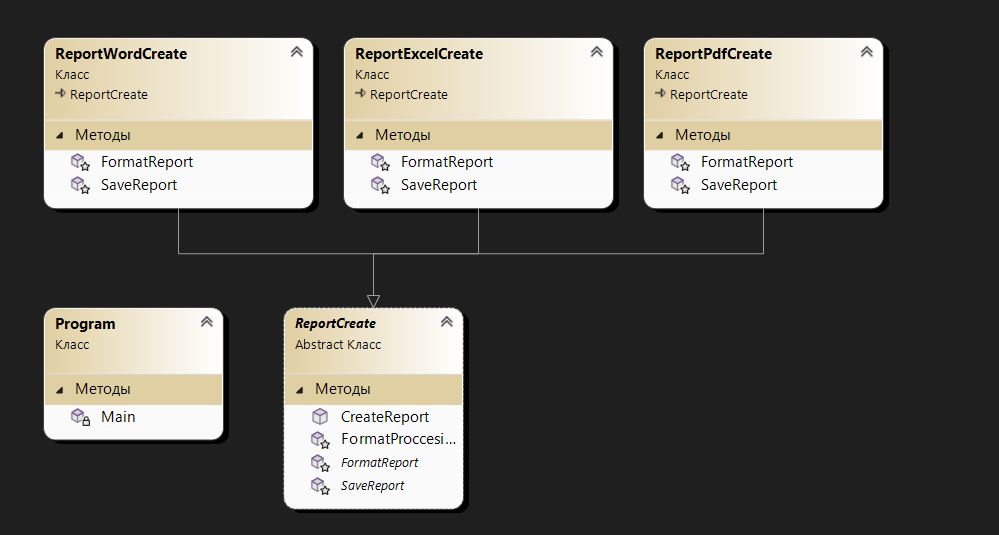
**Отчёт по ПЗ 03.01 Работа с поведенческими паттернами**

**Выполнил Хамитов Айнур 3пк2**

**Вариант 8**

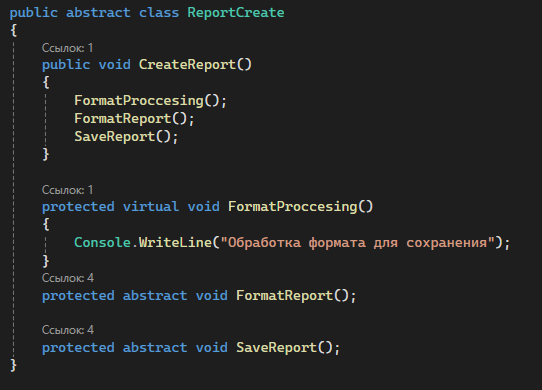
**1. CreateReportApp**

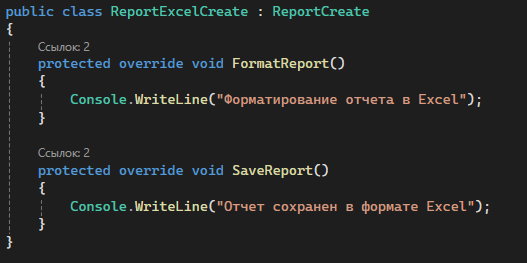
1.1 Генератор отчетов, где отчеты могут быть различных форматов (PDF, Word, Excel) с использованием шаблонного метода для обработки общих шагов.

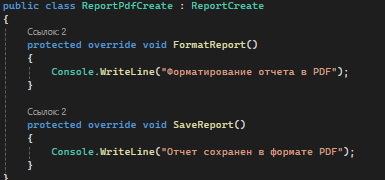
1.2 UML диаграмма

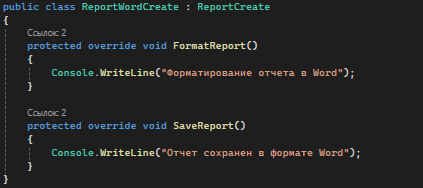
1.3 Паттерн Шаблонный метод был успешно применен для реализации генератора отчетов, где отчеты могут быть различных форматов (PDF, Word, Excel). В базовом классе ReportCreate определен общий алгоритм создания отчета. Подклассы (ReportExcelCreate, ReportPdfCreate, ReportWordCreate) реализуют специфические шаги для каждого формата отчета.

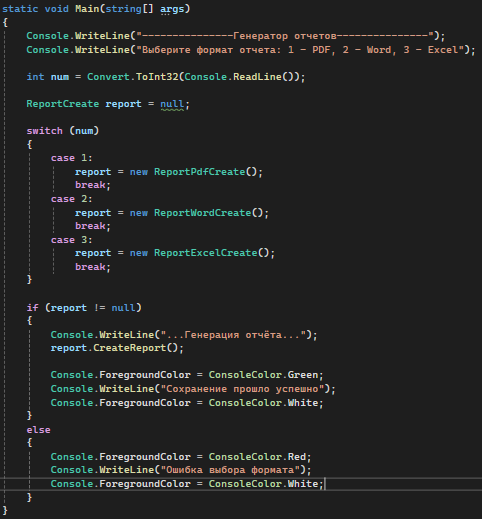
1.4 Листинг кода

****

****

****

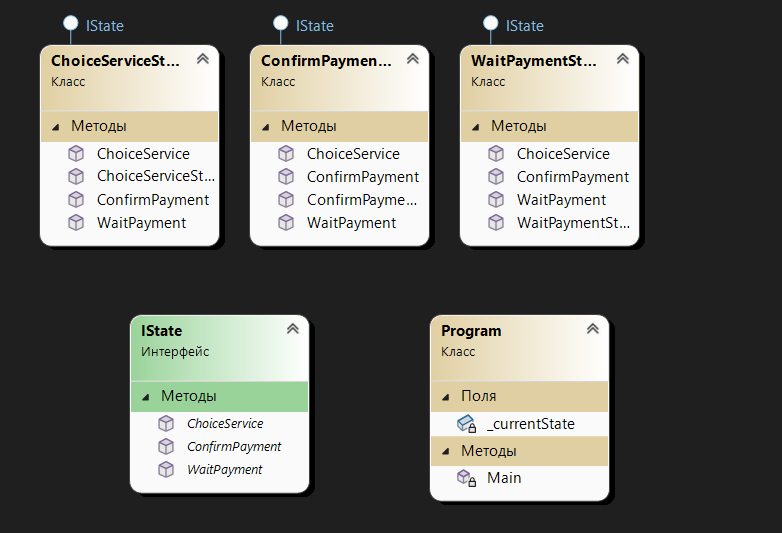
****

****

**2. PayStateApp**

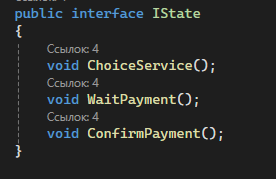
2.1 Приложение для оплаты, где можно находиться в состояниях (выбор услуги, ожидание оплаты, подтверждение оплаты). Доступные действия зависят от состояния.

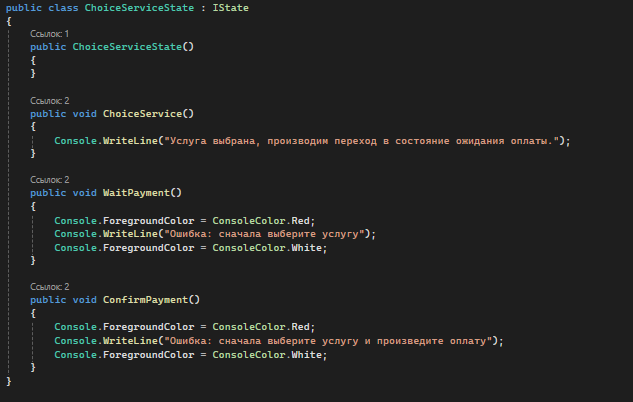
2.2 UML диаграмма

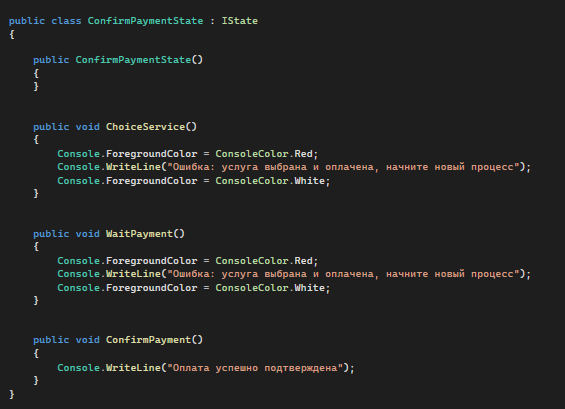


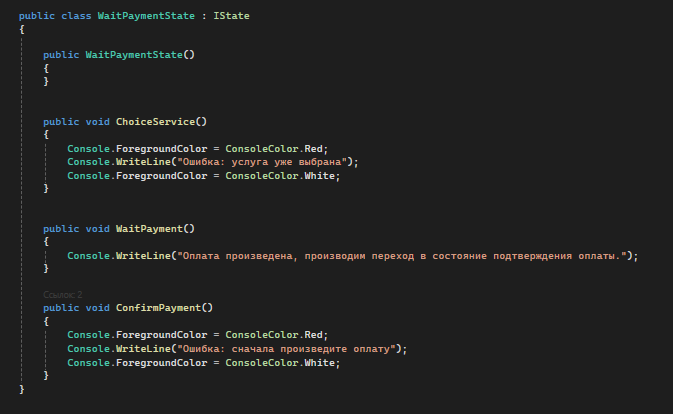
2.3 Паттерн State был успешно применен для разработки приложения для оплаты, где пользователь может находиться в различных состояниях: выбор услуги, ожидание оплаты и подтверждение оплаты. Каждое состояние реализовано в отдельном классе, который реализует интерфейс IState. В каждом состоянии доступны только те действия, которые имеют смысл в данном контексте (например, нельзя подтвердить оплату, если она еще не произведена).

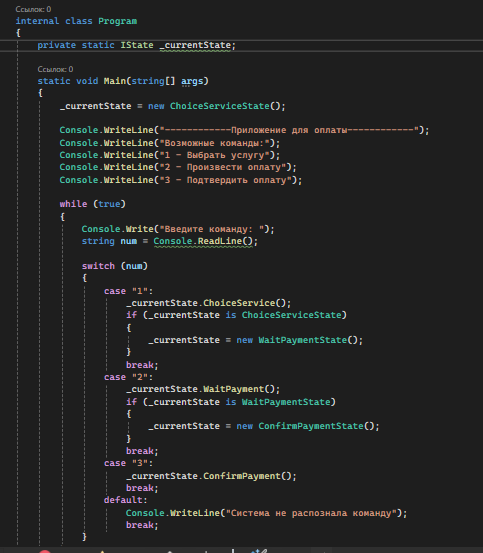
2.4 Листинг кода







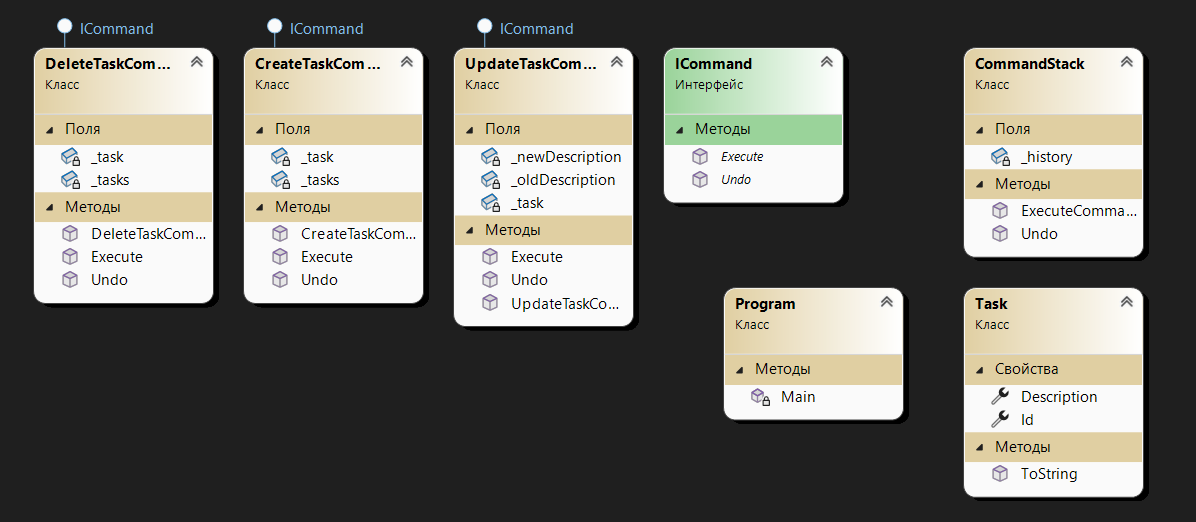




**3. ProjectManagerApp**

3.1 Приложение для управления проектами, позволяющее выполнять команды для создания, обновления и удаления задач проекта, с историей изменений.

3.2 UML диаграмма



3.3 Паттерн Command был успешно применен для разработки приложения для управления задачами. Каждая операция (создание, обновление, удаление задачи) реализована в виде отдельного класса, реализующего интерфейс ICommand. Класс CommandStack управляет историей выполненных команд и позволяет отменять последнюю команду. Добавление новых команд не требует изменения существующего кода. Достаточно создать новый класс команды и добавить его в систему. Паттерн Command позволяет инкапсулировать каждую операцию в отдельный объект и легко управлять выполнением и отменой операций.

3.4 Листинг кода

