МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ

по 3 части курсовой работы
по дисциплине «Объектно-ориентированное программирование»
Тема: «Модель проектного отдела»
Вариант №7

Студент гр. 2301 _____ Комиссаров П.Е.

Преподаватель: _____ Новакова Н.Е.

Санкт-Петербург
2023

Цель работы

Целью курсовой работы является закрепление теоретических знаний и практических навыков разработки программного обеспечения на основе объектно-ориентированного подхода.

Выполнение курсовой работы должно базироваться на объектной модели, являющейся концептуальной базой для объектно-ориентированной парадигмы. В курсовой работе должны быть отражены ключевые концепций объектной модели: абстрагирование, передача сообщений, инкапсуляция, модульность, полиморфизм и наследование.

1. Формулировка задания

В проектном отделе разрабатывают новые изделия. Этим занимаются 4 проектировщика, каждый из которых специализируется на своём этапе проектирования. Весь процесс проектирования можно разбить на 4 этапа, за каждый из которых отвечает отдельный проектировщик.

Задания на проектирование поступают через каждые $A \pm B$ дней. Проектирование на каждом этапе занимает $C_k \pm D_k$ дней, где k — номер проектировщика (номер этапа).

Обычно проектирование протекает от 1-го этапа ко 2-му и т. д. Но может прийти срочный заказ и тогда необходимо выполнить в первую очередь его. 1-й проектировщик откладывает выполнявшийся ранее проект и начинает заниматься новым, а остальные пока продолжают заниматься прежними проектами. Когда материалы срочного проекта доходят до очередного проектировщика, он начинает заниматься им, откладывая предыдущую работу. После окончания срочной работы каждый проектировщик возвращается к своей прежней работе и заканчивает её.

Проанализировать работу над 10 проектами, из которых 2 оказываются срочными (выбор — случайным образом). Два последовательных срочных проекта выстраиваются в очередь.

Результат сбора статистики должен быть выведен в текстовый файл.

2. Теоретический аспект задачи

Описание задач:

1) Написать несколько классов и интерфейсов по заданной предметной области.

2) На основе созданных классов, написать программу, которая реализует модель проектного отдела.

3) Проектный отдел должен состоять из 4 проектировщиков, которые работают над обычными и срочными проектами. Поступление проектов и время их выполнения выбирается случайно (имеется промежуток в котором выбираются случайные числа)

4) Проанализировать работу над 10 проектами, из которых 2 оказываются срочными.

5) Результат сбора статистики должен быть выведен в текстовый файл.

Программа является классическим вариантом дискретно-событийной имитационной модели.

Для решения этой задачи можно использовать модель очереди, где каждый проектировщик будет представлять собой отдельную очередь.

В начале модели, каждый проектировщик будет иметь свою пустую очередь, и по мере ее заполнения должен выполнить проекты из нее.

1. Каждые A±B дней проектировщики получают новые задания.
2. Каждый проектировщик начинает работу над своим первым проектом из очереди.
3. Если приходит срочный проект, то он добавляется в начало очереди и проектировщик, который сейчас работает над своим проектом, откладывает его и начинает работать над срочным проектом.
4. Каждый проектировщик заканчивает свой проект и переходит к следующему в своей очереди.
5. Если очередь пуста, проектировщик просто ждет новых заданий.

6. Для сбора статистики можно использовать структуру данных, которая будет хранить информацию о времени начала и окончания работы над каждым проектом, а также о времени, которое было потрачено на каждый этап проектирования. Также будет записана краткая информация о проекте (его номер и срочность выполнения)

7. После проведения моделирования, результаты будут выведены в текстовый файл.

Этот подход позволяет учесть все описанные в задании факторы, такие как случайный порядок поступления заданий, отложенные проекты и многопоточность работы над проектами.

3. Формализация задачи

В программе должно быть реализовано:

1. Работа с проектировщиками. Каждый проектировщик является очередью. Каждый день проектировщик выполняет первый в очереди проект (при появлении строчного, он добавляется в начало очереди, поэтому проектировщик начинает работать над ним, откладывая текущий). Это все реализуется с помощью класса designer.

2. Работа с проектами. У каждого проекта есть параметры, статусы проработки каждого проекта, время выполнения на каждом этапе. Это все реализуется с помощью класса project.

3. Работа со сменой дней. Дни автоматически будут меняться в течение недели (новую неделю может запускать пользователь). Также будет реализован счетчик дней. Это все реализуется с помощью класса daysSwitch.

4. Генерация случайных чисел. Генератор будет отвечать за часть параметров проекта (время выполнения, дата появления, номер проекта из текстового файла). Генератор также будет отвечать за вероятность выпадения того или иного события. Это все реализуется с помощью класса randomGenerator.

5. Работа со статистикой. Все параметры проекта, время выполнения проекта на каждом этапе, время выполнения проекта в целом структурированно выводится в текстовый файл. Это все реализуется в классе statisticText.

В программе “Модель проектного отдела” было создано … классов (… из которых абстрактных) и … интерфейса.

Структура СМО модели проектного отдела представлена на Рисунок 1.

Диаграмма действий представлена на Рисунок 2.

Диаграмма классов представлена на Рисунок 3.

Структура СМО:

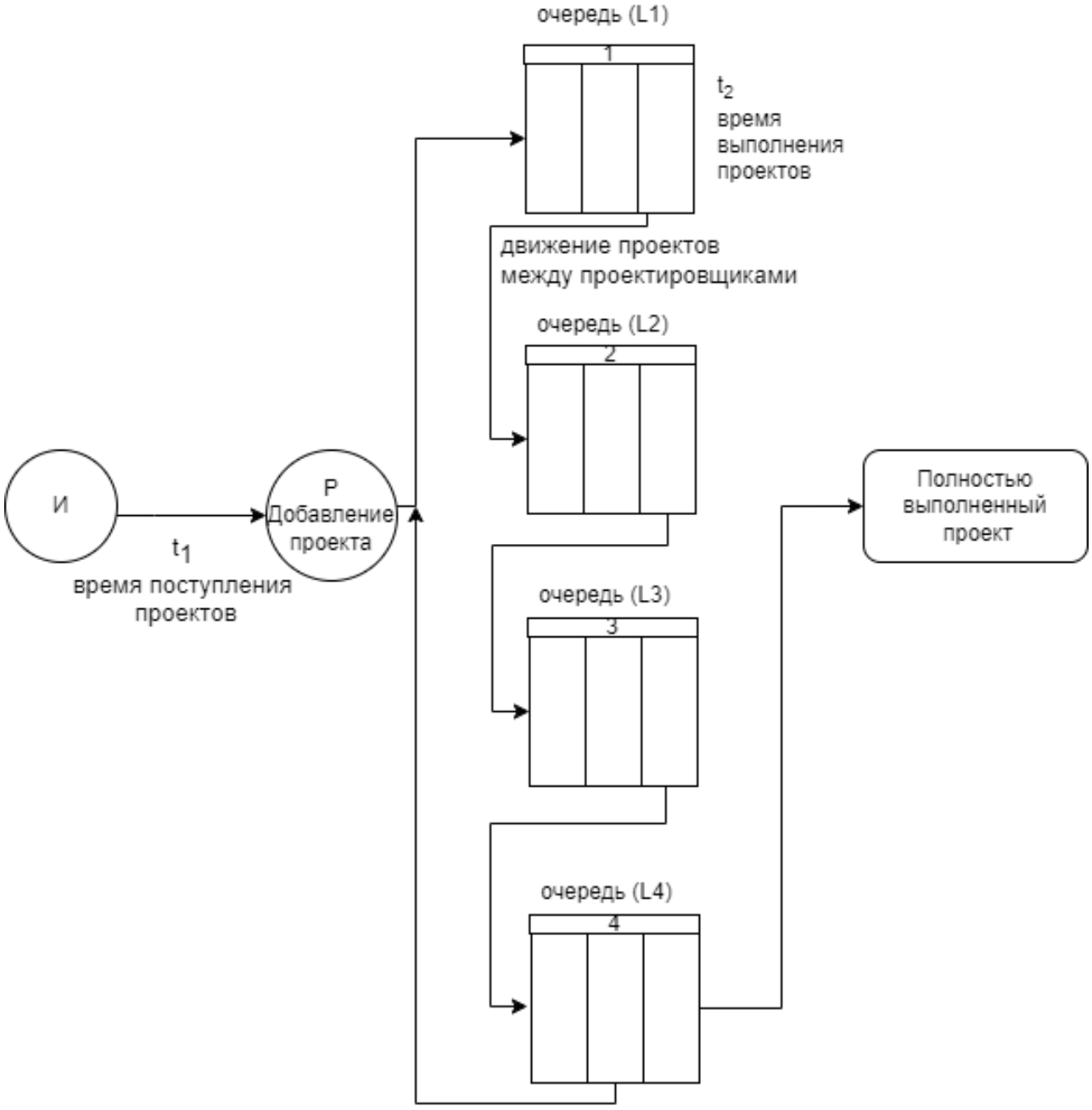


Рисунок 1 – Структура СМО модели проектного отдела

Диаграмма действий:

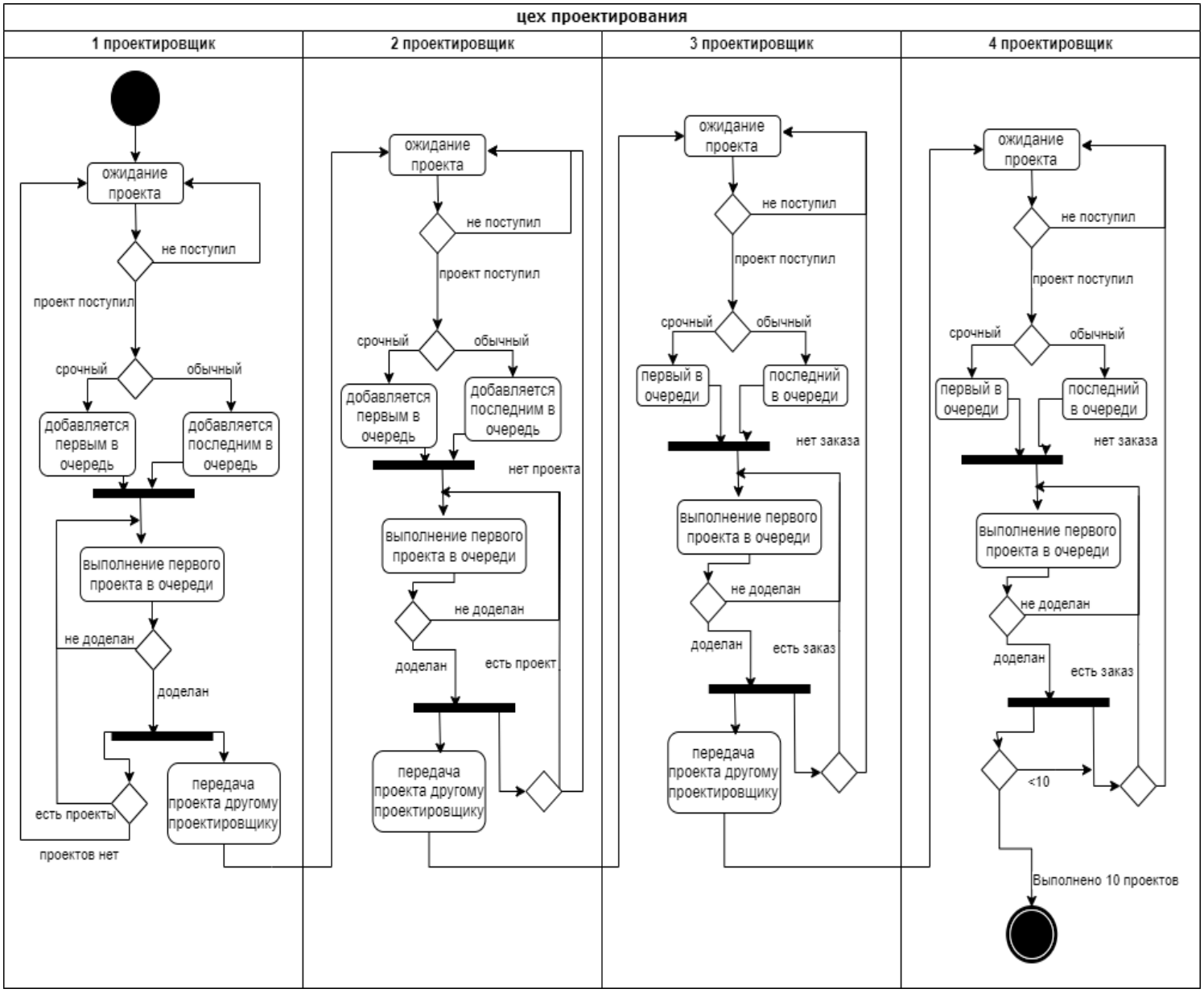


Рисунок 2 – Диаграмма действий

Диаграмма классов UML:

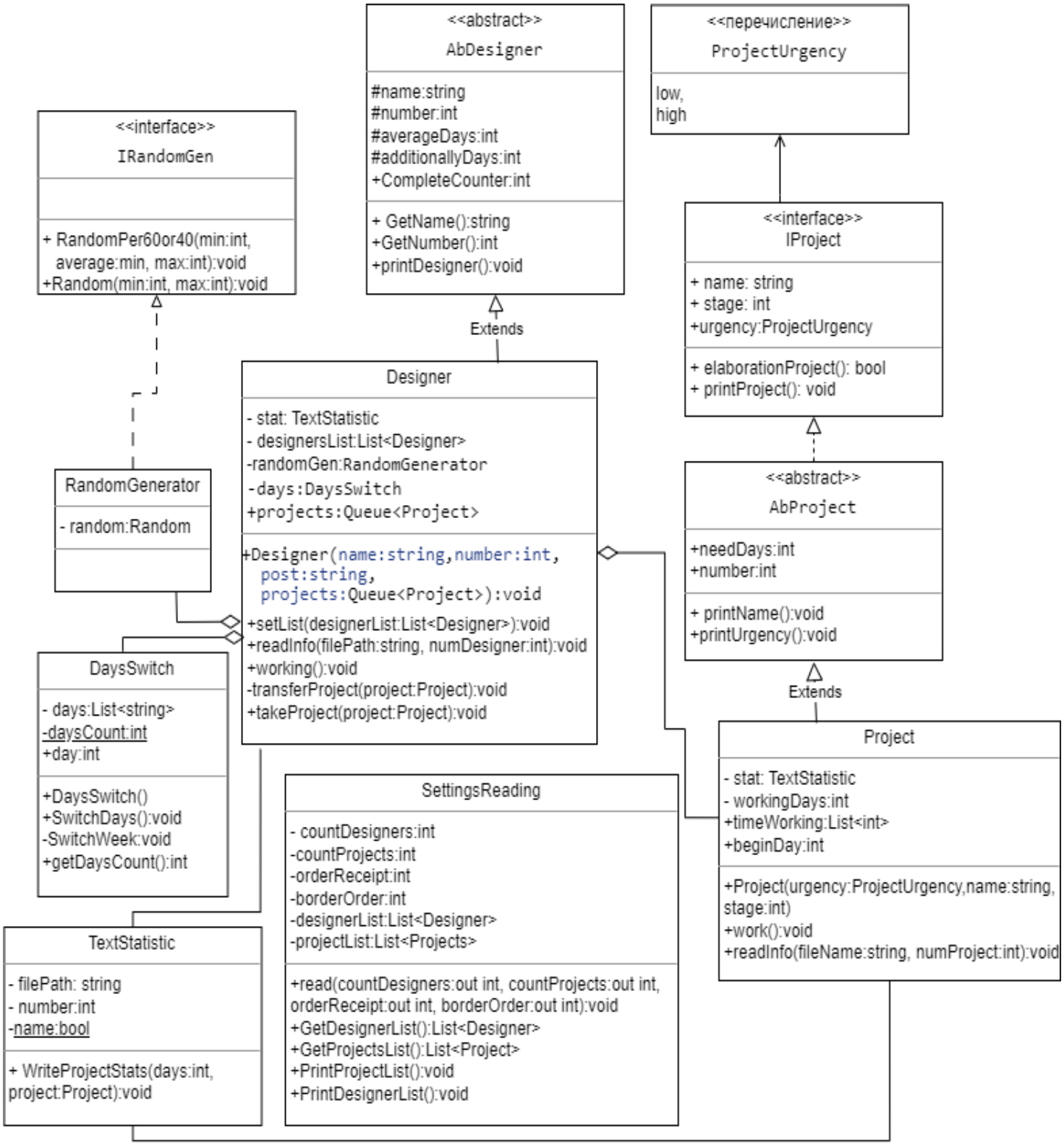


Рисунок 3 – Диаграмма классов UML

4. Спецификация программы

В программе будет использован переменный шаг времени, так как в моделировании и симуляции компьютерных систем переменный шаг времени используется, когда события происходят в непредсказуемом порядке, и временной интервал между ними не может быть предсказан.

Например, в моделировании системы очередей (что является моей реализацией), где задания приходят в непредсказуемом порядке.

В переменном шаге времени временной интервал между событиями может быть любым, от нескольких микросекунд до нескольких часов, и это может быть определено случайным образом, используя различные распределения вероятностей.

В текстовом файле будут перечислены вероятности выпадения того или иного события (в процентах).

Все события будут иметь свои временные границы.

Описание методов класса Program представлено в табл.1.1

Таблица 1.1

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
main	void	static	string[] args		Точка входа приложения

Описание методов интерфейса IRandomGen представлено в табл.1.2

Таблица 1.2

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
Random	int	-	int min, int average, int max	-	генератор случайных чисел
RandomPer60or40	int	-	-	-	генератор случайных чисел в процентах

Описание методов интерфейса IProject представлено в табл.1.3

Таблица 1.3

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
printProject	void	public	-	-	вывод проекта на консоль
elaborationProject	bool	public	-	-	проверка на готовность проекта

Описание методов абстрактного класса AbProject представлено в табл.1.4

Таблица 1.4

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Назначение
printName	void	public	-	название проекта на консоль
printUrgency	void	public	-	вывод срочности проекта
elaborationProject	bool	abstract public	-	проработка проекта
printProject	void	abstract public	-	вывод проекта на консоль

Описание методов абстрактного класса AbDesigner представлено в табл.1.5

Таблица 1.5

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
getName	string	public	-	-	получить фио проектировщика
getNumber	int	public	-	-	получить номер этапа проектировщика

Описание методов класса Project представлено в табл.1.6

Таблица 1.6

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Назначение
work	void	public	-	выполнение проекта
readInfo	void	public	string filename, int numProject	чтение информации о проекте из файла
printProject	void	public	-	вывод проекта на консоль

Описание методов класса Designer представлено в табл.1.7

Таблица 1.7

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Назначение
Designer	-	public	string name int number string post Queue<Project> projects	конструктор
setList	void	public	List<Designer> designerList	присваивание списка проектов
printDesigner	void	public	-	вывод проектировщика
readInfo	void	public	string filename int numDesigner	чтение информации из файла
working	void	public	-	работа
transferProject	void	private	Project project	передача проекта след проектировщику
takeProject	void	public	Project project	получение проекта первым проектировщиком

Описание методов класса RandomGenerator представлено в табл.1.8

Таблица 1.8

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
RandomPer60or40	int	public	int min, int average, int max	-	случайное число из интервала с вероятностью
Random	int	public	int min, int max	-	случайное число из интервала

Описание методов класса daysSwitch представлено в табл.1.9

Таблица 1.9

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
SwitchDay	void	public	-	-	смена дня
SwitchWeek	void	private	-	-	смена недели
getDaysCount	int	public	-	-	получить номер рабочего дня

Описание методов класса settingsReading представлено в табл.1.10

Таблица 1.10

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
read	void	public	-	int countDesigners int countProject int orderReceipt int borderOrder	чтение основных настроек программы
GetDesignersList	List<Designer>	public	-	-	создание списка проектировщиков
GetProjectsList	List<Project>	public	-	-	создание списка проектов
PrintProjectsList	void	public	-	-	вывести список проектов
PrintDesignersList	void	public	-	-	вывести список проектировщиков

Описание методов класса TextStatistic представлено в табл.1.11

Таблица 1.11

Метод	Возвращаемый тип	Модификатор доступа	Входные параметры	Выходные параметры	Назначение
WriteProjectStats	void	public	int days, Project project	-	Вывести статистику проекта в файл

Файлы, необходимые для программы(данные отделены запятой):

Название файла	Пример	Что хранится в файле
projects.txt	Согласование проекта схемы школы с губернатором,1,1	Название проекта, срочность, эта проектирования (с которого нужно начать)
designers.txt	Васурин Антон Сергеевич, 1,Проектировщик,4,2	ФИО, этап выполнения, должность, среднее время выполнения, возможное отклонение
random.txt	80,10,5,4,1	Вероятности выпадения событий
countWorkerAndWork.txt	4 10 4,2	Кол-во проектировщиков, кол-во проектов, средний день появления проекта, возможное отклонение
statistic.txt		Файл, в который будет выводиться статистика рабы программы

5. Руководство оператора

Пользователю на консоль выводится информация о проектировщиках и проектах, которые используются в программе.

Далее пользователю предлагается выбор: при появлении нового проекта останавливать программу (чтобы проверять ход работы проектировщиков) или оставить автоматическую работу программы.

На консоль выводится информация о каждом появлении проекта и проектировщике, выполняющем проект: если у проектировщика нет проекта, то он его ожидает; если проект выполняется, то выводится степень его выполнения; когда проект выполнен, выводится сообщение, что проект готов к передаче на следующий этап; после выполнения проекта 4 проектировщиком выводится сообщение, что проект полностью выполнено. Когда завершается 10 проектов, программа заканчивает свою работу.

При полном выполнении проекта в текстовый файл выводится статистика проекта: название, срочность, время выполнения, время работы на каждом этапе, очередность поступившего проекта.

6. Руководство программиста

Характеристика программы:

Программа является консольным приложением Console App (.NET Framework), созданным в среде Microsoft Visual Studio. Используемый язык программирования - C#

Выходные и выходные данные

Входными данными являются текстовые файлы: со списком проектов, с проектировщиками, с продолжительностью дня, с вероятностями событий.

Выходными данными являются сообщения о работе проектировщиков на консоли. Также вывод статистики по проектам в текстовый файл: параметры проекта, время выполнения на каждом этапе, дни начала и конца проектирования.

7. Контрольный пример

Далее представлены результаты выполнения программы (Рисунки 1, 2, 3).

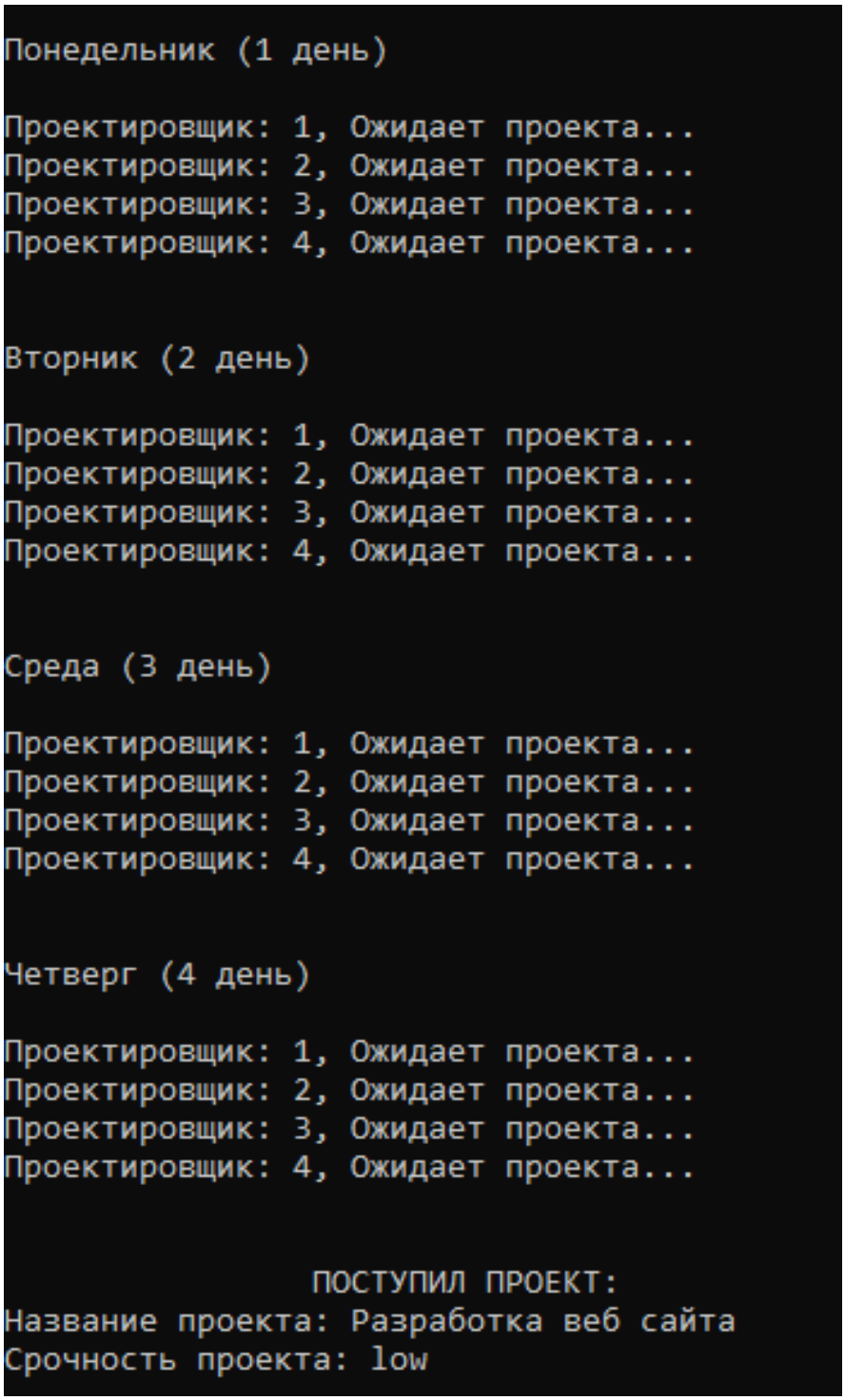
```
                СПИСОК ВСЕХ ПРОЕКТИРОВЩИКОВ:

1. ФИО: Васурин Антон Сергеевич
Порядковый номер: 1

2. ФИО: Добролюбов Григорий Павлович
Порядковый номер: 2

3. ФИО: Иванов Олег Александрович
Порядковый номер: 3

4. ФИО: Самсонов Игорь Алексеевич
Порядковый номер: 4


                СПИСОК ВСЕХ ПРОЕКТОВ:

1. Название проекта: Согласование проекта схемы Школы с губернатором
Срочность проекта: high

2. Название проекта: Разработка плана парка
Срочность проекта: low

3. Название проекта: Контроль над реализацией плана
Срочность проекта: high

4. Название проекта: Оптимизация дизайна здания Суда
Срочность проекта: low

5. Название проекта: Составить план на следующую неделю
Срочность проекта: low

6. Название проекта: Разработка веб сайта
Срочность проекта: low

7. Название проекта: Разработка интерфейса пользователя
Срочность проекта: low

8. Название проекта: Выполнение правок по логотипу для "Вкусно и точка"
Срочность проекта: low

9. Название проекта: Создание логотипа для автомобильной компании
Срочность проекта: low

10. Название проекта: Обсуждение конспекта новой рекламной компании
Срочность проекта: low

Нажмите ENTER, чтобы продолжить
```

Рисунок 1 – вывод списков проектировщиков и проектов

Как видно из Рисунка 1, программа имеет 4 проектировщиков, у которых есть ФИО и их порядковый номер (номер этапа проектирования)

Также есть 10 проектов, 2 из которых имеют высокую срочность выполнения.

Для начала работы проектировщиков нужно нажать Enter.



Риунок 2 – ожидание и появление проекта

Как видно из Рисунка 2, пока у проектировщиков нет работы, они просто “Ожидают проекта…”.

Проекты поступают в случайный день, в случайном порядке. Если проект поступил, то выводится соответствующее сообщение

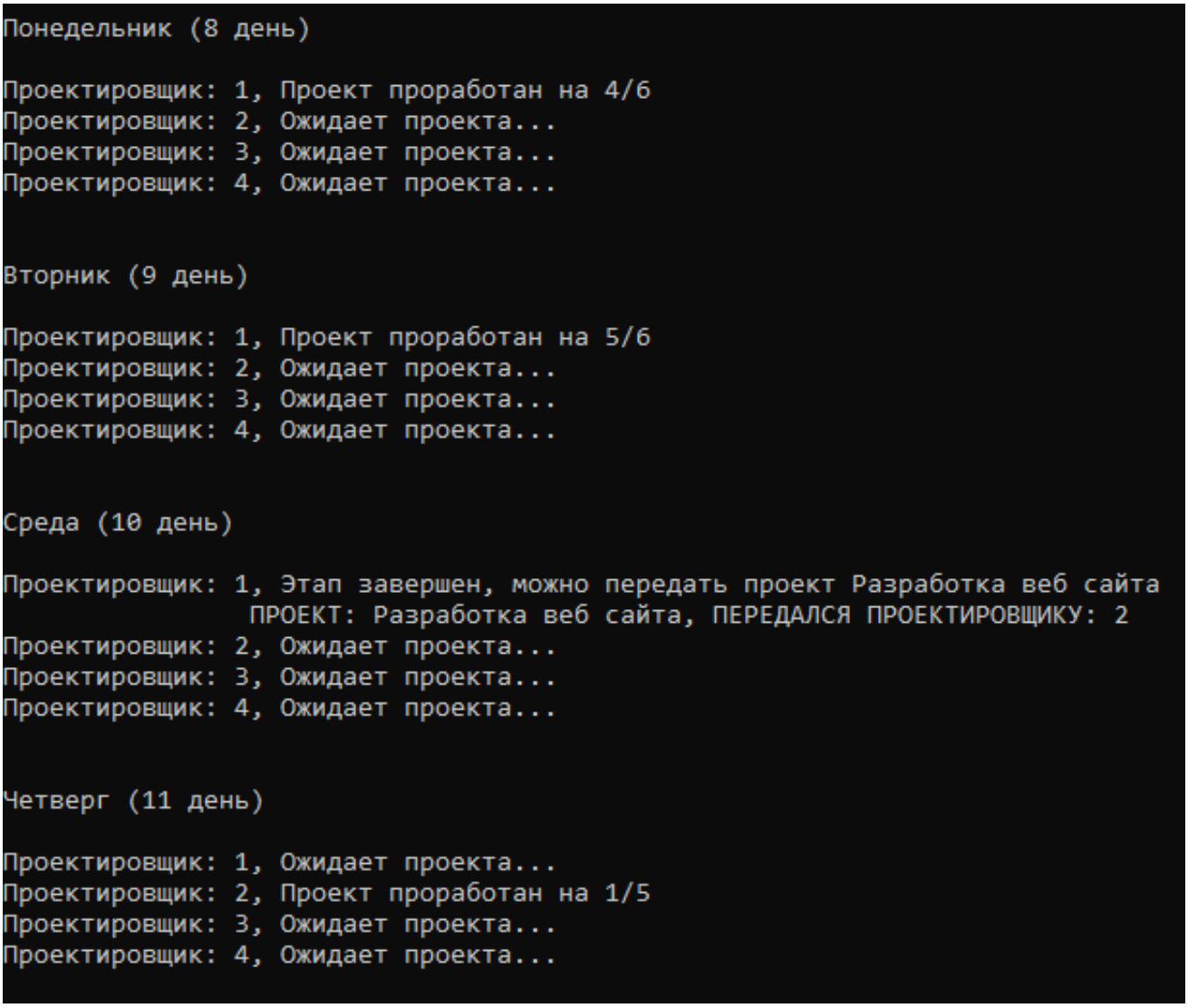


Рисунок 3 – работа над проектом

Как видно из Рисунка 3, каждый проектировщик вырлдняет проект в своем темпе.

Когда проектировщик сделал свою чатсь проекта, он передает проект следующему.

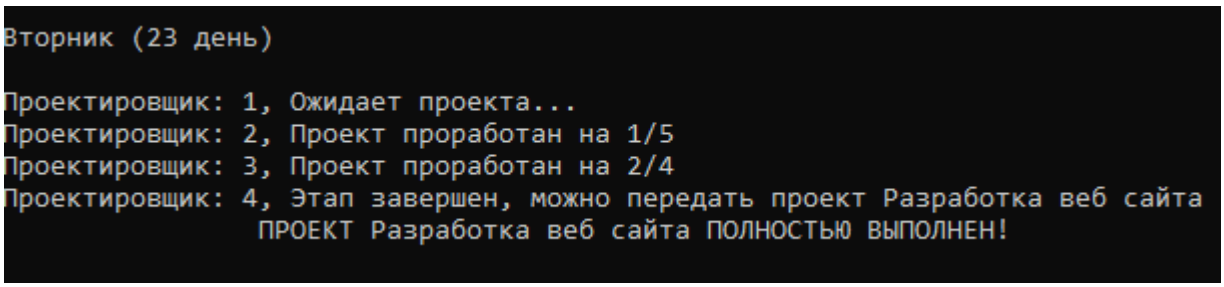


Рисунок 4 – проект полностью выполнен

Как видно из Рисунка 4, когда последний проектировщик заканчивает свою работу, выводится соответствующее сообщение

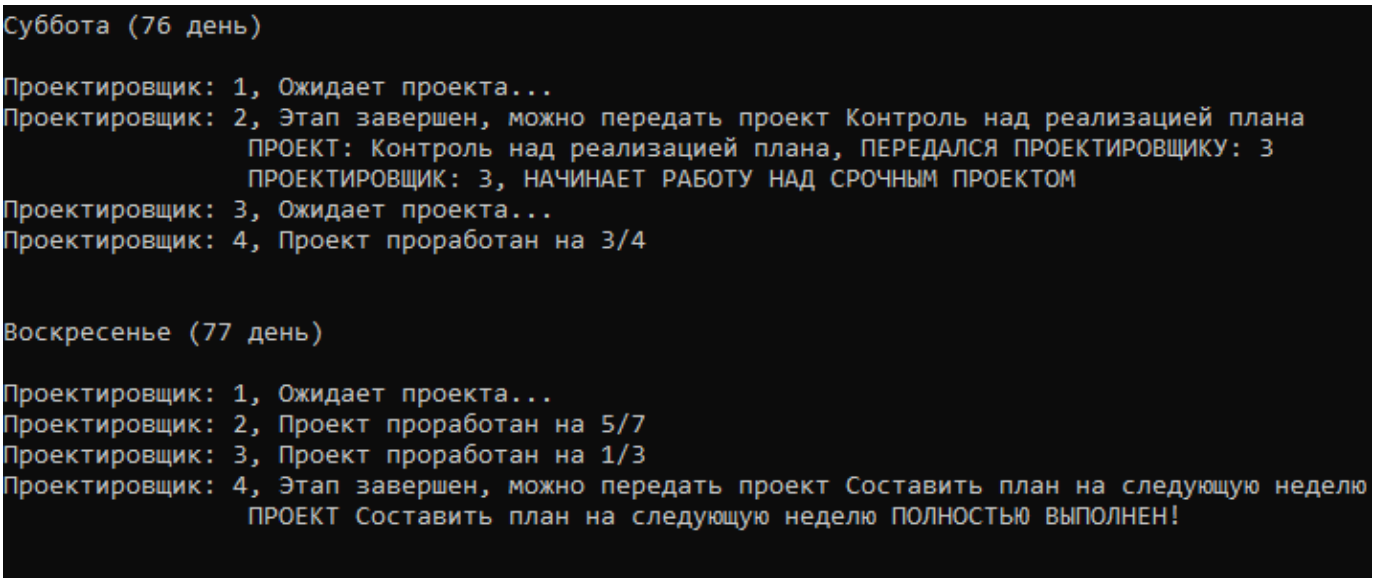


Рисунок 5 – передача срочного проекта

Как видно из Рисунка 5, 2 проектировщик выполнил срочный проект и продолжил заниматься несрочным (после передачи срочного, он прорабатывает несрочный с 4/7)

```
5. Полностью выполнен проект: Разработка плана парка
   С уровнем срочности: low
   День начала проектирования:31
   День окончания проектирования:55
   Колличество дней работы над проектом на каждом этапе:4дн,6дн,4дн,4дн
   По счету проект поступил: 5

6. Полностью выполнен проект: Согласование проекта схемы Школы с губернатором
   С уровнем срочности: high
   День начала проектирования:45
   День окончания проектирования:62
   Колличество дней работы над проектом на каждом этапе:4дн,5дн,3дн,6дн
   По счету проект поступил: 7

7. Полностью выполнен проект: Разработка веб сайта
   С уровнем срочности: low
   День начала проектирования:38
   День окончания проектирования:71
   Колличество дней работы над проектом на каждом этапе:4дн,8дн,3дн,7дн
   По счету проект поступил: 6
```

Рисунок 6 – Пример вывода статистики

Как видно из Рисунка 6, статистика записывает проекты в порядке их выполнения. Проект содержит параметры: название, срочность, день начала проектирования, день окончания проектирования, рвемя работы на каждом этапе проектировния (не считая дни, которые он ждал своей очереди), номер поступления

Также видно, что проект, который выполнился шестым, а поступил седьмым является срочным. Поступивший шестым проект пропустил срочный перед собой, что соответствует заданию.

8. Листинг программы

class Program

```
namespace course3
{
    class Program
    {
        static void Main(string[] args)
        {
            int countDesigners;     //кол-во проектировщиков
            int countProject;       //кол-во проектов
            int numberProject = 0;
            int orderReceipt;       //среднее время для появления проекта
            int borderOrder;        //погрешность во времени
            int orderReceiptDay;    //фактический день появления проекта
            bool checkProjects = false;

            settingsReading settings = new settingsReading();       //настройки
            DaysSwitch days = new DaysSwitch();                     //смена дней
            RandomGenerator random = new RandomGenerator();         //генератор чисел


            List<Designer> designersList = new List<Designer>();      //список проект
ировщиков
            List<Project> projectsList = new List<Project>();         //список проект
ов
            List<Project> projectsListTemp = new List<Project>();     //список проект
ов (который уменьшается)

            settings.read(out countDesigners, out countProject, out orderReceipt, out
 borderOrder); //чтение настроек

            designersList = settings.GetDesignersList();        //получение всех спис
ков
            projectsList = settings.GetProjectsList();
            foreach(Project project in projectsList)
            {
                projectsListTemp.Add(project);
            }


            settings.PrintDesignersList();      //вывести списки на консоль
            settings.PrintProjectsList();

            Console.WriteLine("Нажмите ENTER, чтобы продолжить");
            Console.ReadLine();




            Console.WriteLine("Нужно ли останавливать программу при появлении новых п
роектов? (если надо, напишите 'ДА')");

            if (Console.ReadLine() == "ДА") checkProjects = true;

            Console.WriteLine("\n\n\n");

            orderReceiptDay = random.RandomPer60or40(orderReceipt - borderOrder, orde
rReceipt, borderOrder + orderReceipt); //день выпадения
```

```
            while (designersList[designersList.Count()-
1].CompleteCounter!= countProject)    //пока все проекты не сделаны
            {

                if (days.day == orderReceiptDay && projectsListTemp.Count > 0)    //е
сли проект поступил и проекты не закончились
                {
                    int index = random.Random(0, projectsListTemp.Count() - 1);
                    projectsListTemp[index].number = numberProject += 1;
                    designersList[0].takeProject(projectsListTemp[index]);       //пе
редача проекта 1 проектировщику
                    Console.WriteLine("\t\tПОСТУПИЛ ПРОЕКТ: ");

                    projectsListTemp[index].printProject();
                    if (checkProjects == true)
                    {
                        Console.WriteLine("Нажмите ENTER, чтобы продолжить");
                        Console.ReadLine();
                    }
                    orderReceiptDay = days.day % random.RandomPer60or40(orderReceipt
- borderOrder, orderReceipt, borderOrder + orderReceipt); //день выпадения
                    if (orderReceiptDay == 0) orderReceiptDay = days.day;
                    if (projectsListTemp[index].urgency == ProjectUrgency.high)
                        Console.WriteLine("\t\tПРОЕКТИРОВЩИК: {0}, НАЧИНАЕТ РАБОТУ НА
Д СРОЧНЫМ ПРОЕКТОМ\n", 1);
                    projectsListTemp.RemoveAt(index);        //удаление из списка про
ектов (temp)
                }

                days.SwitchDay();       //смена дня
                for(int i=0; i < designersList.Count() ; i++)
                {
                    designersList[i].working();
                    Thread.Sleep(50);
                }
                Console.WriteLine("\n");

            }
            Console.WriteLine("\t\tВСЕ ПРОЕКТЫ ВЫПОЛНЕНЫ!\n", 1);
            Console.ReadLine();

        }
    }
}
```

interface IProject

```
namespace course3
{
    interface IProject
    {
        string name { get; set; }   //название проекта
        int stage { get; set; }     //этап выполнения
        ProjectUrgency urgency { get; set; }    //срочность
        void printProject();    //вывести проект на экран
        bool elaborationProject();  //степень проработки

    }
}
```

interface IRandomGen

```
namespace course3
{
    interface IRandomGen
    {
        int RandomPer60or40(int min, int average, int max);
        int Random(int min, int max);
    }
}
```

abstract class AbProject

```
namespace course3
{
    abstract class AbProject:IProject
    {

        public string name { get; set; }        //название проекта
        public int stage { get; set; }          //выполняющий проектировщик
        public ProjectUrgency urgency { get; set; } //срочность проекта
        public int needDays { get; set; }       //необходимое кол-
во дней на отработку (на каждом этапе разное)

        public int number { get; set; }         //номер проекта

        public void printName() //вывод названия
        {
            Console.WriteLine("Название проекта: {0}", name);
        }

        public void printUrgency()  //вывод срочности
        {
            Console.WriteLine("Срочность проекта: {0}", Enum.GetName(typeof(ProjectUr
gency), urgency));
        }
        abstract public bool elaborationProject();     //проработка проекта на этапе
        abstract public void printProject();    //вывод всего проекта на консоль
    }
}
```

abstract class AbDesigner

```
namespace course3
{
    abstract class AbDesigner
    {
        protected  int  number { get; set; }        //порядкой номер проектировщика
        protected  string name { get; set; }        //имя проектировщика
        protected int averageDays { get; set; }     //среднее время выполнения проект
a
        protected int additionallyDays { get; set; }//погрешность времени выполнения
        public int CompleteCounter { get; set; }    //счетчик выполненных проектов



        public string  getName()    //получить имя
        {
```

```
            return name;
        }
        public int getNumber ()     //получить порядковый номер
        {
            return number;
        }

    }
}
```

class Project

```
namespace course3
{
    class Project : AbProject
    {
        TextStatistic stat = new TextStatistic();
        int workingDays = 0;        //отработанные дни над проектом
        public List<int> timeWorking = new List<int>();
        public int beginDay;
        public Project(ProjectUrgency urgency, string name, int stage = 0)
        {
            this.urgency = urgency;
            this.name = name;
            this.stage = stage;
        }
        override public bool elaborationProject()      //проработка проекта на этапе
        {
            if (workingDays == needDays)    //если проработан
            {
                timeWorking.Add(workingDays);
                //if (stage == 4) { stat.WriteProjectStats(); }

                workingDays = -1;
                stage += 1;
                needDays = 0;
                return true;
            }
            else return false;

        }
        public void work()      //работа над проектом
        {
            if (workingDays == 1)      //если проект только передался, нужно начать е
го со след. дня
            {
                workingDays += 1;
                Console.WriteLine("Ожидает проекта...");
                return;
            }
            workingDays += 1;
            if (workingDays== needDays)
            {
                Console.WriteLine("Этап завершен, можно передать проект {0}", name);
                return;
            }
            Console.WriteLine("Проект проработан на {0}/{1}", workingDays, needDays);
        }
        public void readInfo(string fileName, int numProject)          //чтение инфор
мации о дизайнере из файла
```

```
        {
            using (StreamReader reader = new StreamReader(fileName))//файл для чтения
            {
                string line;

                while (numProject > 0) { line = reader.ReadLine(); numProject -= 1; }
                line = reader.ReadLine();  //считыванием строку

                string[] parts = line.Split(',');   //делим на части с помощью запято
                                                    //записываем студента из файла
                name = parts[0];
                if (!Enum.TryParse(parts[1], out ProjectUrgency urgencyOut))
                {
                    throw new Exception("Не удалось преобразовать строку в перечислен
ие Urgency");
                }
                urgency = urgencyOut;
                stage = int.Parse(parts[2]);
                reader.Close();
            }
        }
        public override void printProject()
        {
            printName();
            printUrgency();
            Console.WriteLine();
        }

    }
}
```

class Designer

```
namespace course3
{
    class Designer:AbDesigner       //работа с проектировщиком
    {

        TextStatistic stat = new TextStatistic();
        List<Designer> designersList;   //список всех проектировщиков
        RandomGenerator randomGen = new RandomGenerator();  //генератор чисел
        DaysSwitch days = new DaysSwitch();

        public Queue<Project> projects;     //очередь из проектов
        public Designer(string name, int number, string post, Queue<Project> projects
)
        {
            this.name = name;       //фио
            this.number = number;   //порядковый номер

            this.projects = projects;   //присваивание очереди
            CompleteCounter = 0;        //счетчик выполненных проектов
        }
        public void setList(List<Designer> designersList)
        {
            this.designersList = designersList;
        }   //присваивание списка проектов
        public void printDesigner()     //вывод проектировщика
        {
            Console.WriteLine("ФИО: {0}\nПорядковый номер: {1}", name, number);
```

```
            Console.WriteLine();
        }
        public void readInfo(string fileName, int numDesigner)   //чтение информации
о дизайнере из файла
        {
            using (StreamReader reader = new StreamReader(fileName))   //файл для чте
ния
            {
                string line;
                while (numDesigner > 0) { line = reader.ReadLine(); numDesigner -
= 1; }
                line = reader.ReadLine();  //считыванием строку

                    string[] parts = line.Split(',');   //делим на части с помощью за
пятой
                                                        //записываем проектировщика
                    name = parts[0];
                    number =int.Parse(parts[1]);

                averageDays = int.Parse(parts[3]);  //среднее кол-во дней выполнения
                additionallyDays = int.Parse(parts[4]); //погрешность дней
                reader.Close();

            }
        }

        public void working()       //работа
        {

            if (projects.Count() == 0)      //если очередь пустая
            {

                Console.WriteLine("Проектировщик: {0}, Ожидает проекта...", number);
                return;
            }
            Project activeProject = projects.Peek();    //активный проект тот, которы
й первый в очереди

            if (activeProject.needDays == 0)        //установка кол-
ва дней для работы над проектом
            {
                activeProject.needDays=randomGen.RandomPer60or40(averageDays - additi
onallyDays, averageDays, additionallyDays + averageDays);
                //70% на то, что выпадет среднее значение, 30% на выпадение из диапаз
она (не включая среднее)
            }

            Console.Write("Проектировщик: {0}, ", number);
            activeProject.work();   //проект прорабатывается
            if (number==designersList.Count() && activeProject.elaborationProject())
      //если проект полностью выполнен (4 проектировщик сделал)
            {

                Console.WriteLine("\t\tПРОЕКТ {0} ПОЛНОСТЬЮ ВЫПОЛНЕН!", projects.Peek
().name);
                CompleteCounter += 1;

                stat.WriteProjectStats(days.getDaysCount(), activeProject);
                projects.Dequeue();
                return;
            }
```

```
            if (activeProject.elaborationProject()) //если проект сделан
            {

                transferProject(activeProject); //передача проекта другому проектиров
щику
                projects.Dequeue();             //удаления сделанного проекта
                CompleteCounter += 1;           //счетчик выполненных проектов+=1
                return;
            };

        }
        void transferProject(Project project)   //передача проектов
        {
            Console.WriteLine("\t\tПРОЕКТ: {0}, ПЕРЕДАЛСЯ ПРОЕКТИРОВЩИКУ: {1}", proje
ct.name, designersList[project.stage - 1].number);
            if (project.urgency==ProjectUrgency.low) designersList[project.stage-
1].projects.Enqueue(project);
            //если проекта несрочный, то просто вставить его в конец очереди
            else
            {
                Queue<Project> projectsTemp = new Queue<Project>();  //создание новой
 очереди для перезаписи (чтобы вставить проект в начало)
                if (designersList[project.stage - 1].projects.Count() > 0)   //если о
чередь не пустая, то нужно проверить, какой проект первый в очереди
                {
                    if (designersList[project.stage - 1].projects.Peek().urgency == P
rojectUrgency.high)  //если активный проект тоже срочный
                    {
                        projectsTemp.Enqueue(designersList[project.stage - 1].project
s.Dequeue());   //активный остается активным
                        projectsTemp.Enqueue(project);              //новый срочный п
осле активного
                        foreach (Project item in projects)          //перезапись прое
ктов
                        {
                                projectsTemp.Enqueue(item);
                        }
                        designersList[project.stage - 1].projects.Clear();
                        foreach (Project item in projectsTemp)
                        {
                            designersList[project.stage - 1].projects.Enqueue(item);
                        }
                        return;
                    }
                }

                Console.WriteLine("\t\tПРОЕКТИРОВЩИК: {0}, НАЧИНАЕТ РАБОТУ НАД СРОЧНЫ
М ПРОЕКТОМ", designersList[project.stage - 1].number);

                projectsTemp.Enqueue(project);              //вставка срочного проекта
 в начало доп. очереди
                foreach (Project item in designersList[project.stage-1].projects)
                {
                    projectsTemp.Enqueue(item);     //перезапись всех остальных проек
тов в доп очередь
                }
                designersList[project.stage-
1].projects.Clear();    //очиста очереди у след проектировщика
                foreach (Project item in projectsTemp)
                {
```

```
                    designersList[project.stage-
1].projects.Enqueue(item);  //перезапись проектов проектировщика из доп очереди
                }

            }
        }

        public void takeProject(Project project)        //получение проекта (для перв
ого проектировщика)
        {
            project.beginDay = days.getDaysCount();

            if (project.urgency == ProjectUrgency.low) projects.Enqueue(project);   /
/если несрочный, то в конец очереди
            else        //если срочный, вставить в начало очереди
            {

                Queue<Project> projectsTemp = new Queue<Project>();
                if (projects.Count() != 0)
                {
                    if (projects.Peek().urgency == ProjectUrgency.high)  //если актив
ный проект тоже срочный
                    {
                        projectsTemp.Enqueue(projects.Peek());   //активный остается
активным
                        projectsTemp.Enqueue(project);              //новый срочный п
очле активного
                        projects.Dequeue();
                    } else projectsTemp.Enqueue(project);      //еси очередь пуста ил
и первый в очереди несрочный, то новый проект становится активным
                } else projectsTemp.Enqueue(project);      //еси очередь пуста или пе
рвый в очереди несрочный, то новый проект становится активным


                foreach (Project item in projects)
                {
                    projectsTemp.Enqueue(item);
                }

                projects.Clear();
                foreach (Project item in projectsTemp)
                {
                    projects.Enqueue(item);
                }

            }
        }
    }
}
```

class RandomGenerator

```
namespace course3
{
    class RandomGenerator:IRandomGen        //генеработр случайного числа
    {
        Random random = new Random((int)DateTime.UtcNow.Ticks);
```

```
        public int RandomPer60or40(int min, int average, int max)   //выпадение того
или иного события с вероятностью (60 на 40)
        {
            Random random = new Random((int)DateTime.UtcNow.Ticks);

            int num = random.Next(1, 11); // генерируем случайное число между 1 и 10

            if (num <= 7)      //вероятность выпадения 70%
            {
                return average;
            }
            else               //вероятность 30%
            {
                random = new Random((int)DateTime.UtcNow.Ticks);
                num = random.Next(min, max+1);

                while (num==average) num = random.Next(min, max + 1);   //исключаем м
омент выпадения числа с вероятностью 60%
                return  num;
            }

        }

        public int Random(int min, int max)
        {
            Random random = new Random((int)DateTime.UtcNow.Ticks);
            int num = random.Next(min, max + 1);    //генерация случайного в диапазон
е числа
            return num;
        }
    }
}
```

enum ProjectUrgency

```
namespace course3
{
    enum ProjectUrgency     //срочность проекта
    {
        low,               //несрочный
        high               //срочный
    }
}
```

class DaysSwitch

```
namespace course3
{
    class DaysSwitch        //смена дней
    {
        List<string> days = new List<string>();
        static int daysCount=0;
        public int day { get; set; }


        public DaysSwitch()     //создание списка дней ннедели
        {

            days.Add("Понедельник");
```

```
            days.Add("Вторник");
            days.Add("Среда");
            days.Add("Четверг");
            days.Add("Пятница");
            days.Add("Суббота");
            days.Add("Воскресенье");
            days.Add("Неделя окончена");
            day = 0;
        }
        public void SwitchDay()     //смена дней
        {
            if (day==7) Console.WriteLine("{0}\n", days[day]);
            else
                Console.WriteLine("{0} ({1} день)\n", days[day], daysCount+1);
            day += 1;
            Thread.Sleep(50);
            daysCount += 1;
            if (day == 8) SwitchWeek(); //когда вся неделя прошла, запустить смену не
дели
        }
         void SwitchWeek()      //смена недели
        {
            daysCount -= 1;
            Console.WriteLine();
            Thread.Sleep(100);
            day = 0;
            SwitchDay();
        }
        public int getDaysCount()
        {

           return daysCount;
        }
    }
}
```

class settingsReading

```
namespace course3
{
    class settingsReading
    {
        int countDesigners { get; set; }     //кол-во проектировщиков
        int countProject;       //кол-во проектов
        int orderReceipt;
        int borderOrder;


        List<Designer> designersList = new List<Designer>();      //списки проектиров
щиков и проектов
        List<Project> projectsList = new List<Project>();


        public void read(out int countDesigners, out int countProject, out int orderR
eceipt, out int borderOrder)
        {

            StreamReader reader = new StreamReader("countWorkerAndWork.txt");   //фай
л для чтения настроек
            countDesigners = int.Parse(reader.ReadLine());
```

```
            countProject = int.Parse(reader.ReadLine());
            string line = reader.ReadLine();
            string[] parts = line.Split(',');
            orderReceipt = int.Parse(parts[0]);
            borderOrder = int.Parse(parts[1]);
            reader.Close();

            this.countDesigners = countDesigners;
            this.countProject = countProject;
            this.designersList = designersList;
            this.projectsList = projectsList;
            this.borderOrder = borderOrder;
            this.orderReceipt = orderReceipt;
        }
        public List<Designer> GetDesignersList()
        {
            for (int i = 0; i < countDesigners; i++)       //создание списка проектир
овщиков
            {
                Queue<Project> queue = new Queue<Project>();
                Designer designer = new Designer("", 0, "", queue);
                designersList.Add(designer);
                designersList[i].readInfo("designers.txt", i);
            }
            for (int i = 0; i < countDesigners; i++)       //присвоение каждому проек
тировщику списка сотрудников
            {

                designersList[i].setList(designersList);
            }
            return designersList;
        }

        public List<Project> GetProjectsList()
        {
            for (int i = 0; i < countProject; i++)       //создание списка проектов
            {
                Project project = new Project(0, "");
                projectsList.Add(project);
                projectsList[i].readInfo("projects.txt", i);
            }
            return projectsList;
        }

        public void PrintProjectsList()
        {
            Console.WriteLine("\n\t\tСПИСОК ВСЕХ ПРОЕКТОВ:\n");       //вывод всех пр
оектов на экран
            for (int i = 0; i < countProject; i++)
            {
                Console.Write("{0}. ", i + 1);
                projectsList[i].printProject();
            }
        }
        public void PrintDesignersList()
        {
            Console.WriteLine("\n\t\tСПИСОК ВСЕХ ПРОЕКТИРОВЩИКОВ:\n");        //вывод
 всех проектировщиков на экран

            for (int i = 0; i < countDesigners; i++)
            {
```

```
                Console.Write("{0}. ", i + 1);
                designersList[i].printDesigner();

            }
        }


    }
}
```

class TextStatistic

```
namespace course3
{
    class TextStatistic
    {
        string filePath = "statistic.txt";
        int number = 1;
        static bool name = false;



        public void WriteProjectStats(int days, Project project)
        {
            using (StreamWriter writer = File.AppendText(filePath))
            {
                if (name == false)
                {
                    writer.WriteLine("");
                    writer.WriteLine("_____________________________________________________
_____________________________________________");
                    writer.WriteLine("СБОР СТАТИСТИКИ О ПРОЕКТАХ:");
                    writer.WriteLine("_____________________________________________________
_____________________________________________");
                    name = true;
                }
                writer.WriteLine("_____________________________________________________
_____________________________________________");
                writer.WriteLine("{0}. Полностью выполнен проект: {1}\n   С уровнем с
рочности: {2}\n   День начала проектирования:{3}\n   День окончания проектирования:{4
}", number, project.name, project.urgency, project.beginDay + 1, days);
                writer.WriteLine("   Колличество дней работы над проектом на каждом э
тапе:{0}дн,{1}дн,{2}дн,{3}дн", project.timeWorking[0], project.timeWorking[1], projec
t.timeWorking[2], project.timeWorking[3]);
                writer.WriteLine("   По счету проект поступил: {0}", project.number);
                writer.WriteLine("_____________________________________________________
_____________________________________________");
                number += 1;
            }
        }
    }
}
```

Список используемой литература

1. Microsoft Learn – сеть разработчиков Microsoft. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: -)

2. Moodle – виртуальный образовательный кластер: https://vec.etu.ru/moodle/course/view.php?id=14794 (дата обращения: 24.11.2023)

3. Горячев А.В., Кравчук Д.К., Новакова Н.Е. Объектно-ориентированное моделирование. Учеб. Пособие. СПб.: Изд-во СПбГЭТУ “ЛЭТИ”, 2010. (дата обращения: 3.12.2023)

4. Буч Г., Ивар Я. Введение в UML от создателей языка. Издательство: ДМК Пресс, 2015 (дата обращения: 14.12.2023)