Учреждение образования БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет непрерывного и дистанционного обучения

Специальность «Информационные системы и технологии в бизнес менеджменте»

Дисциплина: «Средства и технологии анализа и разработки информационных систем»

Отчет по контрольной работе №1

Тема : Разработка автоматизированной системы поддержки управления продажами в электронной коммерции

Выполнил: Галкин И.В.

Группа: 894351

Вариант: 21



Проверила: Унучек Т.М.

Минск 2021

**СОДЕРЖАНИЕ**

[**1. Краткие теоретические сведения** 3](#_Toc73237433)

[**2. Результаты моделирования** 6](#_Toc73237434)

[**Выводы** 14](#_Toc73237435)

[**Список использованной литературы** 15](#_Toc73237436)

# **1. Краткие теоретические сведения**

# **Краткие теоретические сведения**

Поведенческие шаблоны проектирования определяют общие закономерности связей между объектами, реализующими данные паттерны. Следование этим шаблонам уменьшает связность системы и облегчает коммуникацию между объектами, что улучшает гибкость программного продукта.

В данной работе был выбран паттерн Стратегия для реализации.

**Паттерн Стратегия**

**Стратегия** — это поведенческий паттерн проектирования, который определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс, после чего алгоритмы можно взаимозаменять прямо во время исполнения программы.

Паттерн Стратегия предлагает определить семейство схожих алгоритмов, которые часто изменяются или расширяются, и вынести их в собственные классы, называемые стратегиями.

Вместо того, чтобы изначальный класс сам выполнял тот или иной алгоритм, он будет играть роль контекста, ссылаясь на одну из стратегий и делегируя ей выполнение работы. Чтобы сменить алгоритм, будет достаточно подставить в контекст другой объект-стратегию.

Важно, чтобы все стратегии имели общий интерфейс. Используя этот интерфейс, контекст будет независимым от конкретных классов стратегий. С другой стороны, можно изменять и добавлять новые виды алгоритмов, не трогая код контекста.

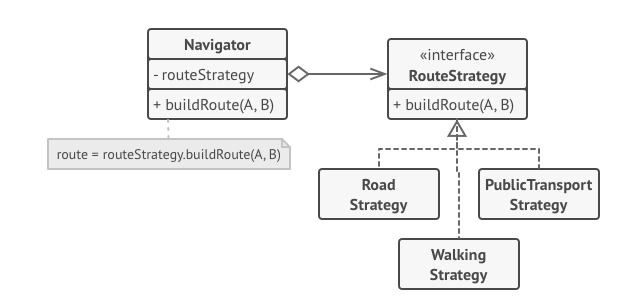


Рисунок 1.1 – Паттерн стратегия

**Паттерн стратегия можно применять когда:**

1. Когда нужно использовать разные вариации какого-то алгоритма внутри одного объекта. Стратегия позволяет варьировать поведение объекта во время выполнения программы, подставляя в него различные объекты-поведения (например, отличающиеся балансом скорости и потребления ресурсов).
2. Когда есть множество похожих классов, отличающихся только некоторым поведением. Стратегия позволяет вынести отличающееся поведение в отдельную иерархию классов, а затем свести первоначальные классы к одному, сделав поведение этого класса настраиваемым.
3. Когда необходимо скрыть детали реализации алгоритмов для других классов. Стратегия позволяет изолировать код, данные и зависимости алгоритмов от других объектов, скрыв эти детали внутри классов-стратегий.
4. Когда различные вариации алгоритмов реализованы в виде развесистого условного оператора. Каждая ветка такого оператора представляет собой вариацию алгоритма.  Стратегия помещает каждую лапу такого оператора в отдельный класс-стратегию. Затем контекст получает определённый объект-стратегию от клиента и делегирует ему работу. Если вдруг понадобится сменить алгоритм, в контекст можно подать другую стратегию.

**Шаги реализации:**

1. Определите алгоритм, который подвержен частым изменениям. Также подойдёт алгоритм, имеющий несколько вариаций, которые выбираются во время выполнения программы.
2. Создайте интерфейс стратегий, описывающий этот алгоритм. Он должен быть общим для всех вариантов алгоритма.
3. Поместите вариации алгоритма в собственные классы, которые реализуют этот интерфейс.
4. В классе контекста создайте поле для хранения ссылки на текущий объект-стратегию, а также метод для её изменения. Убедитесь в том, что контекст работает с этим объектом только через общий интерфейс стратегий.
5. Клиенты контекста должны подавать в него соответствующий объект-стратегию, когда хотят, чтобы контекст вёл себя определённым образом.

**Преимущества и недостатки:**

Преимущества:

* Горячая замена алгоритмов на лету.
* Изолирует код и данные алгоритмов от остальных классов.
* Уход от наследования к делегированию.
* Реализует принцип открытости/закрытости.

Недостатки:

* Усложняет программу за счёт дополнительных классов.
* Клиент должен знать, в чём состоит разница между стратегиями, чтобы выбрать подходящую.

# **2. Руководство пользователя со скриншотами**

После запуска приложения, пользователю предлагается войти в систему одним из трех способов:

* Войти как пользователь
* Войти как администратор
* Зарегистрироваться

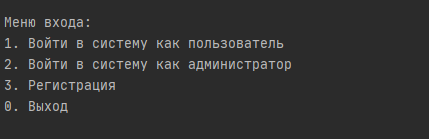


Рисунок 2.1 – Выбор способа входа в систему

После успешного прохождения процесса аутентификации, пользователю, в зависимости от его типа, представляется главное меню приложения.

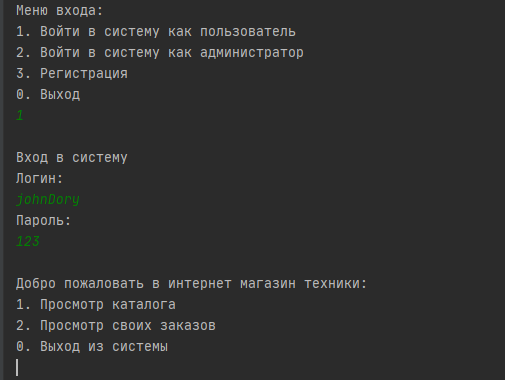


Рисунок 2.2 – Вход в систему в роли пользователя

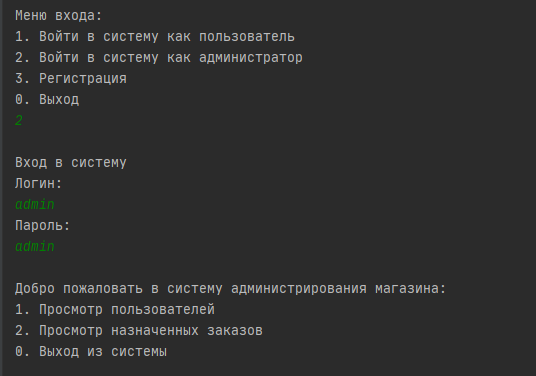


Рисунок 2.3 – Вход в систему в роли администратор

Пользователь может:

1. Просмотреть каталог товаров
2. Добавить желаемые товары в корзину
3. Оформить заказ
4. Выбрать адрес доставки из существующих, либо добавить новый
5. Подтвердить заказ
6. В случае выхода из каталога с заполненной корзиной, у пользователя остается возможность восстановить выбранные ранее товары
7. Посмотреть оформленные на него заказы

# 

Рисунок 2.4 – Добавление товара в корзину

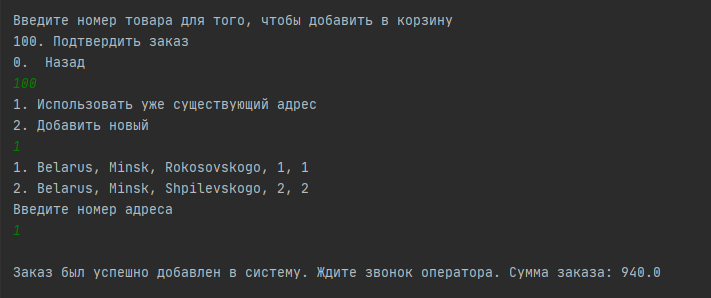


Рисунок 2.5 – Оформление заказа

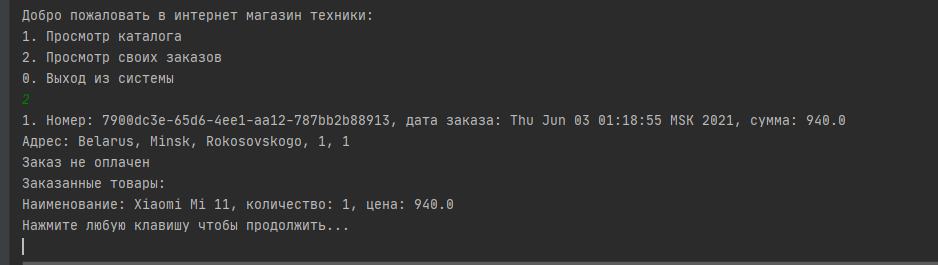


Рисунок 2.6 – Просмотр добавленного заказа

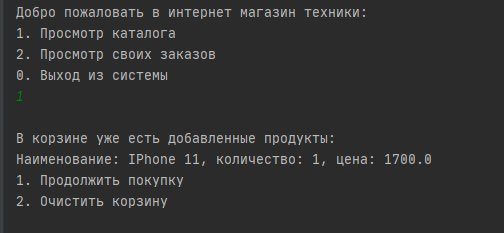


Рисунок 2.7 – Предложение восстановить добавленные заранее в корзину товары

Администратор может:

1. Просматривать пользователей системы
2. Просматривать заказы определенного пользователя
3. Просматривать заказы назначенные на него
4. Изменять статус заказов

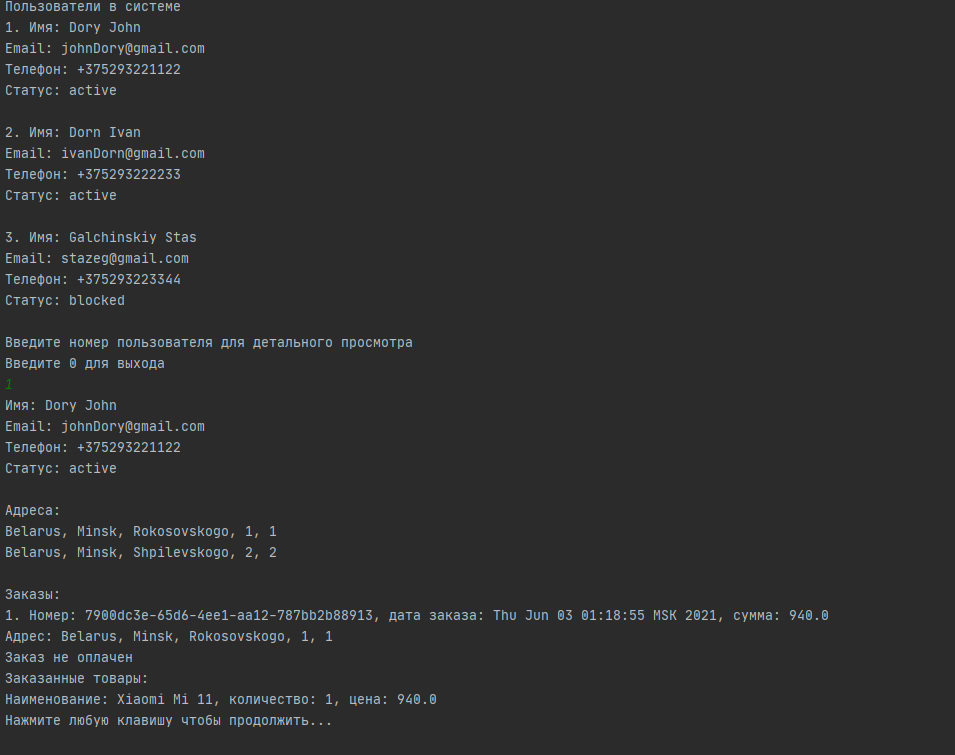


Рисунок 2.8 – Просмотр пользователей в системе и детальный просмотр конкретного

# 

Рисунок 2.9 – Просмотр назначенных на администратора заказов и изменение статуса конкретного

В результате проведенной операции, появилась информация об оплате заказа. Позже, можно будет добавить указывание конкретной информации об оплате.

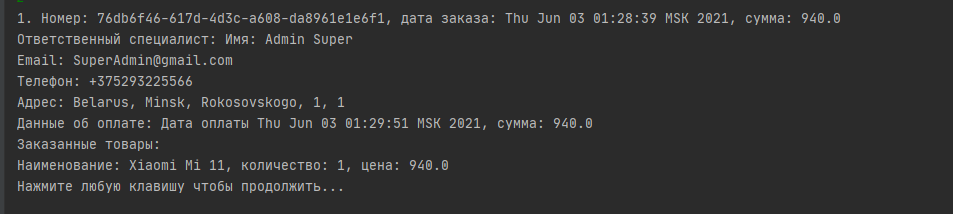
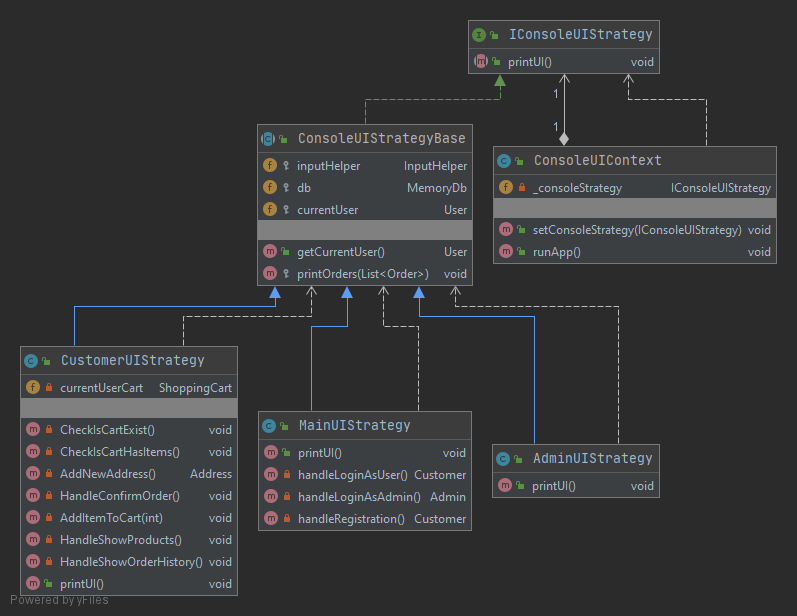


Рисунок 2.10 – Проверка результатов

Паттерн стратегия был реализован в переключении интерфейсов пользователя: если пользователь вошел как администратор, то используется стратегия AdminUIStrategy, в противном случае будет использоваться стратегия CustomerUIStrategy. Стратегия MainUiStrategy используется для формы логина.

При запуске приложения создается объект ConsoleUIContext на который назначается MainUIStrategy через метод setConsoleStrategy.

Далее, в зависимости от типа аутентификации выбирается следующая стратегия и таким же образом назначается в контекст.



# **Выводы**

В данной работе были изучены поведенческие паттерны проектирования. На практике реализован паттерн стратегия.

# **Список использованной литературы**

[1] Стратегия [Электронный ресурс]: – Режим доступа: https://refactoring.guru/ru/design-patterns/strategy