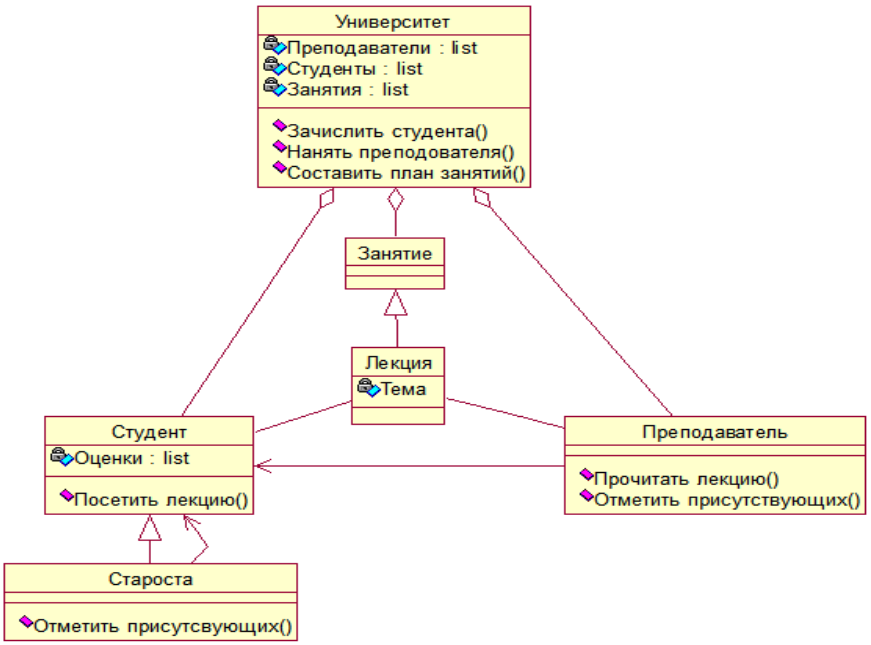

Рисунок 61 – Диаграмма классов

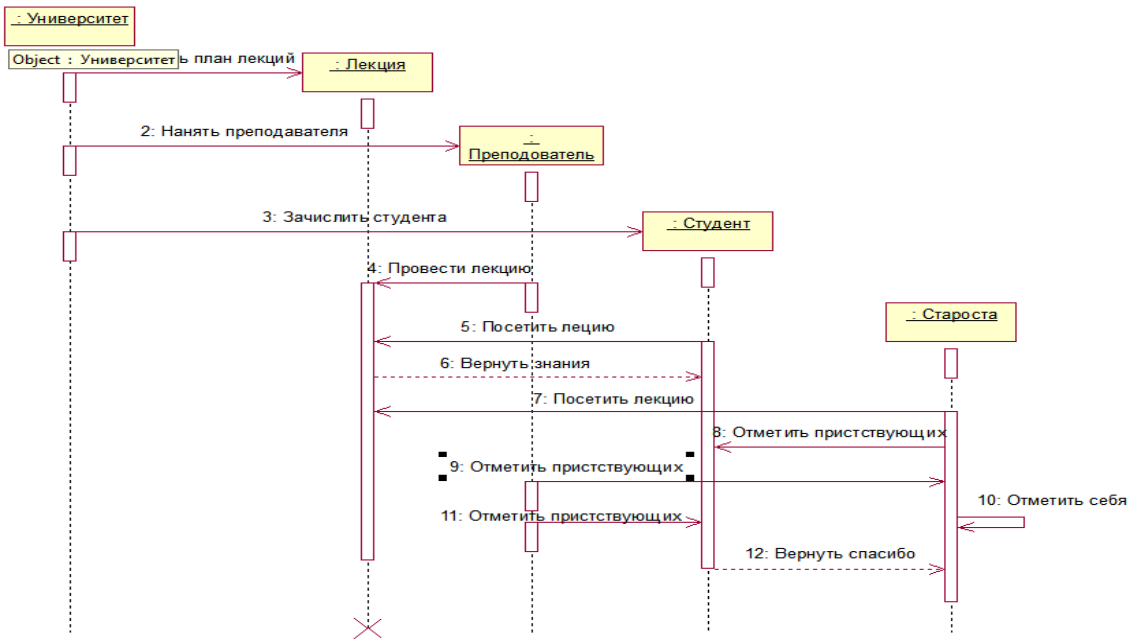

Рисунок 62 – Диаграмма последовательности для приложения

Вариант 2

Реализовать классы Выражение, Число и Комплексное число в алгебраическом виде. Класс выражение может вычислить результат сложения или вычитания двух комплексных чисел, породив новый экземпляр класса Комплексное число в алгебраическом виде. Тип операции зашифрован и определяется при создании класса Выражение.

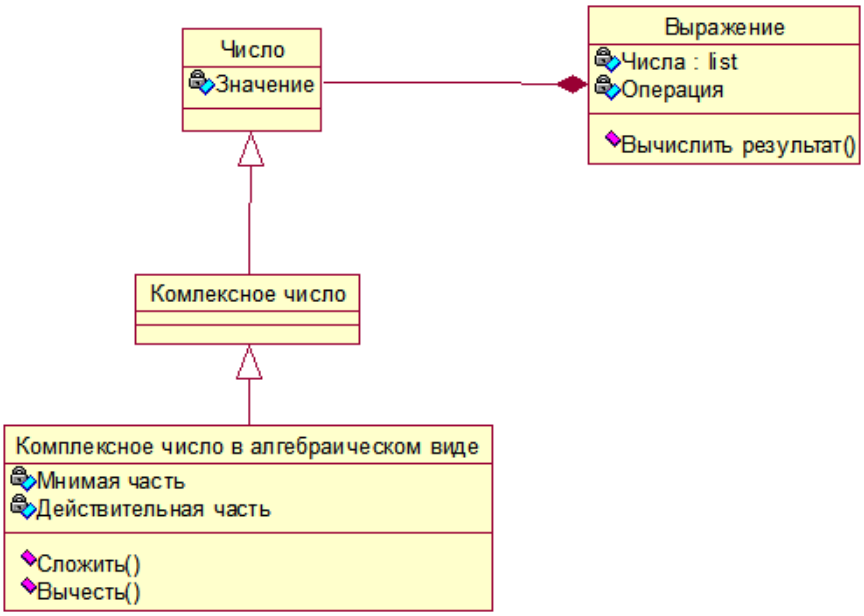

Рисунок 63 – Диаграмма классов

Рисунок 64 – Диаграмма последовательности для приложения

Вариант 3

Реализовать классы Завод, Рабочий, Спичка, Автомобиль. Класс Завод может нанять рабочего, создав его экземпляр. Завод может выпускать продукт. При этом он проходит по коллекции рабочих и вызывает у каждого метод работать(). Результатом исполнения метода является создание экземпляра класса Автомобиль. После создания экземпляра рабочий вызывает метод курить(), в ходе которого он вызывает метод зажечься() у класса Спичка.

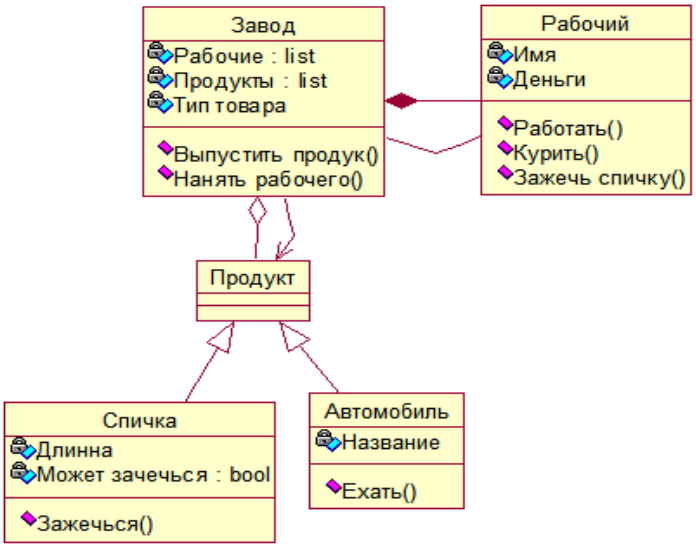

Рисунок 65 – Диаграмма классов

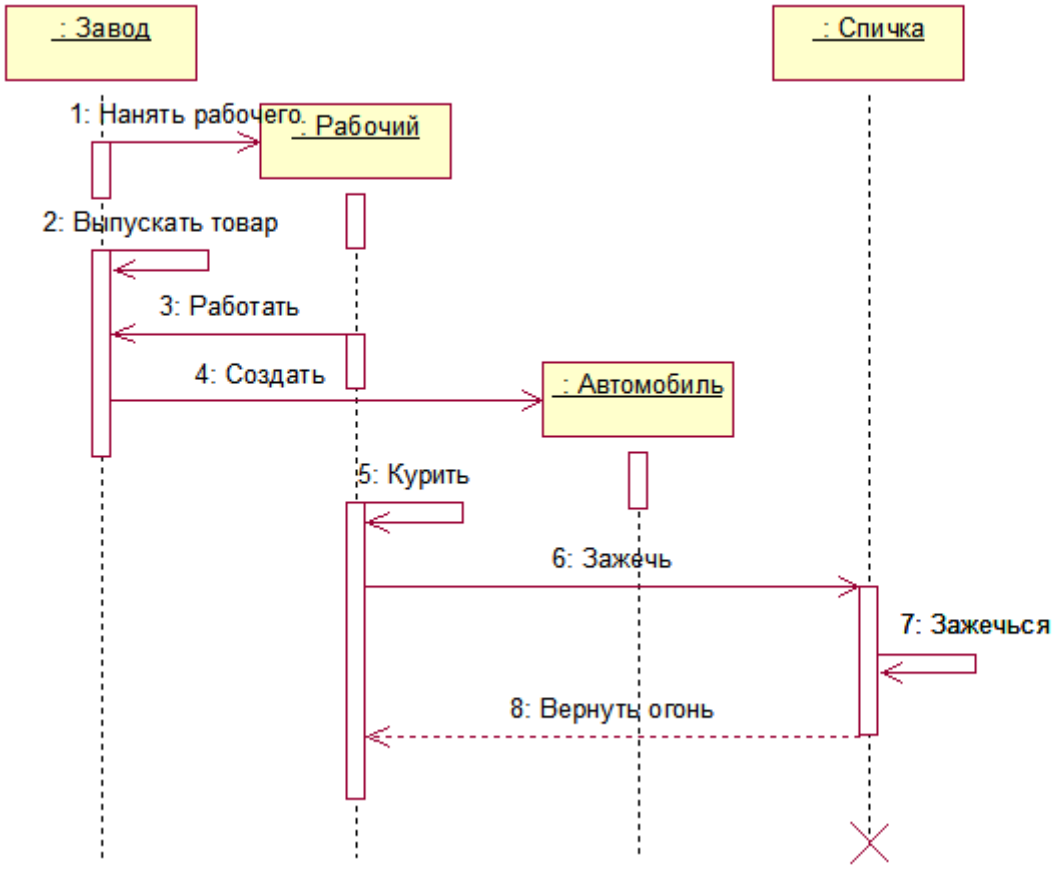

Рисунок 66 – Диаграмма последовательности для приложения

Вариант 4

Реализовать класс голова, являющийся контейнером для классов, наследующих от класса Орган. Класс Мозг может принимать сигналы от других объектов и вызывать у них их методы. Классы Рот и Нос после выполнения метода нюхать() и есть() возвращают строку с результатом.

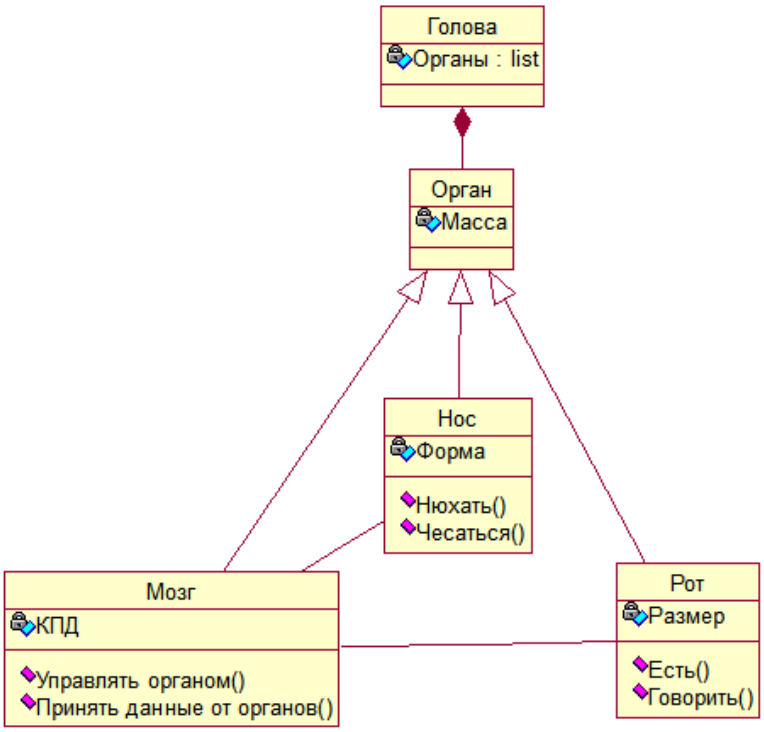

Рисунок 67 – Диаграмма классов

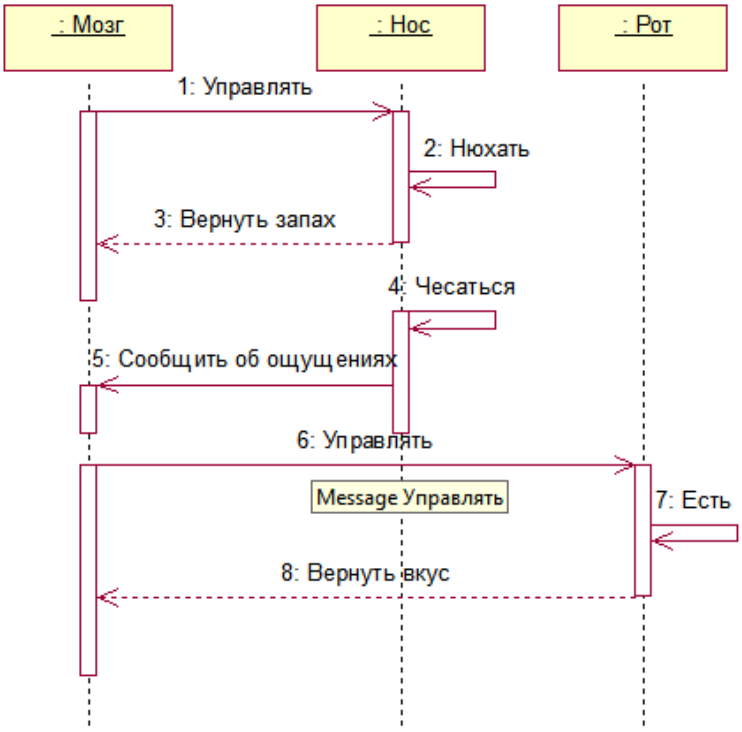

Рисунок 68 – Диаграмма последовательности для приложения

Вариант 5

Реализовать классы Пользователь, Компьютер, Браузер и Плеер. Пользователь может использовать класс Браузер для того, чтобы скачать программу. После этого он может использовать класс Компьютер, чтобы установить программу. Также он может использовать класс компьютер, чтобы удалить программу. При установке или удалении программы объект наследующий от Программа добавляется в коллекцию, хранящийся в объекте Компьютер. Класс плеер обладает методом Воспроизвести фильм, который выводит на экран название фильма.

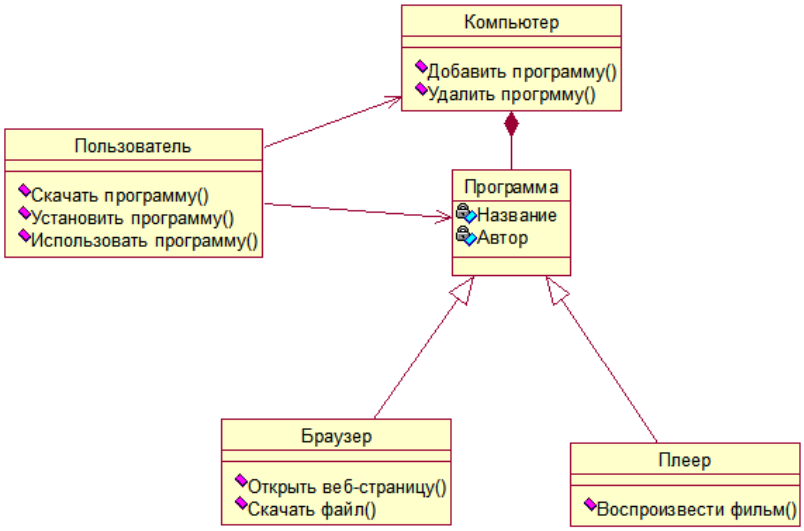

Рисунок 69 – Диаграмма классов

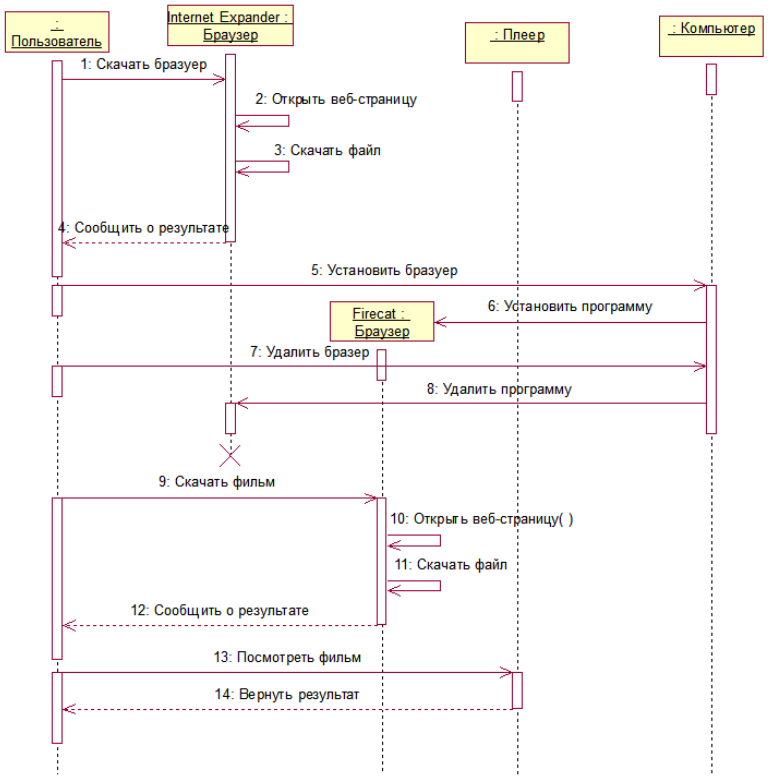

Рисунок 70 – Диаграмма последовательности для приложения

Вариант 6

Реализовать классы Повар, Программист, Еда. Повар может создавать экземпляры класса Еда. Программист может писать код, выводя на экран случайные символы. Программист и Повар могут съесть Еду, прочитав её поле вкус и уничтожив объект.

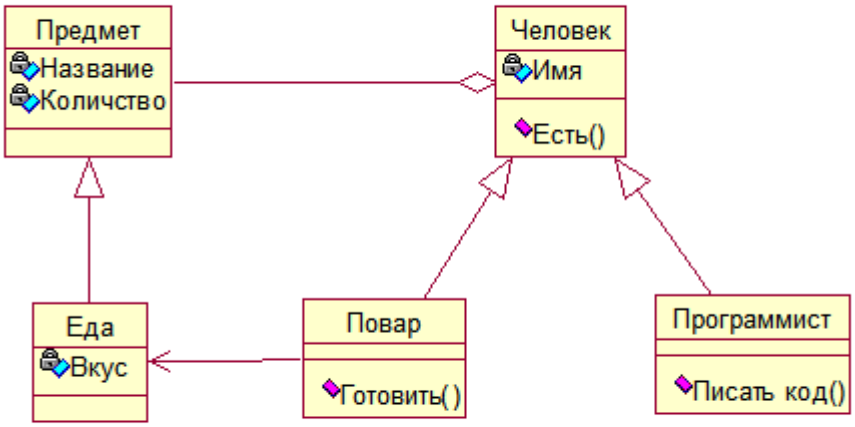

Рисунок 71 – Диаграмма классов

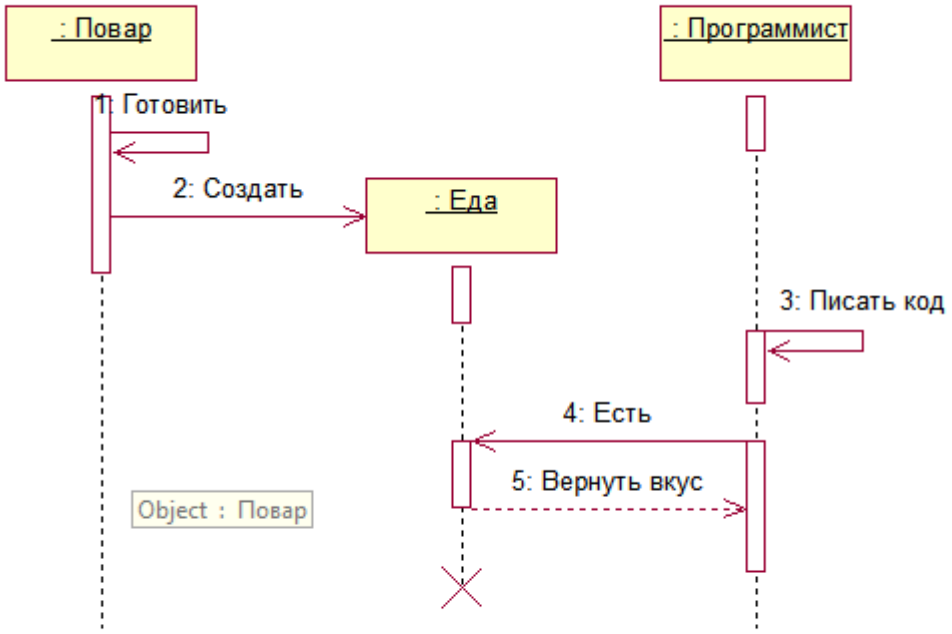

Рисунок 72 – Диаграмма последовательности для приложения

Вариант 7

Реализовать класс автомобиль, являющийся контейнером для классов, наследующих от класса Деталь. При вызове у Автомобиля метода ехать() он вызывает у объекта Двигатель метод работать(). Если метод завестись() не был ранее вызван, то Двигатель его у себя вызывает. Далее двигатель вызывает методы подать топливо() и вращаться() у объектов Топливный бак и Колесо.

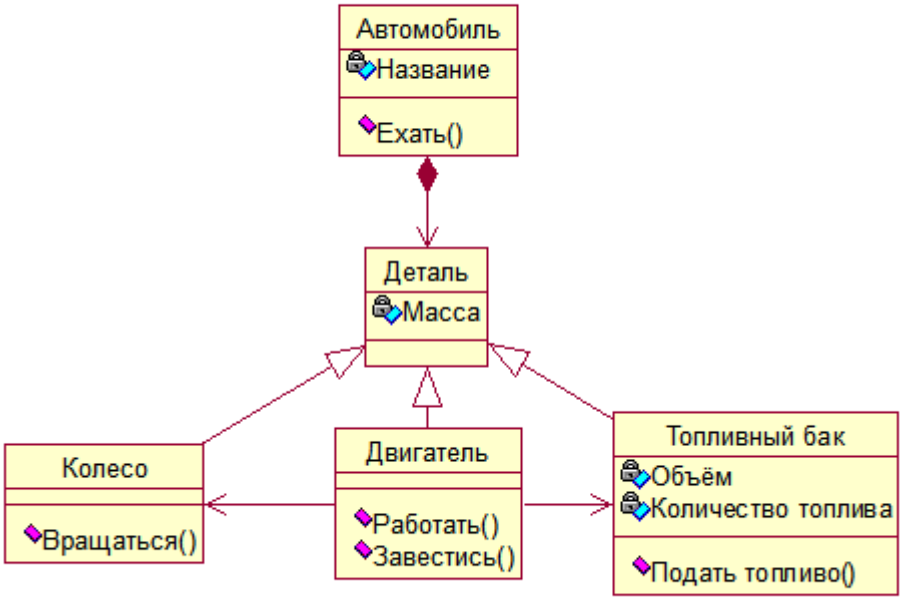

Рисунок 73 – Диаграмма классов

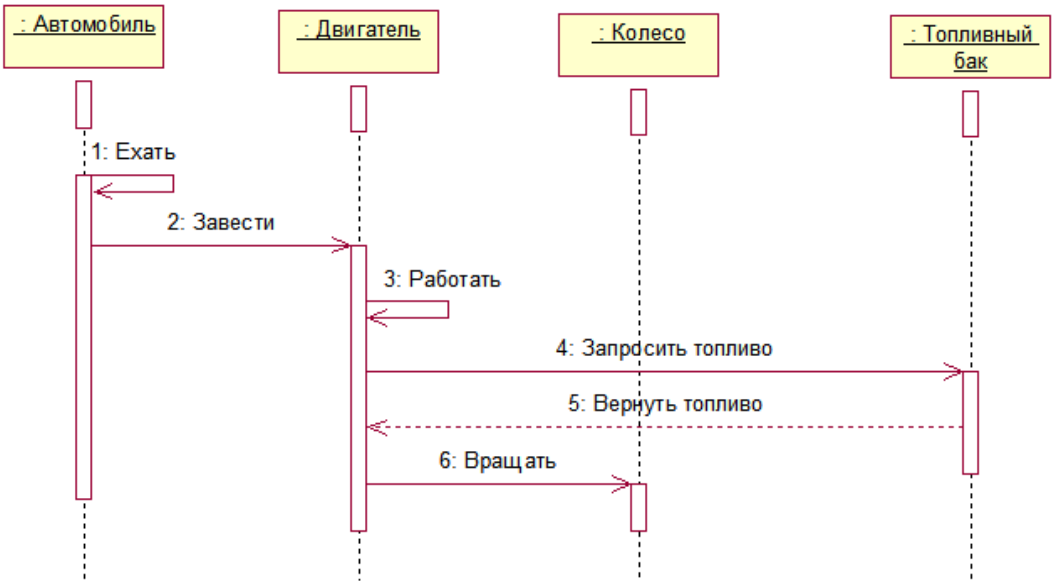

Рисунок 74 – Диаграмма последовательности для приложения

Вариант 8

Реализовать классы Юридическое лицо, Физическое лицо, наследующие от класса Клиент, Компания и Заказ. Классы, наследующие от класса Клиент, обладают методом заказать(), который вызывает метод Получить заказ класса Компания. Она создаёт экземпляр класса Заказ, исполняет его, меняя его состояние, и возвращает экземпляр Клиенту. Клиент после этого вызывает у себя метод заплатить()

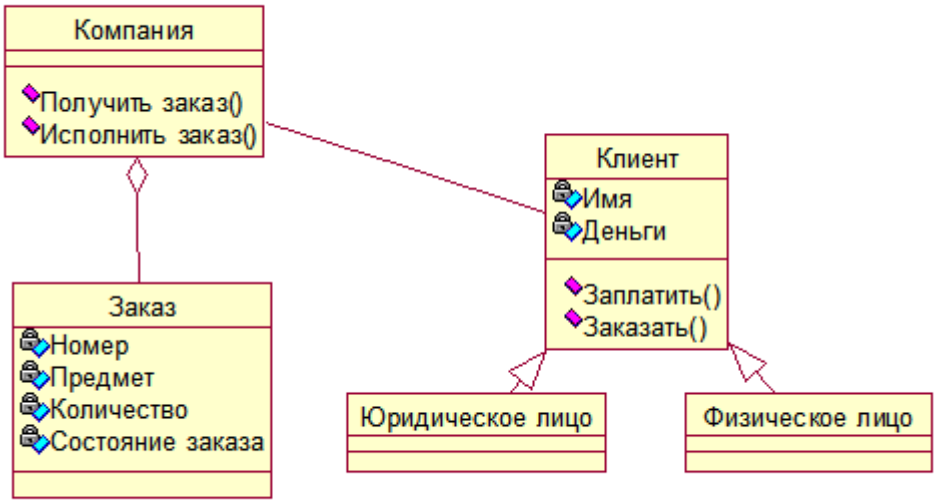

Рисунок 75 – Диаграмма классов

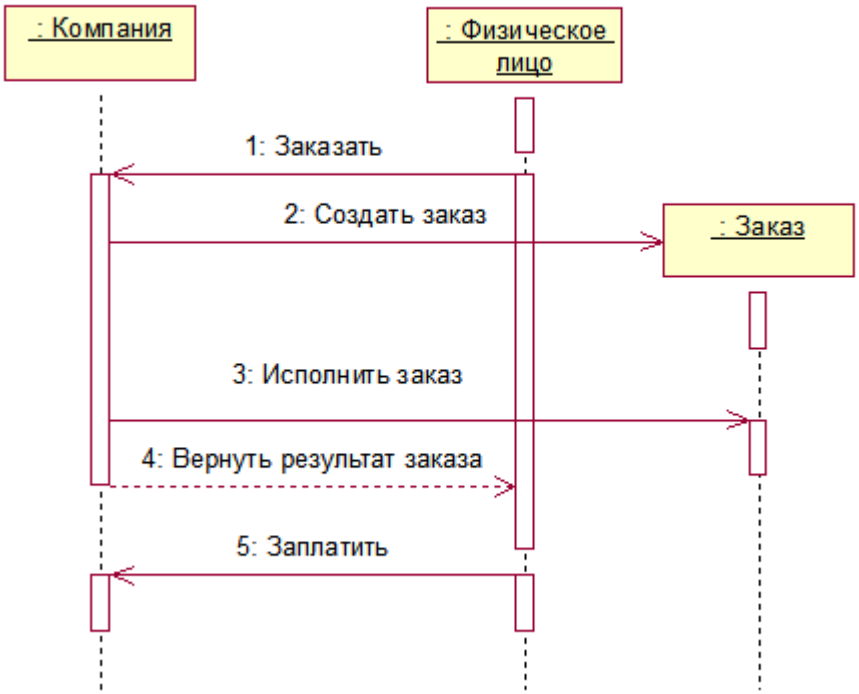

Рисунок 76 – Диаграмма последовательности для приложения

Вариант 9

Реализовать класс Компания, и классы Директор, Программист, Уборщик, наследующие от класса Сотрудник. Компания может нанимать Сотрудников, создавая объекты, наследующие от этого класса. Директор может менять поле задачи класса Компания, которые потом будет исполнять программист, получая значение поля задача как входной параметр метода Писать код(), который в свою очередь будет возвращать готовую задачу. Класс уборщик обладает методом убирать() который, будет выводить на экран текст.

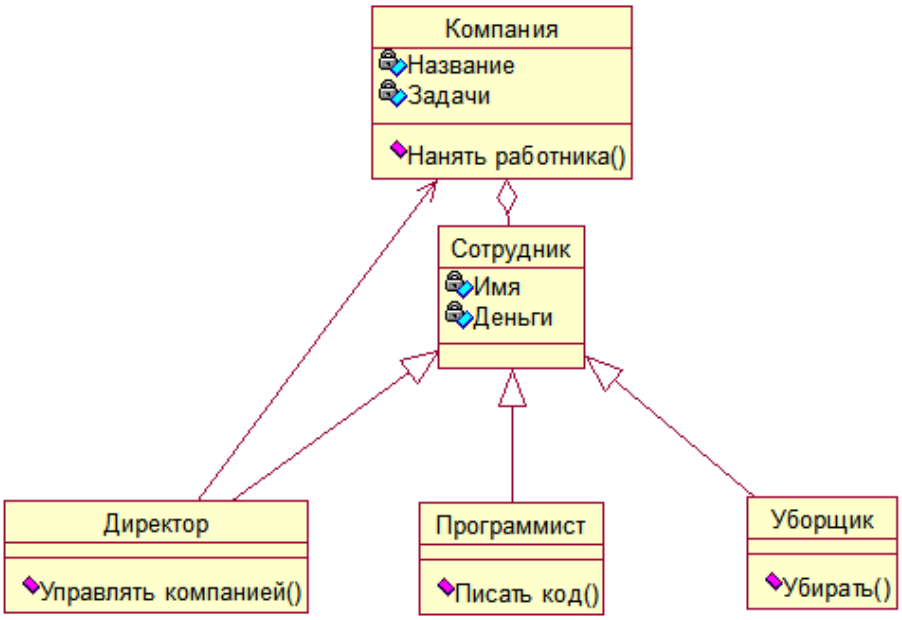

Рисунок 77 – Диаграмма классов

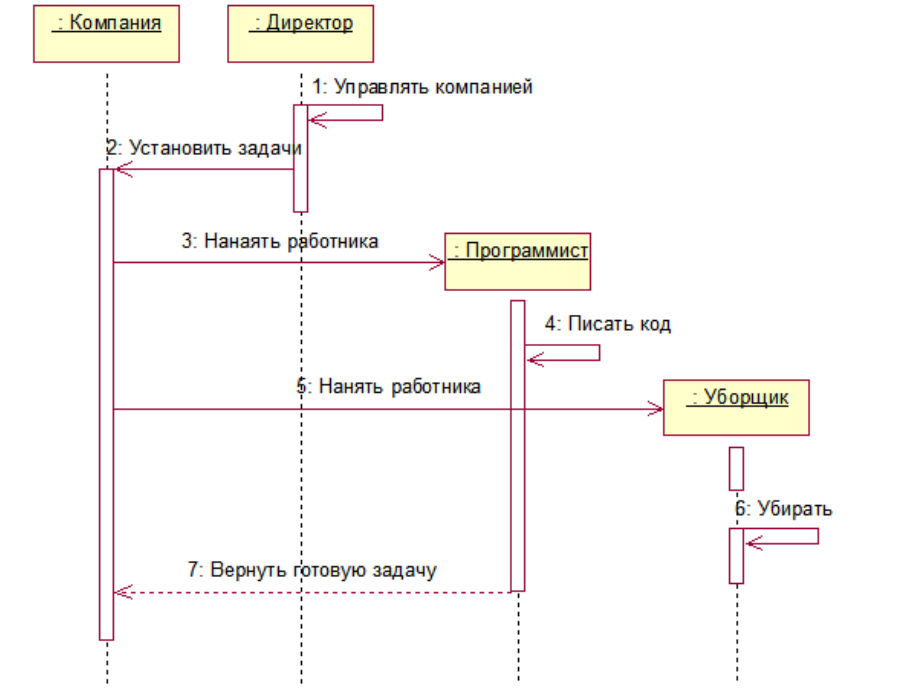

Рисунок 78 – Диаграмма последовательности для приложения

Вариант 10

Реализовать классы Университет, Преподаватель, Инженер, Студент, Зачётка. Университет может зачислить студента(), создав объект этого класса, и выдать зачётку() сообщив классу Студент ссылку на объект Зачётка. Студент может внести в Зачётку свои данные. Также Студент может сдать экзамен(), вызвав у преподавателя метод принять экзамен(), после чего преподаватель изменяет поле Оценки класса Зачётка.

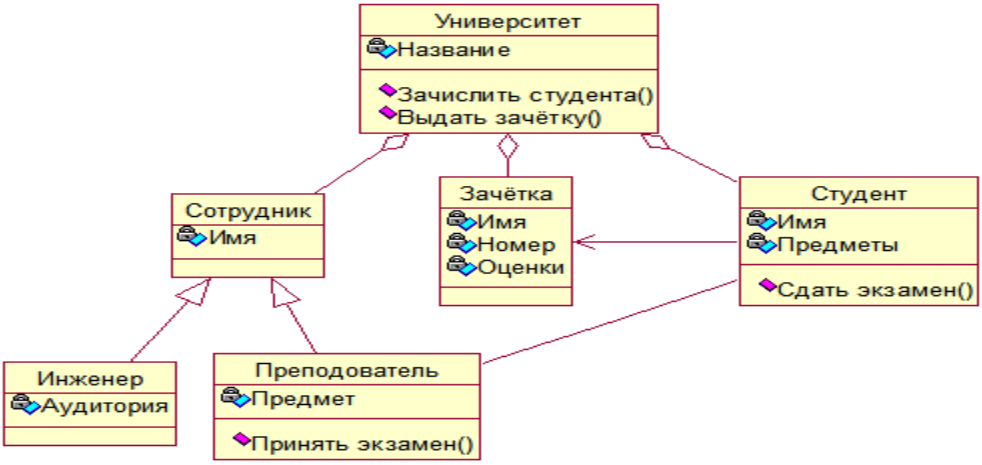

Рисунок 79 – Диаграмма классов

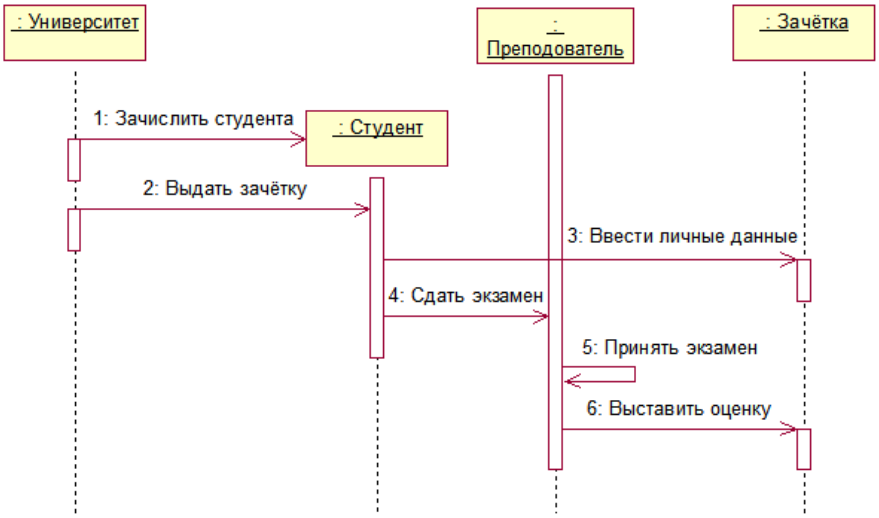

Рисунок 80 – Диаграмма последовательности для приложения

Вариант 11

Реализовать классы Рабочий, Отвёртка, Пила, Рубанок, Заготовка. Метод Рабочего работать() вызывает методы Пилить(), стругать(), закрутить(), раскрутить() у классов Пила, Рубанок, Отвёртка, которые в свою очередь меняют поле Форма класса Заготовка. Методы закрутить(), раскрутить() форму не меняют, а выводят на экран текст “закручено” и “раскручено”.

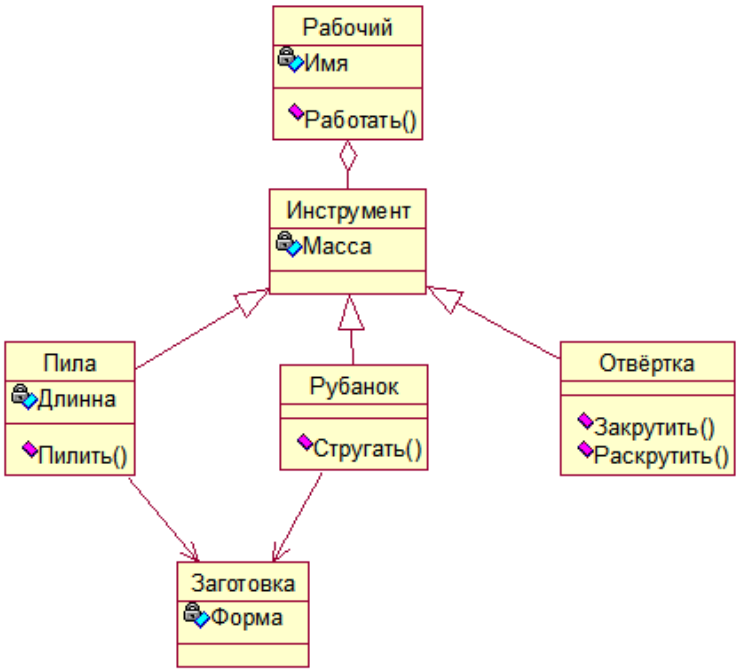

Рисунок 81 – Диаграмма классов

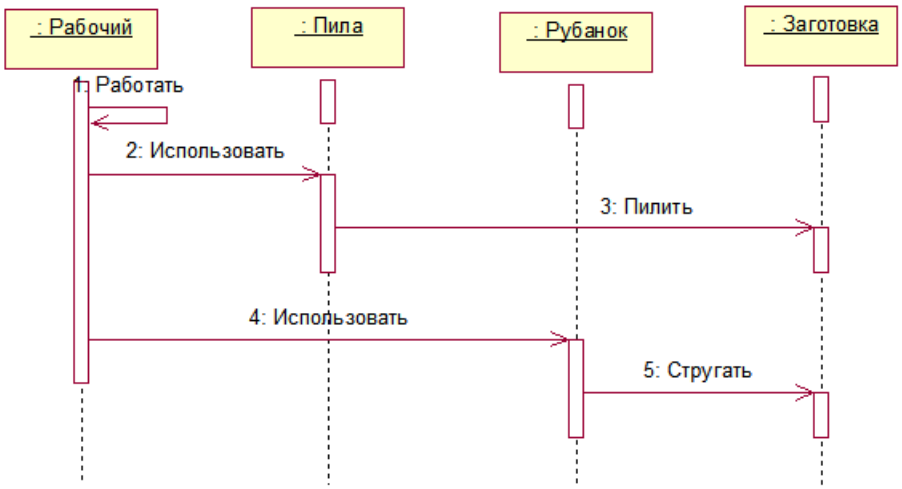

Рисунок 82 – Диаграмма последовательности для приложения

Вариант 12

Реализовать классы Студент, Чашка, Кофе, Сахар, Чайник. Которые реализуют процесс приготовления кофе. Метод сделать кофе() класса Студент Насыпать() класса Сахар, который вызывает Наполнить() класса Чашка. После этого вызывается метод Насыпать() класса Кофе, вызывающий такой же метод класса Чашка. После этого Вызывается метод Наполнить водой() класса Чайник, который в свою очередь вызывает метод Налить(), а он в свою очередь вызывает метод наполнить() класса Чашка.

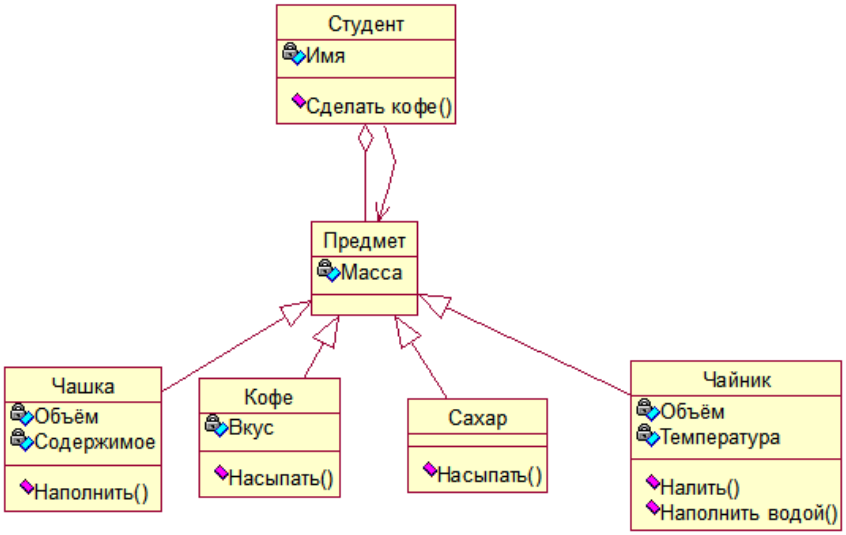

Рисунок 83 – Диаграмма классов

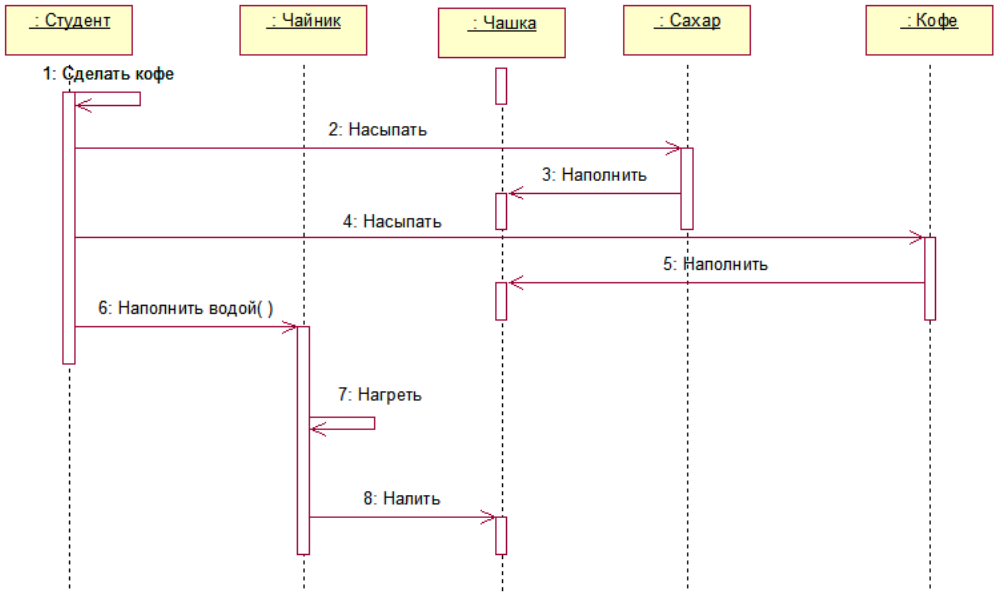

Рисунок 84 – Диаграмма последовательности для приложения

Вариант 13

Реализовать классы Хозяин, и Собачий корм, а также классы Пудель, Терьер и Овчарка, наследующие от класса Собака. Хозяин содержит у себя ссылки на объекты Собака и Собачий корм. Хозяин обладает методом Дать команду, который вызывает метод Исполнить команду класса Собака. Метод Исполнить команду() выводит на экран текст. Собака обладает методом Просить еды(), который вызывает метод Накормить класса Хозяин, возвращающий Собаке ссылку на объект Собачий Корм.

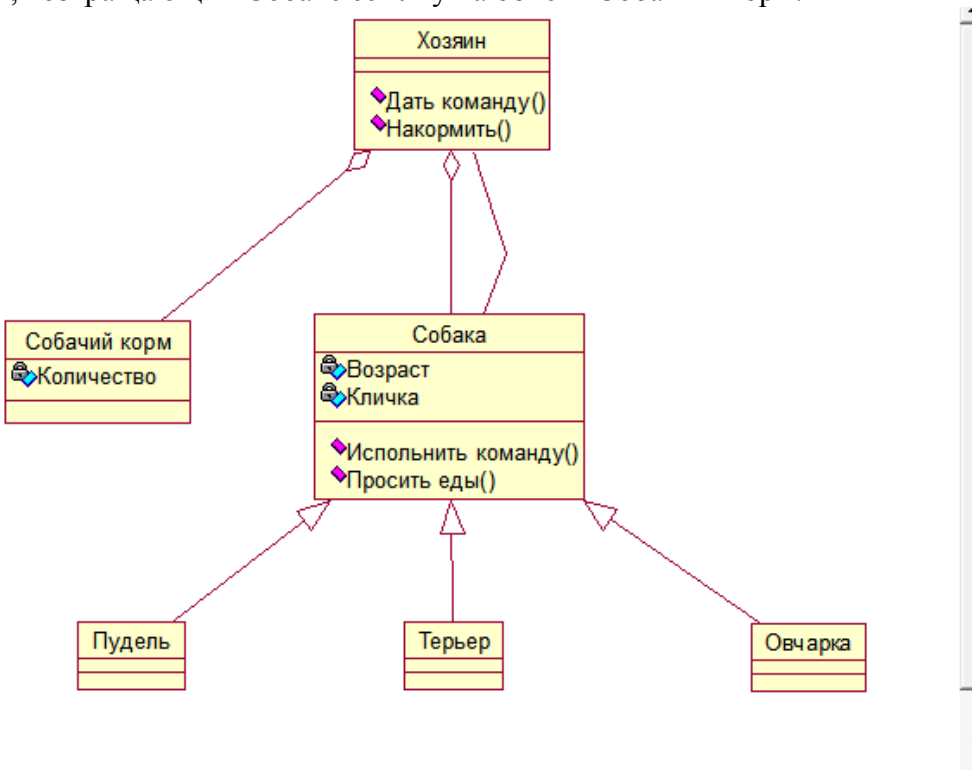

Рисунок 85 – Диаграмма классов

Рисунок 86 – Диаграмма последовательности для приложения

Вариант 14

Реализовать классы Пациент, Врач и Болезнь. Пациент обладает методом жаловаться(), который вызывает метод диагностировать() класса Врач. Этот метод читает поля класса Болезнь, содержащиеся в классе Пациент. А потом Вызывает у себя метод Лечить().

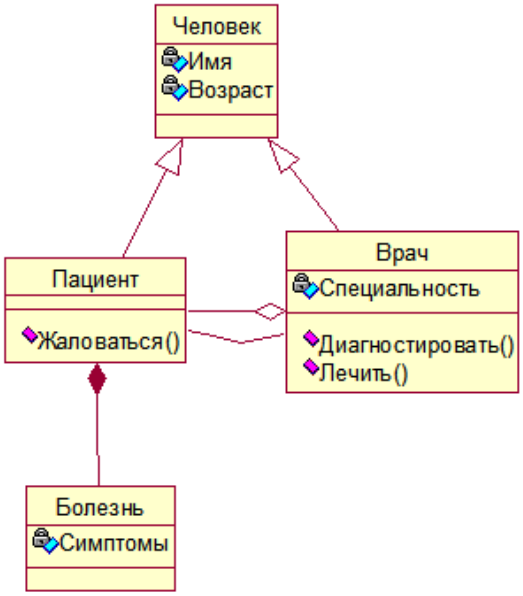

Рисунок 87 – Диаграмма классов

Рисунок 88 – Диаграмма последовательности для приложения

Вариант 15

Реализовать классы Человек, Фиалка, Роза, бутон. При вызове метода Полить() класса Человек, вызывается метод Создать бутон() у класса Роза. У каждого класса роза может быть не больше одного бутона. При вызове метода Понюхать() класса Человек, на экран выводится значение поля запах класса Бутон.

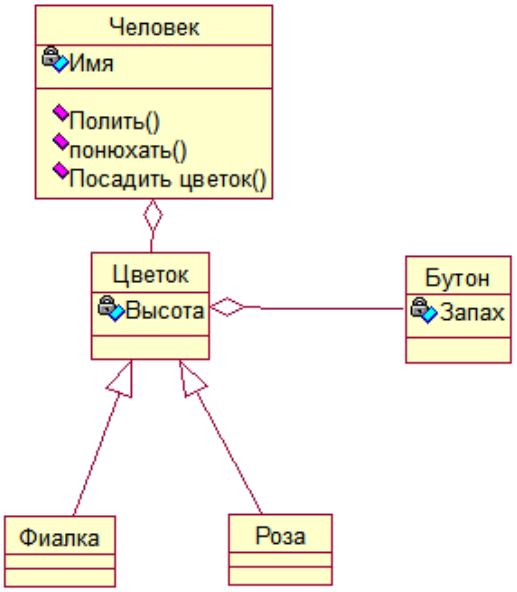

Рисунок 89 – Диаграмма классов

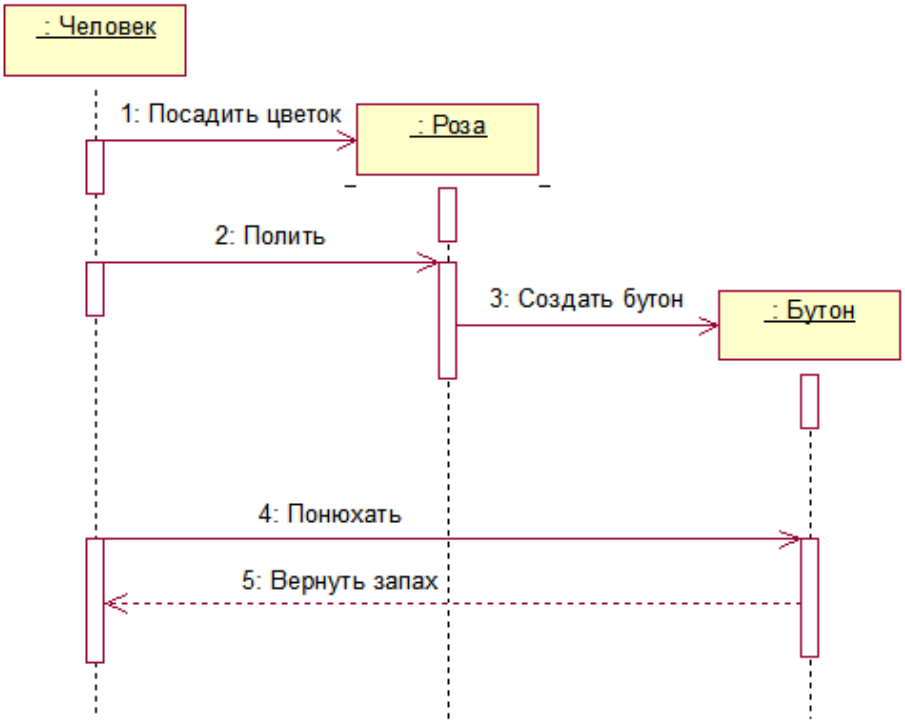

Рисунок 90 – Диаграмма последовательности для приложения

Вариант 16

Реализовать Классы Писатель, Читатель и Книга. Писатель может создавать экземпляры класса Книга. Читатель может Прочитать() класс книга, получив количество страниц и вызвав у себя метод написать отзыв(), который отправляет строчку с отзывом классу Писатель().

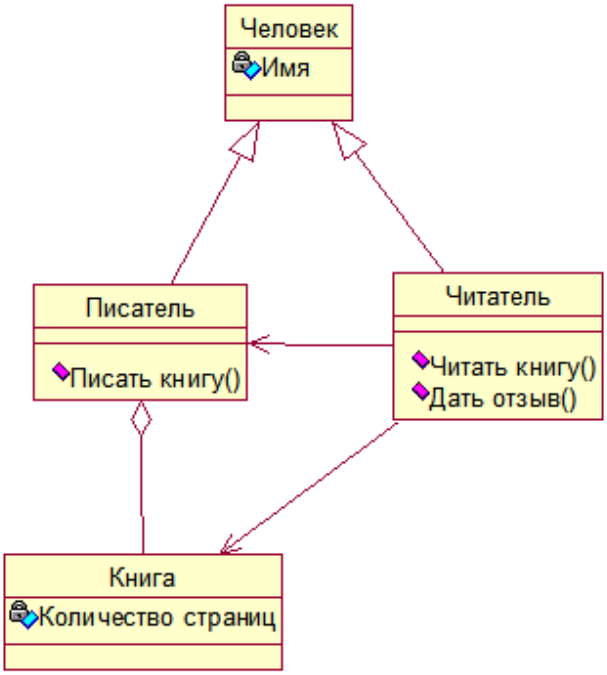

Рисунок 91 – Диаграмма классов

Рисунок 92 – Диаграмма последовательности для приложения

Вариант 17

Реализовать Классы Самолёт, Пассажир, Пилот, Стюардесса. Метод пассажира Зайти на самолёт() получает у класса самолёт номер свободного места и запоминает его. Метод Управлять самолётом() класса Пилот вызывает методы Потерять высоту() и Набрать высоту() класса Самолёт(). Метод разносить еду() класса стюардесса выводит на экран текст.

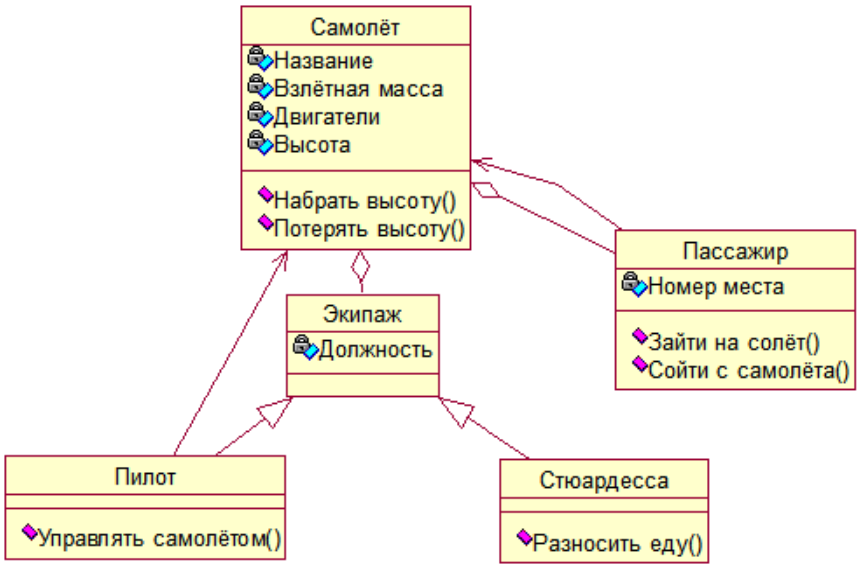

Рисунок 93 – Диаграмма классов

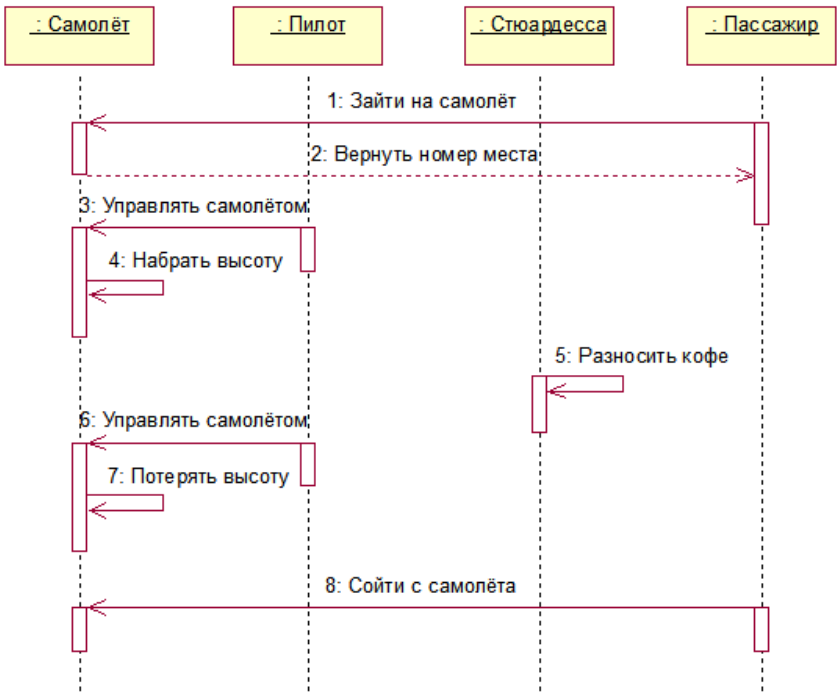

Рисунок 94 – Диаграмма последовательности для приложения

Вариант 18

Реализовать классы Бедный клиент, Богатый клиент и Студент, наследующие от класса Клиент, а также классы Банкомат и Деньги. При вызове метода Снять деньги со счёта() класса клиент вызывается метод проверить пароль() класса Банкомат, который возвращает случайное число. После этого Классу клиент передаётся ссылка на класс Деньги.

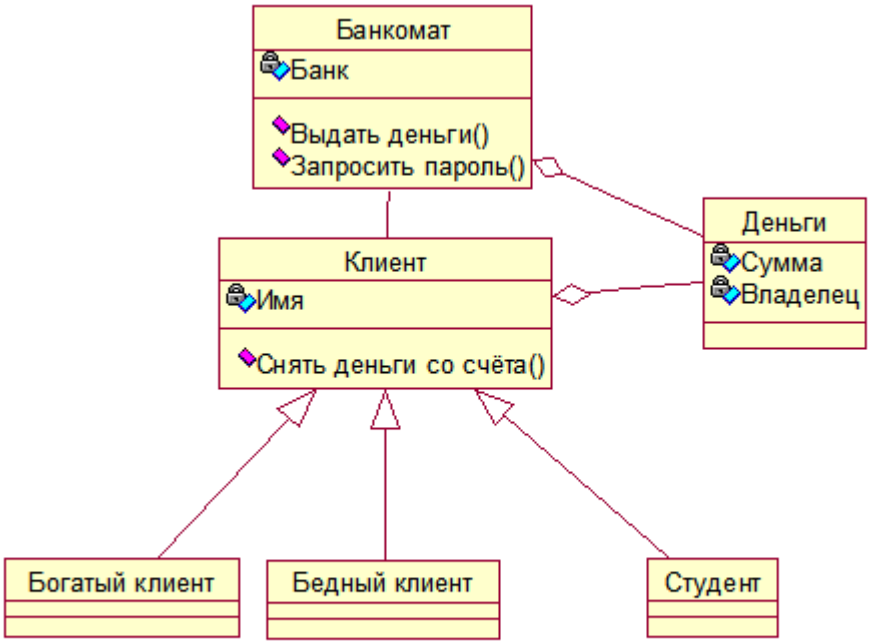

Рисунок 95 – Диаграмма классов

Рисунок 96 – Диаграмма последовательности для приложения

Вариант 19

Реализовать классы Эсминец, Авианосец, Крейсер, наследующие от класса Корабль. Метод Запустить торпеду() класса Эсминец вызывает метод взорваться класс Торпеда. Метод запустить самолёт класса Авианосец вызывает методы Взлететь(), провести бомбардировку(), сесть() класса Самолёт. Метод сесть класса Самолёт вызывает метод Принять самолёт() класса Авианосец.

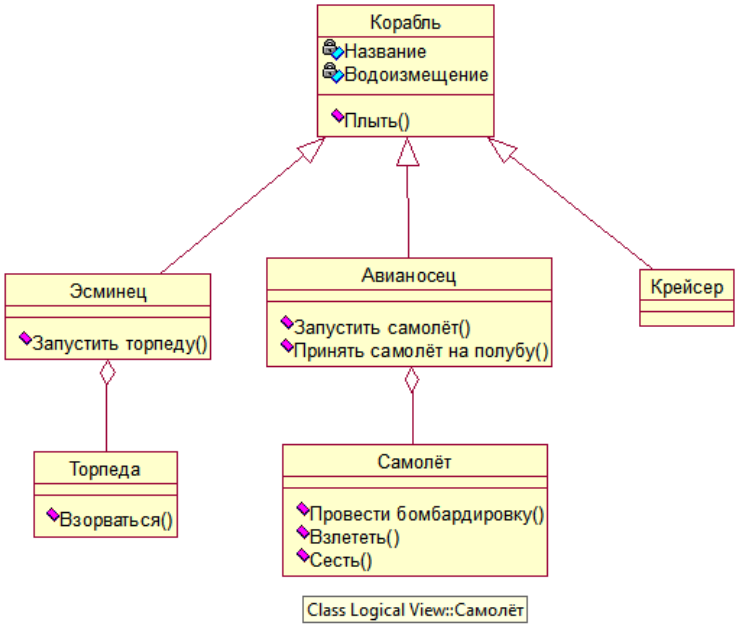

Рисунок 97 – Диаграмма классов

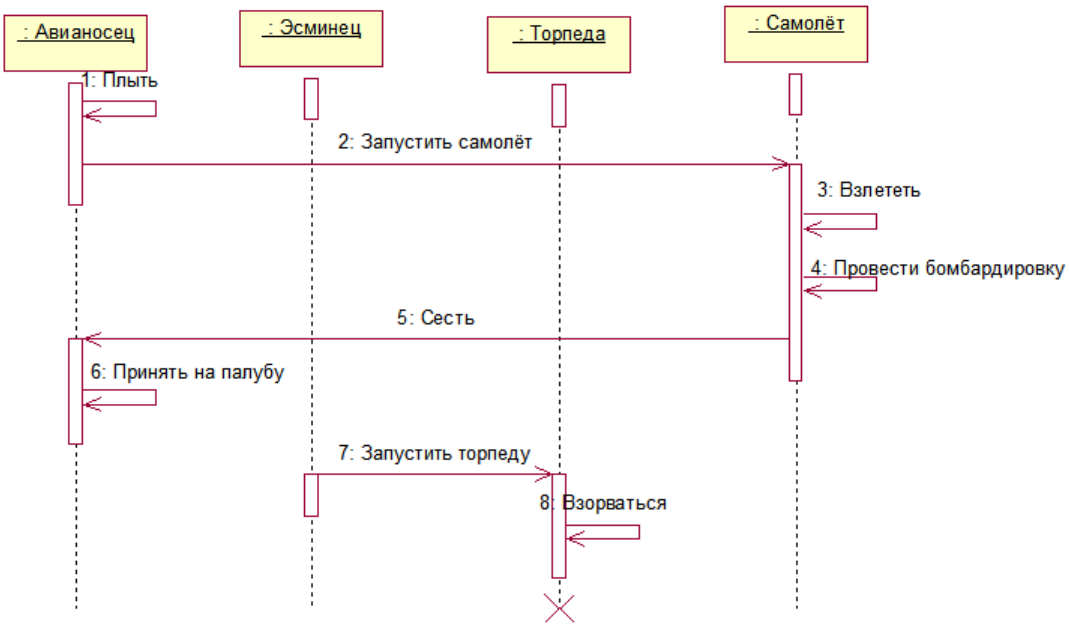

Рисунок 98 – Диаграмма последовательности для приложения

Вариант 20

Реализовать классы Человек, Принтер и классы Офисная бумага, Чертёжная бумага, Фотобумага, наследующие от класса Бумага. Метод заправить бумагой() класса Человек передаёт ссылку на класс Бумага классу Принтер. Метод Печатать текст() класса принтер изменяет поля класса Офисная бумага, а метод Печатать изображения изменяет поля класса Фотобумага.

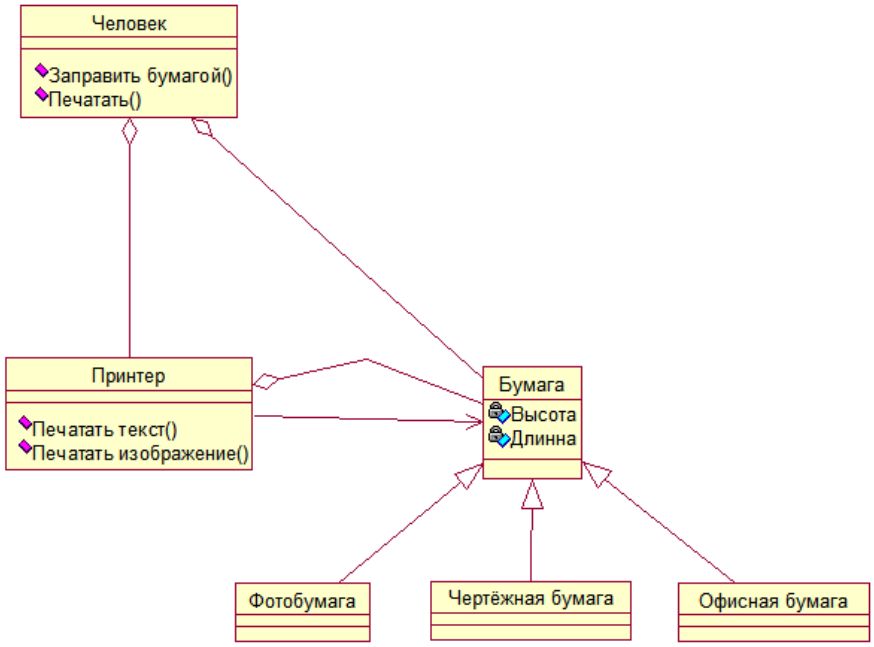

Рисунок 99 – Диаграмма классов

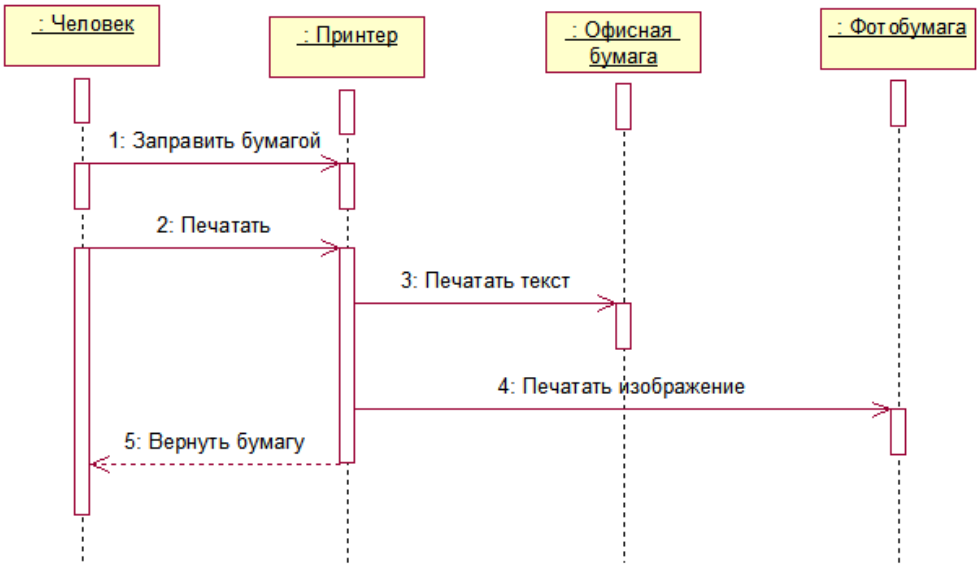

Рисунок 100 – Диаграмма последовательности для приложения

Вариант 21

Реализовать класс Адрес, а также классы Студент и Рабочий, наследующие от класса Человек. Метод заселиться() добавляет ссылку на Класс Адрес в поле класса Человек. Метод выселиться() эту ссылку убирает. Методы работать и учиться выводят на экран текст.

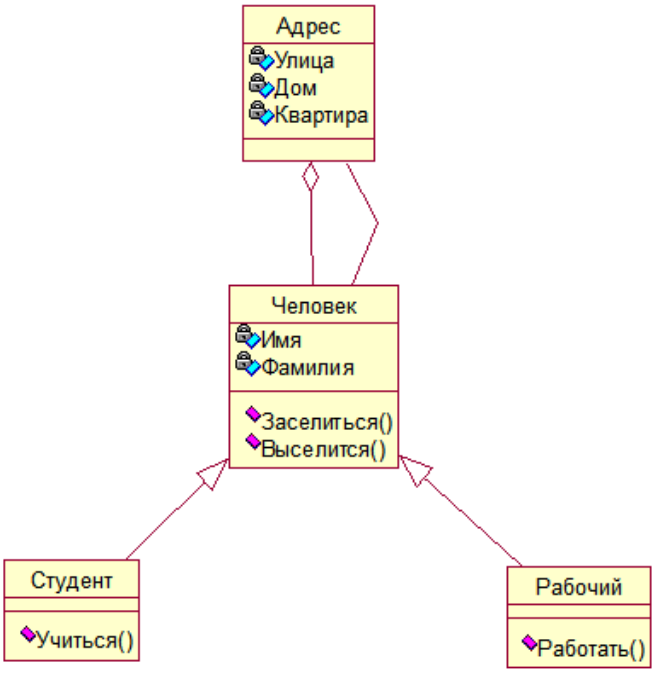

Рисунок 101 – Диаграмма классов

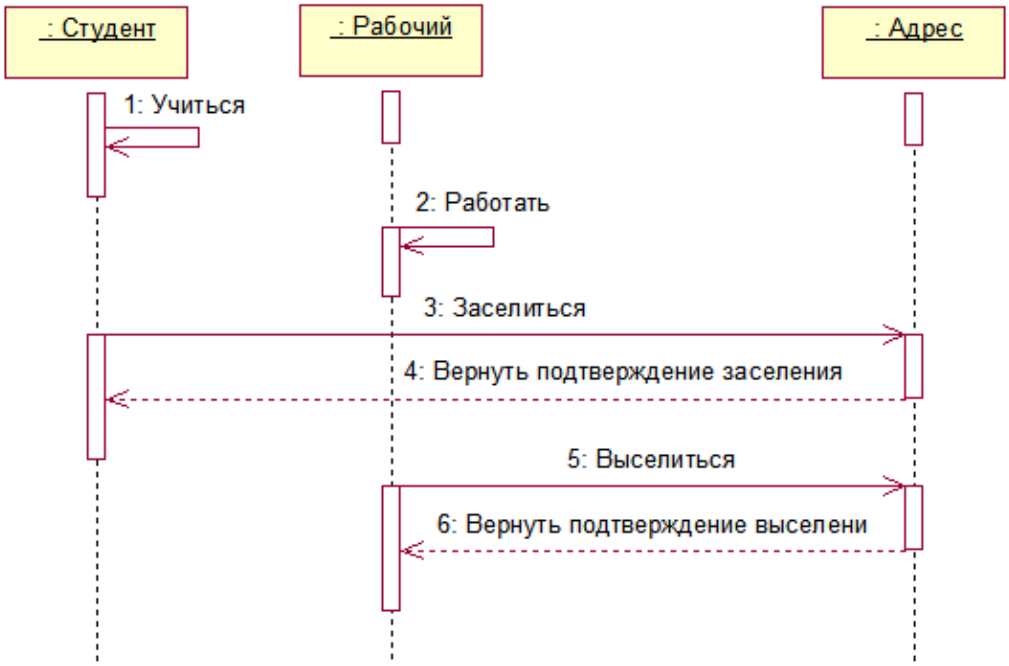

Рисунок 102 – Диаграмма последовательности для приложения

Вариант 22

Реализовать Классы Рабочий, Зоопарк, Страус, Слон. Класс Страус наследует от класса Птица, Слон наследует от Млекопитающее, Птица и млекопитающее наследуют от Животное. Метод Кормить() класса Рабочий вызывает метод Есть() класса животное, который выводит на экран текст и возвращает строку.

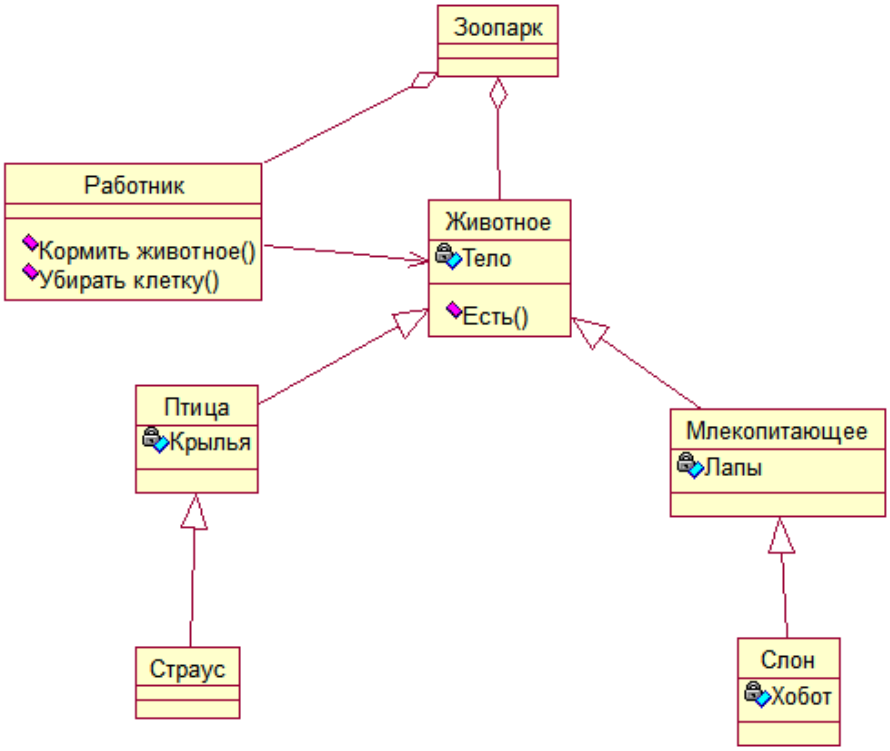

Рисунок 103 – Диаграмма классов

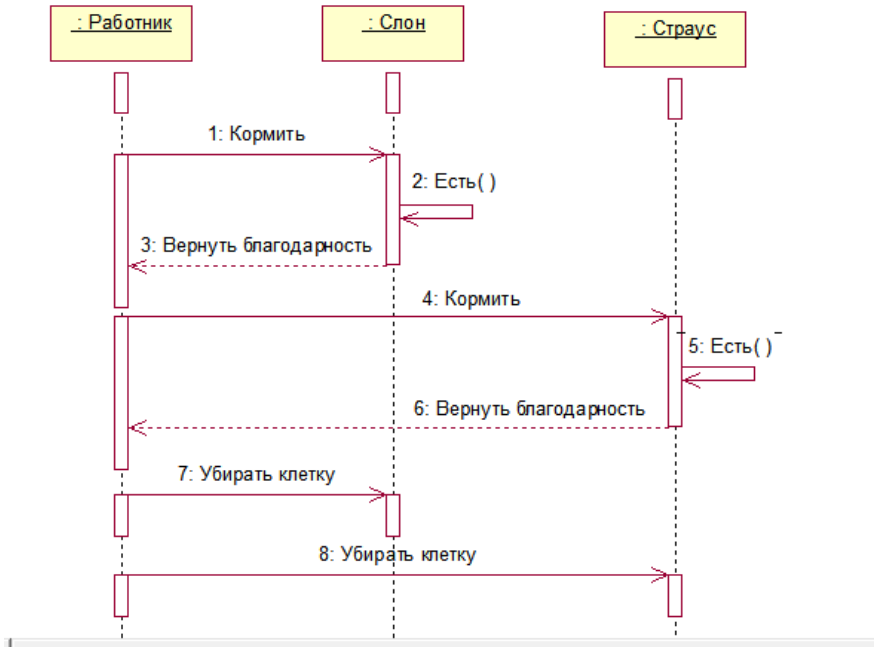

Рисунок 104 – Диаграмма последовательности для приложения

Вариант 23

Реализовать классы Школьник и Студент, наследующие от класса Учащийся и классы Школа и Университет, наследующие от класса Учреждение образования. Учащийся может учиться(), что выводит на экран текст. Если метод учиться вызван достаточно раз, то Школьник вызывает метод отчислиться со школы(), который вызывает метод Отчислить учащегося() класса Школа, который в свою очередь убирает ссылку на школьника из класса Школа. Далее школьник может сдать вступительный экзамен(), что вызовет метод провести экзамен() класса университет. Если оценка достаточна, то создаётся класс Студент. Школьник разрушается.

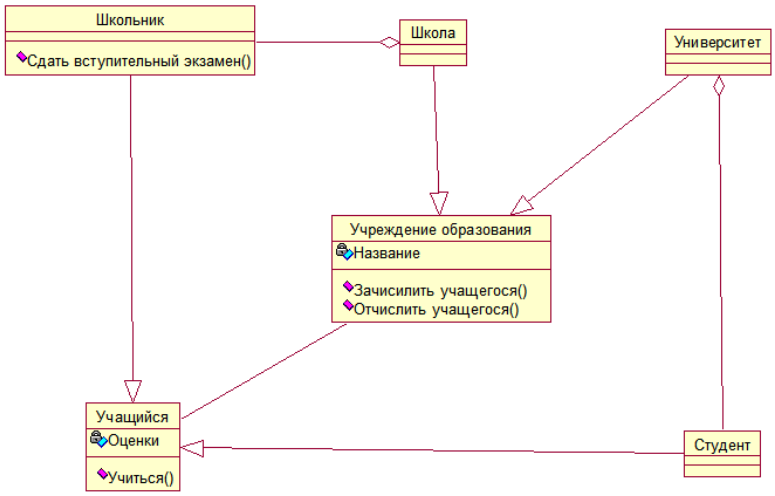

Рисунок 105 – Диаграмма классов

Рисунок 106 – Диаграмма последовательности для приложения

Вариант 24

Реализовать классы Человек, Чашка, Бутылка, Вода, Сок. Классы Чашка и бутылка наследуют от класса Сосуд, а классы Вода и Сок наследуют от класса Жидкость. Класс Сосуд содержит ссылку на класс Жидкость. Метод перелить жидкость перемещает ссылку на жидкость из одного объекта в другой. Метод вылить жидкость() убирает ссылку из объекта и возвращает её.

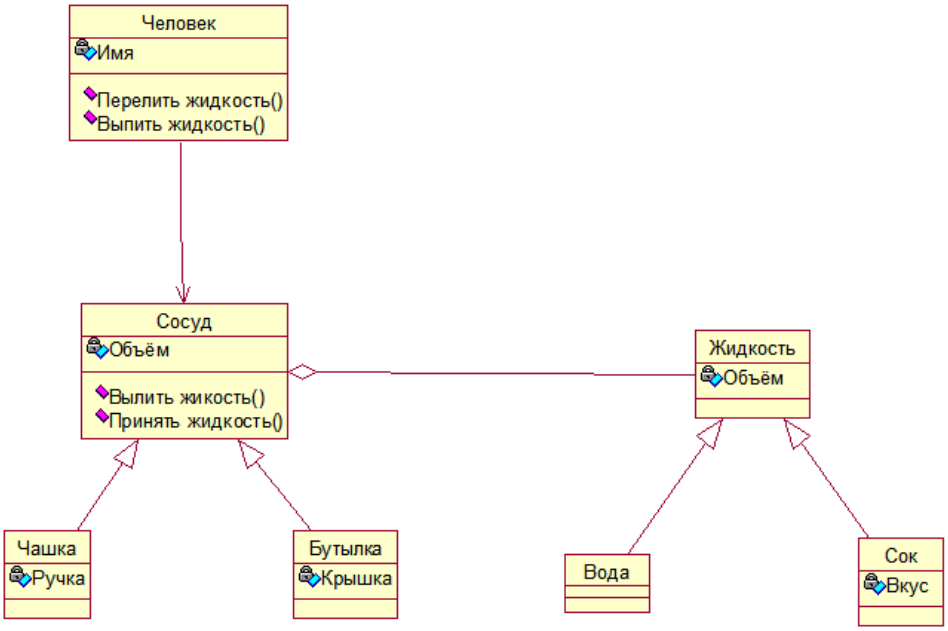

Рисунок 107 – Диаграмма классов

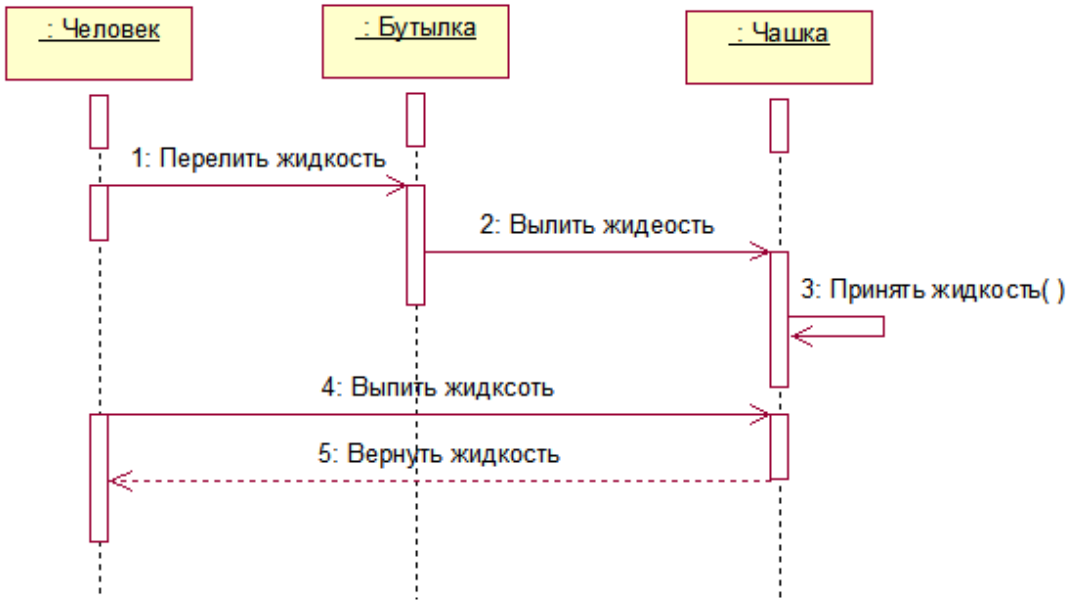

Рисунок 108 – Диаграмма последовательности для приложения

Вариант 25

Реализовать методы Покупатель, Магазин, Тележка, Товар. Метод взять товар() запоминает ссылку на объект класса Товар. Метод положить товар() вызывает метод принять товар() класса Тележка, который запоминает ссылку на Товар. Метод купить товар() вызывает метод попросить деньги() класса Магазин, который вызывает метод заплатить() класса Покупатель, который отнимает некую сумму из поля деньги и возвращает её.

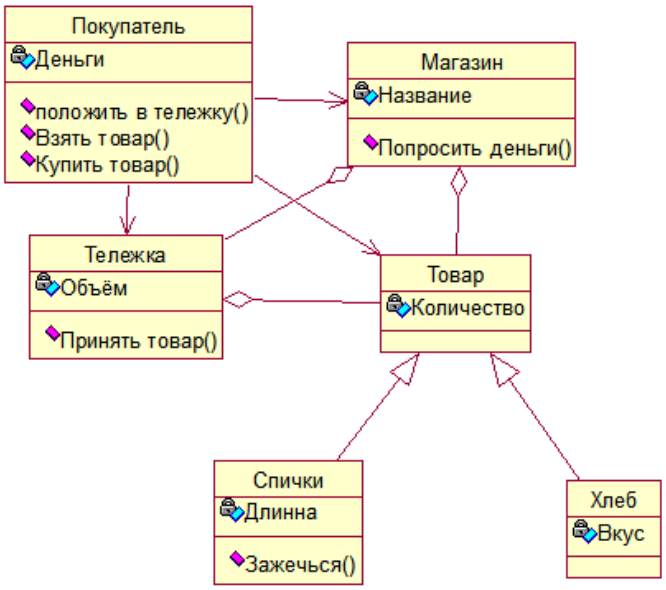

Рисунок 109 – Диаграмма классов

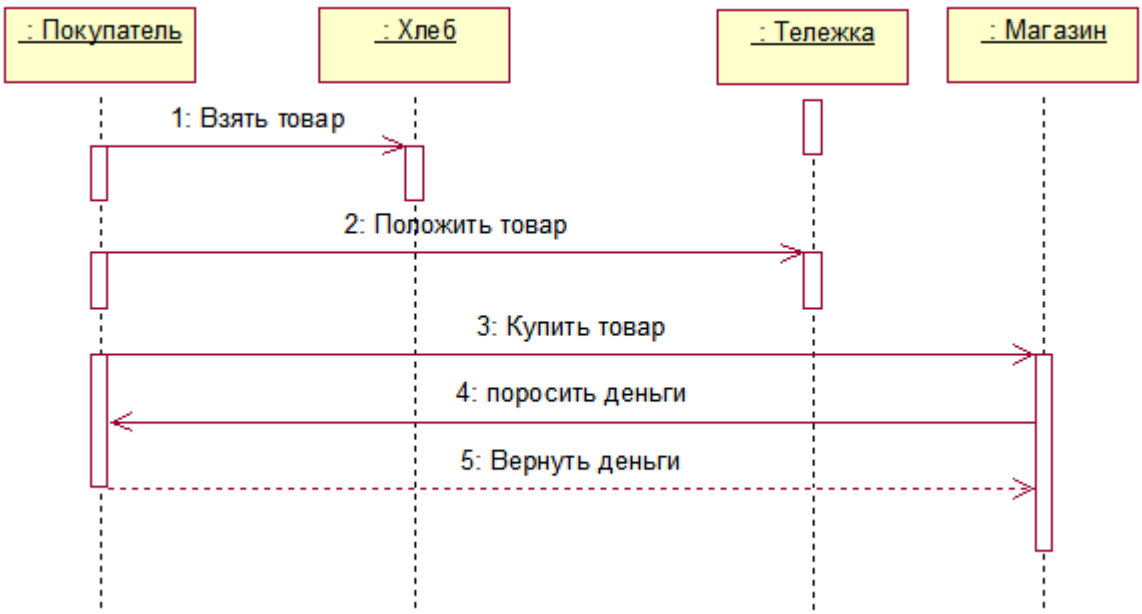

Рисунок 110 – Диаграмма последовательности для приложения

Вариант 26

Реализовать классы Студент, Портной, Ткань, Одежда. Метод Студента оставить заказ() вызывает метод снять мерки() класса Портной, который читает поля класса Студент. После этого класс Портной создаёт экземпляр класса Одежда.

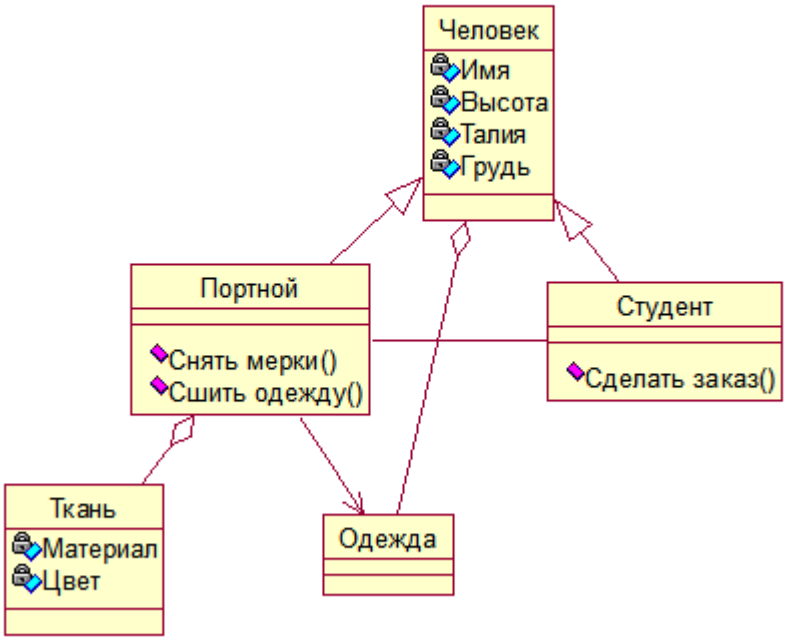

Рисунок 111 – Диаграмма классов

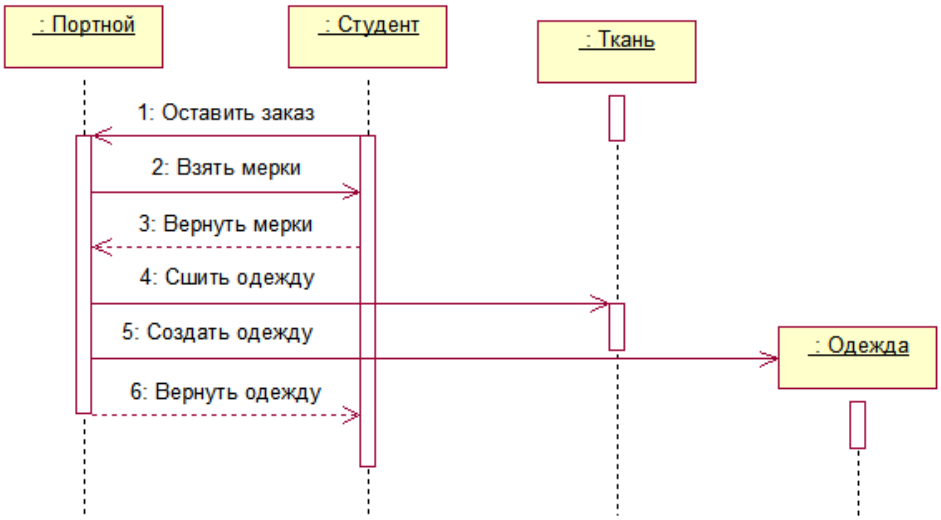

Рисунок 112 – Диаграмма последовательности для приложения

Вариант 27

Реализовать классы Персонаж, Квест на убийство, Квест на перемещение. Метод Персонажа Взять квест() вызывает метод Активировать квест() класса Квест, который возвращает строку с текстом. При вызове метода убить монстра() достаточное количество раз вызывается метод завершить квест(), который выводит на экран текст()

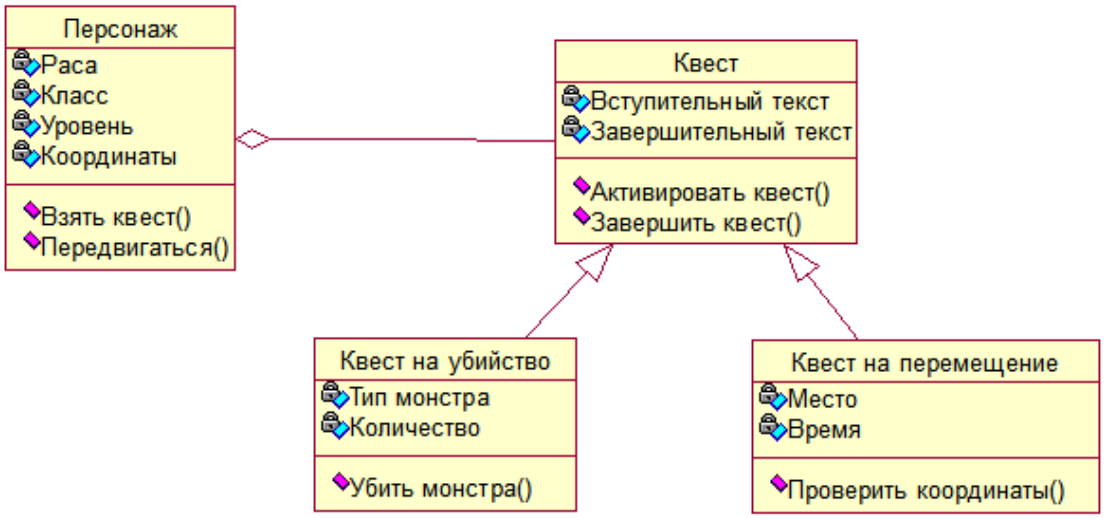

Рисунок 113 – Диаграмма классов

Рисунок 114 – Диаграмма последовательности для приложения

Вариант 28

Реализовать классы Пользователь, Зарегистрированный пользователь, Сообщение, Форум, Администратор. Зарегистрированный пользователь наследует от Пользователя, Администратор наследует от Зарегистрированного пользователя. При вызове метода создать сообщение() создаётся объект Класса сообщение и Пользователь получает на него ссылку. Метод Получить список сообщений() возвращает ссылку на коллекцию сообщений. Метод удалить сообщение() разрушает объект класса сообщение.

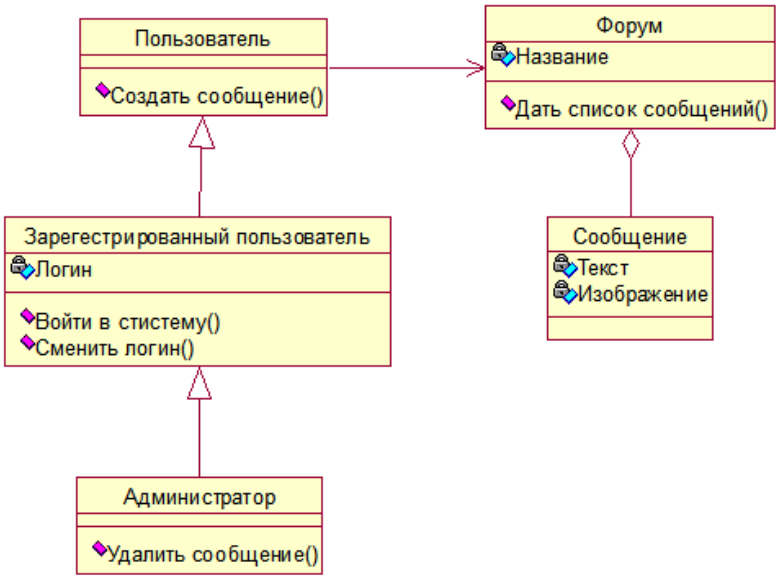

Рисунок 115 – Диаграмма классов

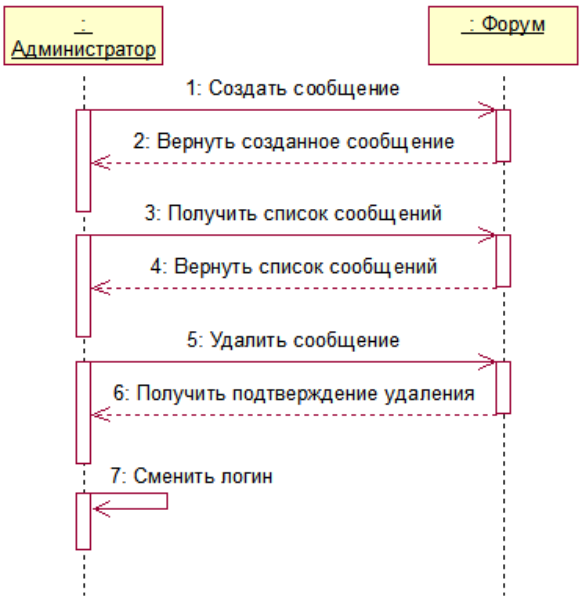

Рисунок 116 – Диаграмма последовательности для приложения

Вариант 29

Реализовать классы Пользователь, Зарегистрированный пользователь, Система авторизации, Информация о пользователе. Информация о пользователе является контейнером. При вызове метода Зарегистрироваться класса Пользователь. Создаётся Информация о пользователе и новый объект Зарегистрированный пользователь, которому сообщается ссылка на контейнер с данными. При вызове метода Войти в систему() объекту Форум посылается копия объекта Информация о пользователе, который ищет объект с такими же полями и возвращает true или false.

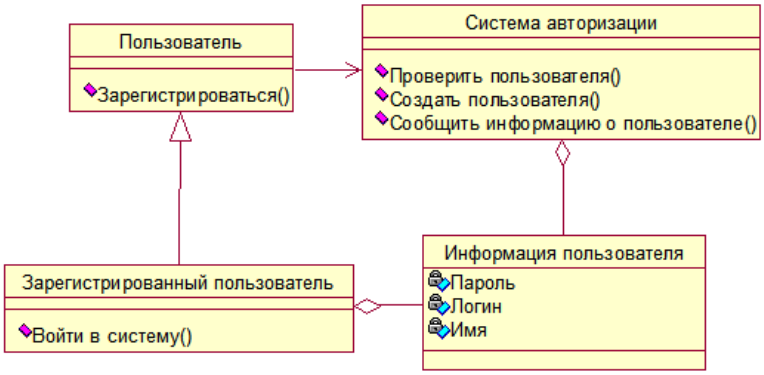

Рисунок 117 – Диаграмма классов

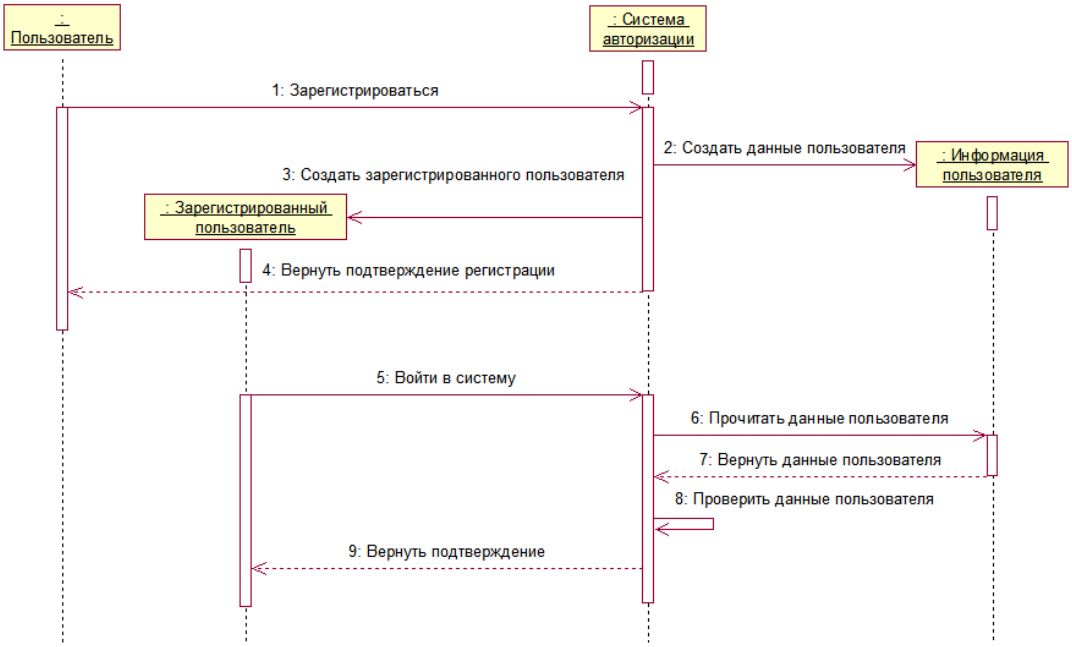

Рисунок 118 – Диаграмма последовательности для приложения

Вариант 30

Реализовать классы Пользователь, Видеохостинг, Видео с котиками, Видео с собачками. Метод создать видео() класса Пользователь создаёт объект класса Видео вызывает метод Добавить видео() класса Видеохостинг, сообщая методу ссылку на Видео. Метод получить список видео возвращает коллекцию Видео. А метод посмотреть видео получает ссылку на Видео.

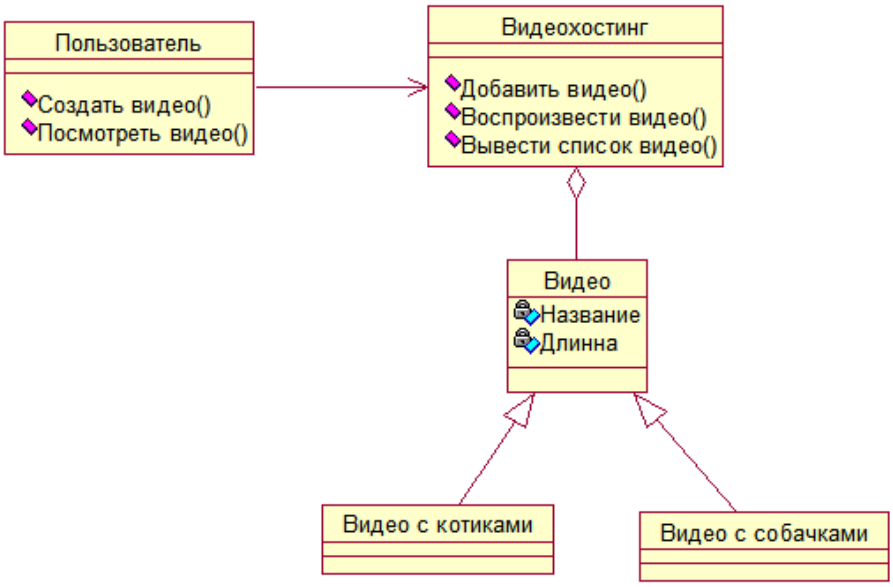

Рисунок 119 – Диаграмма классов

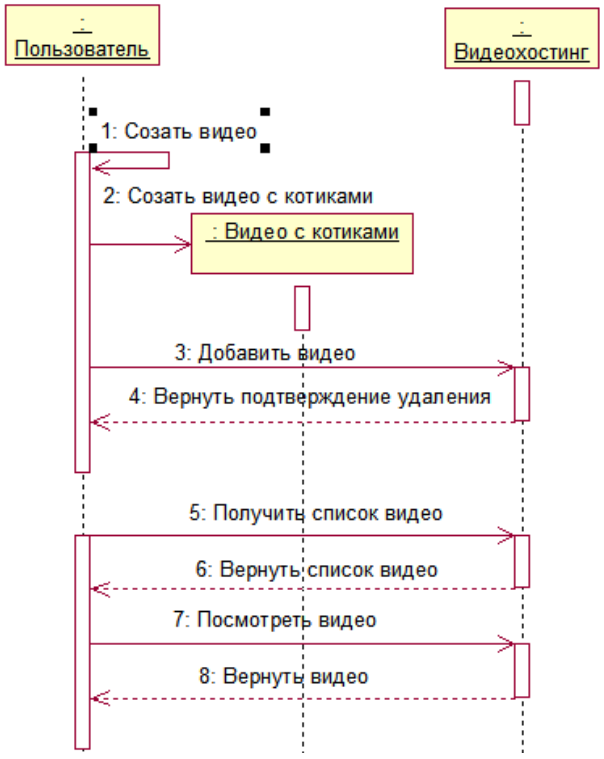

Рисунок 120 – Диаграмма последовательности для приложения

Лабораторная работа №3

Разработка программ с использованием модульного подхода

Задание к лабораторной работе:

– необходимо разработать графический интерфейс с использованием менеджера компоновки;

– представление графического интерфейса разрабатывается с учётом предоставления всей функциональности предложенного варианта;

– объекты и их взаимоотношения, имеющиеся в варианте задания, должны быть реализованы;

– функциональная часть приложения, представленная диаграммой последовательности, должна быть реализована;

– графический интерфейс должен быть создан посредством одной из следующих библиотек: Swing, SWT, JavaFX;

– разбить функционал приложения на несколько пакетов придерживаясь логики.

Вариант 1

Добавить класс журнал и организовать взаимодействие с ним. Чтобы про вызове метода Отметить присутствующих класса Староста в классе Строка журнала обозналось присутствие студента на лекции. А при вызове метода Отметить присутствующих класса Преподаватель эти данные считывались и сравнивались с данными объекта Преподаватель.

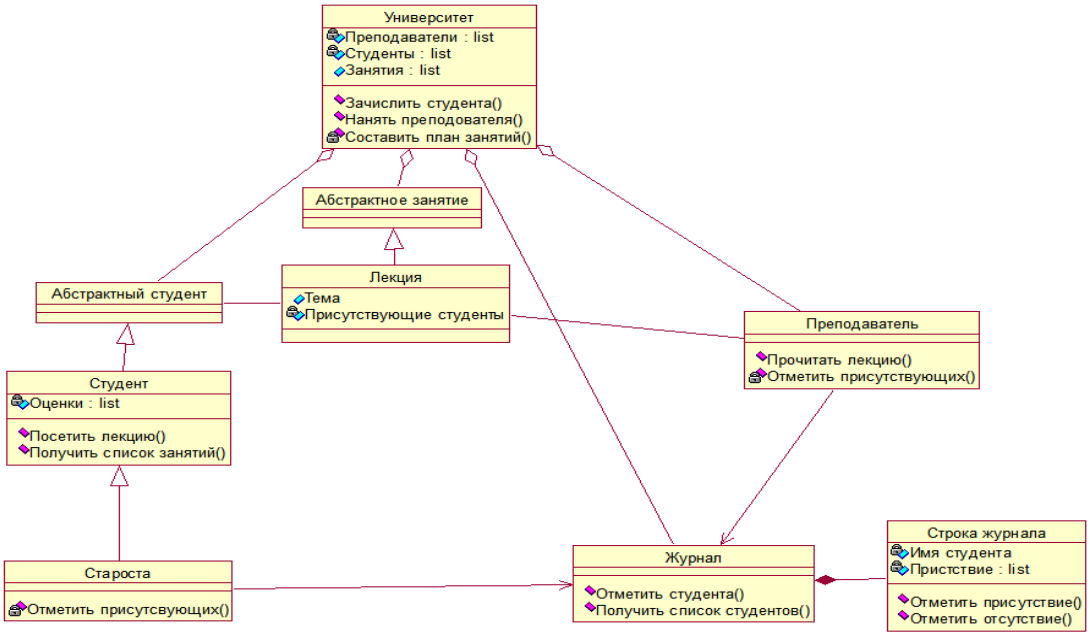

Рисунок 121 – Диаграмма классов

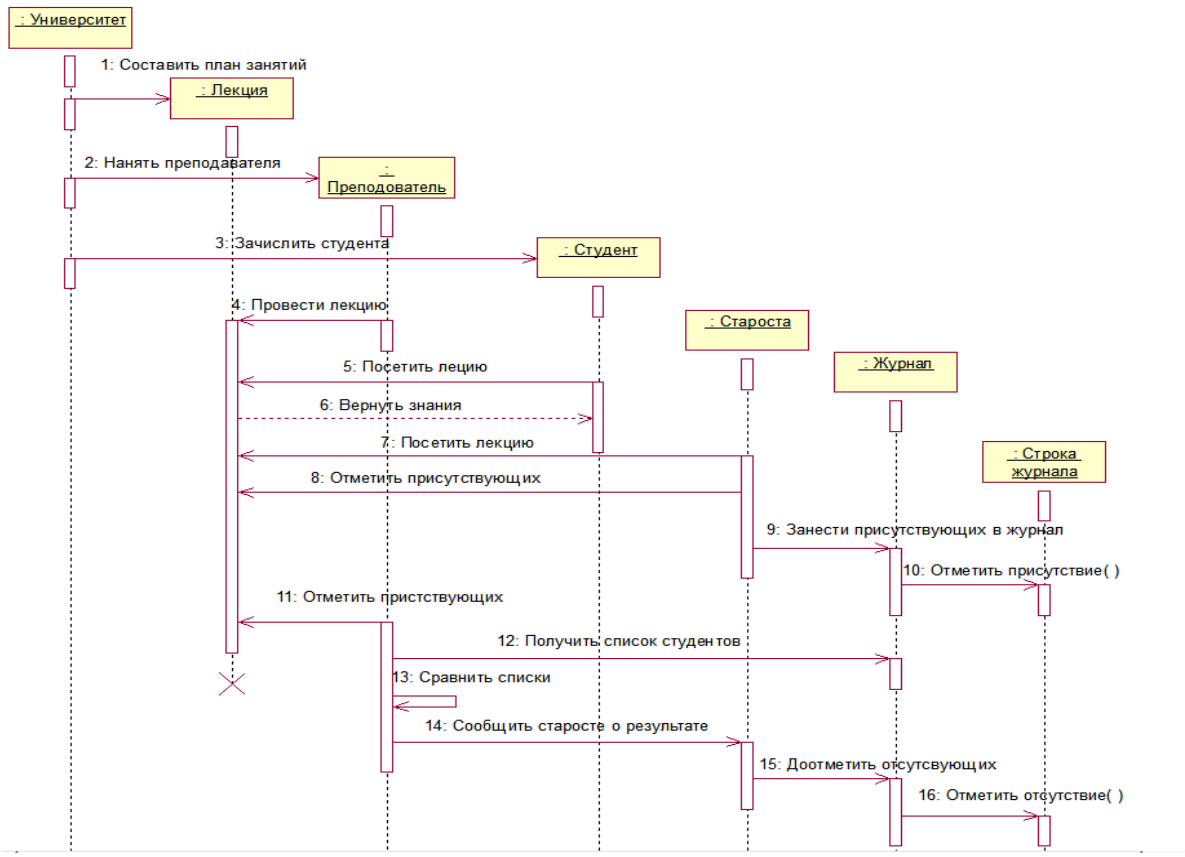

Рисунок 122 – Диаграмма последовательности

Вариант 2

Добавить класс Комплексное число в показательном виде, организовать взаимодействие с классом Комплексное число в алгебраическом виде. Реализовать действие умножения и деления. Реализовать класс Человек, управляющий классом Выражение.

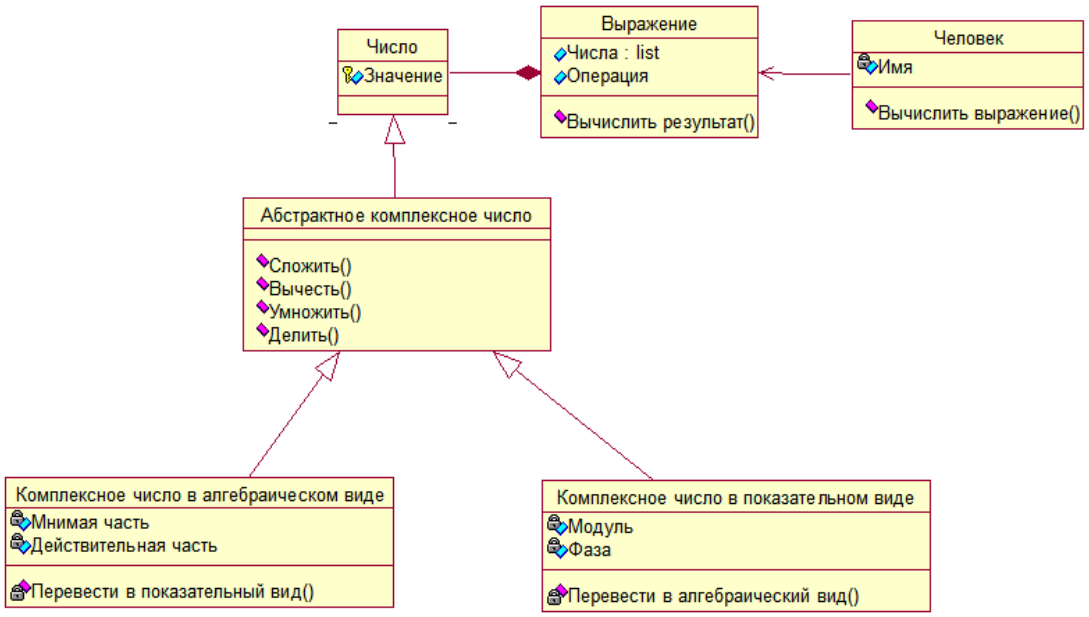

Рисунок 123 – Диаграмма классов

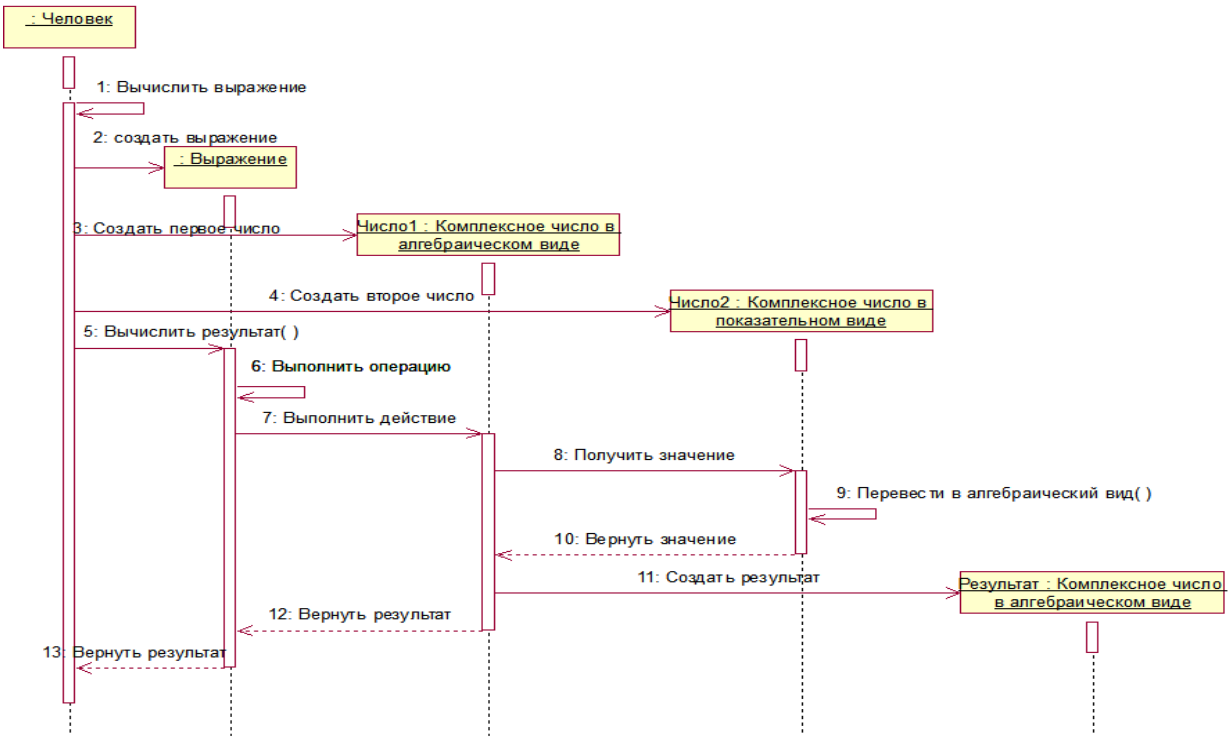

Рисунок 124 – Диаграмма последовательности

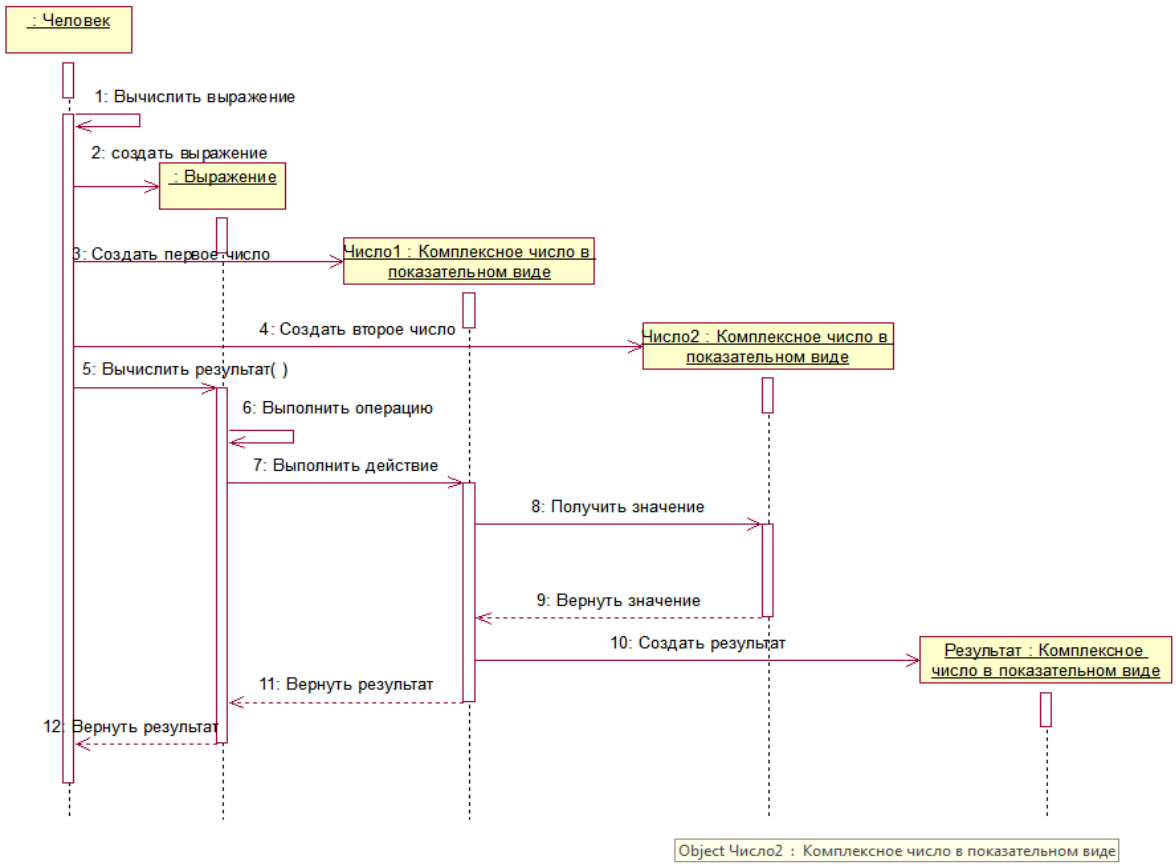

Рисунок 125 – Диаграмма последовательности

Вариант 3

Добавить класс Директор, управляющий классом Завод. Добавить класс Магазин, содержащий ссылку на класс товар, который производит класс Завод. Добавить новые классы, наследующие от класса Продукт.

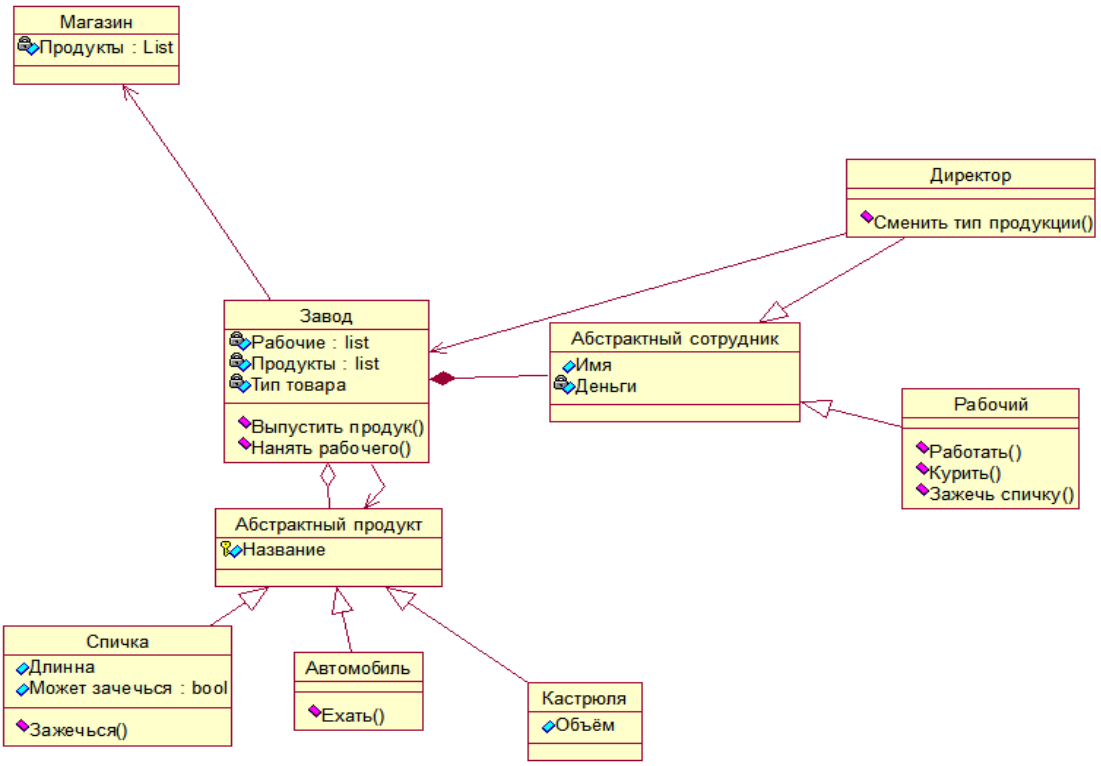

Рисунок 126 – Диаграмма классов

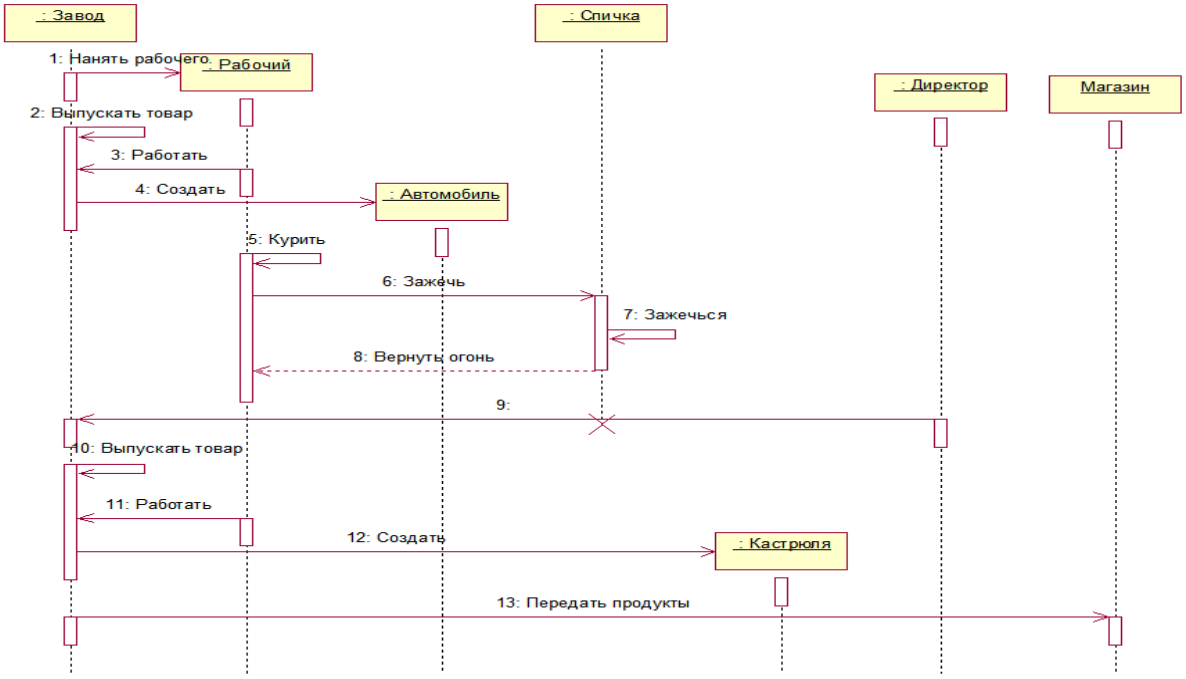

Рисунок 127 – Диаграмма последовательности

Вариант 4

Добавить классы Орган тела и Абстрактный орган. Добавить класс ухо, наследующий от класса Орган головы, класс Желудок, наследующий от класса Орган тела. Организовать взамодействие класса Рот с классом Желудок и класса Ухо с классом Мозг.

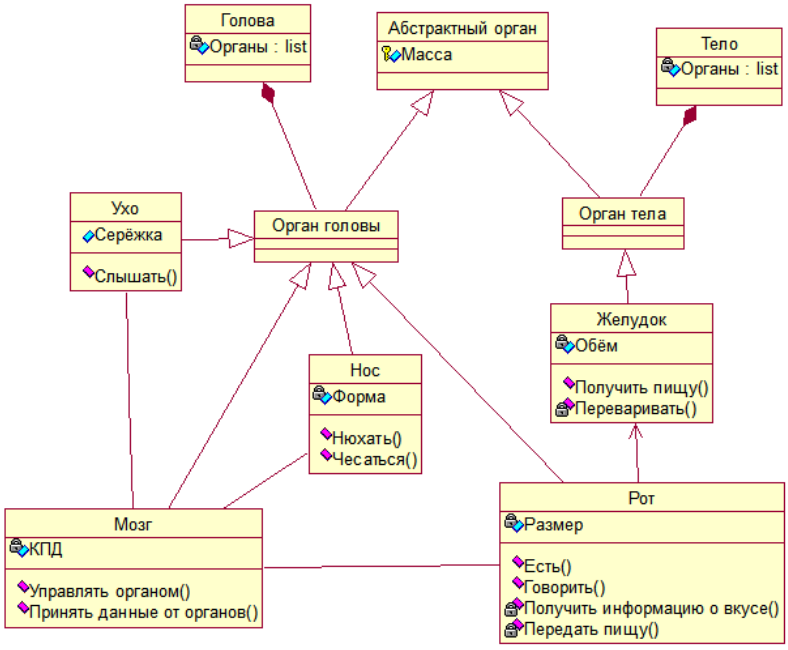

Рисунок 128 – Диаграмма классов

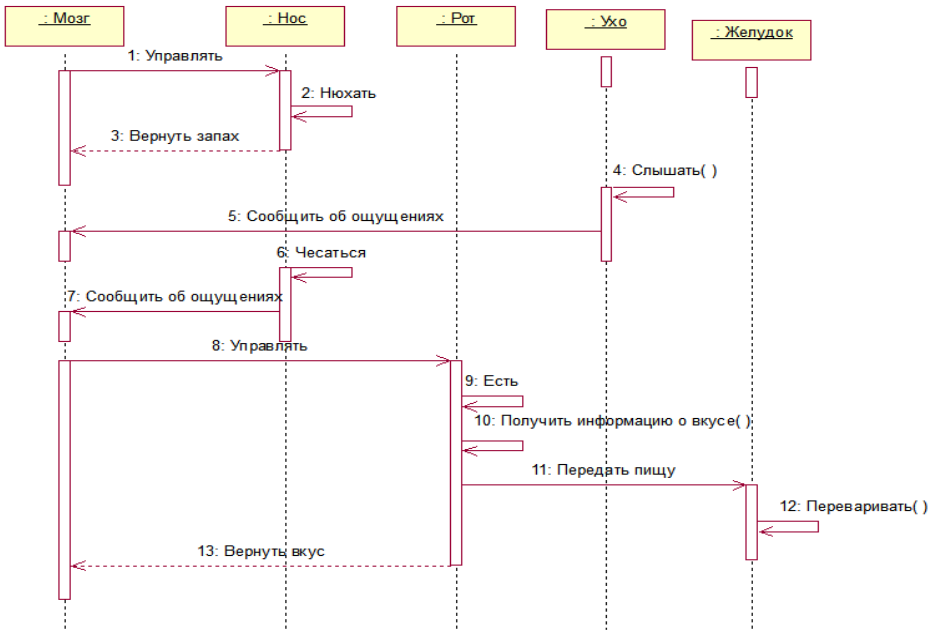

Рисунок 129 – Диаграмма последовательности

Вариант 5

Добавить класс Абстрактный файл, от которого наследует класс Программа. Добавить класс Видеофайл, и организовать его взаимодействие с классом Плеер и классом Браузер.

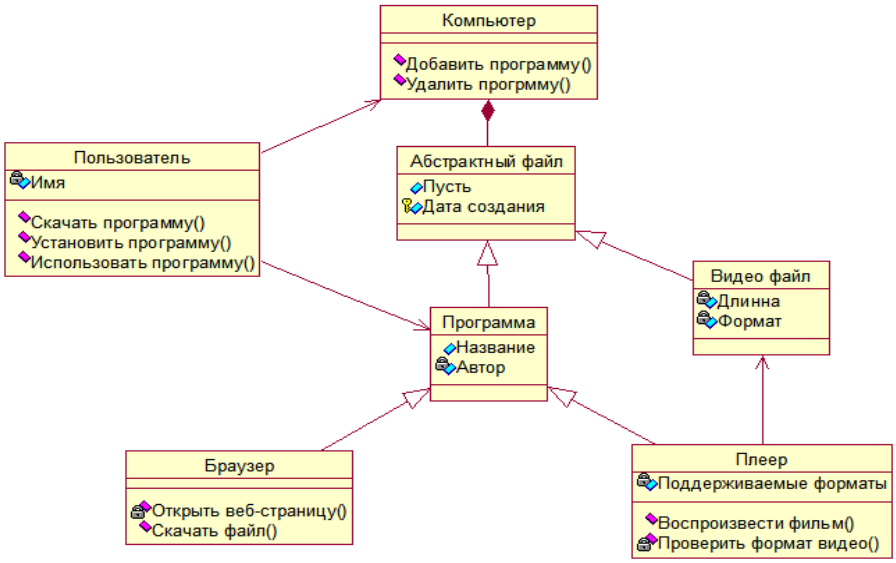

Рисунок 130 – Диаграмма классов

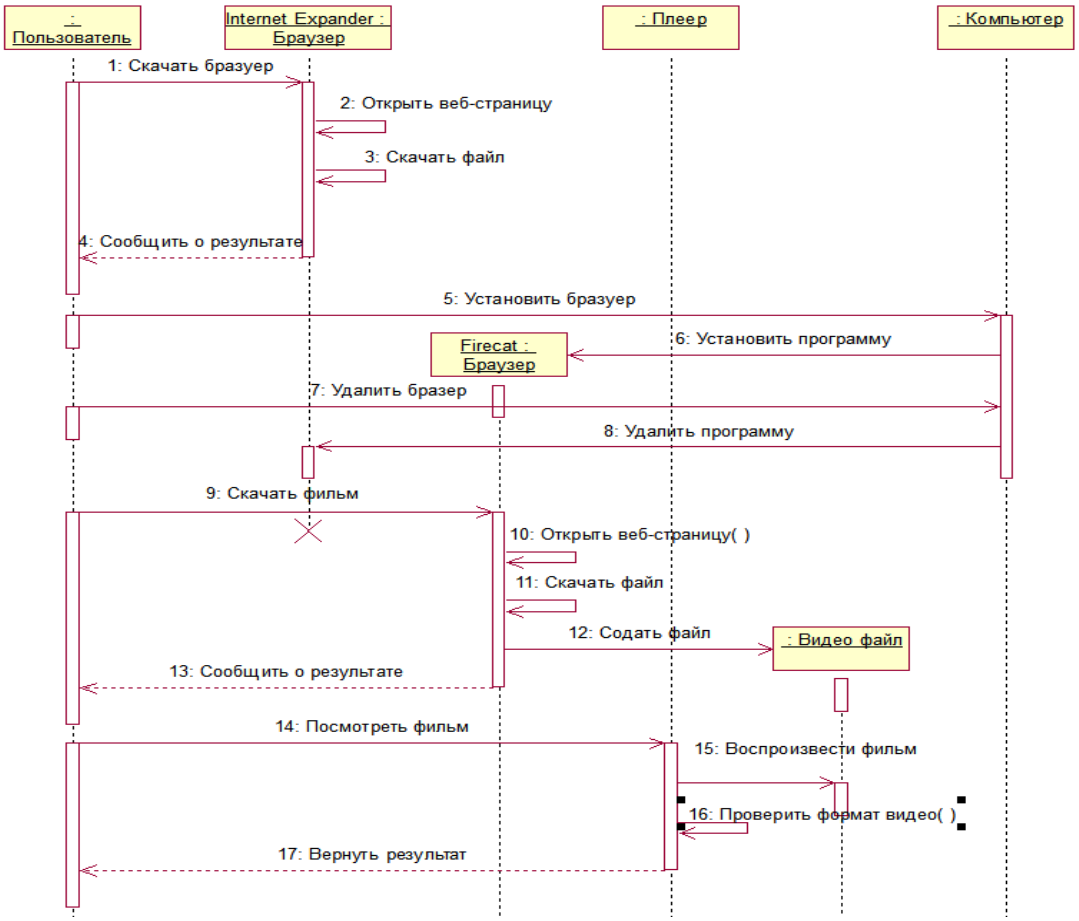

Рисунок 131 – Диаграмма последовательности

Вариант 6

Добавить класс Фермер, наследующий от класса Абстрактный человек и производящий экземпляры класса Овощ. Эти экземпляры потом использует Повар для производства объектов Еда и разрушает их после производства.

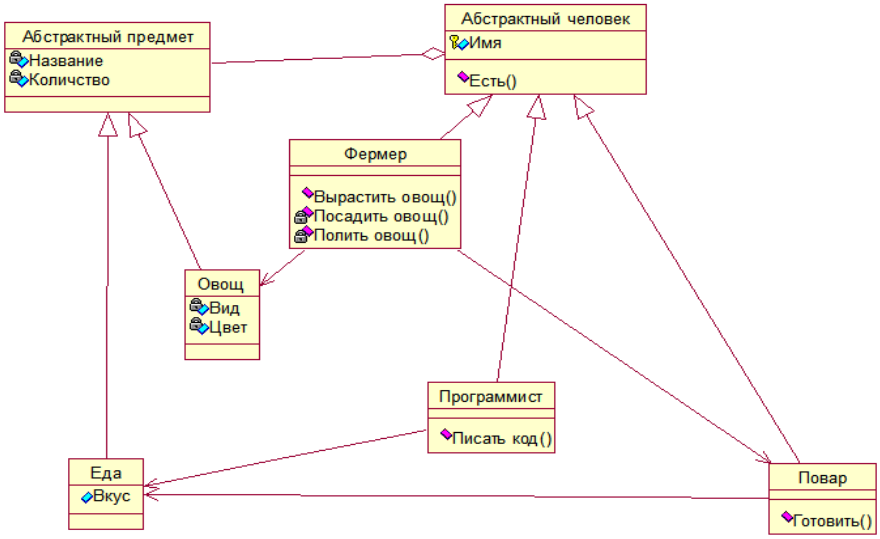

Рисунок 132 – Диаграмма классов

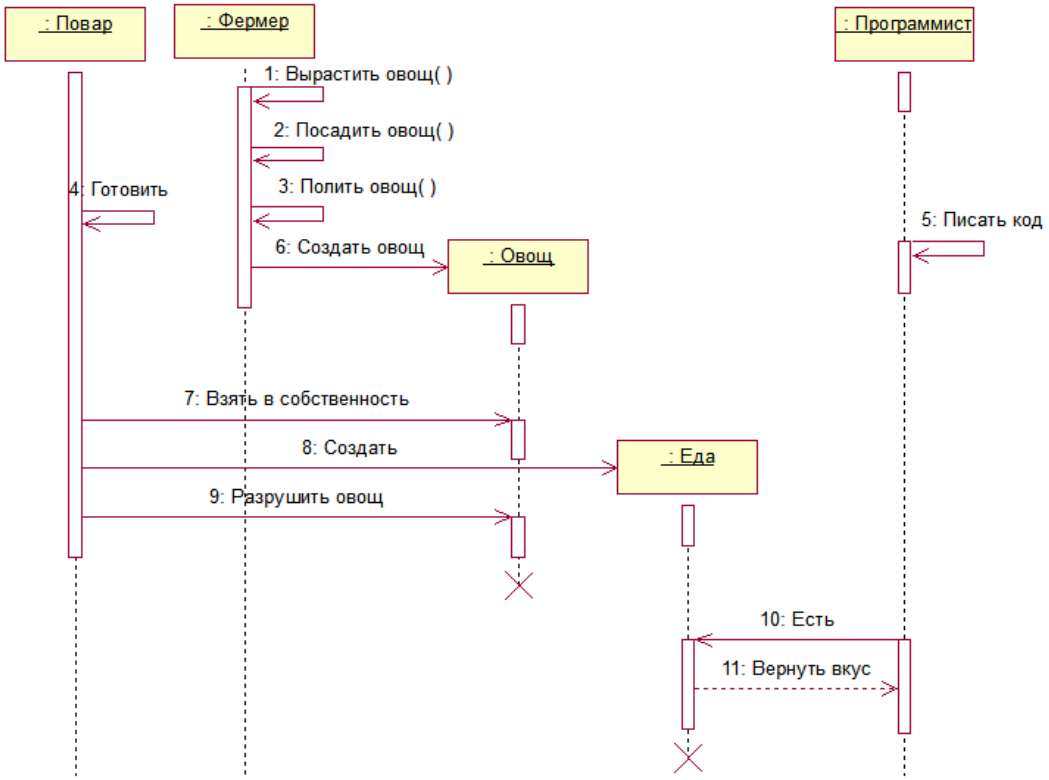

Рисунок 133 – Диаграмма последовательности

Вариант 7

Добавить класс Трактор, наследующий от класса Автомобиль, обладающий методом убирать снег(). Добавить класс Водитель, управляющий классом Автомобиль.

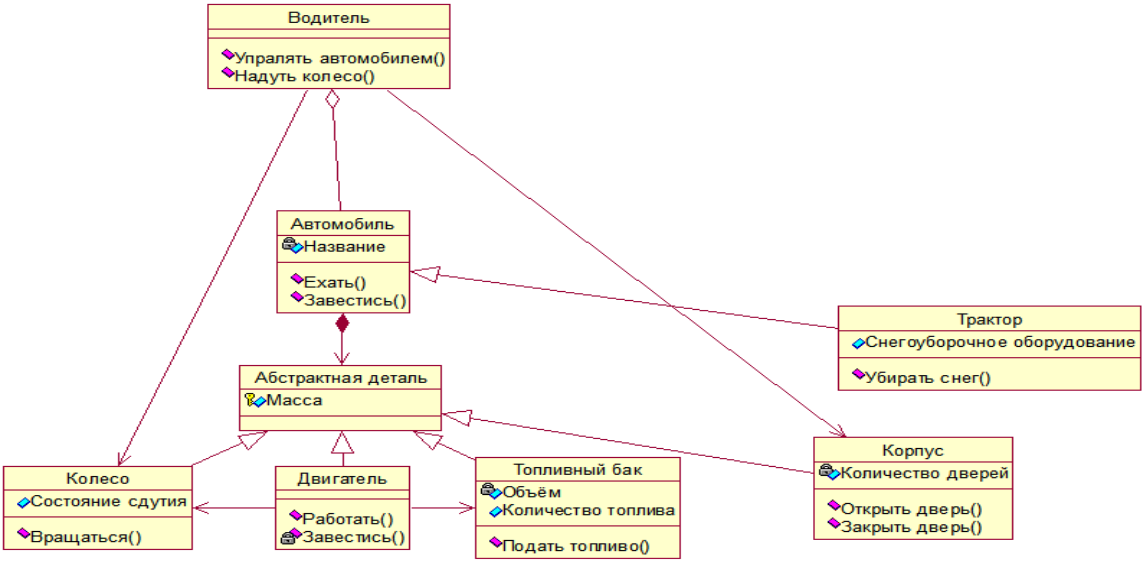

Рисунок 134 – Диаграмма классов

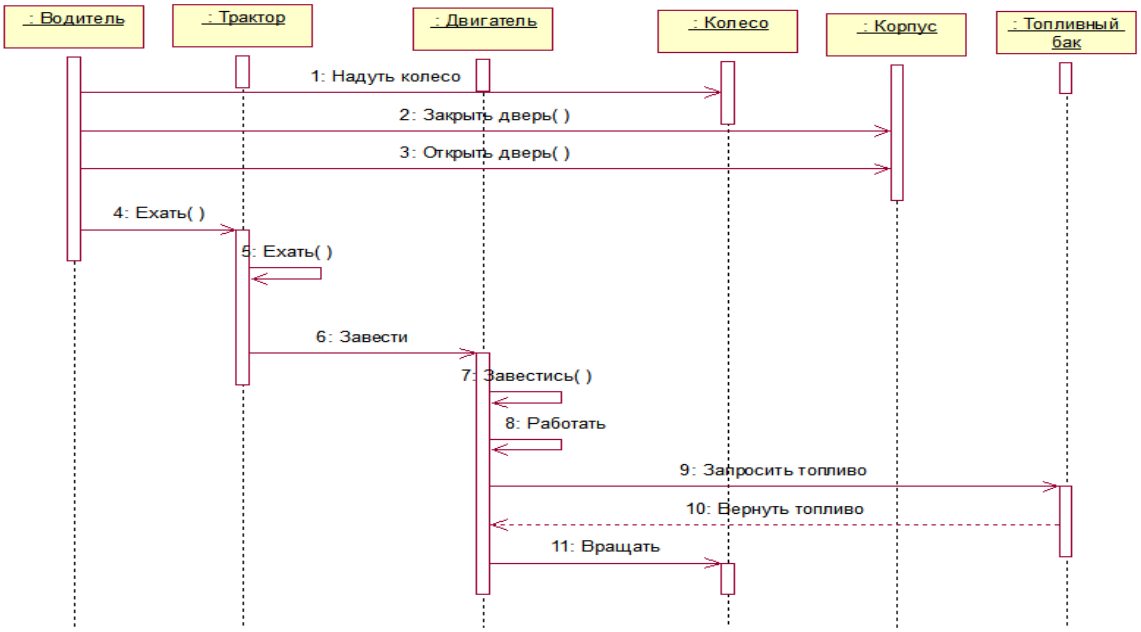

Рисунок 135 – Диаграмма последовательности

Вариант 8

Добавить класс Руководитель Компании, управляющий классом Компания. Добавить класс Вип-клиент, наследующий от класса клиент и класс Срочный заказ, наследующий от класса Заказ.

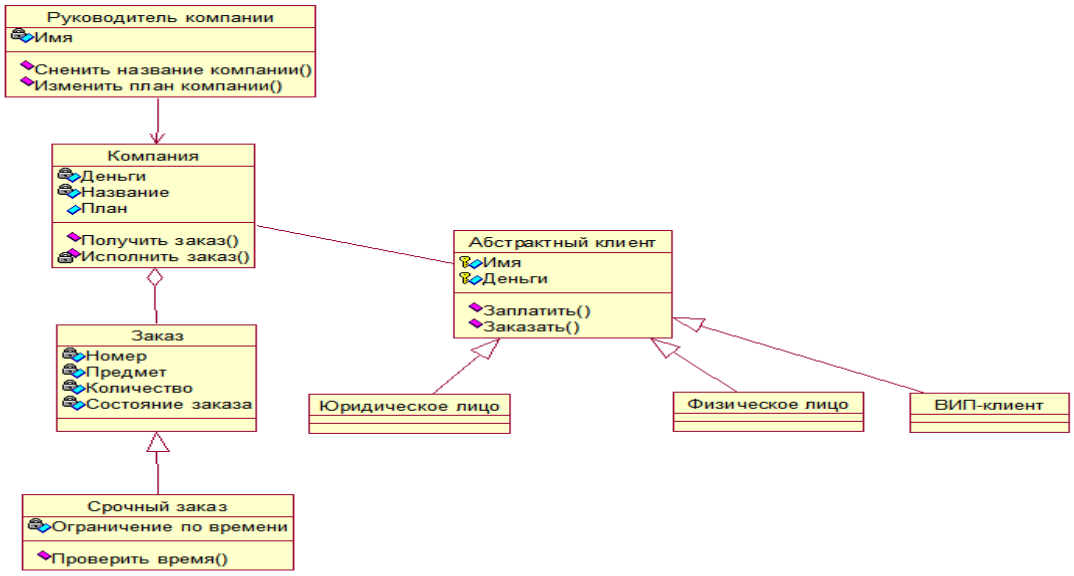

Рисунок 136 – Диаграмма классов

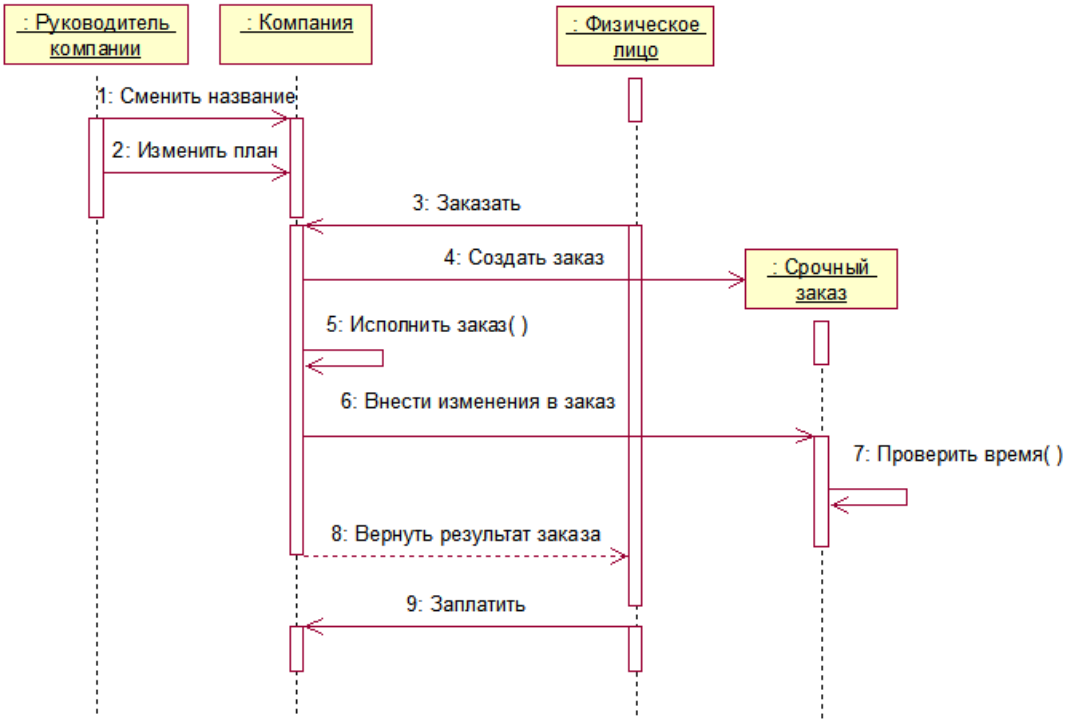

Рисунок 137 – Диаграмма последовательности

Вариант 9

Расширить взаимодействие классов. Организовать получение Работниками зарплаты и дать им возможность попросить увеличить зарплату. Результат просьбы увеличить зарплату должен зависеть от количества выполненных заказов.

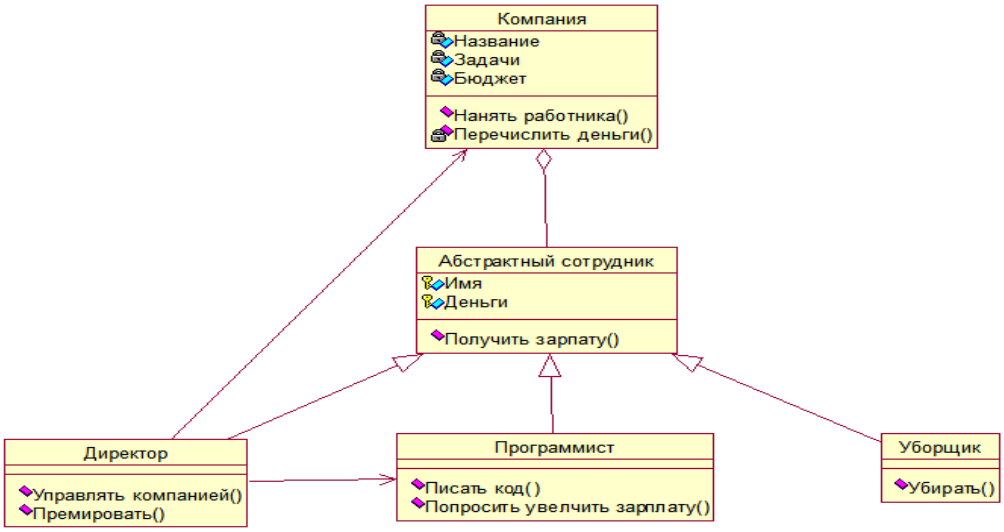

Рисунок 138 – Диаграмма классов

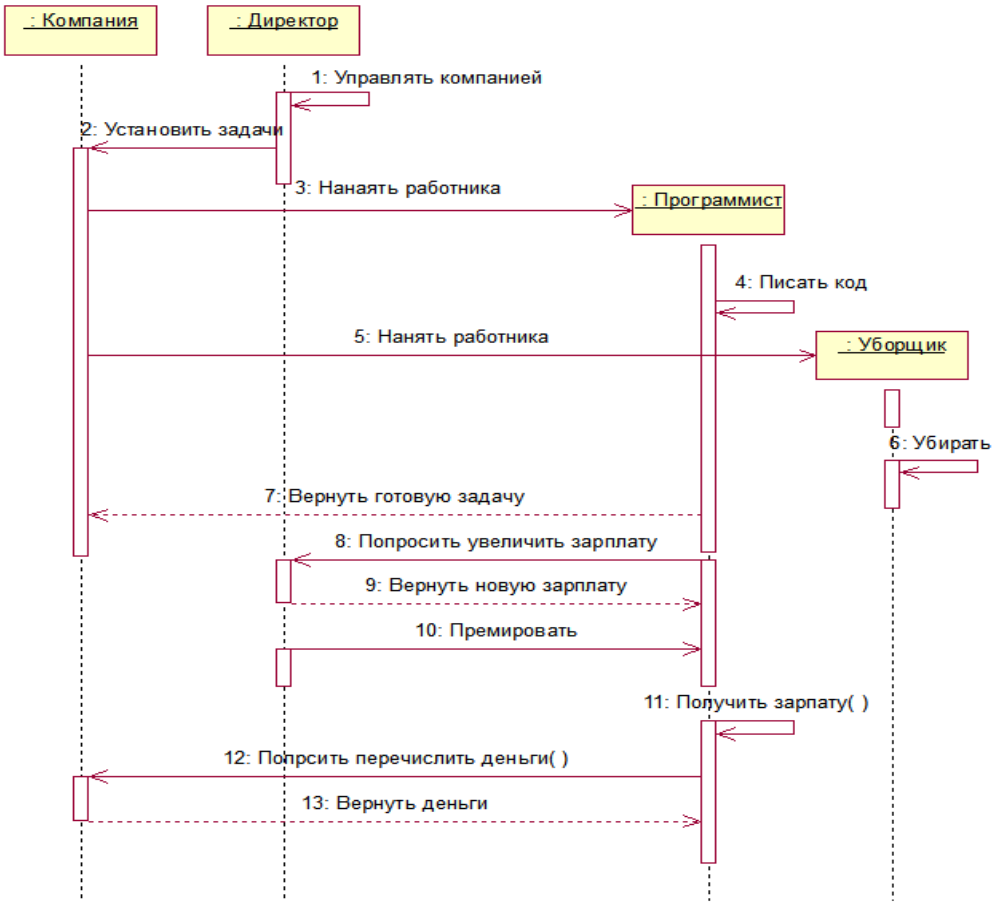

Рисунок 139 – Диаграмма последовательности

Вариант 10

Добавить класс Заведующий кафедрой, который управляет классом университет и отвечает за создание объектов класса экзамен. У класса экзамен есть два состояния сдан и не сдан.

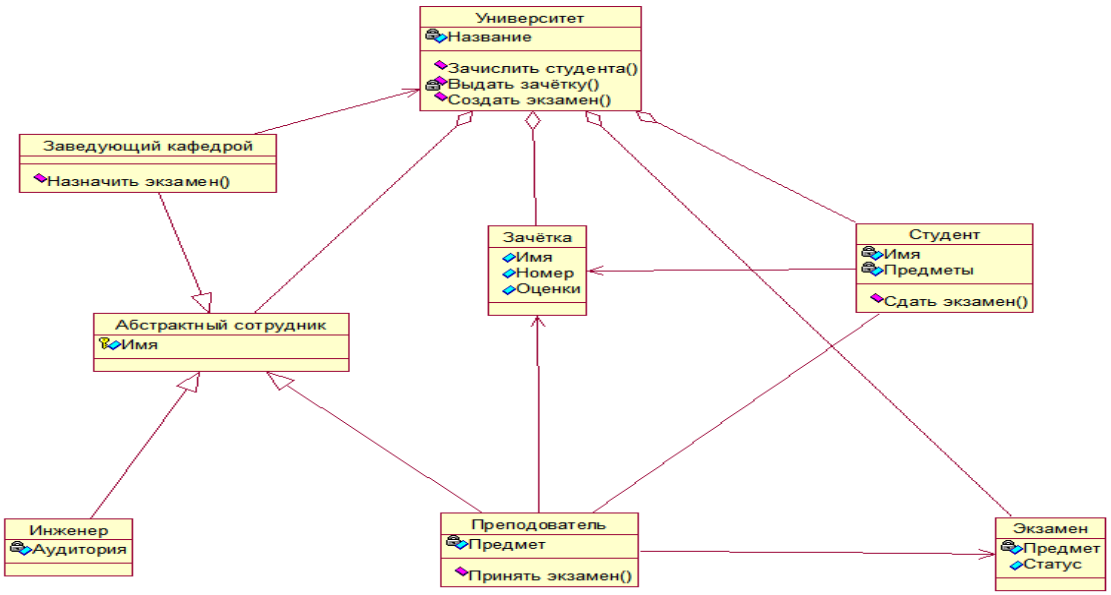

Рисунок 140 – Диаграмма классов

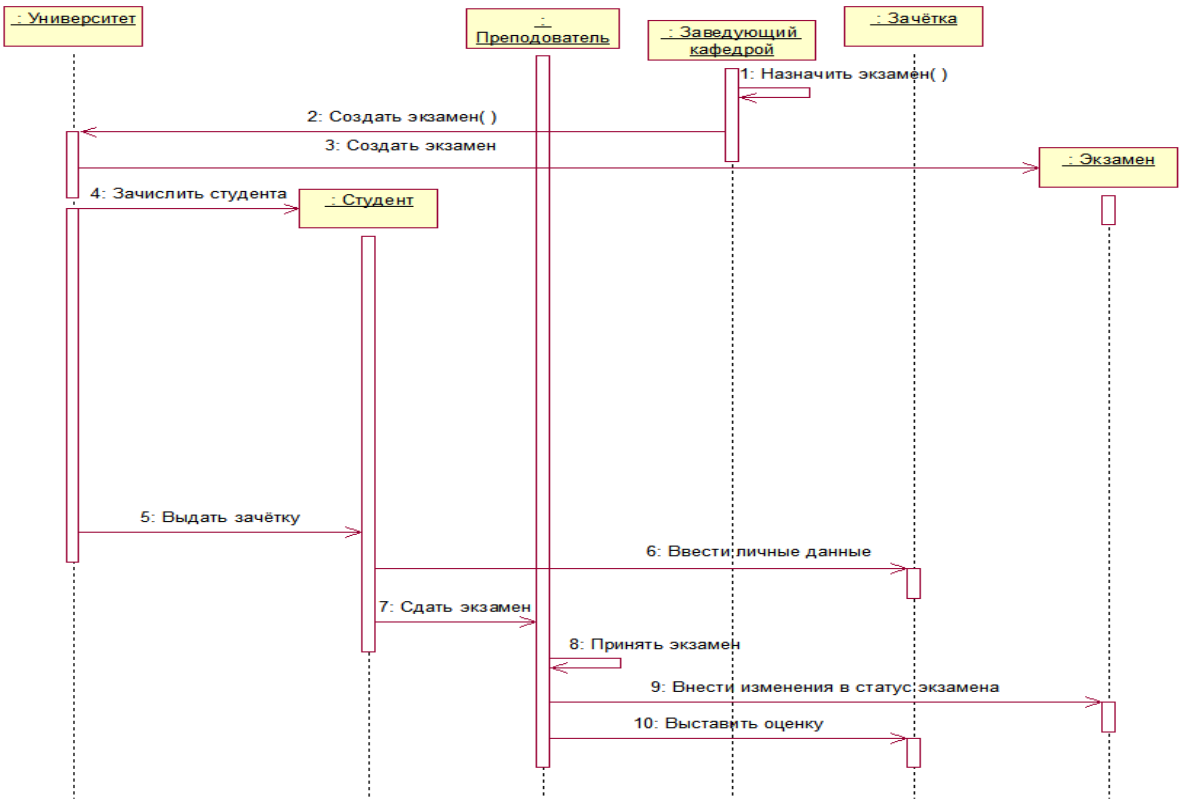

Рисунок 141 – Диаграмма последовательности

Вариант 11

Добавить класс заготовка и обеспечить его взаимодействие с классом отвёртка. Объект класса отвёртка при вызове метода закрутить изменяет статус готовности объекта класса Изделие.

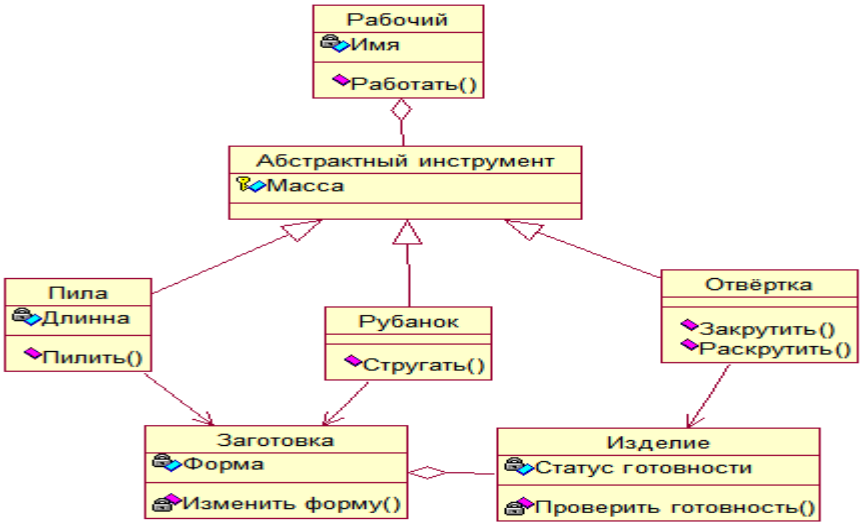

Рисунок 142 – Диаграмма классов

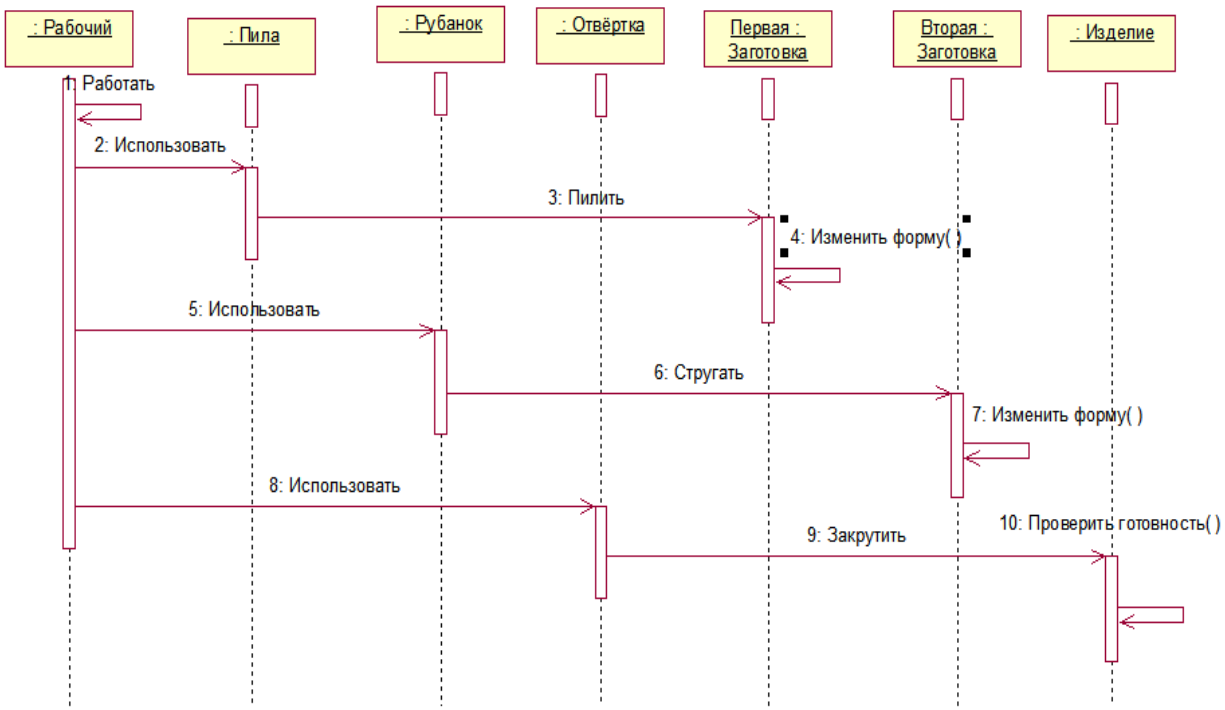

Рисунок 143 – Диаграмма последовательности

Вариант 12

Для классов кофе и сахар сделать родительский класс сыпучий предмет. Добавить наследующий от него класс Чай. Добавить класс молоко. Обеспечить класс Чайник методом Нагреть. Классу студент добавить метод Сделать кофе с молоком() и Сделать чай().

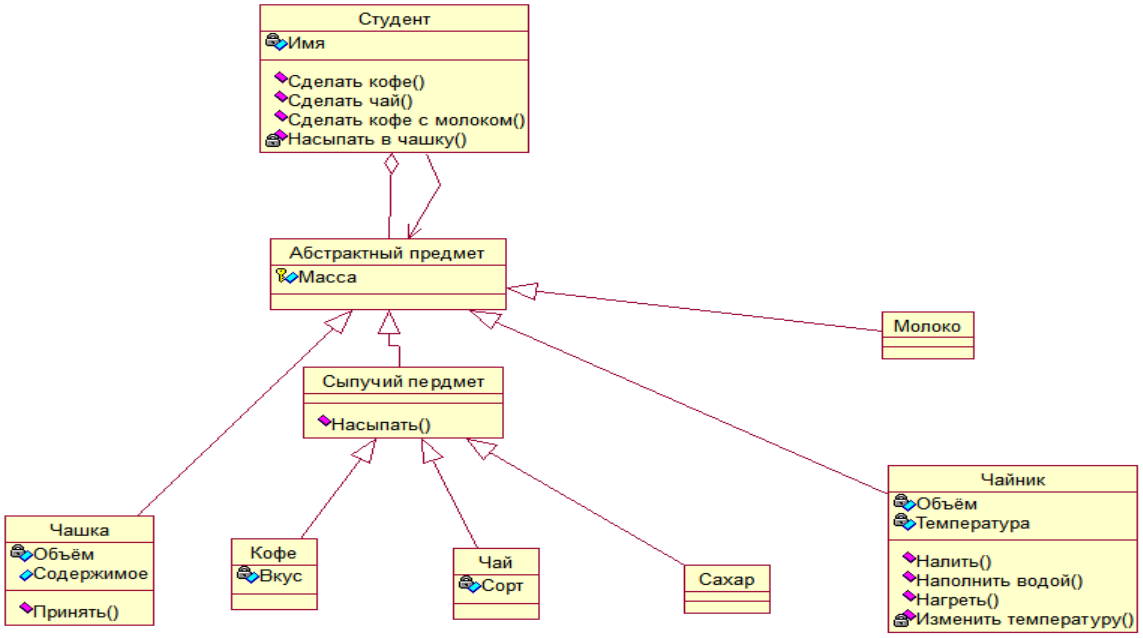

Рисунок 144 – Диаграмма классов

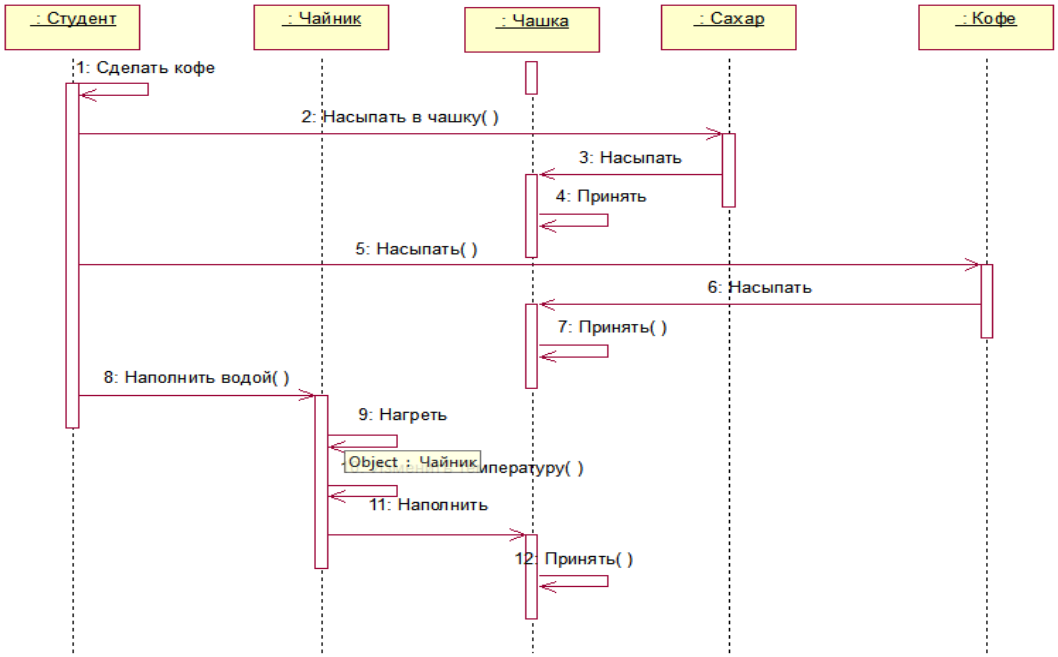

Рисунок 145 – Диаграмма последовательности

Вариант 13

Добавить класс Абстрактный питомец, родительский класс для класса Абстрактная собака и Кошка. Классу хозяин добавить методы Завести собаку() и Завести кошку() создающие объекты классов, наследующих от класса Абстрактная собака и объекты класса Кошка. Классу Хозяин добавить метод Выгулять собаку().

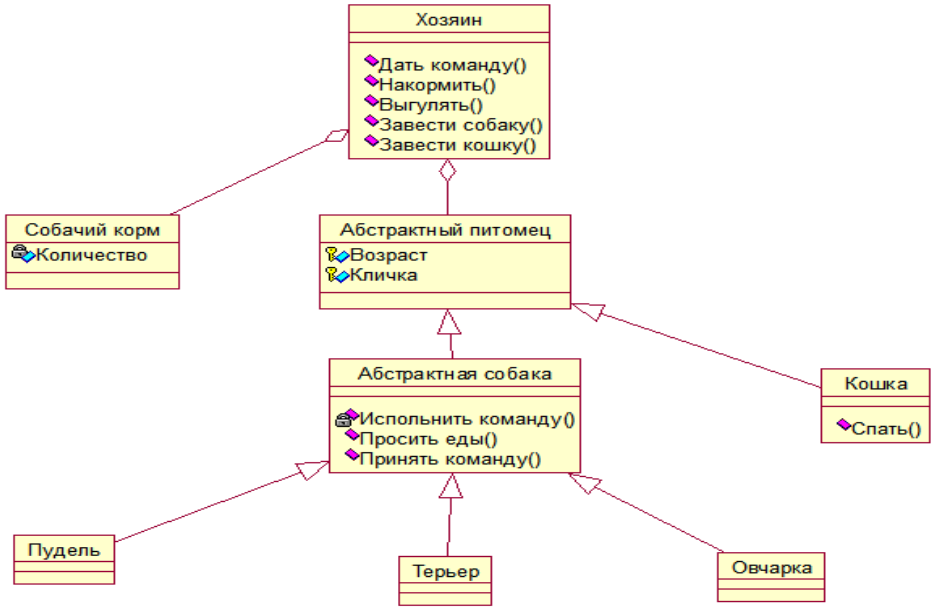

Рисунок 146 – Диаграмма классов

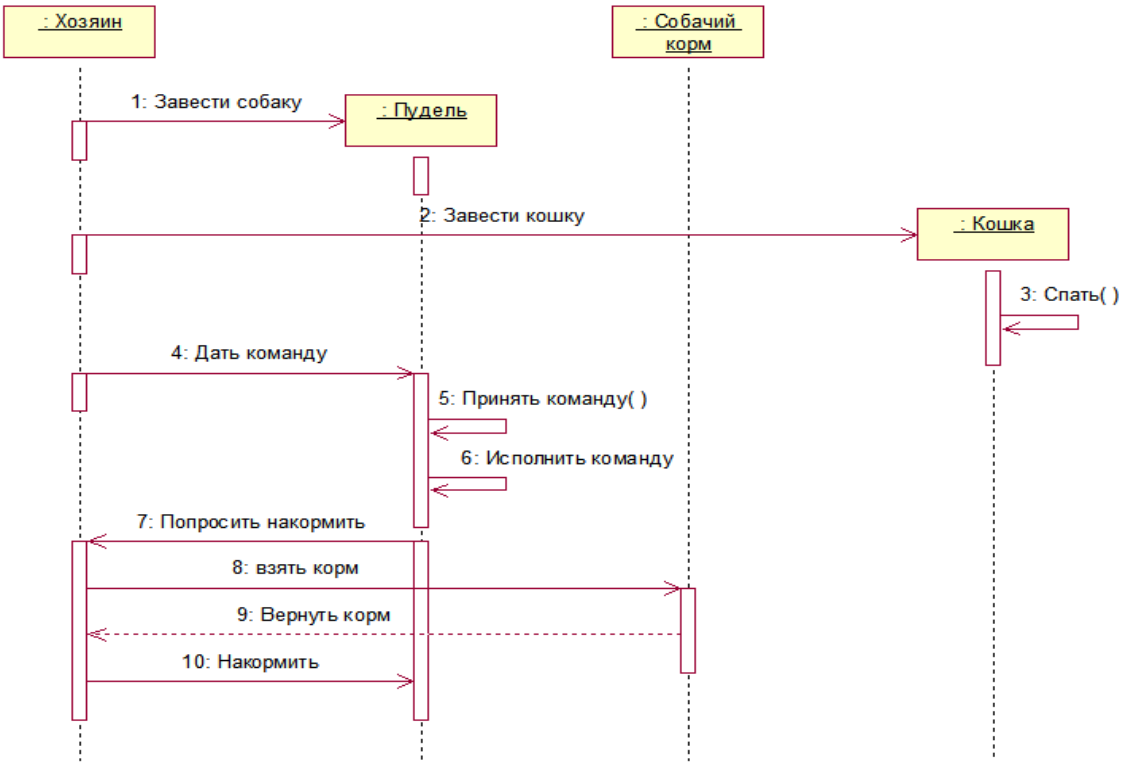

Рисунок 147 – Диаграмма последовательности

Вариант 14

Добавить классы Ангина и Грипп, наследующие от класса Абстрактная болезнь. Пациенту добавить горло и лёгкие. Также добавить метод принять лекарство () разрушающий Болезнь. Добавить класс Лекарство.

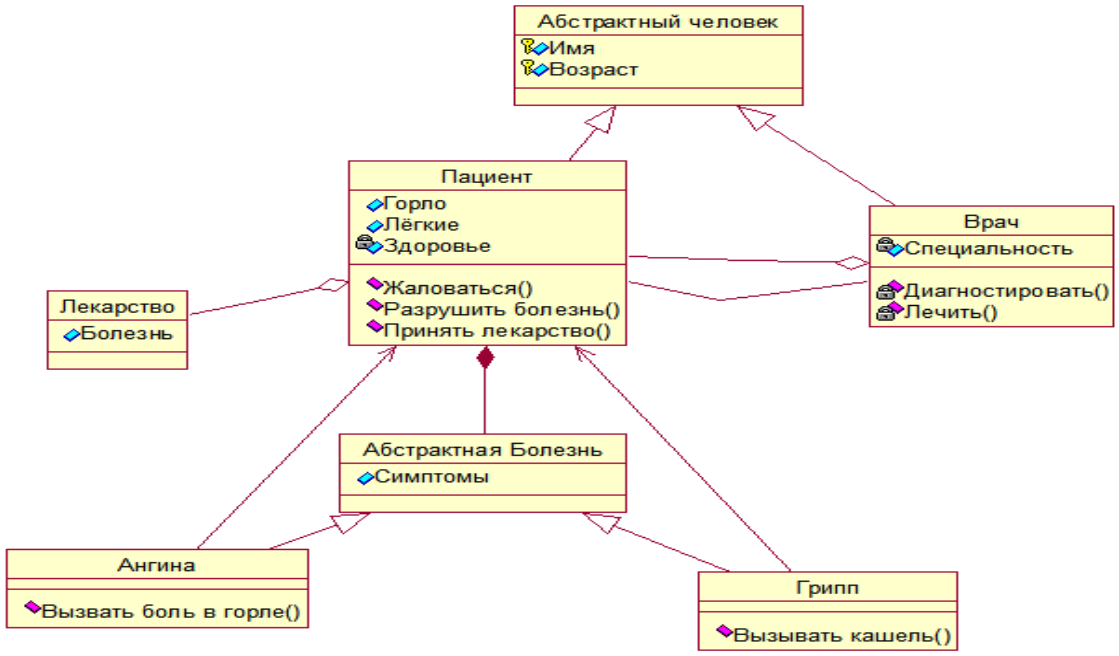

Рисунок 148 – Диаграмма классов

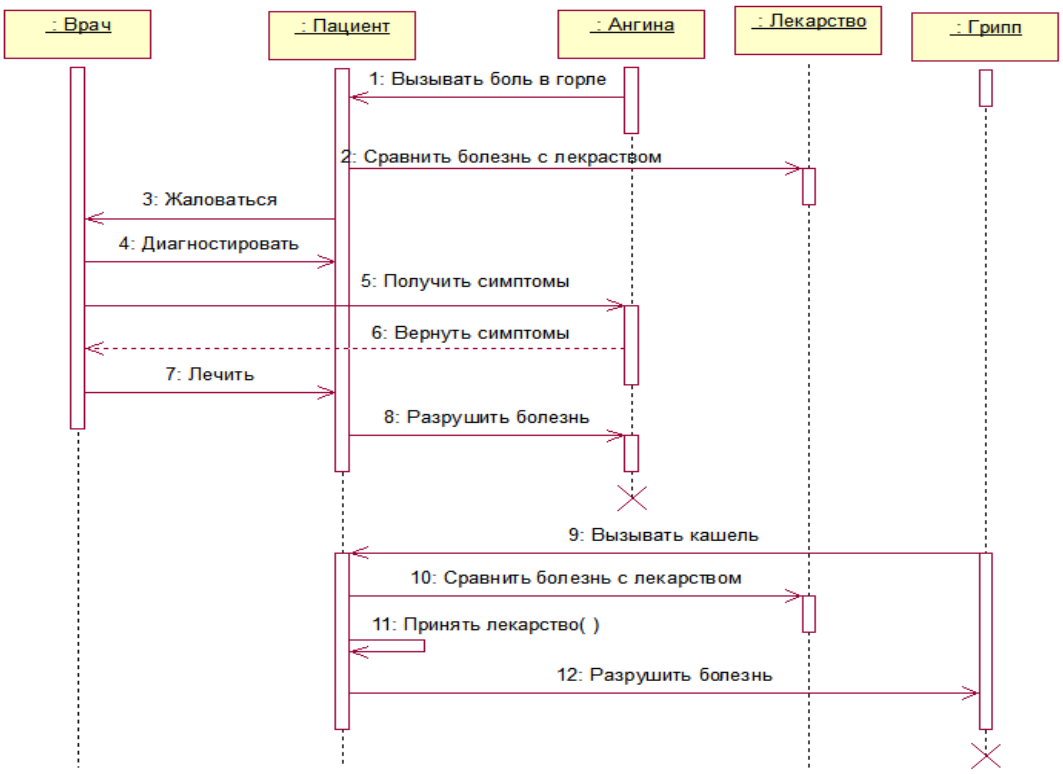

Рисунок 149 – Диаграмма последовательности

Вариант 15

Добавить класс Гладиолус. Добавить класс Лепесток, принадлежащий классу Бутон и класс Лист, принадлежащий классу Цветок. Обеспечить создание Объектов Лепесток одновременно с объектами Бутон. Реализовать метод Засохнуть() у класса Цветок.

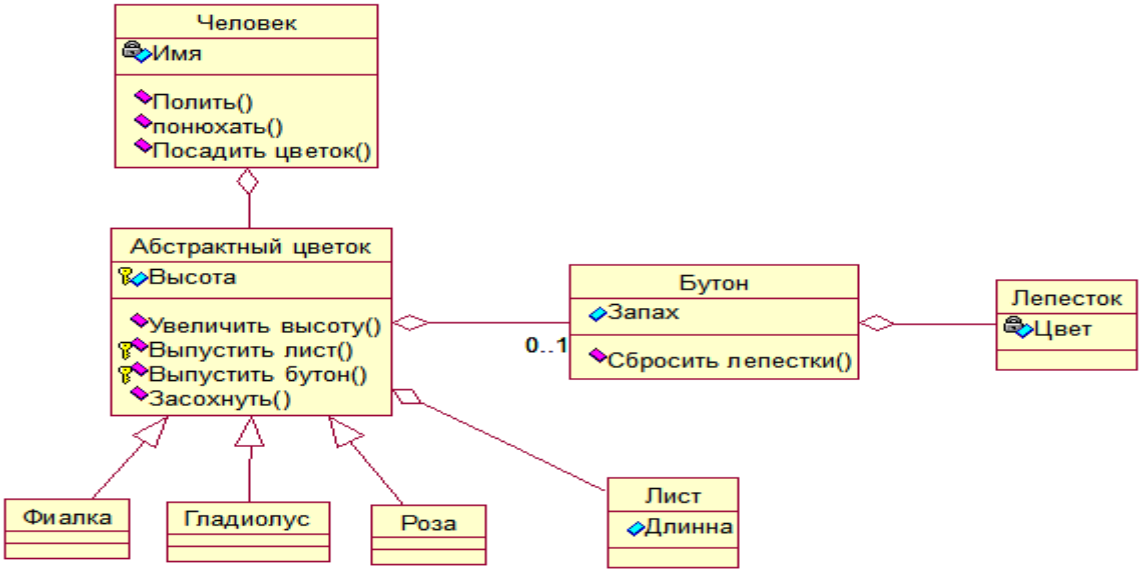

Рисунок 150 – Диаграмма классов

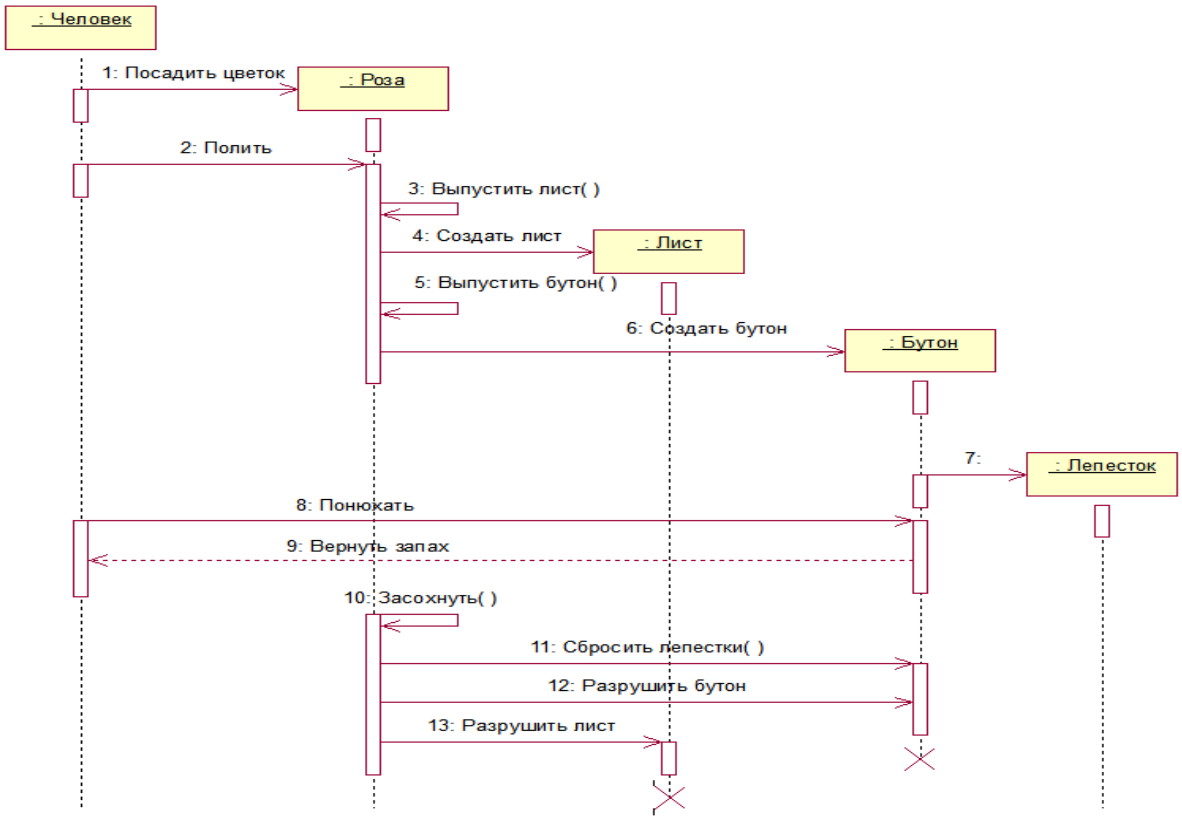

Рисунок 151 – Диаграмма последовательности

Вариант 16

Добавить класс Критик с Методом написать рецензию() и класс Рецензия, который он создаёт. Также создать классы Роман и Рассказ, наследующие от класса Абстрактная книга.

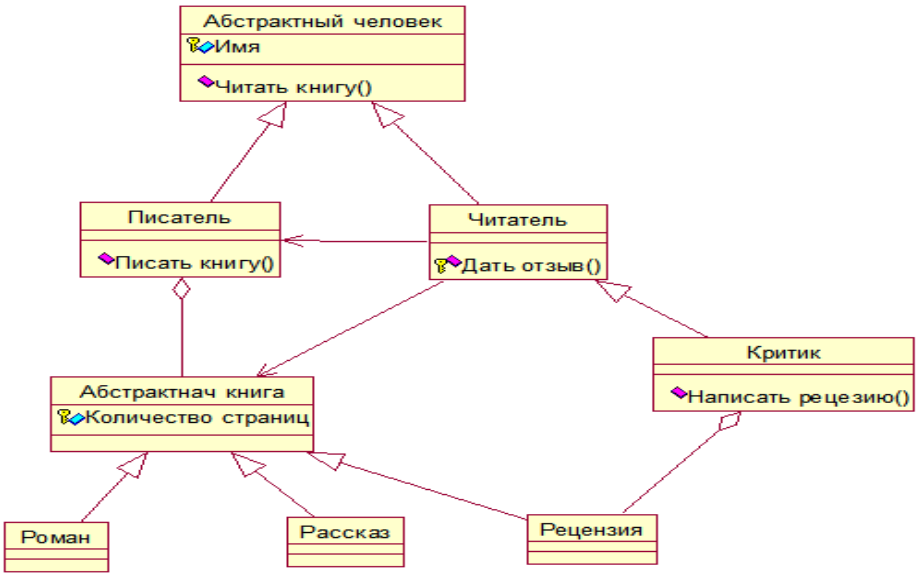

Рисунок 152 – Диаграмма классов

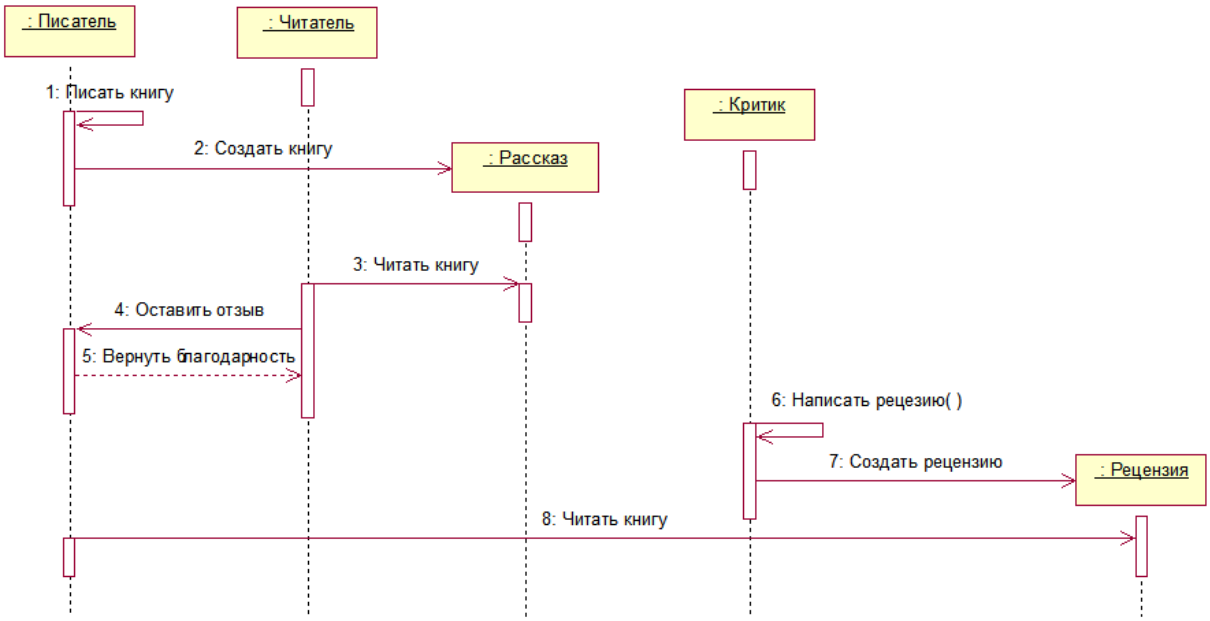

Рисунок 153 – Диаграмма последовательности

Вариант 17

Создать классы Пассажир эконом класса и Пассажир первого класса, наследующие от класса Пассажир. Добавить взаимодействие класса Пассажир первого класса с Классом Стюардесса.

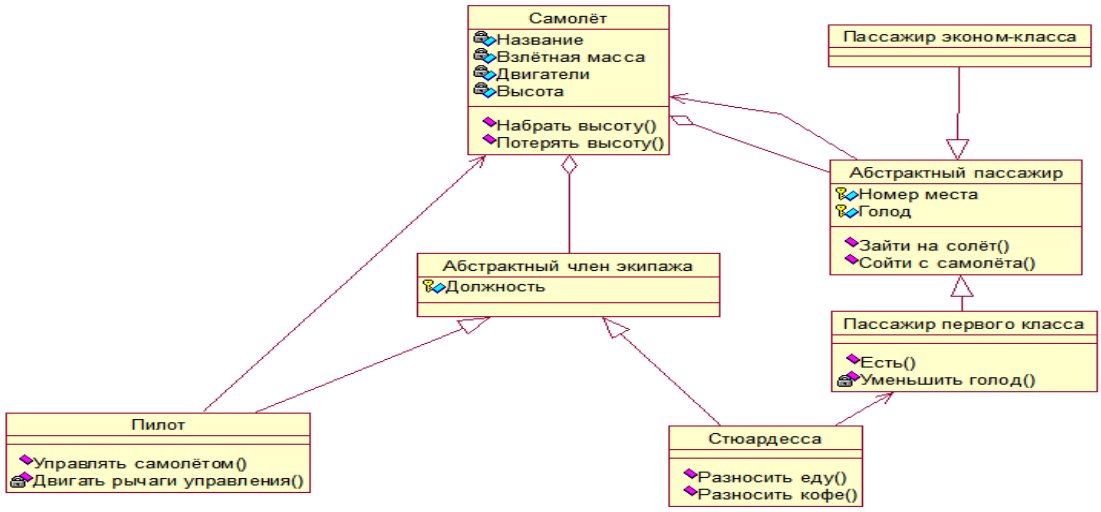

Рисунок 154 – Диаграмма классов

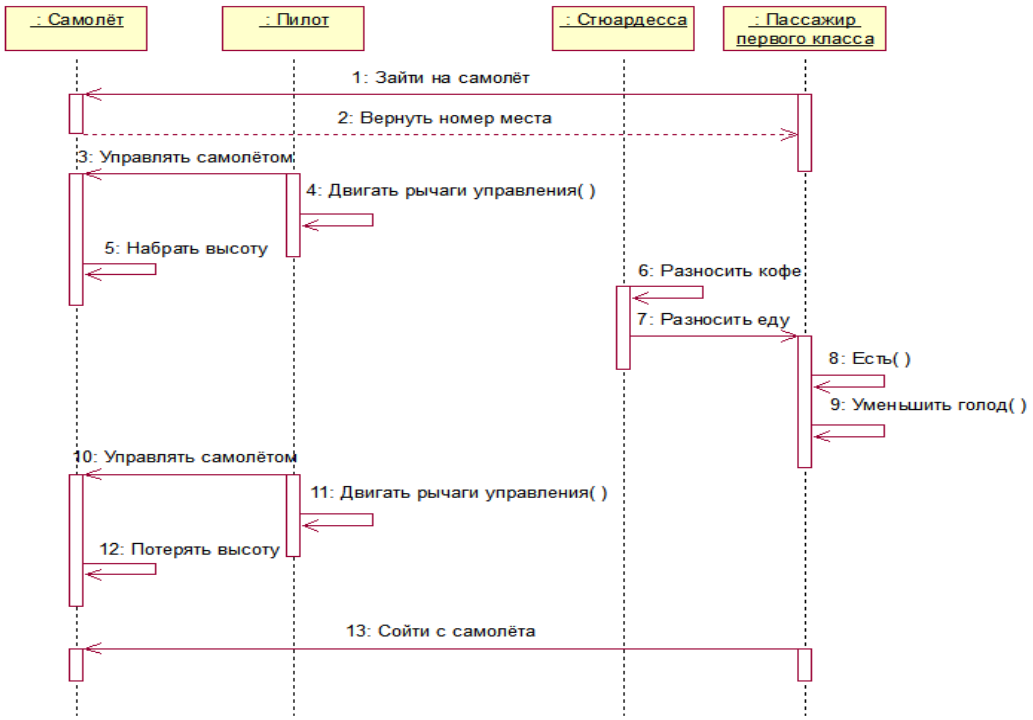

Рисунок 155 – Диаграмма последовательности

Вариант 18

Добавить класс Банк и организовать его взаимодействие с классом Банкомат. В частности реализовать запрос денег Банкоматом из Банка.

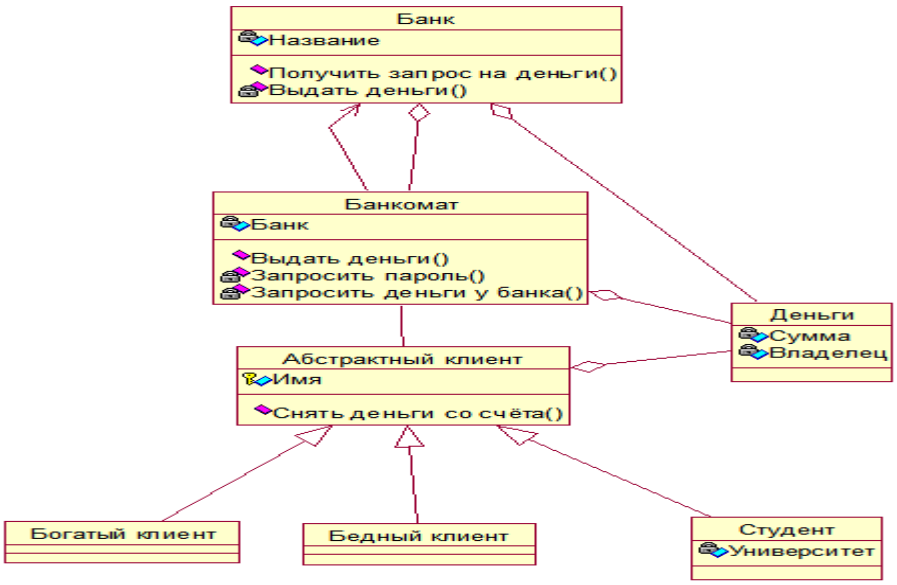

Рисунок 156 – Диаграмма классов

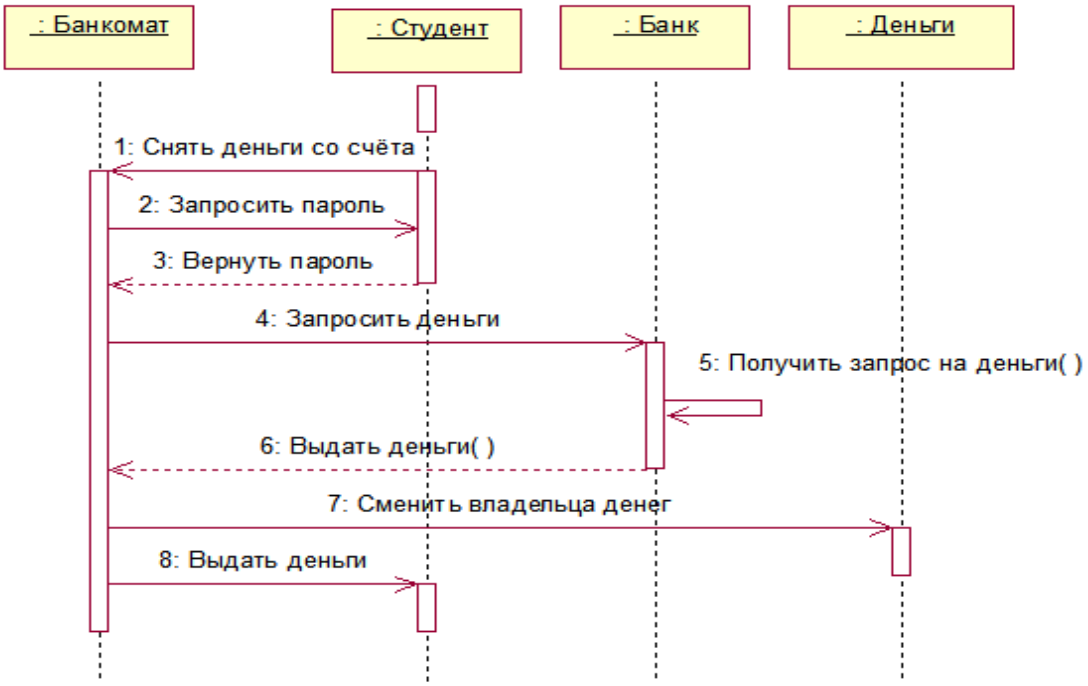

Рисунок 157 – Диаграмма последовательности

Вариант 19

Классу Корабль добавить метод Вести пушечный огонь() и реализовать классы Зенитная пушка и Пушка главного калибра.

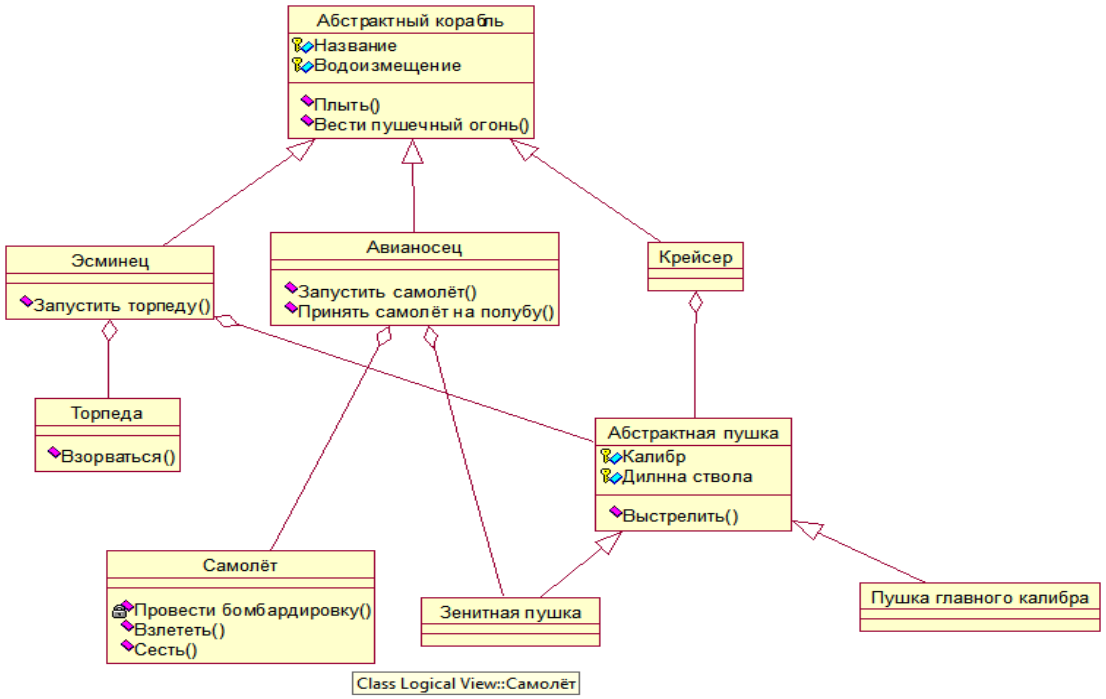

Рисунок 158 – Диаграмма классов

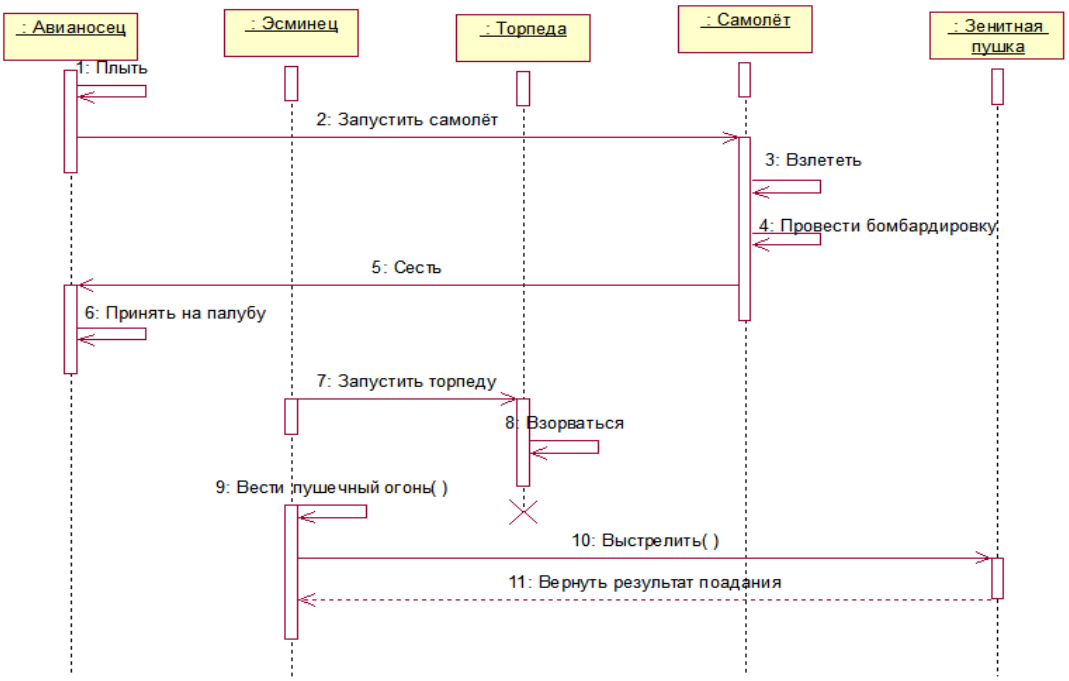

Рисунок 159 – Диаграмма последовательности

Вариант 20

Преобразовать класс Бумага. Добавить в него поле содержимое, которое будет хранить написанное на бумаге. Добавить класс Ксерокс, который будет создавать копии с бумаги.

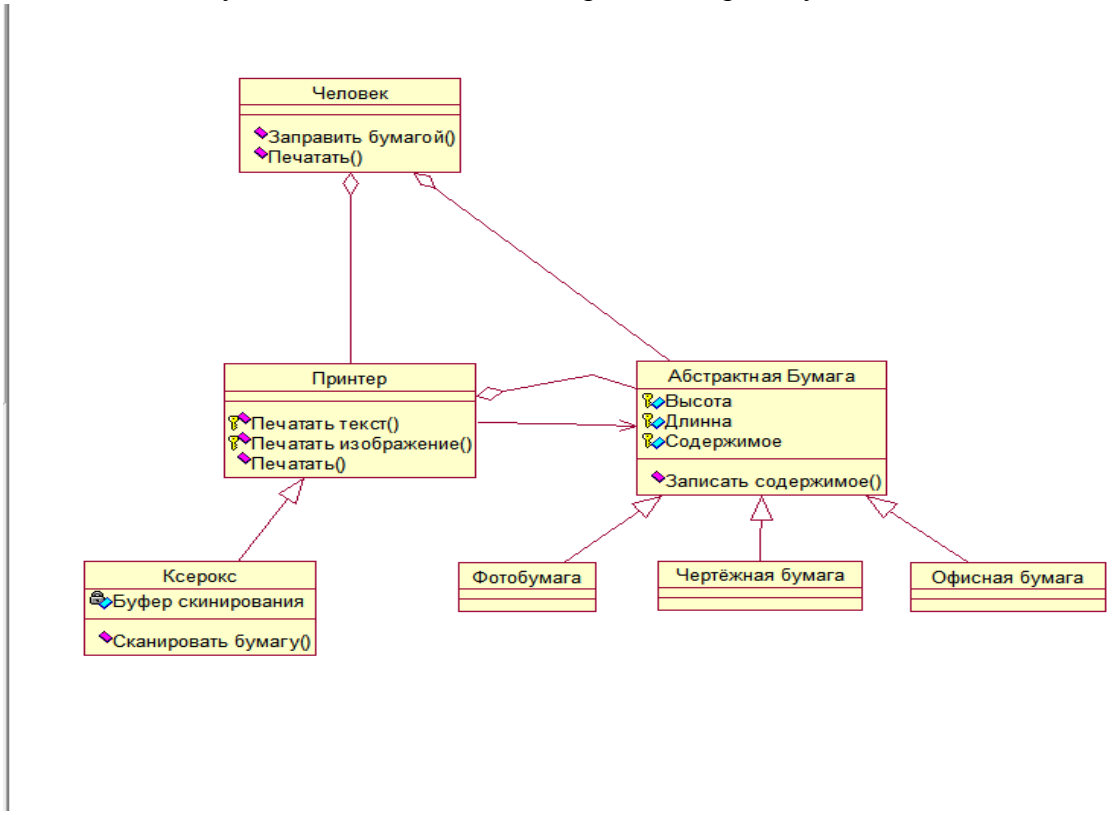

Рисунок 160 – Диаграмма классов

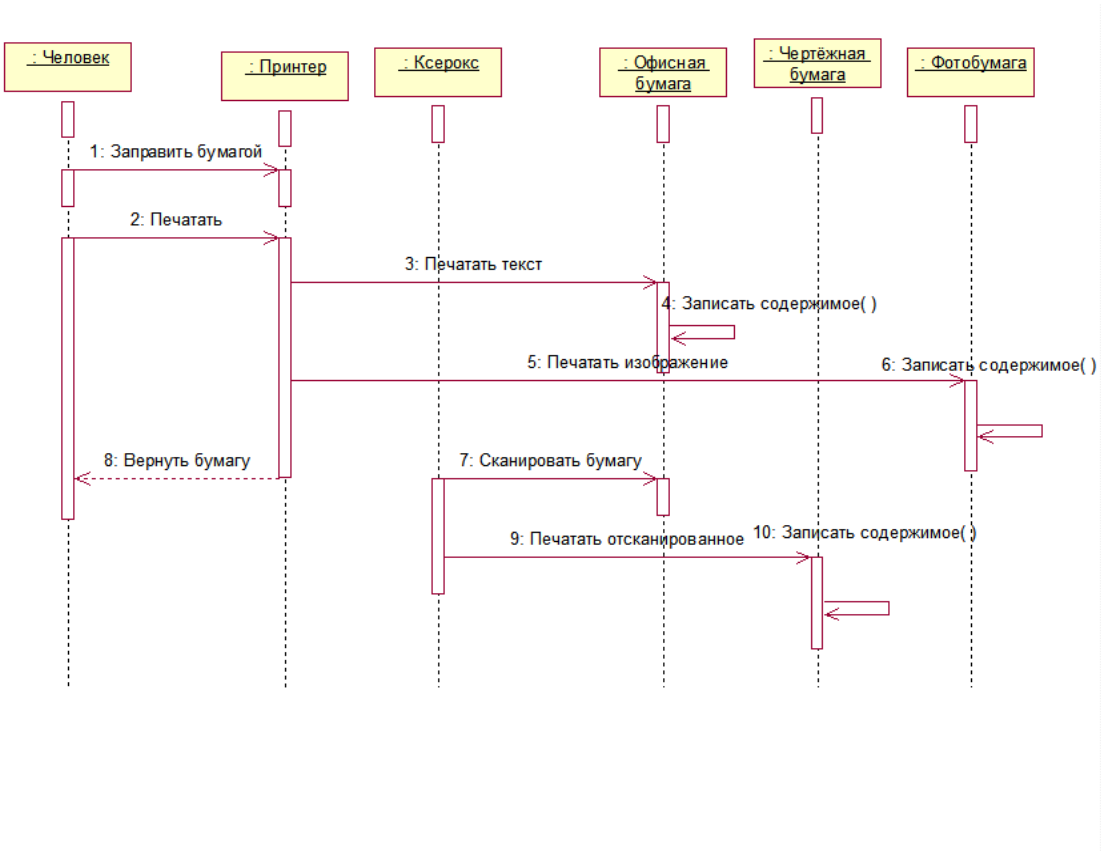

Рисунок 161 – Диаграмма последовательности

Вариант 21

Добавить класс Квартира, хранящий в себе класс Адрес и организовать взаимодействие с ним: вселение жильцов в комнату и выселение.

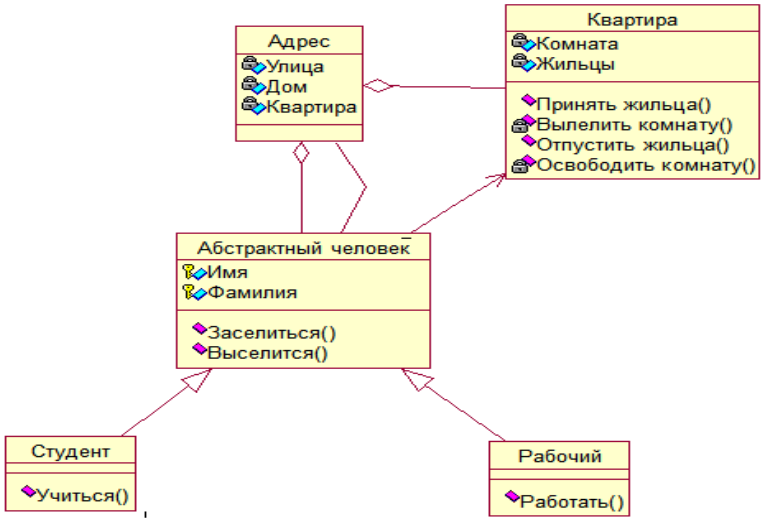

Рисунок 162 – Диаграмма классов

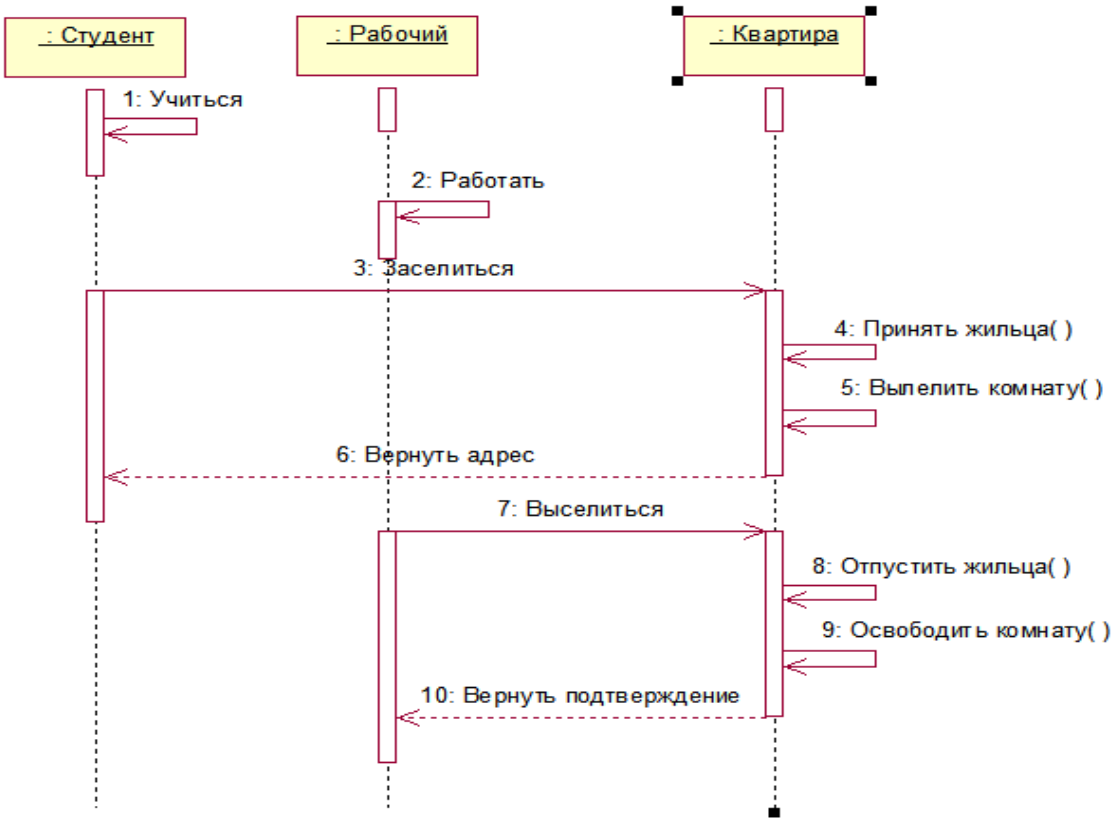

Рисунок 163 – Диаграмма последовательности

Вариант 22

Добавить классы Тигр и Фламинго по аналогии с классами Слон и Страус. Также добавить класс Зритель. Его метод посмотреть на животное() должен получать копию класса Животное и отображать на экран информацию о классе.

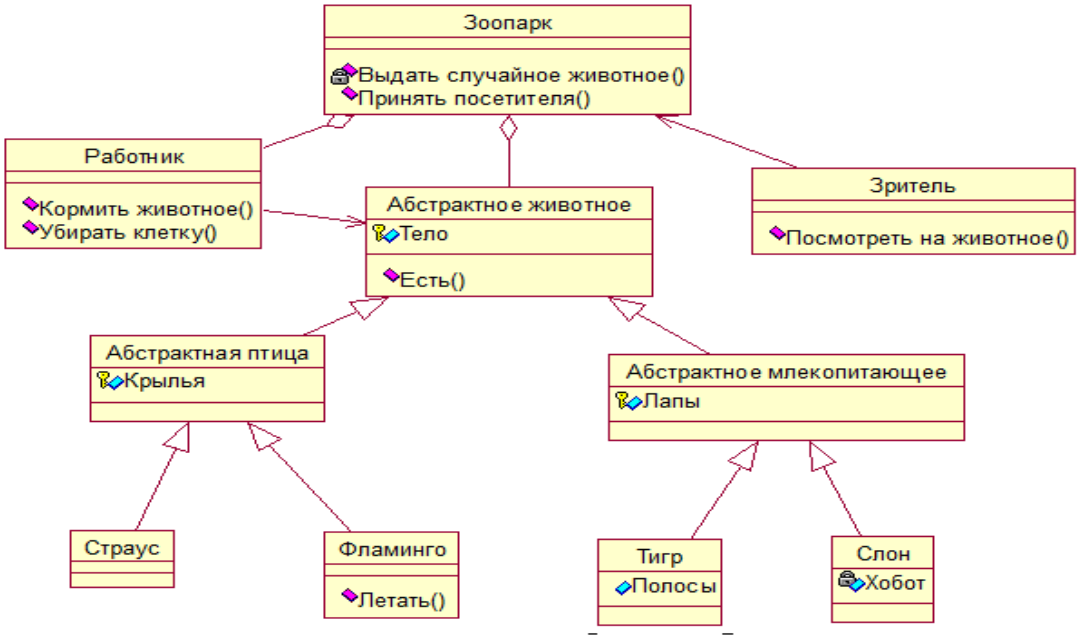

Рисунок 164 – Диаграмма классов

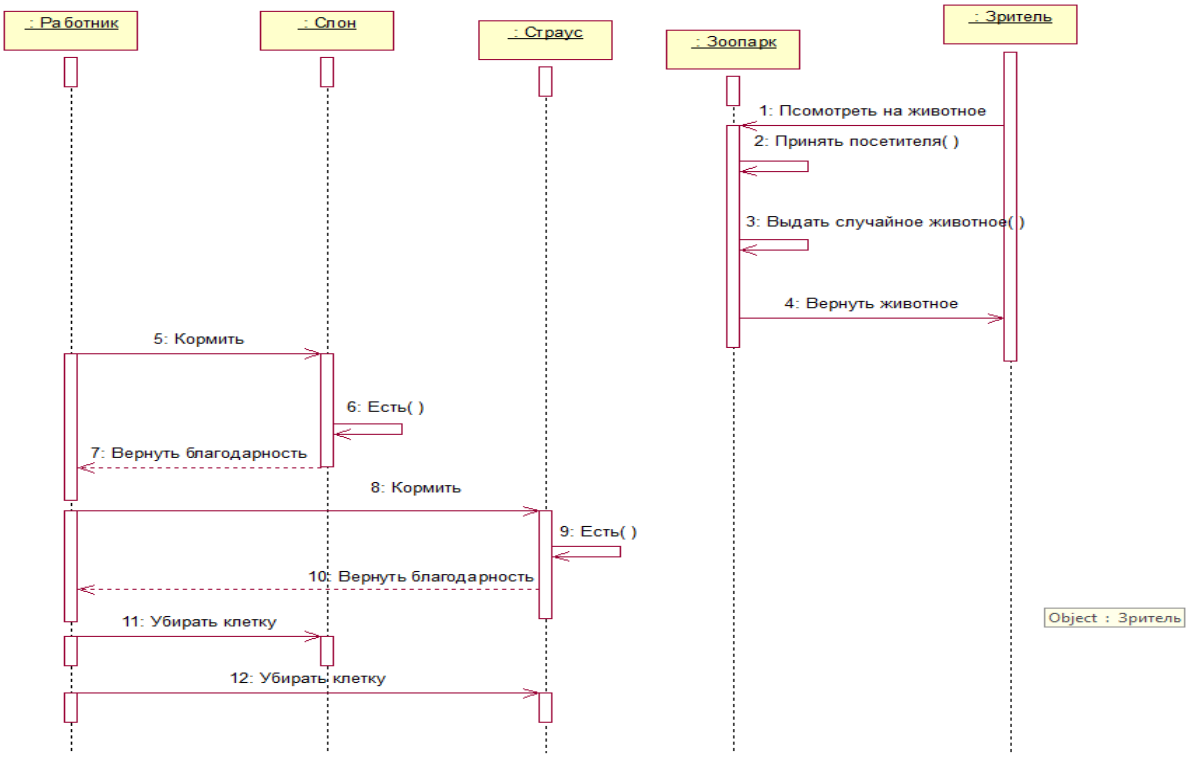

Рисунок 165 – Диаграмма последовательности

Вариант 23

Классу Университет добавить методы Провести лекцию(), Провести семинар(), Провести лабораторный практикум(). Эти методы должны менять поля знания и оценки класса Студент. Добавить класс Детский сад.

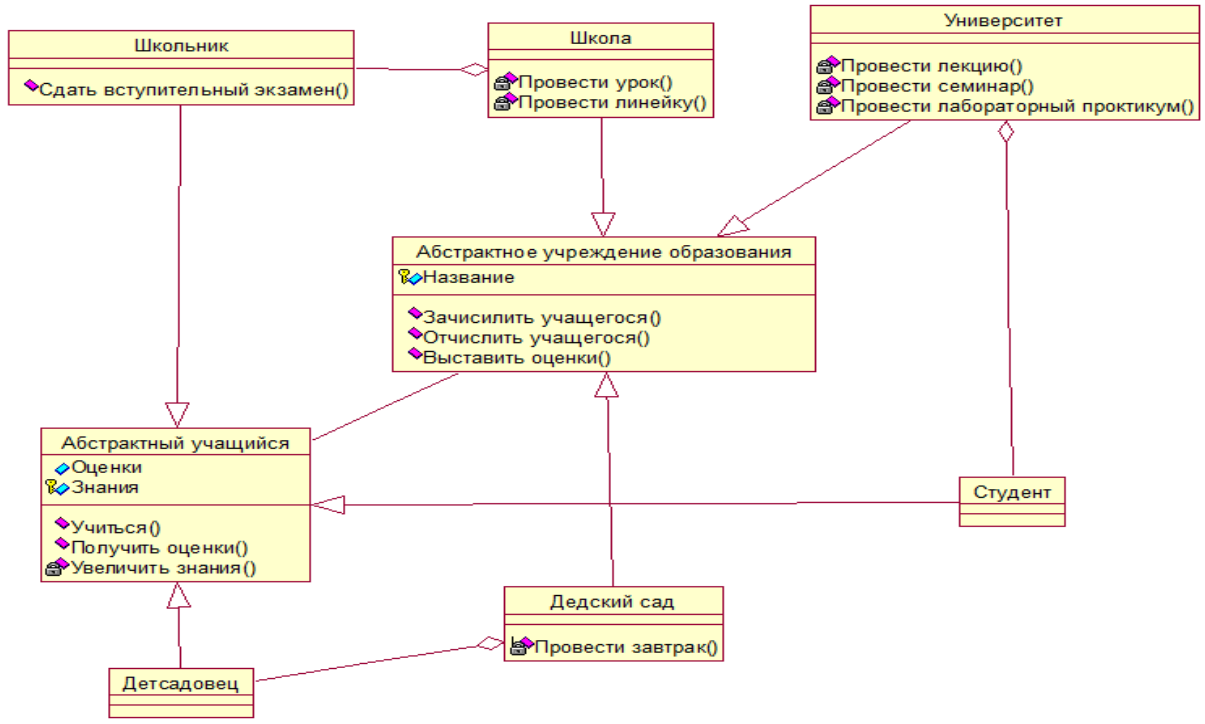

Рисунок 166 – Диаграмма классов

Рисунок 167 – Диаграмма последовательности

Вариант 24

Добавить класс Магазин и организовать взаимодействие класса Человек с ним с помощью метода купить жидкость(). Также добавить классы Чайник, Чай, Вода. И добавить классу Сосуд метод Перелить жидкость().

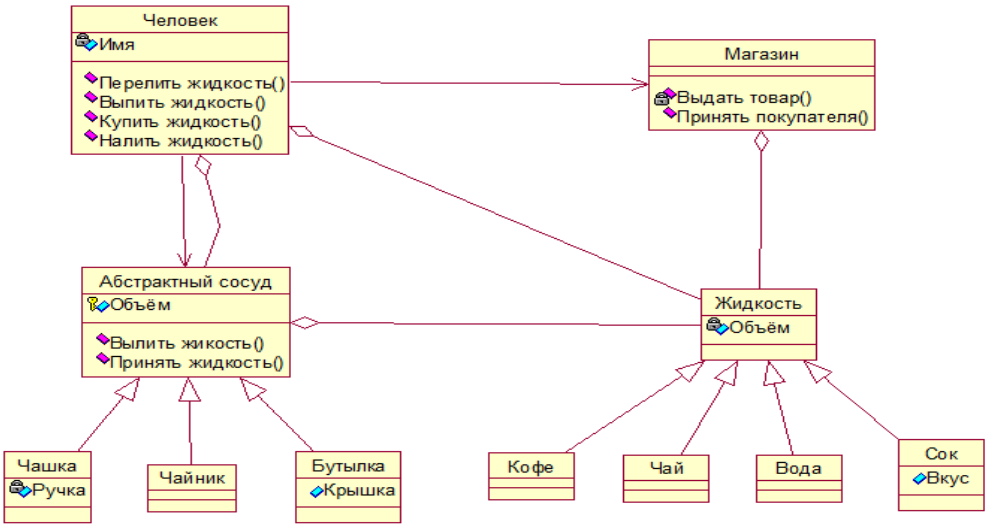

Рисунок 168 – Диаграмма классов

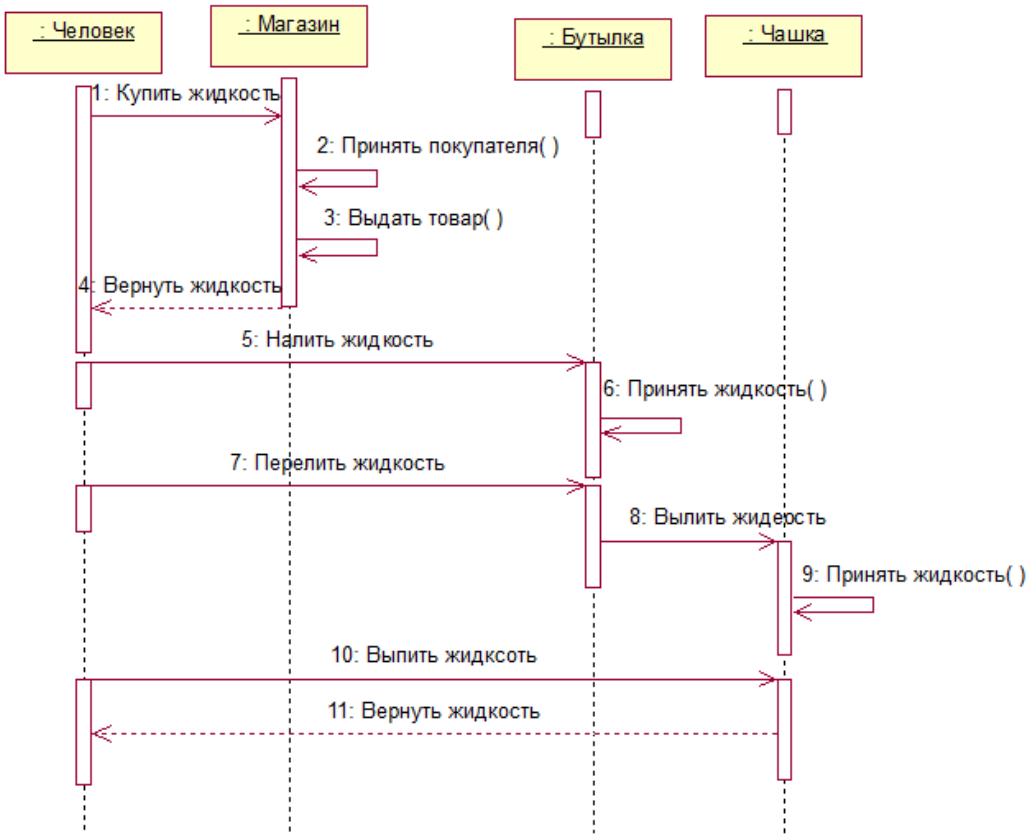

Рисунок 169 – Диаграмма последовательности

Вариант 25

Добавить класс Пакет и организовать возможность помещать туда товар. Добавить классу Тележка метод Выдать товар(). Добавить класс Молоко.

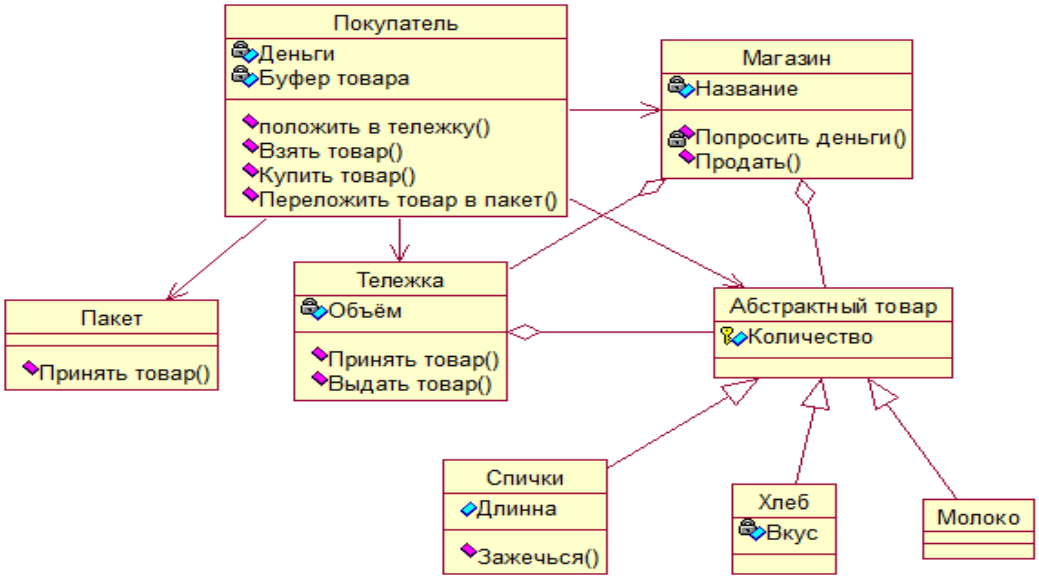

Рисунок 170 – Диаграмма классов

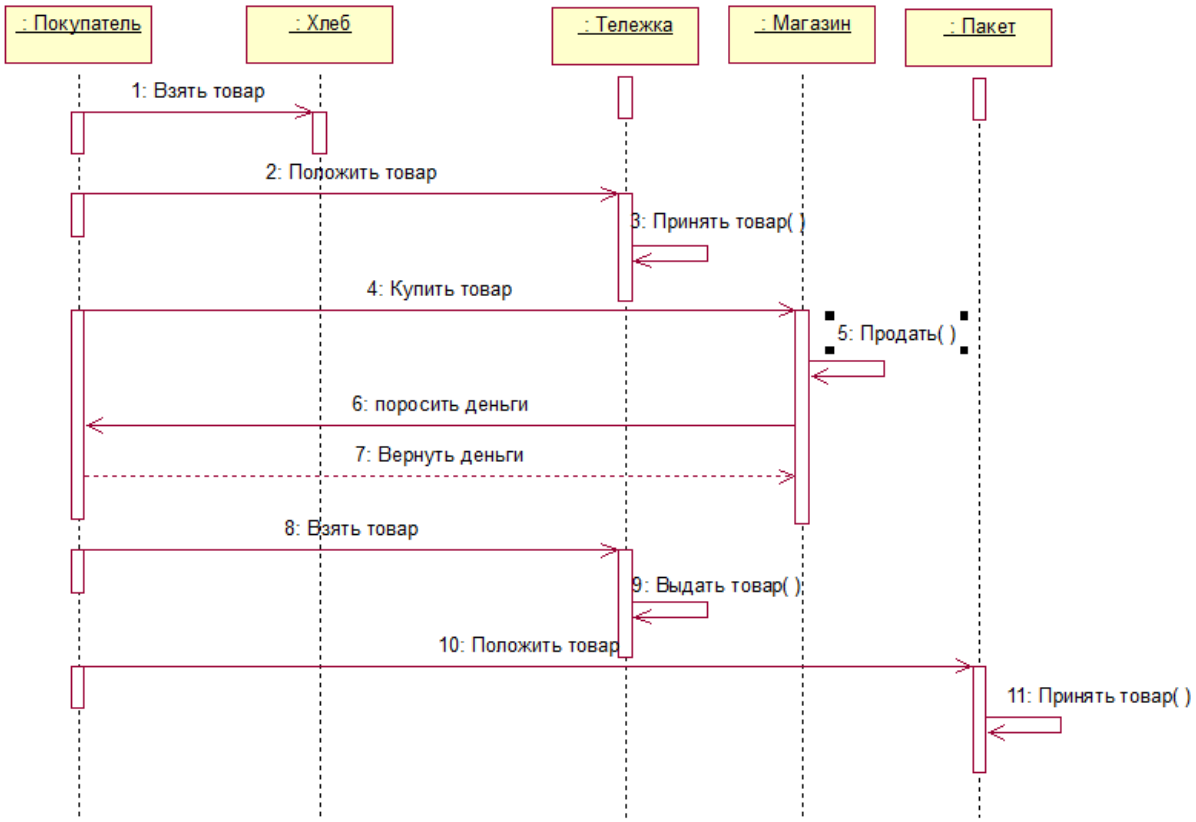

Рисунок 171 – Диаграмма последовательности

Вариант 26

Добавить классы, наследующие от класса одежда, а именно Штаны и Рубашка. Добавить классу Портной фабричные методы, создающие эти классы. Добавить классу Ткань поле количество и организовать его работу.

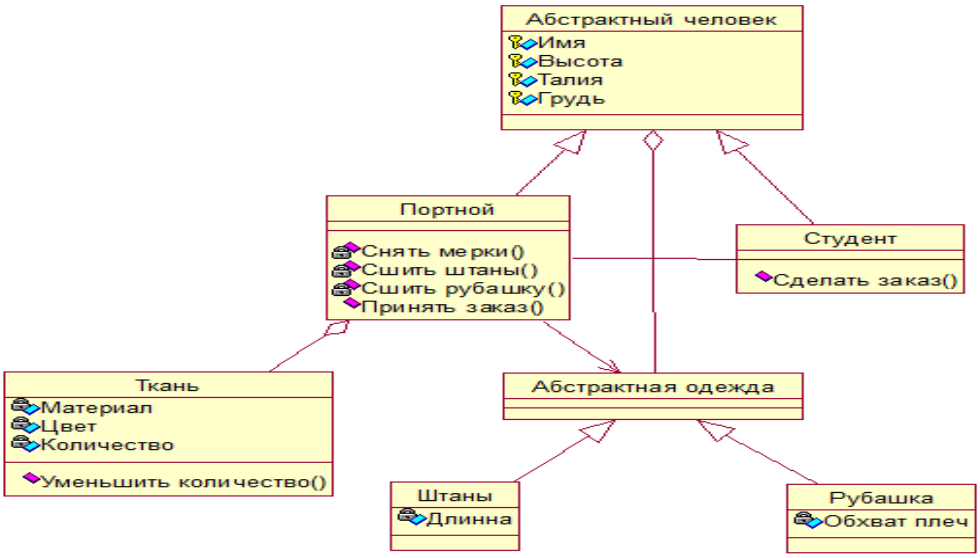

Рисунок 172 – Диаграмма классов

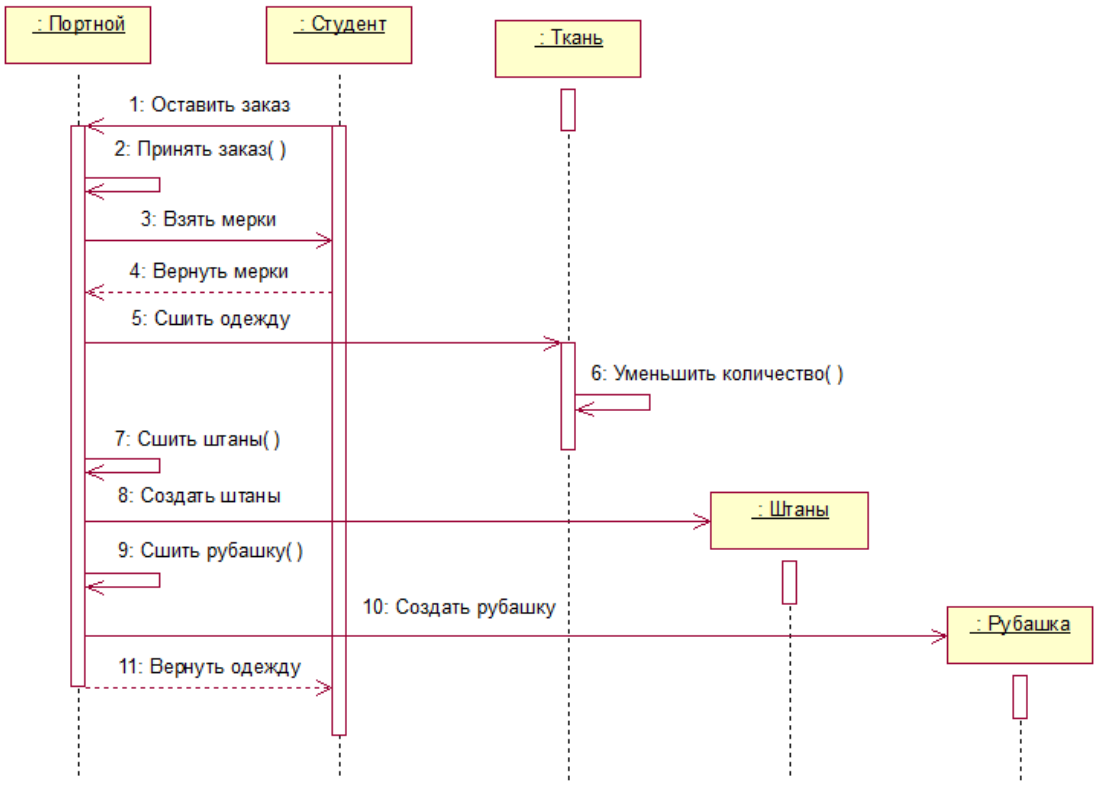

Рисунок 173 – Диаграмма последовательности

Вариант 27

Добавить монстров Кабан и Оборотень. Организовать метод убийства монстра. Переделать работу классов так, чтобы монстры сообщали квесту о своём убийстве.

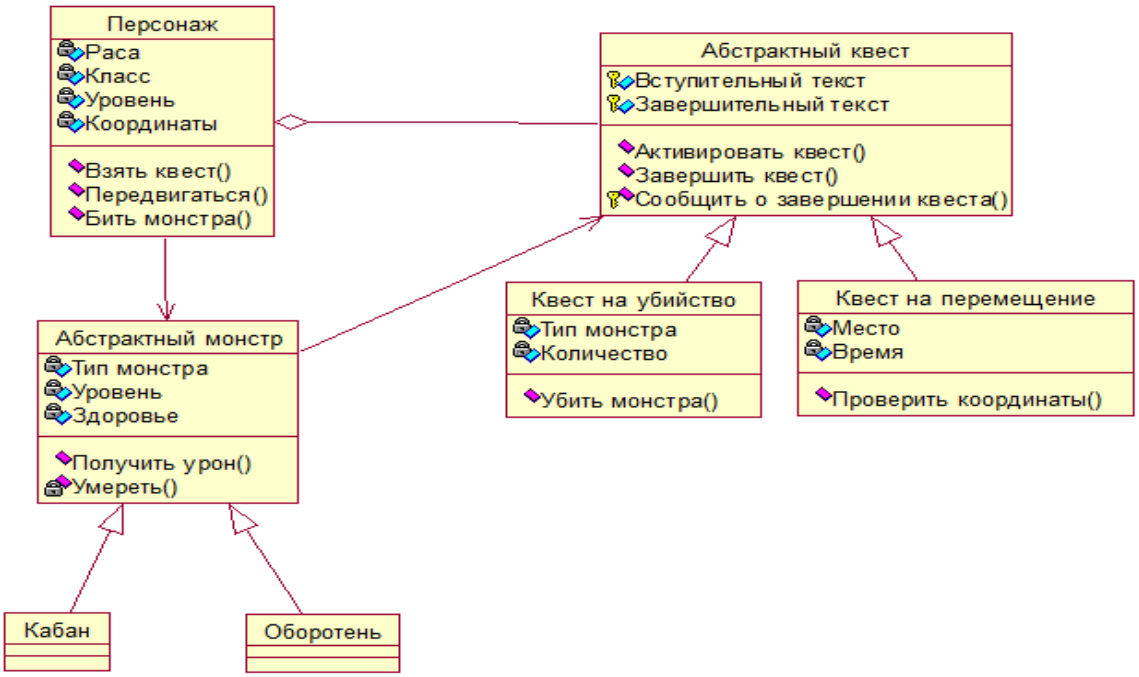

Рисунок 174 – Диаграмма классов

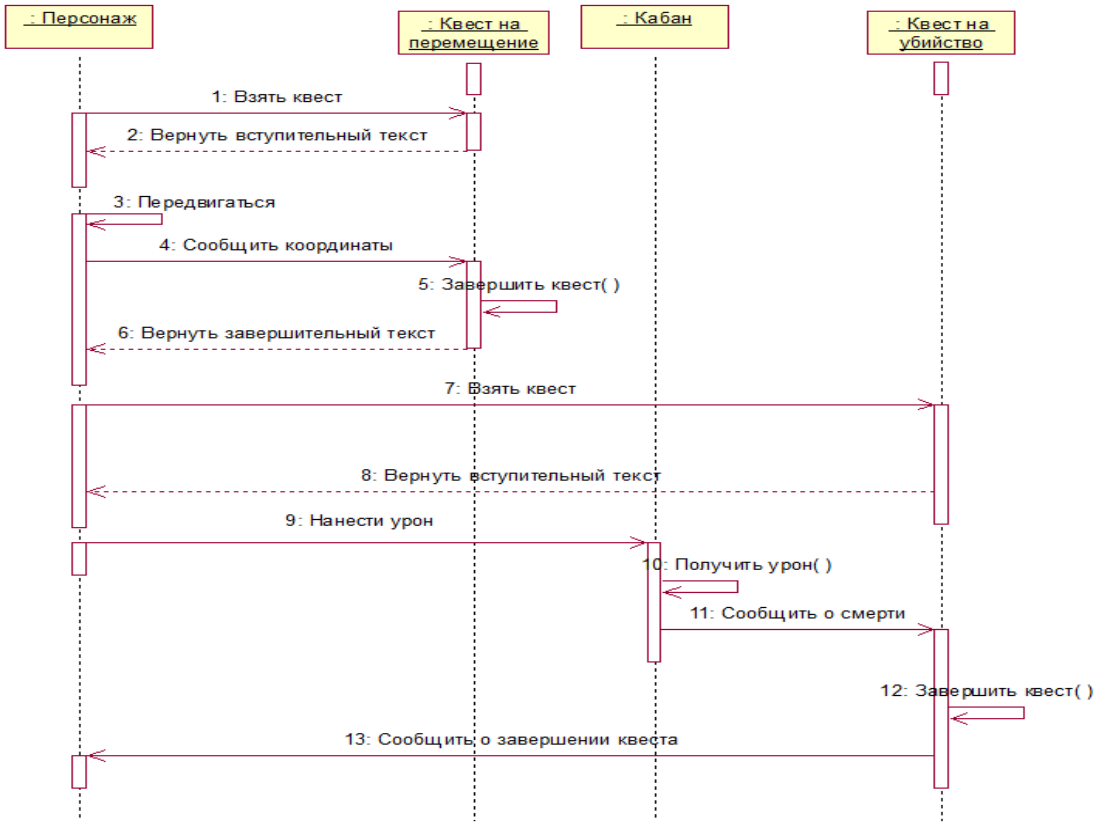

Рисунок 175 – Диаграмма последовательности

Вариант 28

Добавить класс База пользователей, хранящий данные о Зарегистрированных пользователях. Обеспечить управление этим классом с помощью класса Администратор.

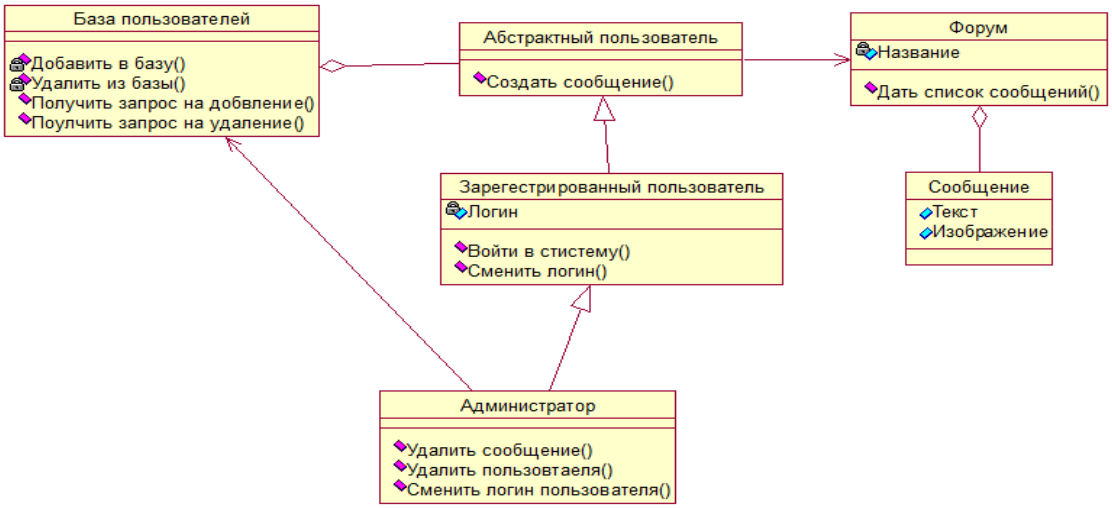

Рисунок 176 – Диаграмма классов

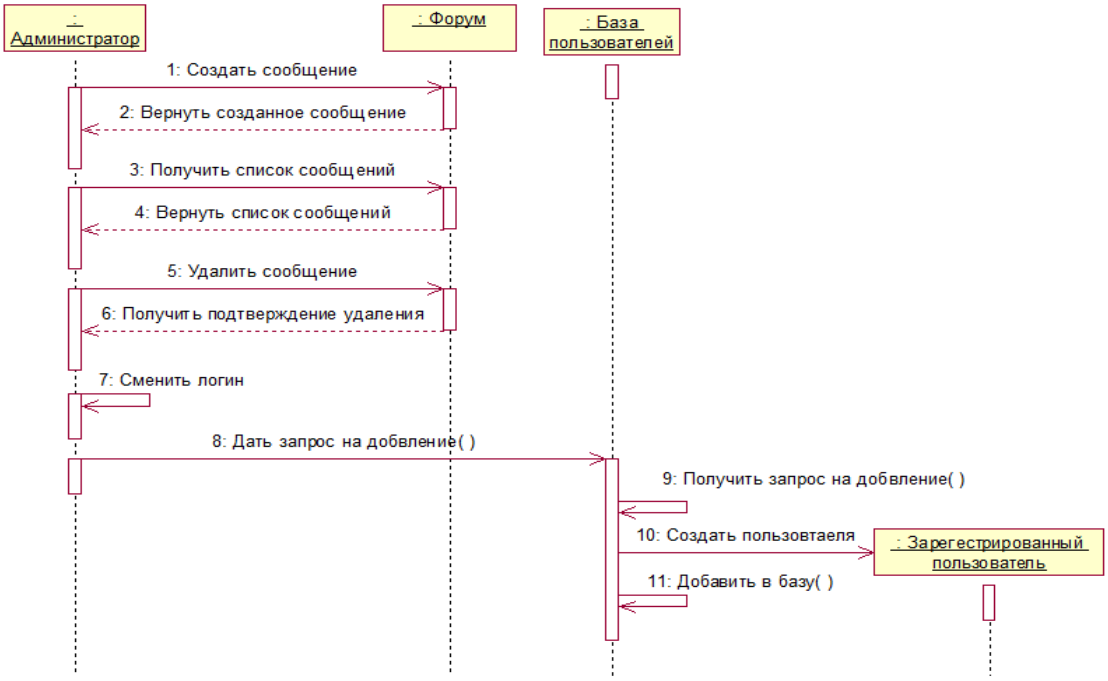

Рисунок 177 – Диаграмма последовательности

Вариант 29

Добавить возможность редактировать информацию о пользователе. В класс Пользователь добавить поля Почта, Ник. Обеспечить работу с ними.

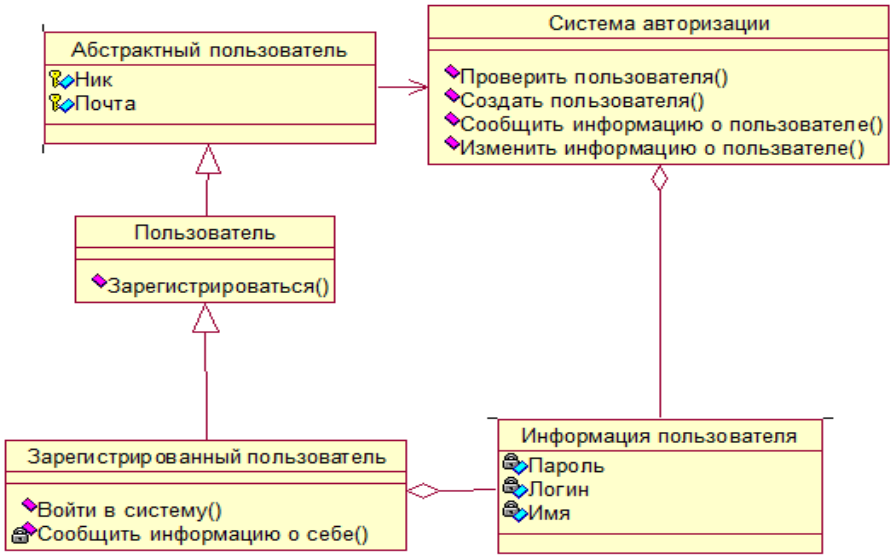

Рисунок 178 – Диаграмма классов

Рисунок 179 – Диаграмма последовательности

Вариант 30

Добавить класс Плейлист, хранящий список видео. Класс должен выдавать следующее видео из списка при вызове метода Воспроизвести плейлист(). Добавить класс Музыкальное видео.

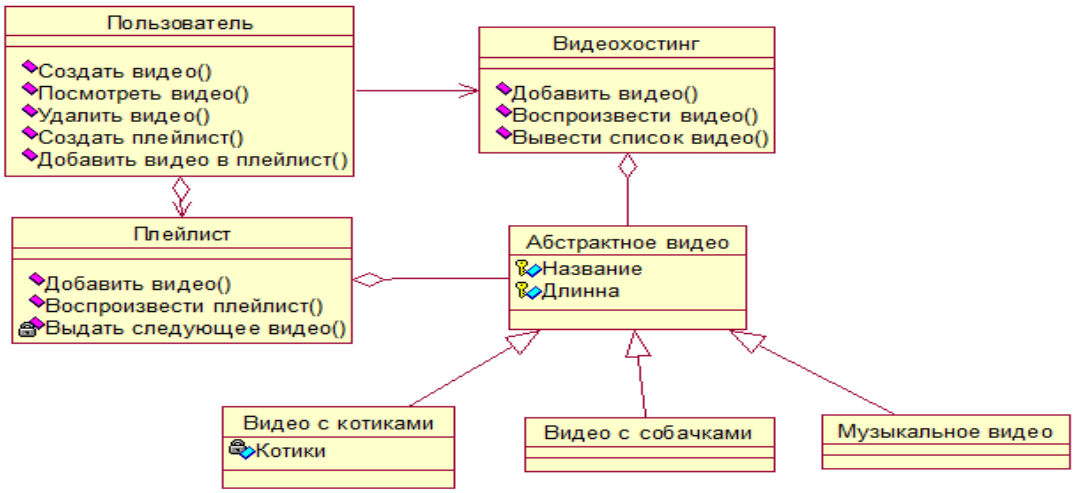

Рисунок 180 – Диаграмма классов

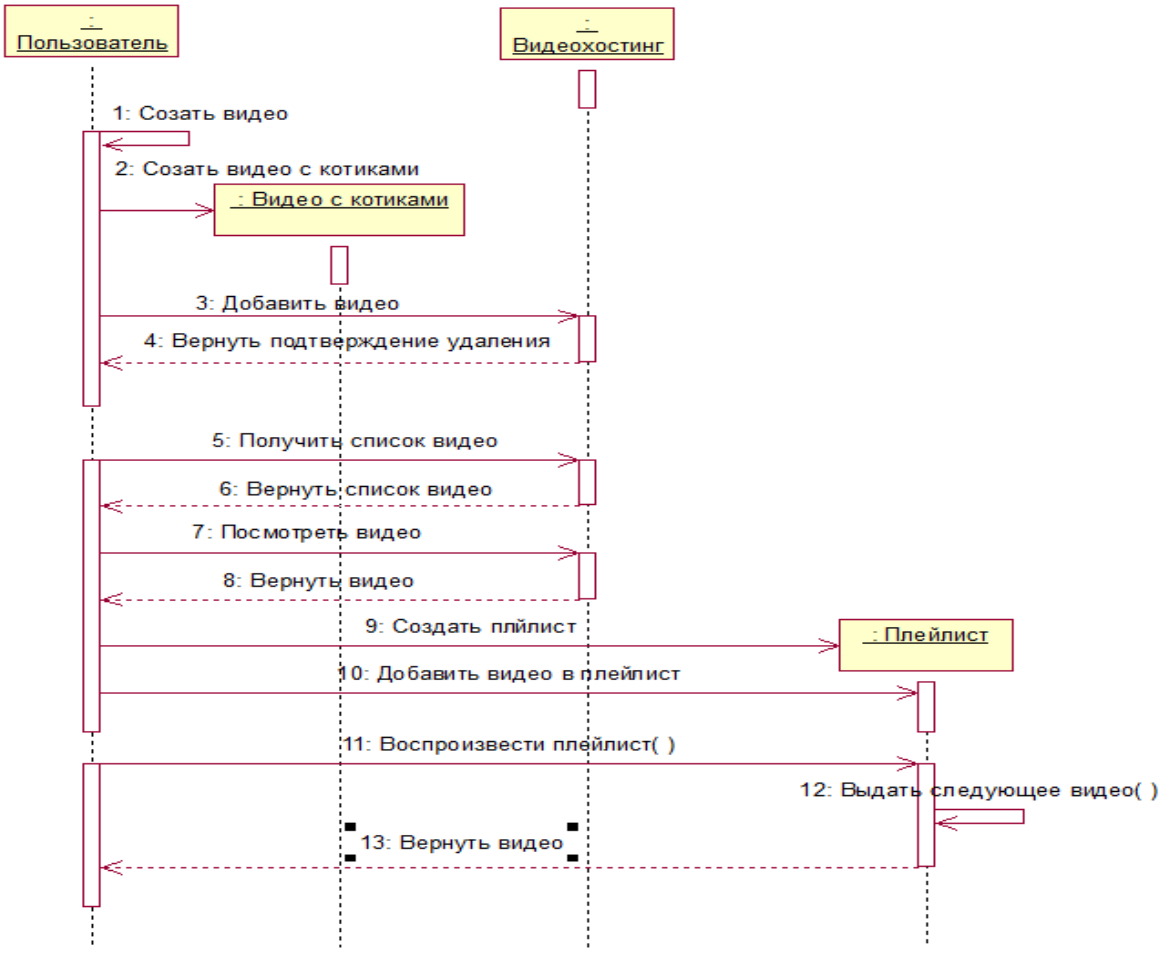

Рисунок 181 – Диаграмма последовательности

Лабораторная работа №4

Разработка многопоточных программ,
использование стандартных примитивов синхронизации

Задание к лабораторной работе:

– необходимо разработать графический интерфейс с использованием менеджера компоновки;

– представление графического интерфейса разрабатывается с учётом предоставления всей функциональности предложенного варианта;

– объекты и их взаимоотношения, имеющиеся в варианте задания, должны быть реализованы;

– функциональная часть приложения, представленная диаграммой последовательности, должна быть реализована;

– графический интерфейс должен быть создан посредством одной из следующих библиотек: Swing, SWT, JavaFX;

– разбить функционал приложения на несколько пакетов придерживаясь логики;

– согласно варианту задания выбрать любую сущность, вынести её в отдельный поток, синхронизировать взаимодействие с ней.

Вариант 1

Машины едут по пятиполосной дороге с разной скоростью. В дороги их скорость замеряется и, если их скорость превышает разрешённую, они убираются с дороги. Иначе они убирают сами себя через секунду. Каждые 50 машин в консоль выводится отчёт, содержащий максимальную скорость и количество нарушений. Каждую машину запускать в отдельном потоке.

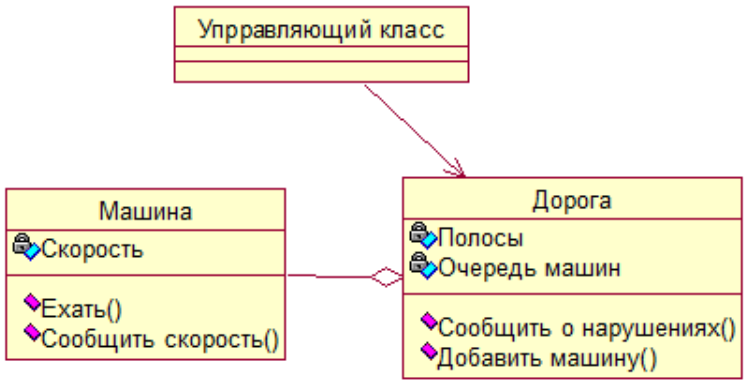

Рисунок 182 – Диаграмма классов

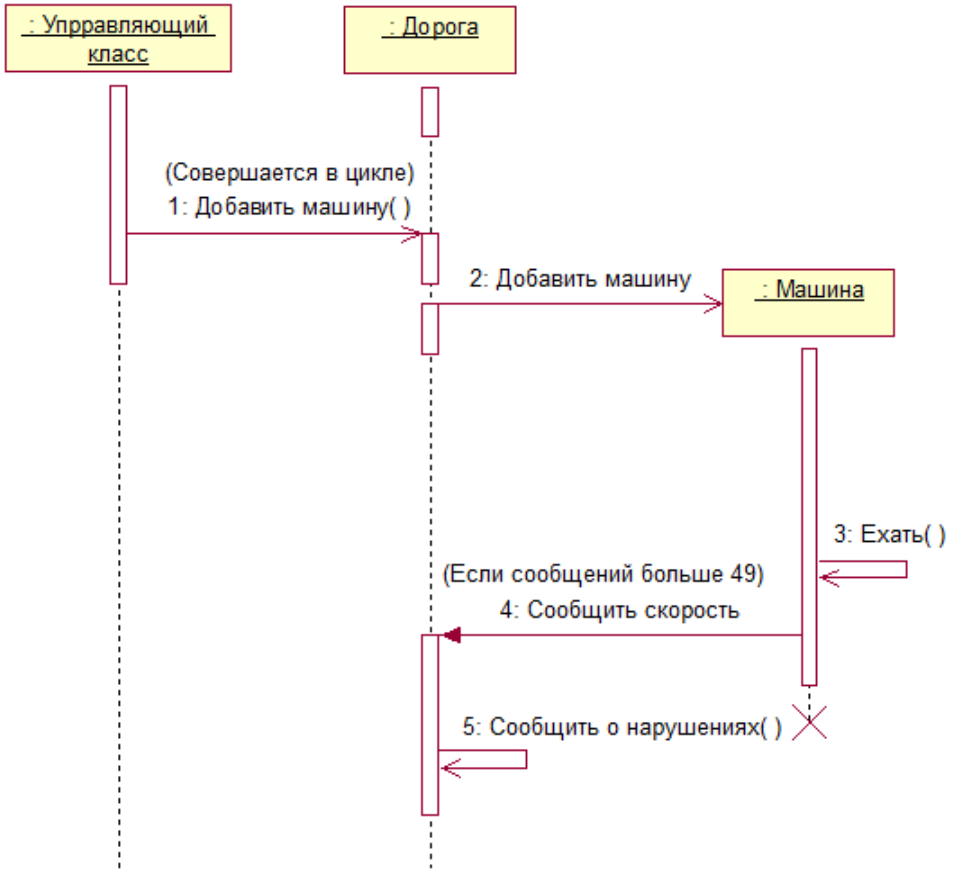

Рисунок 183 – Диаграмма последовательности

Вариант 2

Вычислить интеграл методом трапеций. Площадь каждой трапеции вычислять в отдельном потоке.

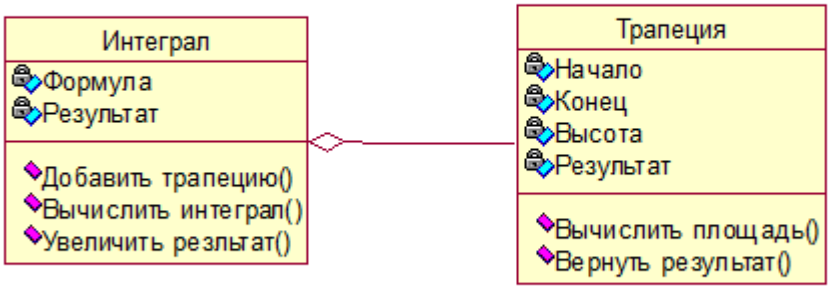

Рисунок 184 – Диаграмма классов

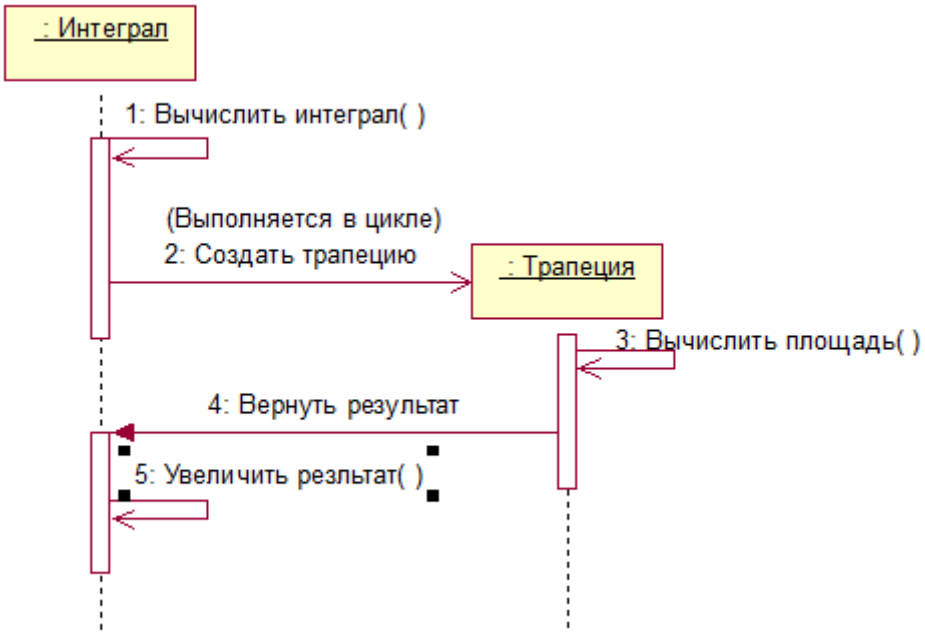

Рисунок 185 – Диаграмма последовательности

Вариант 3

Вычислить сумму двух векторов. Вычисление каждой отдельной суммы реализовать в отдельном потоке.

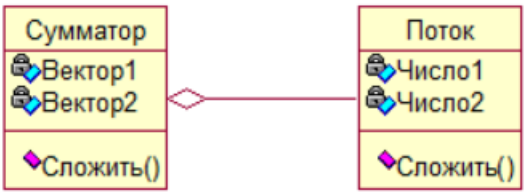

Рисунок 186 – Диаграмма классов

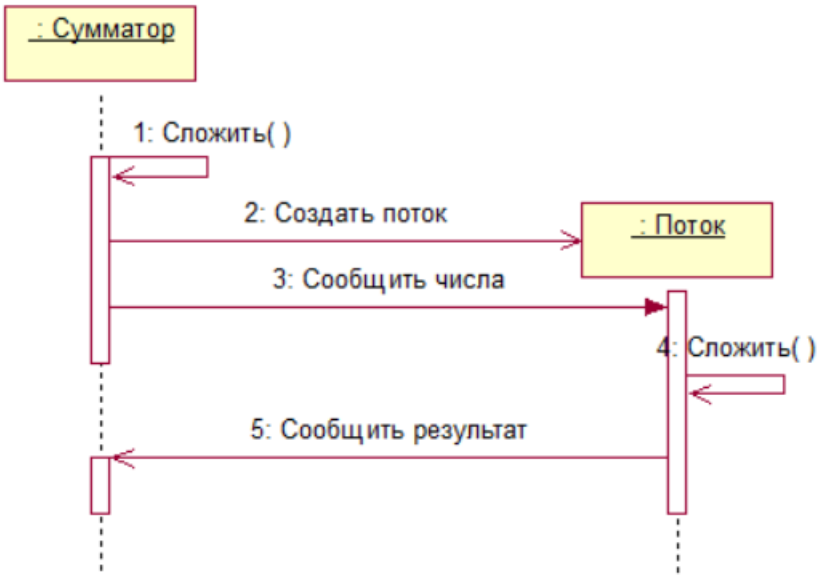

Рисунок 187 – Диаграмма последовательности

Вариант 4

Вычислить алгебраическое произведение двух векторов. Каждое частичное произведение вычислить в отдельном потоке.

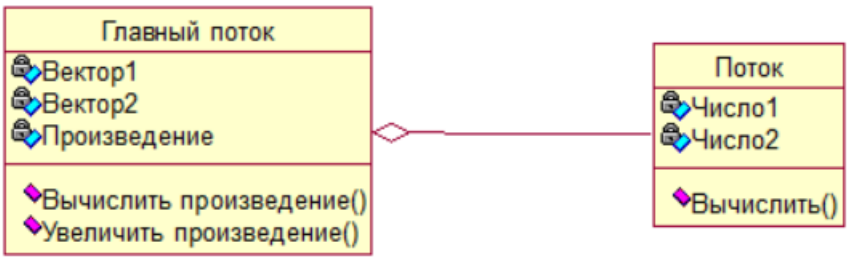

Рисунок 188 – Диаграмма классов

Рисунок 189 – Диаграмма последовательности

Вариант 5

Вычислить произведение матриц. Каждое произведение строки на столбец вычислить в отдельном потоке.

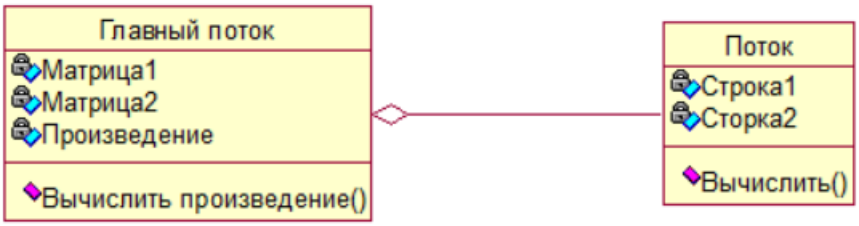

Рисунок 190 – Диаграмма классов

Рисунок 191 – Диаграмма последовательности

Вариант 6

Вывести массив строк в консоль. Вывод каждой строки реализовать в отдельном потоке, по одному символу за раз. После 50 выводов разрушить поток.

Рисунок 192 – Диаграмма классов

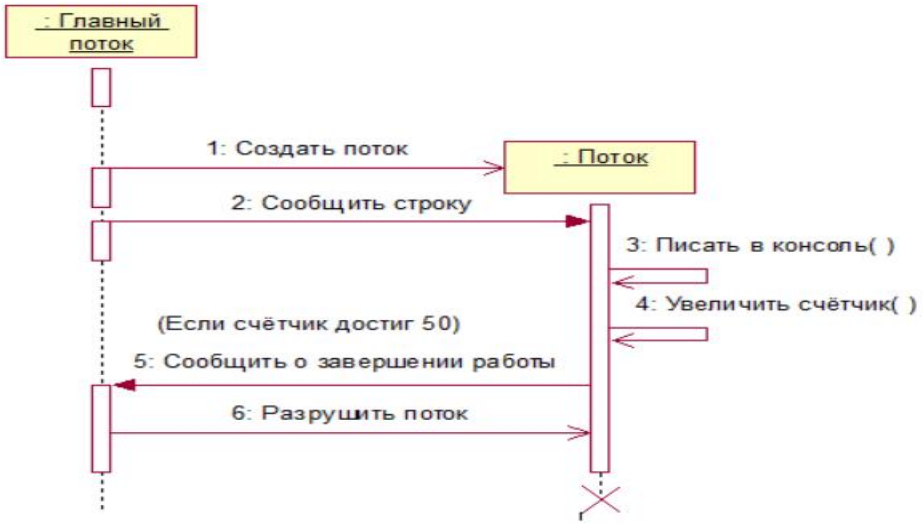

Рисунок 193 – Диаграмма последовательности

Вариант 7

Животные подходят к кормушке, получают из неё еду, уходят. У кормушки одновременно может быть три животного. Каждое животное представить отдельным потоком.

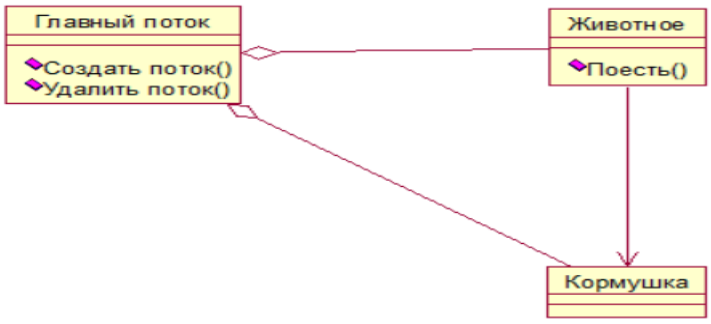

Рисунок 194 – Диаграмма классов

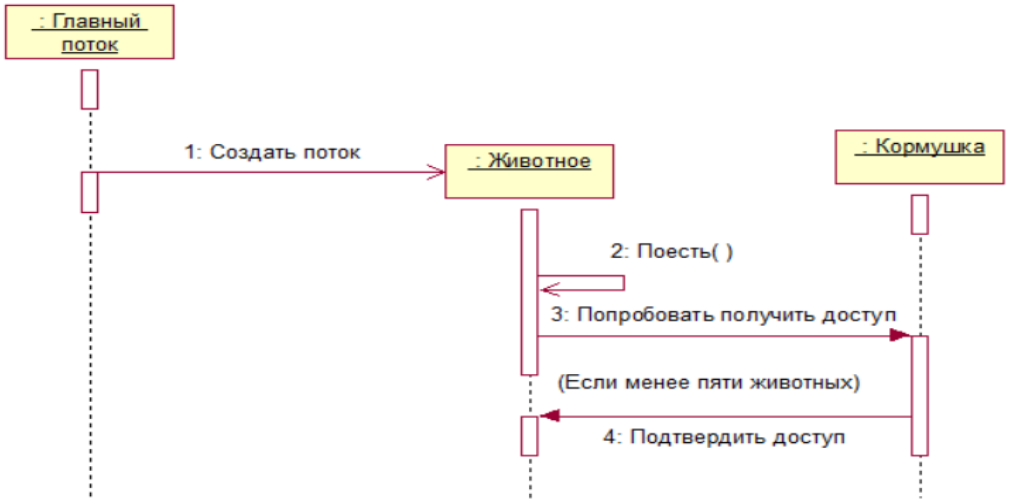

Рисунок 195 – Диаграмма последовательности

Вариант 8

Студенты получают книги в библиотеке и возвращают их после прочтения. Если книги нет, они ждут, пока она появится. Каждого студента реализовать в отдельном потоке.

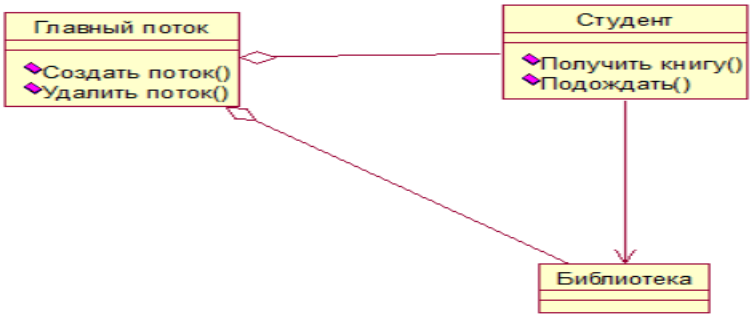

Рисунок 196 – Диаграмма классов

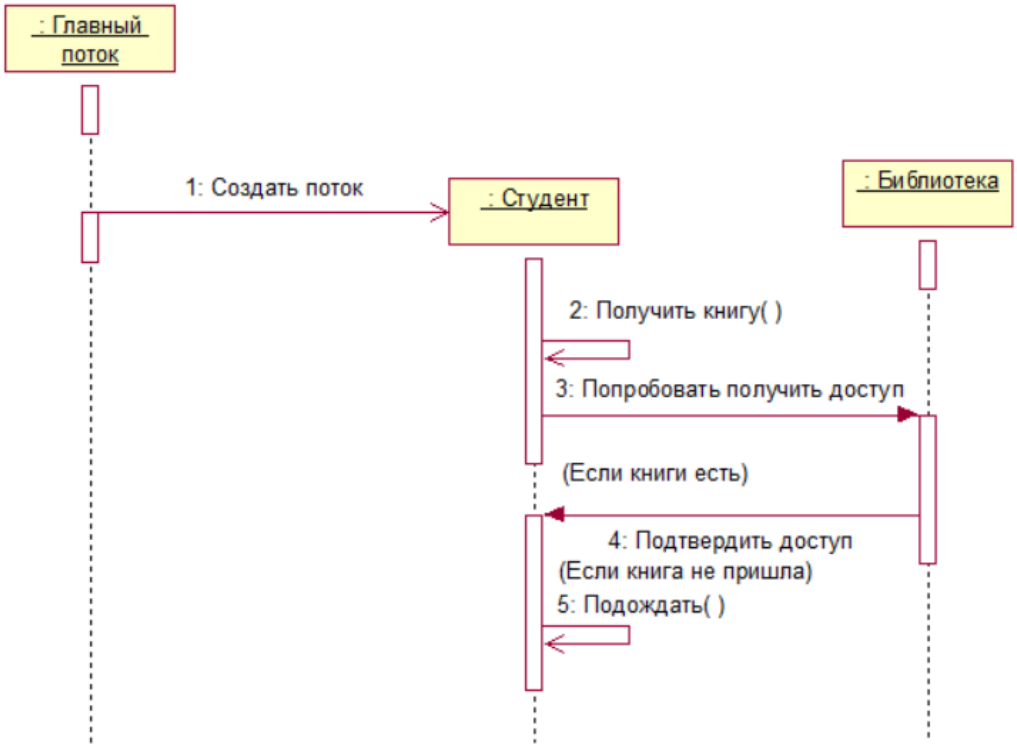

Рисунок 197 – Диаграмма последовательности

Вариант 9

Задача о читателях и писателях. Читатели могут читать данные, если нет активных писателей. Писатели модифицируют данные, если нет активных читателей и писателей. К данным может иметь доступ только один поток одновременно.

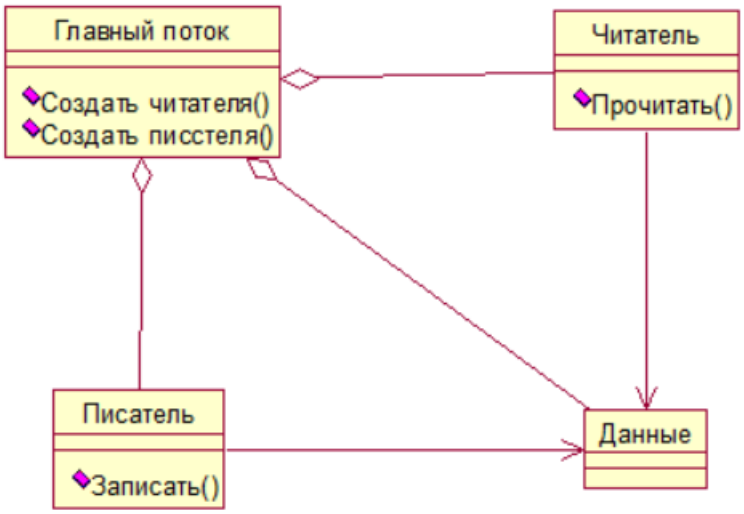

Рисунок 198 – Диаграмма классов

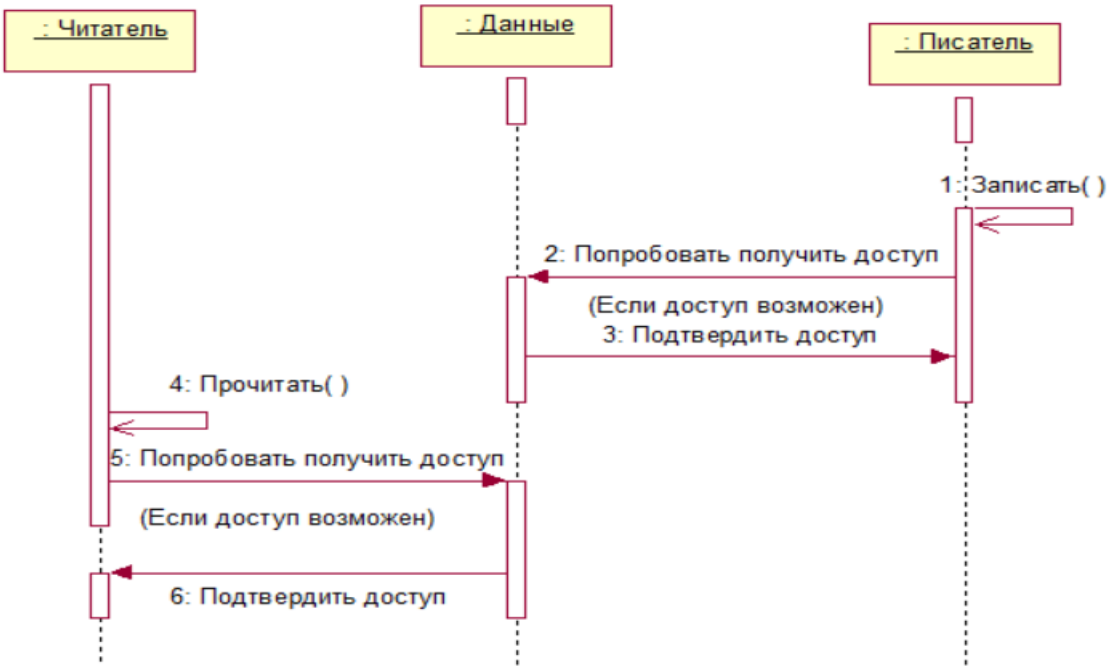

Рисунок 199 – Диаграмма последовательности

Вариант 10

Потоки спорят. Один тип потоков увеличивает общую переменную на один, другой тип потоков уменьшает переменную на один. Когда сумма по модулю достигает некого числа, объявляется победитель и потоки останавливаются.

Рисунок 200 – Диаграмма классов

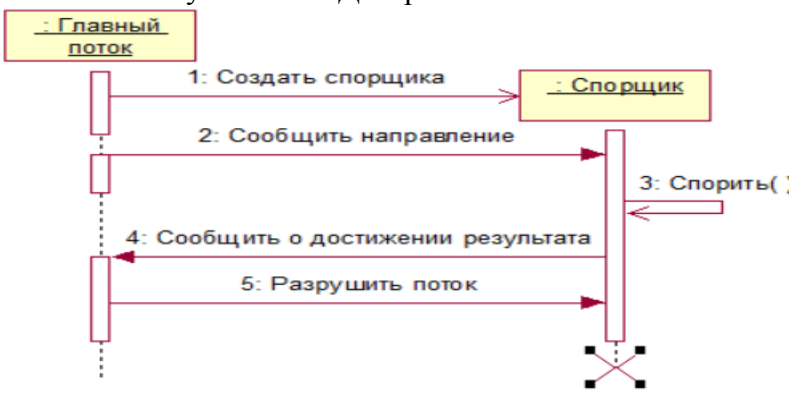

Рисунок 201 – Диаграмма последовательности

Вариант 11

Задача обедающих философов. Философы обедают, сидя за круглым столом. По правую и левую руку философов лежит вилка таким образом, что между двумя соседними философами находится только вилка. Философ может есть только если у него в руках две вилки. Реализовать алгоритм, гарантирующий, что философ начнёт есть за конечный промежуток времени.

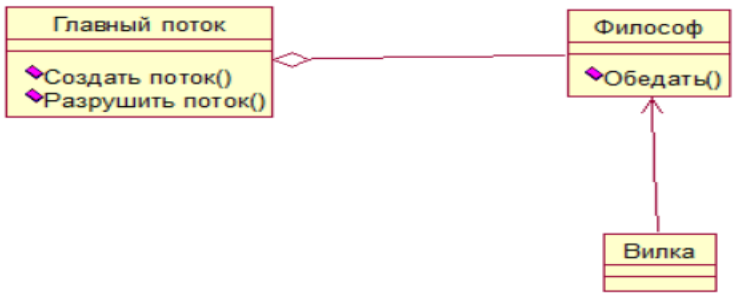

Рисунок 202 – Диаграмма классов

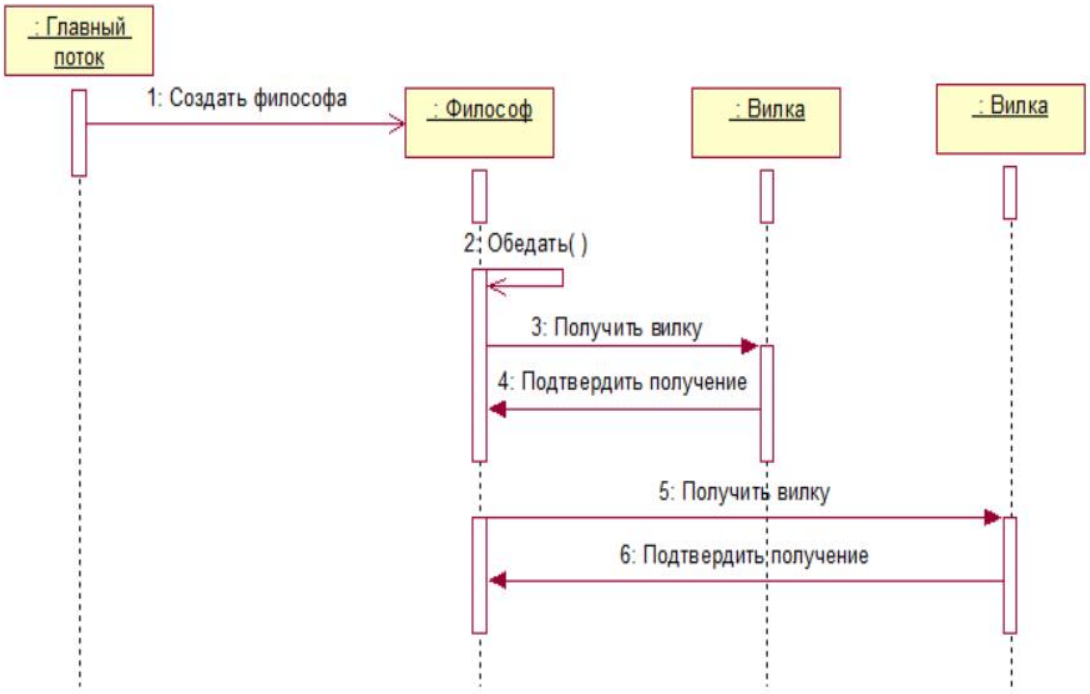

Рисунок 203 – Диаграмма последовательности

Вариант 12

Реализовать нахождение целых корней уравнения методом подбора. Подбор каждого корня сделать в отдельном потоке. После нахождения корня, вывести на экран ответ и остановить все угадывающие потоки.

Рисунок 204 – Диаграмма классов

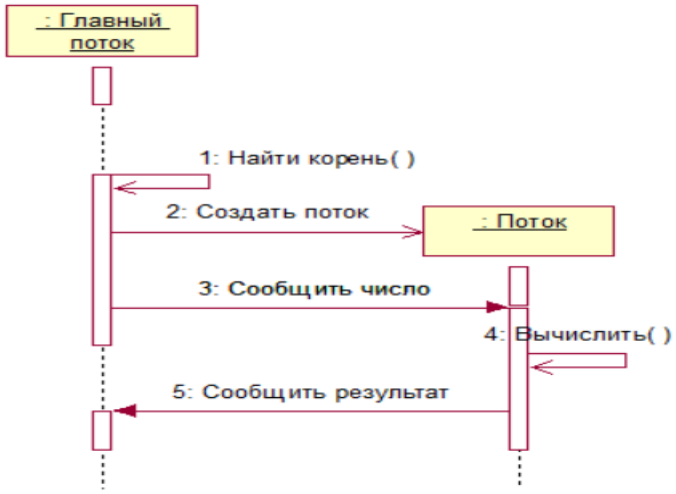

Рисунок 205 – Диаграмма последовательности

Вариант 13

Нахождение комплексных корней N-ной степени. Каждый из N корней должен вычисляться в отдельном потоке. Основной поток выводит ответы по мере готовности.

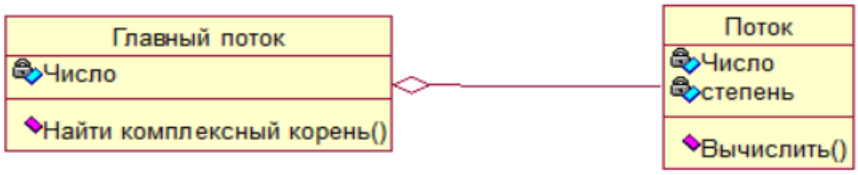

Рисунок 206 – Диаграмма классов

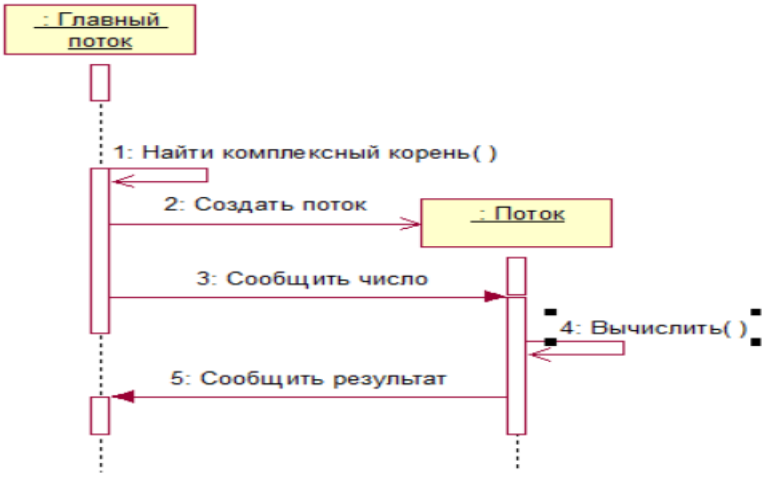

Рисунок 207 – Диаграмма последовательности

Вариант 14

Примитивный чат-бот. Один поток посылает боту строку. Если эта строка известна боту, он отвечает на неё заранее известной строкой. Если не известна, отвечает сообщением об ошибке. Реализовать бота и его собеседника в отдельных потоках. Реализовать возможность одновременной работы бота с несколькими потоками-собеседниками.

Рисунок 208 – Диаграмма классов

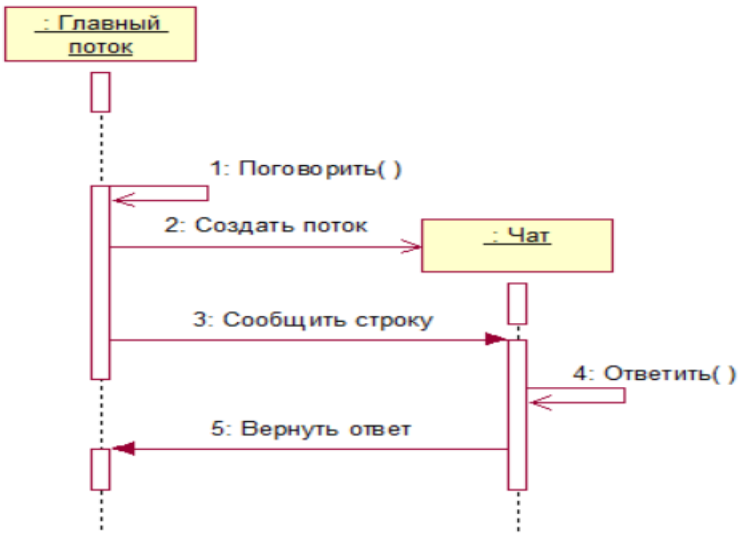

Рисунок 209 – Диаграмма последовательности

Вариант 15

Производители и потребители. Есть два типа потоков: одни пишут в буфер, другие из него читают. Буфер ограниченного размера. Реализовать алгоритм взаимодействия потоков, который не допускает переполнения буфера и не допускает считывания несуществующих данных.

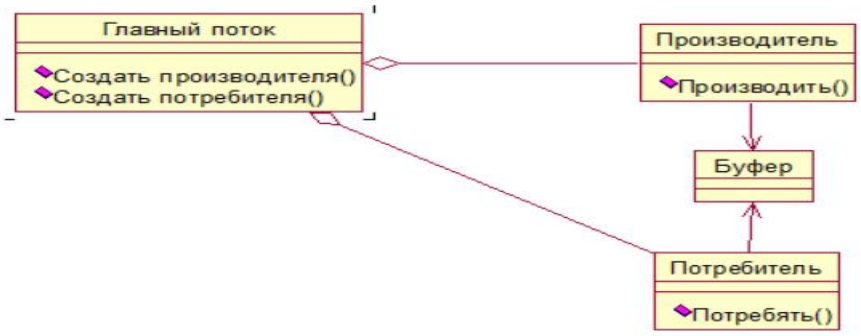

Рисунок 210 – Диаграмма классов

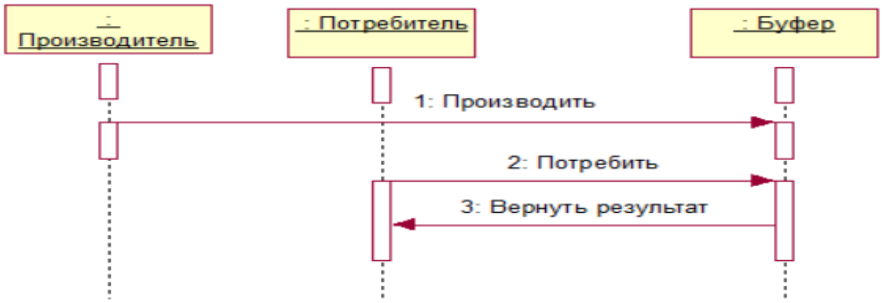

Рисунок 211 – Диаграмма последовательности

Вариант 16

Реализовать лифт. В лифте может быть не более пяти человек. Когда лифт приходит на какой-то этаж, в него могут войти люди, находящиеся на этом этаже и выйти люди, которым надо на этот этаж. Лифт и каждый и людей должен быть отдельным потоком. Организовать алгоритм, гарантирующий доставку человека на его этаж за конечный промежуток времени.

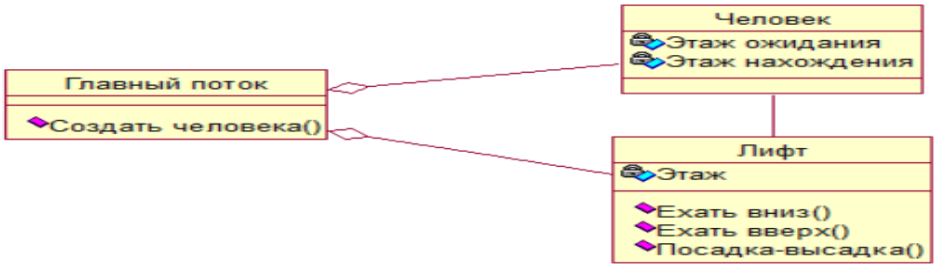

Рисунок 212 – Диаграмма классов

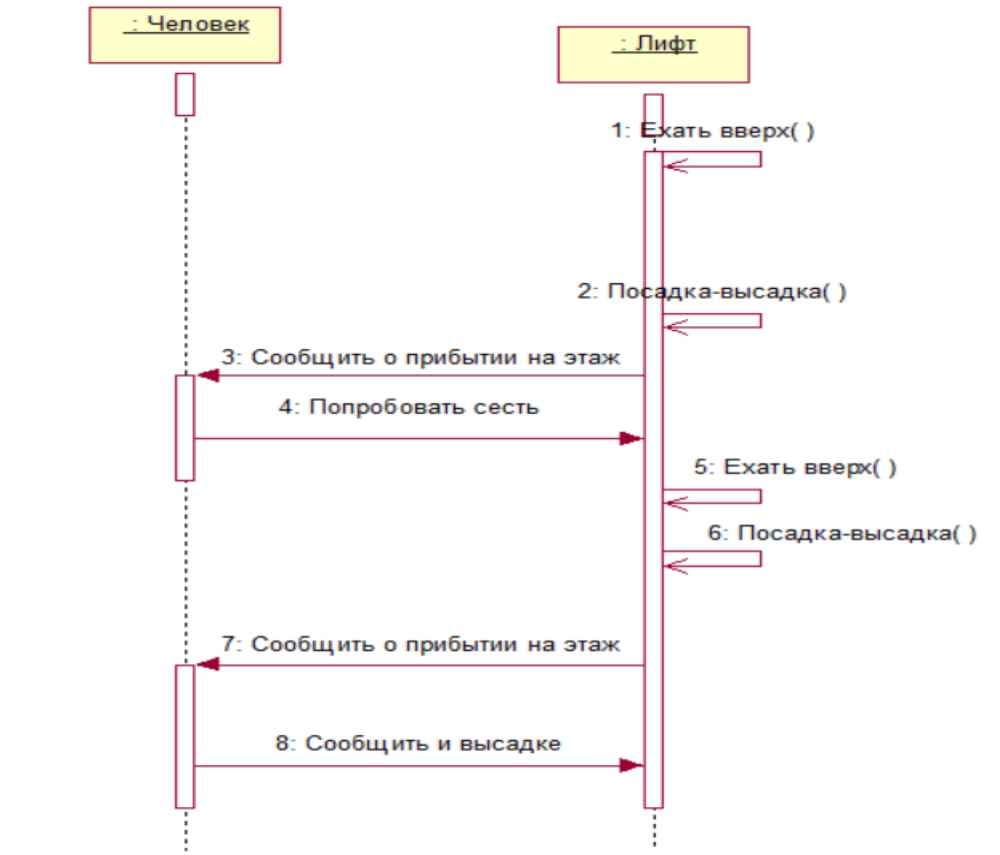

Рисунок 213 – Диаграмма последовательности

Вариант 17

Реализовать душевую. В ней может быть бесконечное количество людей, но только мужчины или только женщины. Каждый человек должен быть представлен отдельным потоком. Реализовать алгоритм, гарантирующий помывку мужчин и женщин.

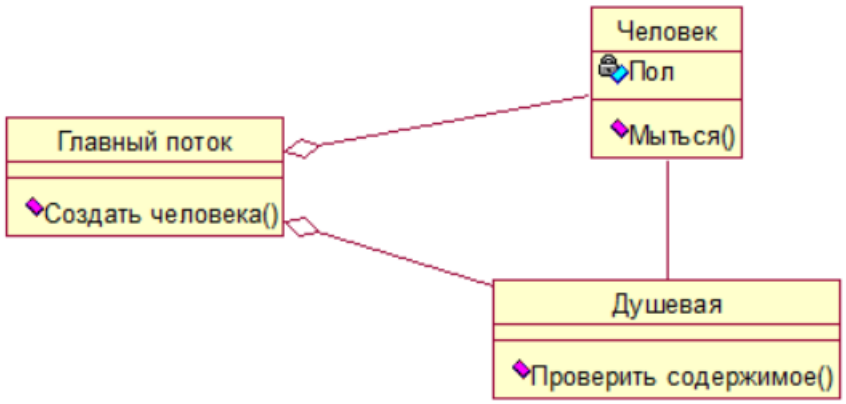

Рисунок 214 – Диаграмма классов

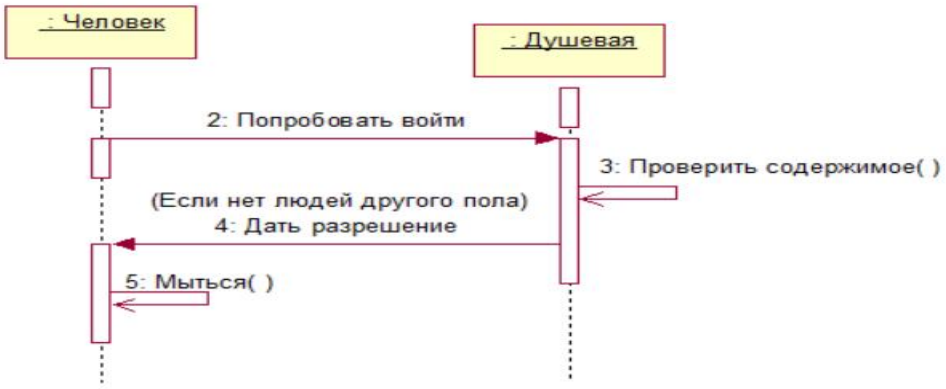

Рисунок 215 – Диаграмма последовательности

Вариант 18

. В доме есть только подключение по интернету через диалап. Одно поколение людей пользуется телефоном, а другое интернетом. Можно пользоваться или телефоном, или интернетом. Телефоном может одновременно пользоваться только один человек, а интернетом бесконечное количество. Каждого человека представить отдельным потоком. Реализовать алгоритм, гарантирующий доступ к телефону и интернету за конечный промежуток времени.

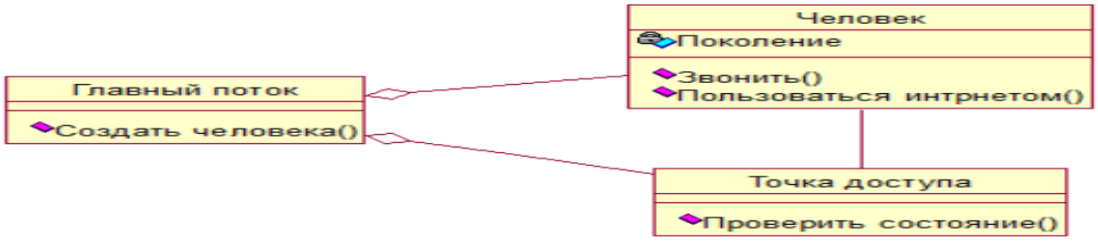

Рисунок 216 – Диаграмма классов

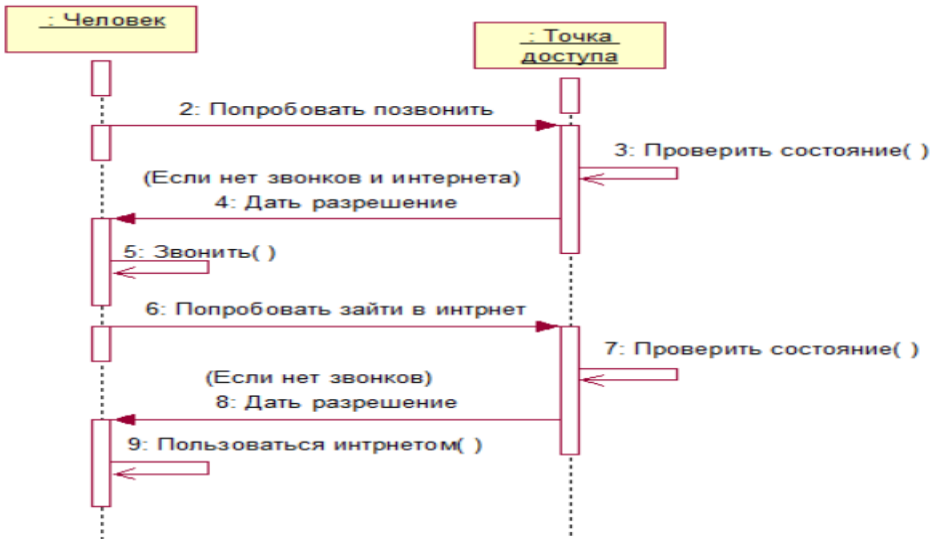

Рисунок 217 – Диаграмма последовательности

Вариант 19

Касса в ресторане быстрого питания. У кассы может находиться только один человек. После заказа, человек отходит от кассы и ждёт, пока кухня его приготовит. Ждать своего заказа может несколько человек сразу. Реализовать алгоритм, гарантирующий получение заказа в конечный промежуток времени. Каждый человек и кухня должны быть представлены отдельным потоком.

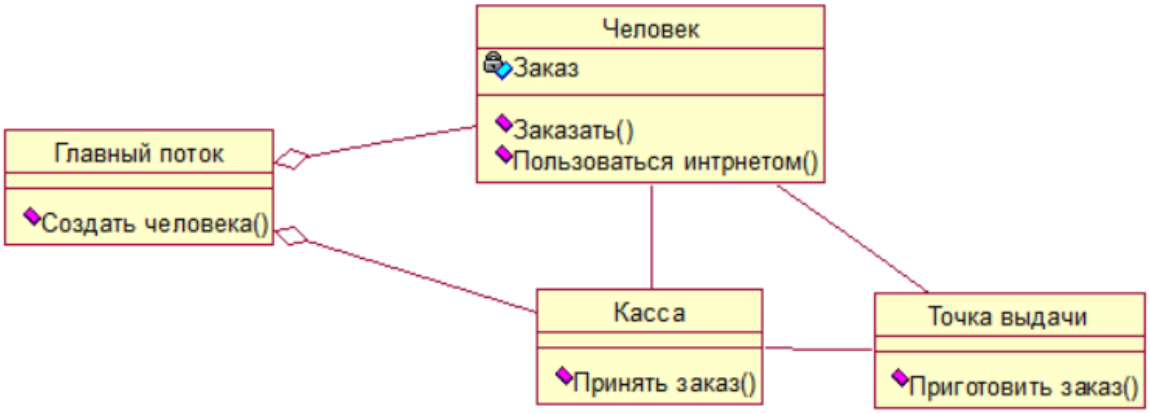

Рисунок 218 – Диаграмма классов

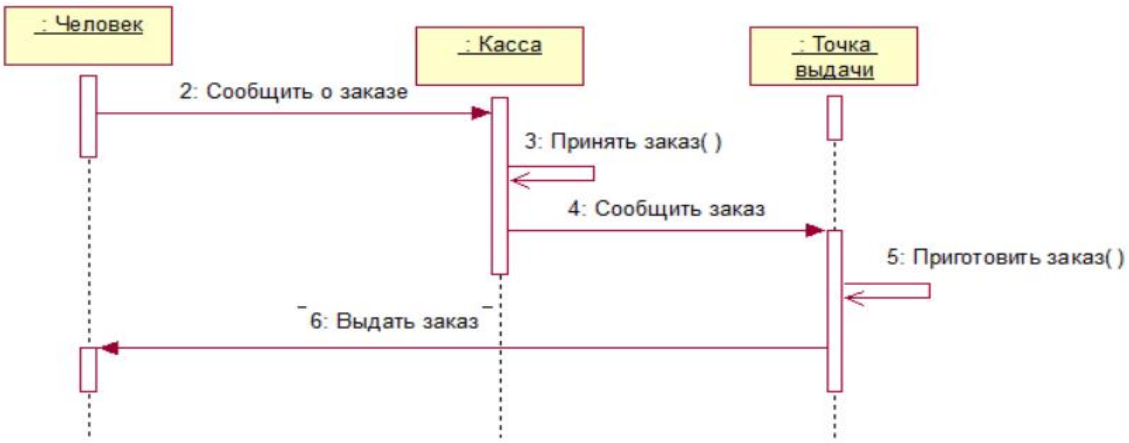

Рисунок 219 – Диаграмма последовательности

Вариант 20

Реализовать Многопоточную сортировку слиянием.

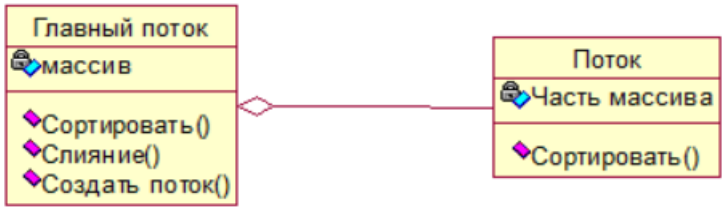

Рисунок 220 – Диаграмма классов

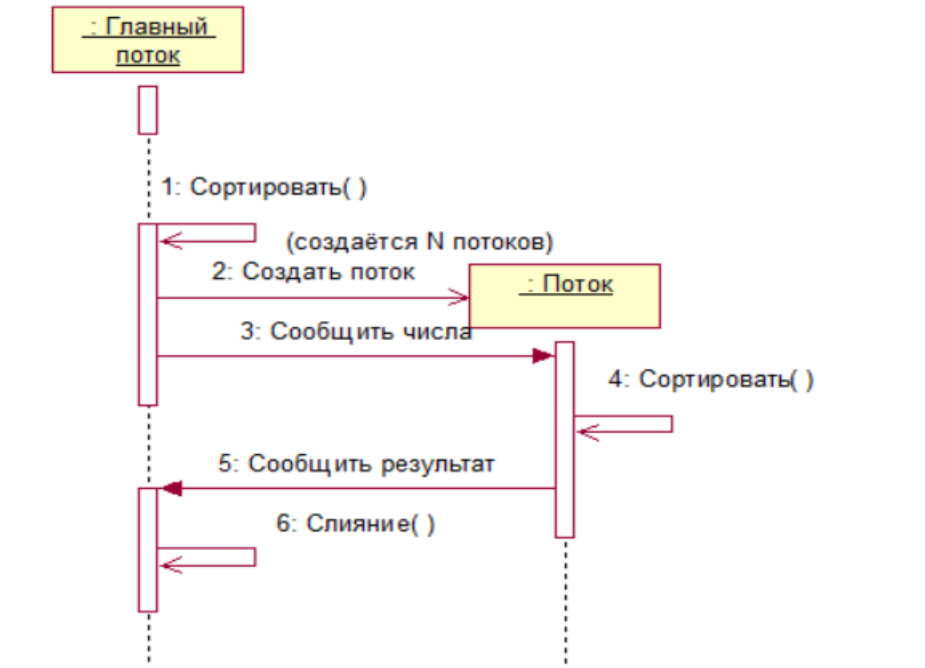

Рисунок 221 – Диаграмма последовательности

Вариант 21

Рассчитать определитель третьего порядка через определители второго порядка. Каждый определитель второго порядка считать в отдельном потоке.

Рисунок 222 – Диаграмма классов

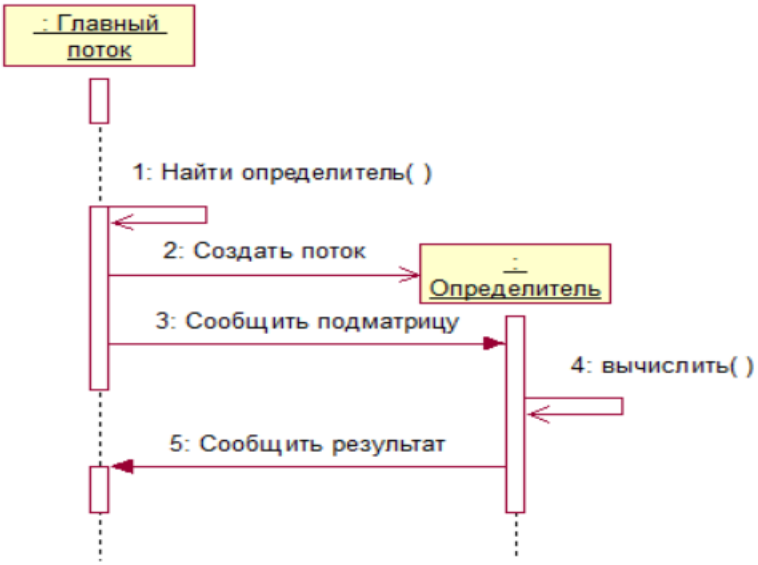

Рисунок 223 – Диаграмма последовательности

Вариант 22

Задана дискретная функция. Найти в этой функции локальные максимумы и минимумы. Каждую точку проверять на максимум и минимум в отдельном потоке.

Рисунок 224 – Диаграмма классов

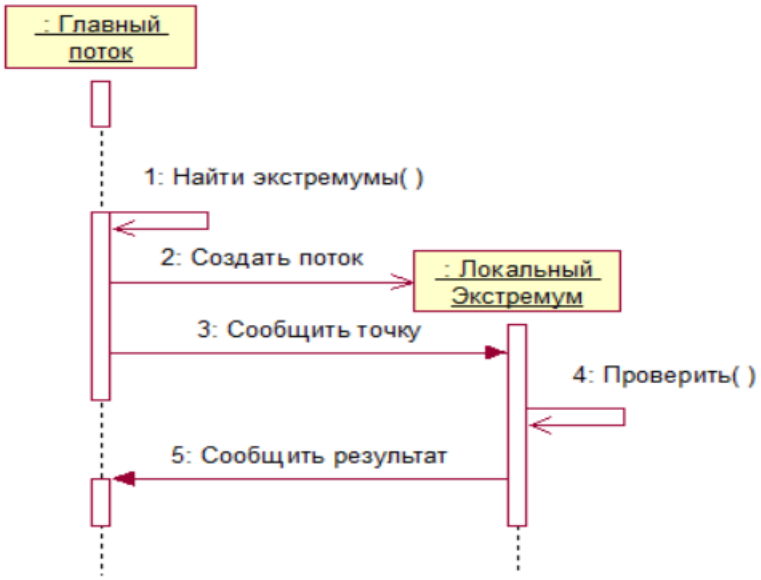

Рисунок 225 – Диаграмма последовательности

Вариант 23

Найти дискретную линейную свёртку двух сигналов. Каждую итерацию цикла вычислять в отдельном потоке.

Рисунок 226 – Диаграмма классов

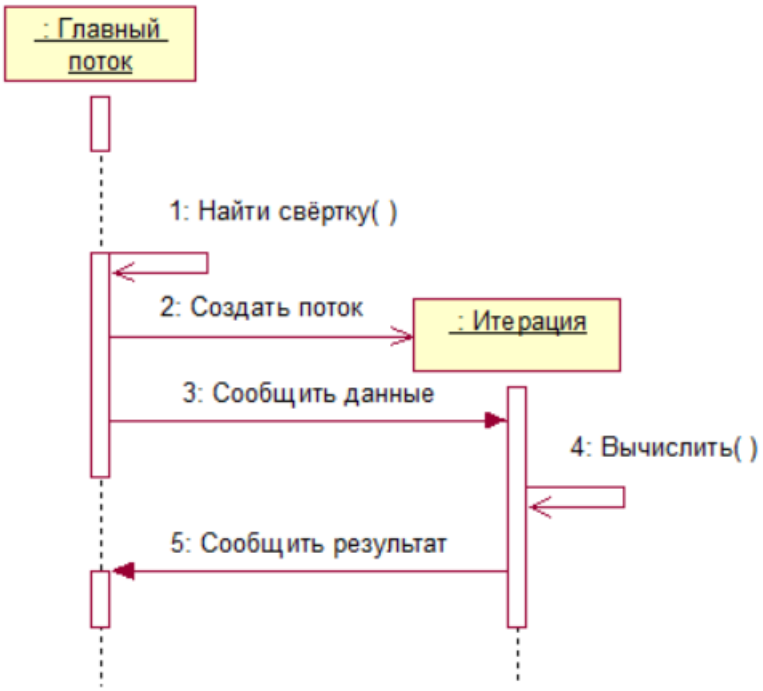

Рисунок 227 – Диаграмма последовательности

Вариант 24

Найти корреляцию двух дискретных сигналов. Каждую итерацию цикла вычислить в отдельном потоке.

Рисунок 228 – Диаграмма классов

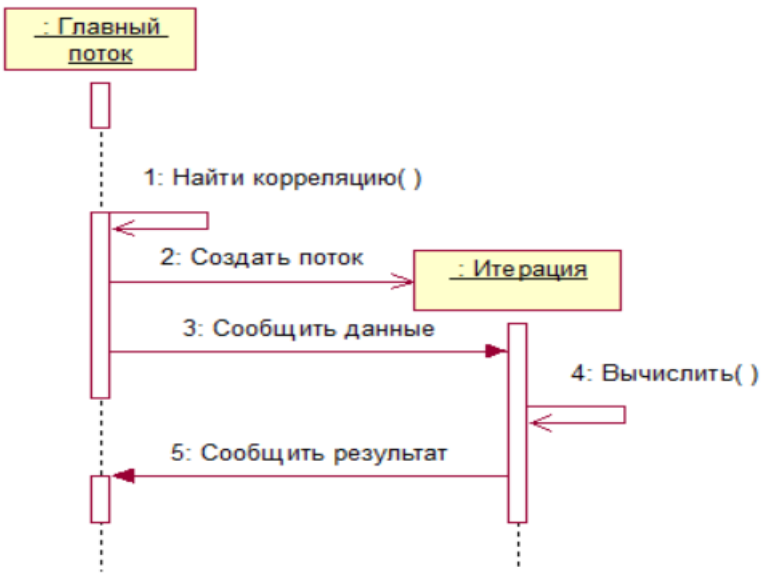

Рисунок 229 – Диаграмма последовательности

Вариант 25

Реализовать многопоточный калькулятор. Несколько потоков клиентов посылают вычисления серверу, а сервер посылает им результат вычислений.

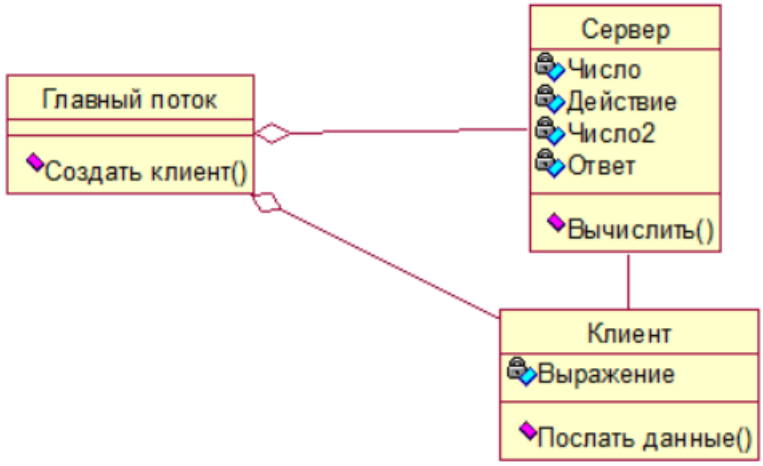

Рисунок 230 – Диаграмма классов

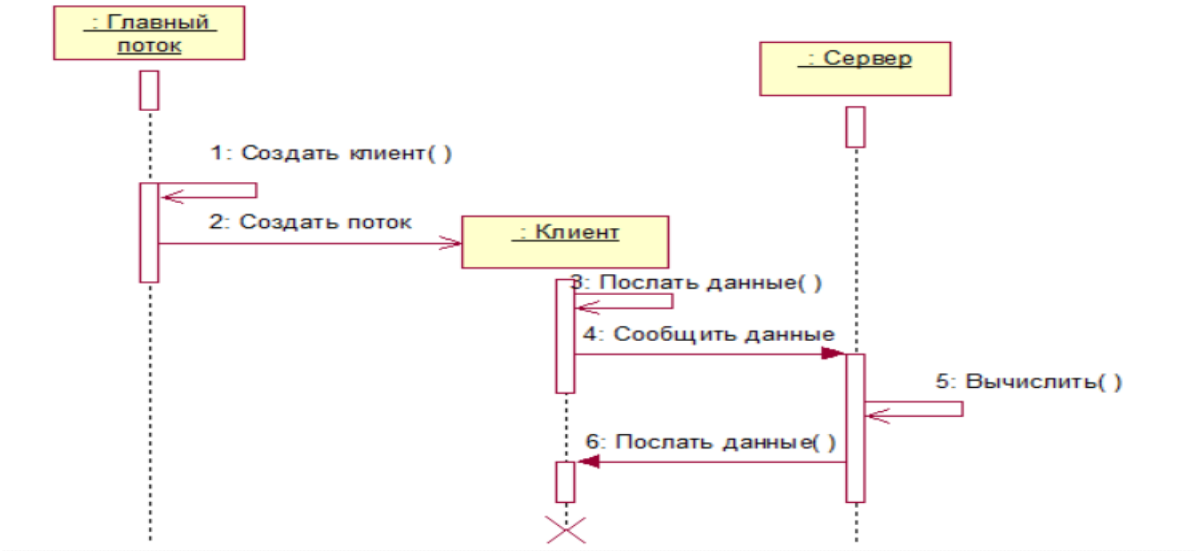

Рисунок 231 – Диаграмма последовательности

Вариант 26

Найти преобразование Фурье дискретного сигнала. Значение каждой точки вычислять в отдельном потоке.

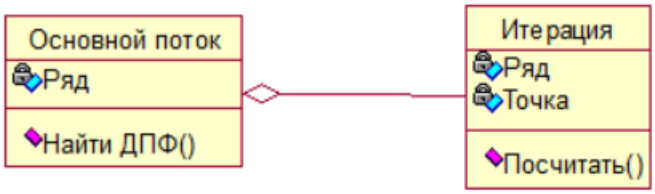

Рисунок 232 – Диаграмма классов

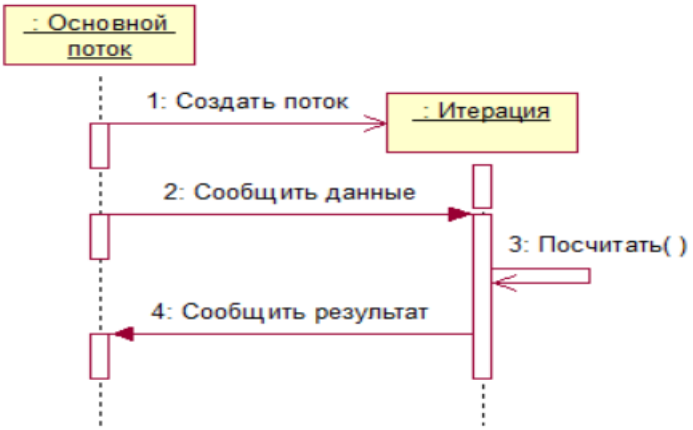

Рисунок 233 – Диаграмма последовательности

Вариант 27

Реализовать модель Солнечной системы. Планеты движутся по круговым орбитам и вращаются вокруг оси. Каждый раз, когда планета совершает полный оборот, на экран отображается поздравление с новым годом и Название планеты. Каждый раз, когда совершается полный оборот вокруг оси, на экран отображается поздравление с добрым утром и Название планеты. Каждую планету реализовать в отдельном потоке.

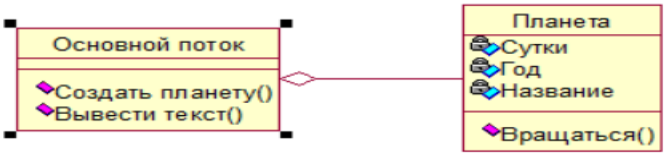

Рисунок 234 – Диаграмма классов

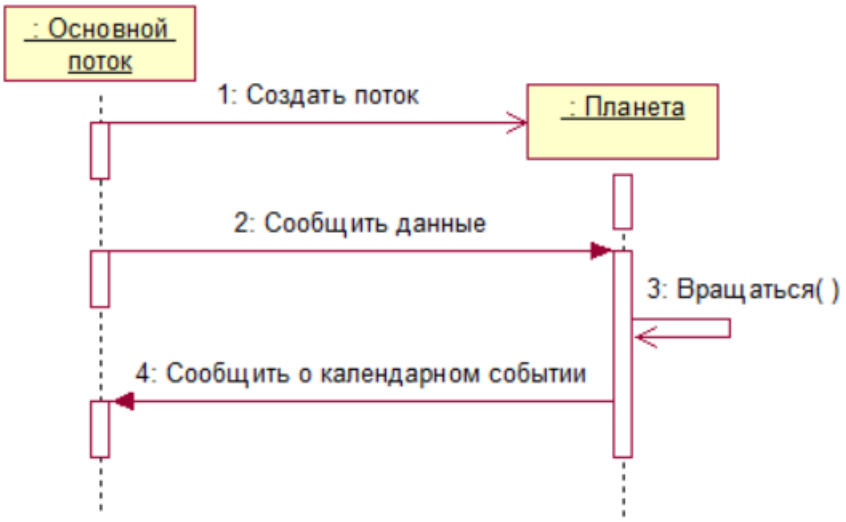

Рисунок 235 – Диаграмма последовательности

Вариант 28

Сравнение строк. Реализовать посимвольное сравнение строк. Если символы отличаются, отобразить какой символ заменён на какой и его позицию. Сравнение каждой пары символов реализовать в отдельном потоке.

Рисунок 236 – Диаграмма классов

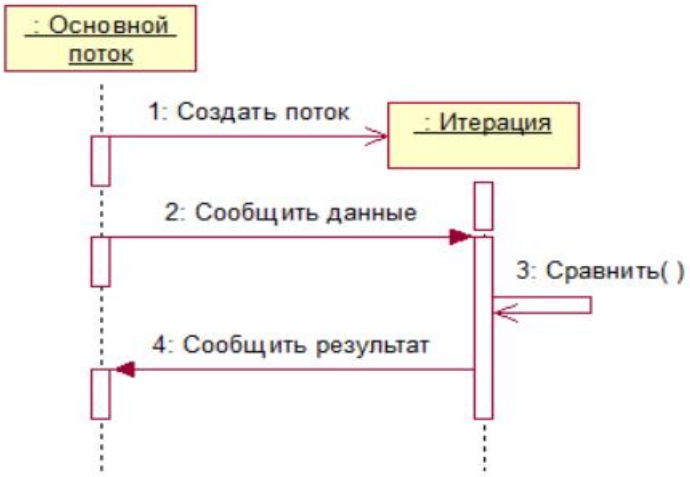

Рисунок 237 – Диаграмма последовательности

Вариант 29

Центр печати документов. В центре есть три принтера А4, 2 принтера А3 и один принтер А2. Он принимает заказы, состоящие из случайной комбинации листов трёх форматов. Один заказ может исполняться на любом количестве принтеров одновременно. Печать одного листа занимает некоторое время. Организовать исполнение заказов и выдачу заказов клиентам.

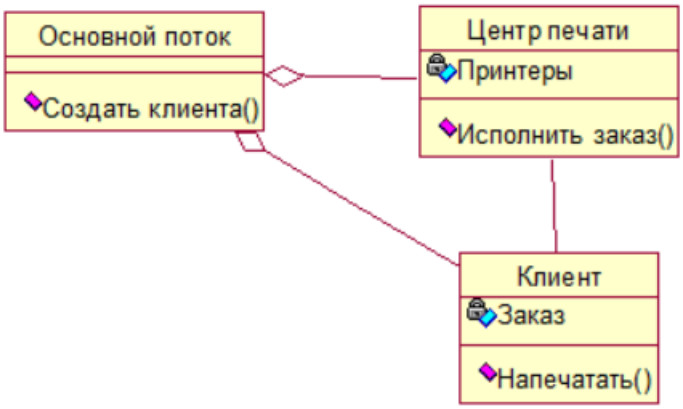

Рисунок 238 – Диаграмма классов

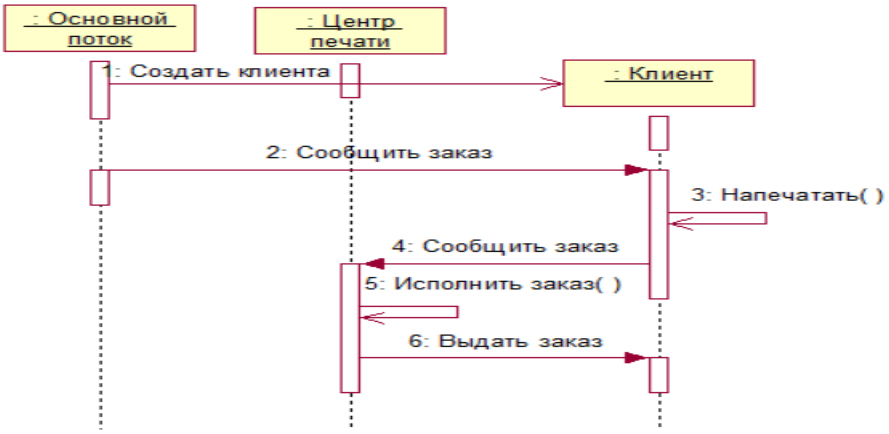

Рисунок 239 – Диаграмма последовательности

Вариант 30

Центр управления полётом. Есть пять полос. В аэропорт прибывают самолёты. Каждый самолёт может находится на полосе случайное время, ограниченное сверху и снизу. Реализовать алгоритм, гарантирующий посадку каждого самолёта в конечный интервал времени.

Рисунок 240 – Диаграмма классов

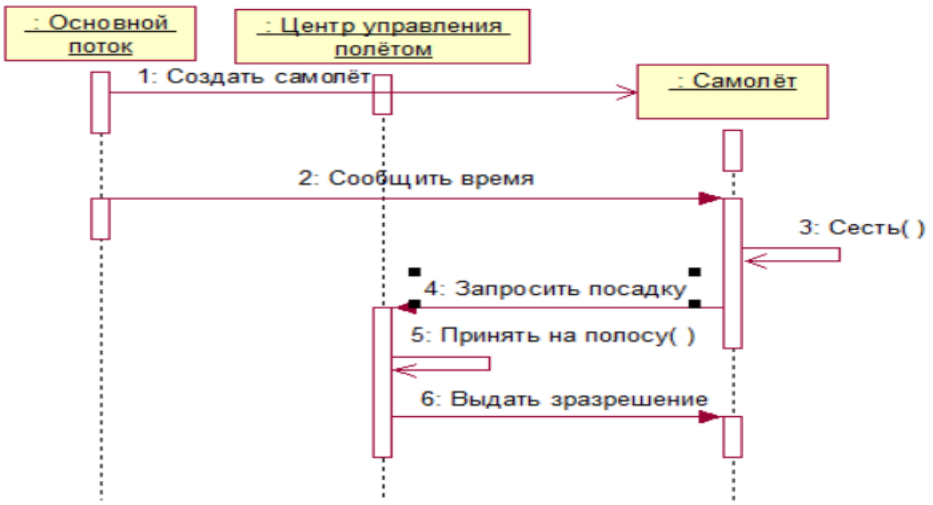

Рисунок 241 – Диаграмма последовательности

Общие требования ко всем лабораторным по Scala:

- единый стиль кода: http://docs.scala-lang.org/style/
- весь реализованный функционал должен быть покрыт тестами при помощи библиотеки ScalaTest;
- если результатом работы функции является коллекция элементов, то и возвращаем этот результат в коллекции;
- весь ключевой функционал не работает с консолью напрямую: функция возвращает единичные значения или коллекции – никакого прямого вывода в консоль;
- пользуйтесь подвыражениями;
- запрещается использовать:
  - isInstanceOf, asInstanceOf;
  - var
  - return
  - ;
  - if (cond) true else false
  - print

Лабораторная работа №5

Функциональное программирование с использованием языка Scala. Хвостовая рекурсия

Задание к лабораторной работе:

- выполнить задание по варианту не используя хвостовую рекурсию;
- выполнить задание по варианту используя хвостовую рекурсию.

1. Реализовать сортировку списка.
2. Найти N чисел Фибоначчи.
3. Найти факториал числа.
4. Найти наибольший общий делитель.
5. Вывести все целые значения между двумя числами.
6. Найти все простые числа в списке.
7. Найти минимальное число в списке.
8. Найти среднее арифметическое.
9. Найти локальные экстремумы ряда.
10. Найти длину самого большого слова в списке.
11. Найти количество вхождений символа в строке.
12. Инвертировать строку.
13. Сложить два вектора.
14. Умножить два вектора.
15. Заменить все строчные буквы на прописные.
16. Разделить все чётные числа на два, а к нечётным прибавить один.
17. Вывести позицию первого элемента массива, равного введённому числу.
18. Возвести число в введённую степень.
19. Перевести десятичное число в двоичную систему счисления.
20. Найти сумму всех чисел в списке.
21. Найти длину наибольшей последовательности одинаковых чисел идущих подряд.
22. Найти все числа в списке, которые меньше разности двух предыдущих.
23. Составить список, содержащий только положительные числа из исходного списка.
24. Найти комбинации двух положительных чисел, дающие в сумме введённое значение.
25. Найти все числа, квадрат которых попадает в заданный промежуток.
26. Найти сумму произведения двух функций на заданном целочисленном промежутке.
27. Найти произведение двух векторов.
28. Найти определённый интеграл методом трапеций.
29. Найти индекс начала наибольшего интервала увеличивающейся последовательности.
30. Составить список на основе оригинального, удалив оттуда повторяющиеся символы.

Лабораторная работа №6

Функции высших порядков. Коллекции и абстракции

Задание к лабораторной работе:

- выполнить задание согласно выданному варианту;
- при реализации использовать функции высших порядков.

1. Удалить из коллекции все повторяющиеся элементы.
2. Посчитать количество заданных элементов в коллекции.
3. Из двух коллекций сделать одну, в которой будут только общие для двух коллекций элементы.
4. Из двух коллекций сделать одну, в которой будут только уникальные элементы.
5. Все чётные элементы в коллекции умножить на два, а нечётные умножить на 3.
6. Из двух несортированных коллекций сделать одну сортированную.
7. Найти среднее арифметическое для всех элементов коллекции.
8. Оставить в коллекции только простые числа.
9. Найти сумму всех положительных чисел в коллекции.
10. Коллекция представляет из себя номерной список студентов, отсортированных по алфавиту. Из двух таких списков сделать один.
11. Разделить коллекцию на две. В первой должны быть только положительные элементы, а во второй только отрицательные.
12. В коллекции хранится информация о футболистах и их голах. Создать две коллекции. В одной отсортировать футболистов по голам, а в другой в алфавитном порядке.
13. Существует набор точек. Найти максимальное расстояние между двумя точками среди точек коллекции.
14. Найти сумму всех элементов коллекции.
15. Подсчитать количество совпадающих символов, стоящих на одинаковых позициях относительно первого элемента, в двух коллекциях.
16. Есть две коллекции с целочисленными элементами. Одну из них изменить таким образом, чтобы содержала поэлементную сумму, вторую – поэлементную разность.
17. Если первый символ после пробела строчная буква, заменить на прописную.
18. Найти длину самого длинного слова в коллекции.
19. Перевернуть все слова в коллекции, оставив неизменным их порядок.
20. Удалить из коллекции все повторяющиеся слова.
21. Отсортировать в коллекции все слова по длине.
22. Найти в коллекции полиномы.
23. Вставить в центр одной коллекции другую.
24. Найти в коллекции символов все слова, состоящие только из цифр.
25. Отсортировать коллекцию пузырьком.
26. Найти в коллекции индексы всех отрицательных элементов.
27. Если индекс простого числа, хранящегося в коллекции, чётный, то умножить его на 10.
28. Найти произведение двух матриц.
29. Транспонировать матрицу.
30. Найти произведения чисел на главной и побочной диагоналях матрицы.

Лабораторная работа №7

Сопоставление с образцом.

Задание к лабораторной работе:

- выполнить задание согласно выданному варианту;
- при реализации использовать сопоставление с образцом.

1. Найти сумму положительных и произведение отрицательных чисел.
2. Все буквы первой половины алфавита сделать строчными, а второй половины прописными.
3. Заменить пробелы на нижнее подчёркивание, а нижнее подчёркивание на пробелы.
4. Если число делиться на 5, умножить его на 2, а если делится на 6, разделить его на 3.
5. Все элементы, стоящие на чётной позиции, заменить на нули, а стоящие на нечётной, на единицы.
6. Все простые числа разделить на 2, а составные разделить на 3.
7. Если остаток от деления числа на три равен двум, умножить число на семь, а если равен одному, то умножить на 8.
8. Заменить цифры в строке на слова. То есть “1” заменить на “One”, “2” на “Two” и так далее.
9. Применить к данным кодирование 8b/10b
10. Раскодировать данные после кодирования 8b/10b
11. Применить к данным кодирование 5b/6b
12. Раскодировать данные после кодирования 5b/6b
13. Применить к данным кодирование кодом Грея. Кодировать каждые 8 бит
14. Раскодировать данные после кодирования кодом Грея. Раскодировать каждые 8 бит.
15. Добавить бит чётности после каждых 8 бит. Дополнять единицы до чётного количества.
16. Проверить информацию на сбои с помощью проверки бита чётности. Он стоит после каждых 8 бит и дополняет нули до чётного количества.
17. Зашифровать текст шифром Цезаря.
18. Расшифровать текст, зашифрованный шифром Цезаря.
19. После каждого слова “Ping” вставить “Pong”.
20. Посчитать количество правых и левых скобок. Если оно не совпадает, доставить недостающие в начало или конце.
21. Вывести все номера телефонов, принадлежащие Беларуси.
22. Посчитать в строке количество слов длинной в 6 символов.
23. Если перед словом стоит 2 пробела, переписать его задом наперёд.
24. Если после точки не стоит пробел и прописная буква, доставить пробел и заменить строчную букву на прописную.
25. Посчитать количество в строке каждого знака препинания.
26. Вывести на экран текст между символами “FE80” и “0001”.
27. Заменить слово “Azimov” на слово “Planck”, а слово “Planck” на слово “Heinlein”.
28. Удалить текст между скобками ().
29. Заменить кириллические символы на латинские. То есть написать транслитом.
30. Заменить латиницу на Leet.

Лабораторная работа №8

Отложенные вычисления. Абстракции многопоточности. Модель акторов. Парсинг. Комбинаторы синтаксического анализа

Задание к лабораторной работе:

– выполнить задание согласно выданному варианту.

1. Логин-сервер. Необходимо реализовать: добавление нового пользователя, задав логин и пароль; изменение пароля существующего пользователя; авторизация пользователя по логину и паролю и отказ авторизации, если пароль не соответствует логину.
2. База данных пользователей. Необходимо реализовать: добавление нового пользователя, задав имя, возраст и страну; изменение возраста или страны; вывод информации о пользователе.
3. Библиотека. Необходимо реализовать: добавление книги в базу; изменение статуса книги на доступную для выдачи и недоступную; выдачу книги на руки или отказ, если она недоступна.
4. Касса магазина. Необходимо реализовать: добавление покупки в чек; выдачу суммы покупки; хранение выданных чеков; выдачу информации о старом чеке по запросу.
5. Бассейн. Необходимо реализовать: открытие абонемента на имя; проверку, открыт ли абонемент на конкретное имя и выдачу подтверждения или отказа; закрытие абонемента после некоторого количества посещений.
6. Портной. Необходимо реализовать: добавление заказа, содержащего имя, одежду и ткань; получение информации о заказе, по имени; удаление заказа.
7. Чат-бот. Необходимо реализовать: ответ на строку; добавление новых ответов на строки; удаление ответов; стандартный ответ на неизвестную строку.
8. Игра крестики-нолики в поле 3 на 3. Необходимо реализовать: ход игрока по двум координатам; отказ в ходе, в случае если там стоит крестик или нолик; сообщение о выигрыше крестиков или ноликов; сохранение игры в памяти; загрузка игры.

Рисунок 242 – Диаграмма последовательности

9. Телефонная книга. Необходимо реализовать: добавление телефона и имени; выдачу телефона по имени; выдачу имени по телефону.

10. Почтовый ящик: добавление письма с обратным адресом, заголовком, текстом письма; вывод информации о следующем непрочитанном письме.
11. Календарь и часы. Необходимо реализовать: ход времени; остановку хода времени; вывод даты и времени; установку нужной даты и времени.
12. Стэк. Необходимо реализовать: добавление на вершину стека; снятие элемента с вершины стека; вывод элемента на вершине без его снятия.
13. Автомобиль. Необходимо реализовать: установку скорости в метрах в секунду; вывод пройденного расстояния; посадку пассажира, содержащего место и имя.
14. Треугольник. Необходимо реализовать: установку значения каждой из сторон; вывод периметра; вывод площади.
15. Банк. Необходимо реализовать: создание нового клиента, содержащего имя, номер счёта и сумму денег; снятие денег со счёта; добавление денег на счёт.
16. Собака. Необходимо реализовать: установку кличку собаки; ответ собаки на свою кличку словом “Гав”; кормление собаки, в ответ должна быть строка “Гав-гав”; исполнение команд “Сидеть” и “Стоять”.
17. Проверка орфографии. Необходимо реализовать: внесение новых слов в словарь; проверку слова на правописание; предложение об исправлении ошибки.
18. Форум. Необходимо реализовать: добавление темы; добавление сообщения в тему; вывод всех сообщений на одну тему.
19. Университет. Необходимо реализовать: зачисление студента по имени и среднему баллу, после зачисления ему выдаётся номер студенческого билета; посещение студентом занятий по имени и номеру билета.
20. Видеохостинг. Необходимо реализовать: добавление видео, содержащего название и видеоряд (строку текста); просмотр видео по названию; удаление видео по названию.
21. Вычислитель интеграла. Необходимо реализовать: назначение пределов интегрирование; назначение шага интегрирование; выбор метода: прямоугольники или трапеции.
22. Вычислитель суммы векторов. Необходимо реализовать: ввод первого вектора, ввод второго вектора, проведение суммирования, вывод результат.
23. Вычислитель комплексных корней. Необходимо реализовать: ввод числа; ввод степени корня; вывод каждого из корней N-ой степени.
24. Сортировщик. Необходимо реализовать: ввод массива; выбор направления сортировки; выбор способа сортировки; вывод результата.
25. Сравниватель строк. Необходимо реализовать: ввод первой строки; ввод второй строки; вывод всех отличных символов и их позиций.
26. Дискретное преобразование Фурье. Необходимо реализовать: ввод ряда для преобразования; проведение преобразования; вывод результата преобразования.
27. Свёртка. Необходимо реализовать: ввод ряда для свёртки; ввод ядра свёртки; копирование ядра свёртки в буфер; Копирование ядра свёртки из буфера; вывод результата свёртки.
28. Решение уравнение. Необходимо реализовать: в уравнении вида x+a=b ввод чисел a и b; решение уравнение; вывод результата.
29. Органайзер. Необходимо реализовать: добавление задачи; пометку задачи как выполненной; вывод всех невыполненных задач.
30. Шифровальщик информации. Необходимо реализовать: ввод информации, её название и пароля к ней; шифрование информации; вывод информации по названию: расшифровка информации по паролю и названию.