

## **МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Юго-Западный государственный университет»  
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ

Проректор по учебной работе

О.Г. Локтионова

« 14 » 02



### **ПАТТЕРН ПРОЕКТИРОВАНИЯ «СТРАТЕГИЯ»**

Методические указания по выполнению лабораторной работы по  
дисциплине "Методология программной инженерии" для студентов  
направления подготовки магистров 09.04.04 "Программная инженерия"

Курск 2018

УДК 004.65

Составители: Т.М. Белова, В.Г. Белов

Рецензент

Кандидат технических наук, доцент кафедры программной инженерии ЮЗГУ И.Н. Ефремова

**Паттерн проектирования «Стратегия»:** методические указания по выполнению лабораторной работы по дисциплине "Методология программной инженерии" для студентов направления подготовки магистров 09.04.04 "Программная инженерия"/ Юго-Зап. гос. ун-т; сост.: Т.М. Белова, В.Г. Белов, – Курск, 2018. – 16 с.: ил. 15.

Изложена последовательность действий по разработке и применению паттерна проектирования «Стратегия» при работе в интегрированной среде разработки Eclipse.

Материал предназначен для студентов направления подготовки магистров 09.04.04 «Программная инженерия», а также будет полезен студентам всех направлений подготовки, изучающим технологии разработки программных продуктов с использованием паттернов.

Текст печатается в авторской редакции.

Подписано в печать 14.02.18. Формат 60х84 1/16.

Усл. печ. л. 0,7. Уч.-изд. л. 0,6. Тираж 50 экз. Заказ 788. Бесплатно.

Юго-Западный государственный университет  
305040, Курск, ул. 50 лет Октября, 94.

## Содержание

1 Цель лабораторной работы .....	4
2 Основные понятия .....	4
3 Порядок выполнения лабораторной работы .....	7
4 Содержание отчета по лабораторной работе .....	15
5 Вопросы к защите лабораторной работы .....	15
6 Индивидуальные задания.....	15
Список использованных источников.....	16

## 1 Цель лабораторной работы

Целью лабораторной работы является приобретение знаний, умений и навыков использования возможностей паттерна проектирования «Стратегия» в разработке информационно-вычислительных систем.

## 2 Основные понятия

При реализации проектов по разработке программных систем и моделированию бизнес-процессов встречаются ситуации, когда решение проблем в различных проектах имеют сходные структурные черты. Попытки выявить похожие схемы или структуры в рамках объектно-ориентированного анализа и проектирования привели к появлению понятия паттерна, которое из абстрактной категории превратилось в неперенный атрибут современных CASE-средств.

Паттерны различаются степенью детализации и уровнем абстракции. Предлагается следующая общая классификация паттернов по категориям их применения:

- архитектурные паттерны,
- паттерны проектирования,
- паттерны анализа,
- паттерны тестирования,
- паттерны реализации.

Архитектурные паттерны (Architectural patterns) - множество предварительно определенных подсистем со спецификацией их ответственности, правил и базовых принципов установления отношений между ними.

Архитектурные паттерны предназначены для спецификации фундаментальных схем структуризации программных систем. Наиболее известными паттернами этой категории являются паттерны GRASP (General Responsibility Assignment Software Pattern). Эти паттерны относятся к уровню системы и подсистем, но не к уровню классов. Как правило, формулируются в обобщенной форме, используют обычную терминологию и не зависят от области приложения. Паттерны этой категории систематизировал и описал К. Ларман.

Паттерны проектирования (Design patterns) - специальные схемы для уточнения структуры подсистем или компонентов программной системы и отношений между ними.

Паттерны проектирования описывают общую структуру взаимодействия элементов программной системы, которые реализуют исходную проблему проектирования в конкретном контексте. Наиболее известными паттернами этой категории являются паттерны GoF (Gang of Four), названные в честь Э. Гаммы, Р. Хелма, Р. Джонсона и Дж. Влиссидеса, которые систематизировали их и представили общее описание. Паттерны GoF включают в себя 23 паттерна. Эти паттерны не зависят от языка реализации, но их реализация зависит от области приложения.

Паттерны анализа (Analysis patterns) - специальные схемы для представления общей организации процесса моделирования.

Паттерны анализа относятся к одной или нескольким предметным областям и описываются в терминах предметной области. Наиболее известными паттернами этой группы являются паттерны бизнес-моделирования ARIS (Architecture of Integrated Information Systems), которые характеризуют абстрактный уровень представления бизнес-процессов. Паттерны анализа конкретизируются в типовых моделях с целью выполнения аналитических оценок или имитационного моделирования бизнес-процессов.

Паттерны тестирования (Test patterns) - специальные схемы для представления общей организации процесса тестирования программных систем.

К этой категории паттернов относятся такие паттерны, как тестирование черного ящика, белого ящика, отдельных классов, системы. Паттерны этой категории систематизировал и описал М. Гранд. Некоторые из них реализованы в инструментальных средствах, наиболее известными из которых является IBM Test Studio. В связи с этим паттерны тестирования иногда называют стратегиями или схемами тестирования.

Паттерны реализации (Implementation patterns) - совокупность компонентов и других элементов реализации, используемых в структуре модели при написании программного кода.

Эта категория паттернов делится на следующие подкатегории: паттерны организации программного кода, паттерны оптимизации программного кода, паттерны устойчивости кода, паттерны разработки графического интерфейса пользователя и др. Паттерны этой категории описаны в работах М. Гранда, К. Бека, Дж. Тидвелла и др. Некоторые из них реализованы в популярных интегрированных средах программирования в форме шаблонов создаваемых проектов. В этом

случае выбор шаблона программного приложения позволяет получить некоторую заготовку программного кода.

Шаблон проектирования или паттерн (англ. design pattern) в разработке программного обеспечения — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Технически, паттерны (шаблоны) проектирования - это абстрактные примеры правильного использования небольшого числа комбинаций простейших техник объектно-ориентированного программирования. Паттерны проектирования - это простые примеры, показывающие правильные способы организации взаимодействий между классами или объектами.

Паттерн "Стратегия" (Strategy) относится к группе, называемой поведенческой (Behavioral). Паттерны из этой группы определяют алгоритмы и взаимодействие между классами и объектами, то есть их поведение, увеличивая таким образом их гибкость.

В группу включены следующие паттерны:

- Цепочка обязанностей (Chain of responsibility),
- Команда (Command),
- Интерпретатор (Interpreter),
- Итератор (Iterator),
- Посредник (Mediator),
- Хранитель (Memento),
- Наблюдатель (Observer),
- Состояние (State),
- Стратегия (Strategy),
- Шаблонный метод (Template method),
- Посетитель (Visitor).

• Паттерн "Стратегия" (Strategy) - определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс. После чего, алгоритмы можно взаимно заменять прямо во время исполнения программы.

### 3 Порядок выполнения лабораторной работы

Данный поведенческий шаблон предназначен, прежде всего, для использования в таких случаях, когда в решаемой задаче в каком-то определенном месте необходимо использовать различные алгоритмы в зависимости от конкретного состояния системы и/или ее окружения в конкретный момент времени. Рассматриваемый пример реализует алгоритмы, с помощью которых в зависимости от ситуации необходимо сложить, вычесть или перемножить два целых числа.

1. Открыть Eclipse и создать новый Java Project с именем Strategy (рисунок 1).

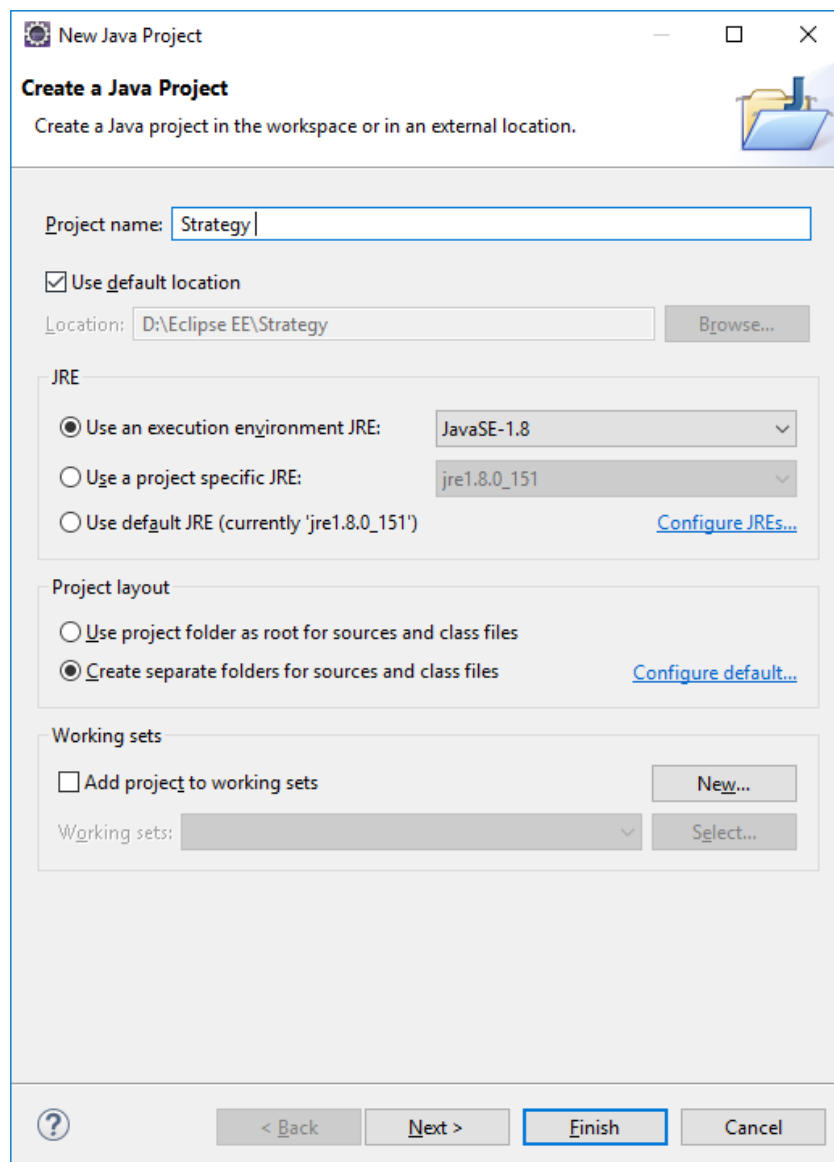


Рисунок 1

2. Схема задачи представлена ниже в виде UML-диаграммы, которая будет постепенно наполняться содержанием. Эта схема создается в этом проекте путем добавления RapyrusModel (рисунок 2).

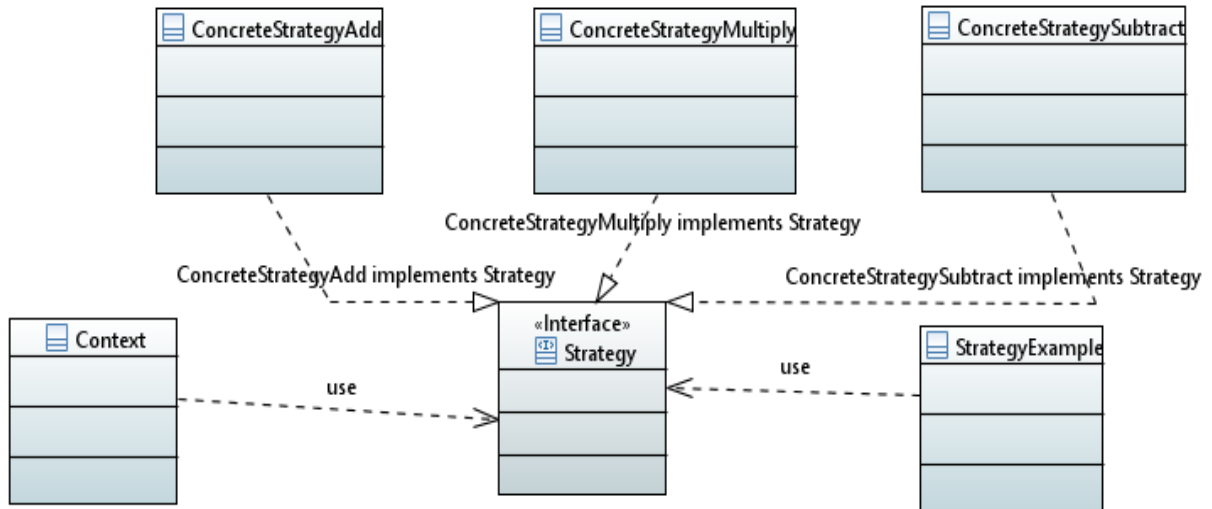


Рисунок 2

3. По этой схеме сгенерировать Java-код. Автоматически создаются заготовки для будущих классов и интерфейса.
4. Добавить в интерфейс Strategy метод execute, принимающий на вход два целых числа a и b (рисунок 3).

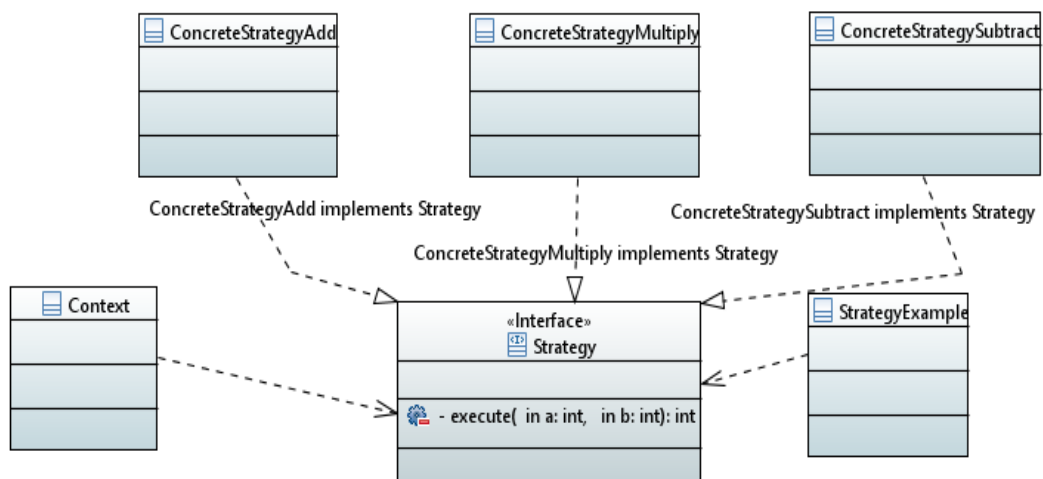


Рисунок 3



Интерфейс будет иметь следующий вид (рисунок 4):

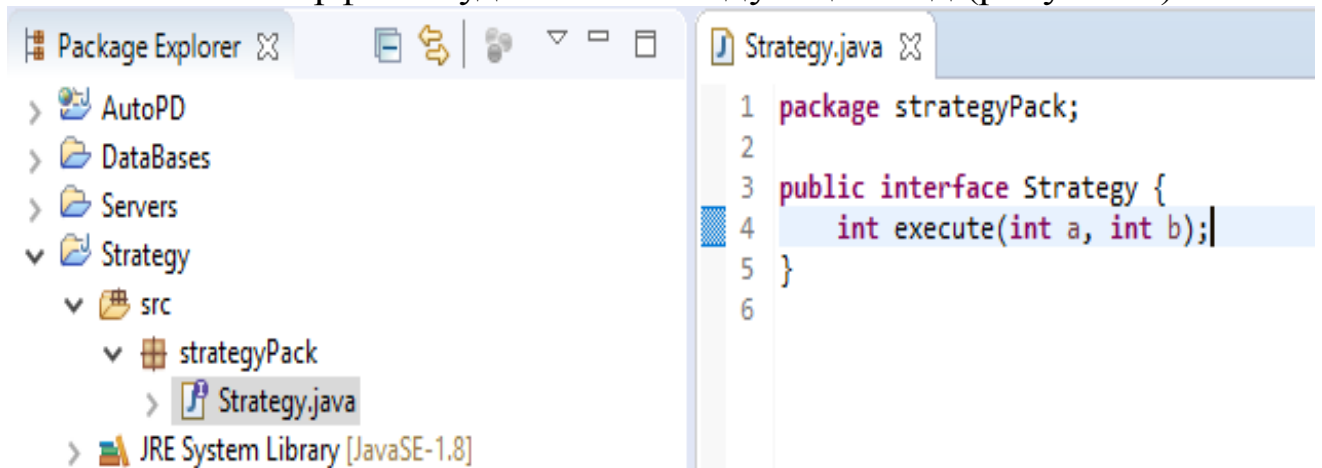


Рисунок 4

5. В классе ConcreteStrategyAdd нужно реализовать метод execute, который будет выводить результат сложения двух целых чисел (рисунок 5).

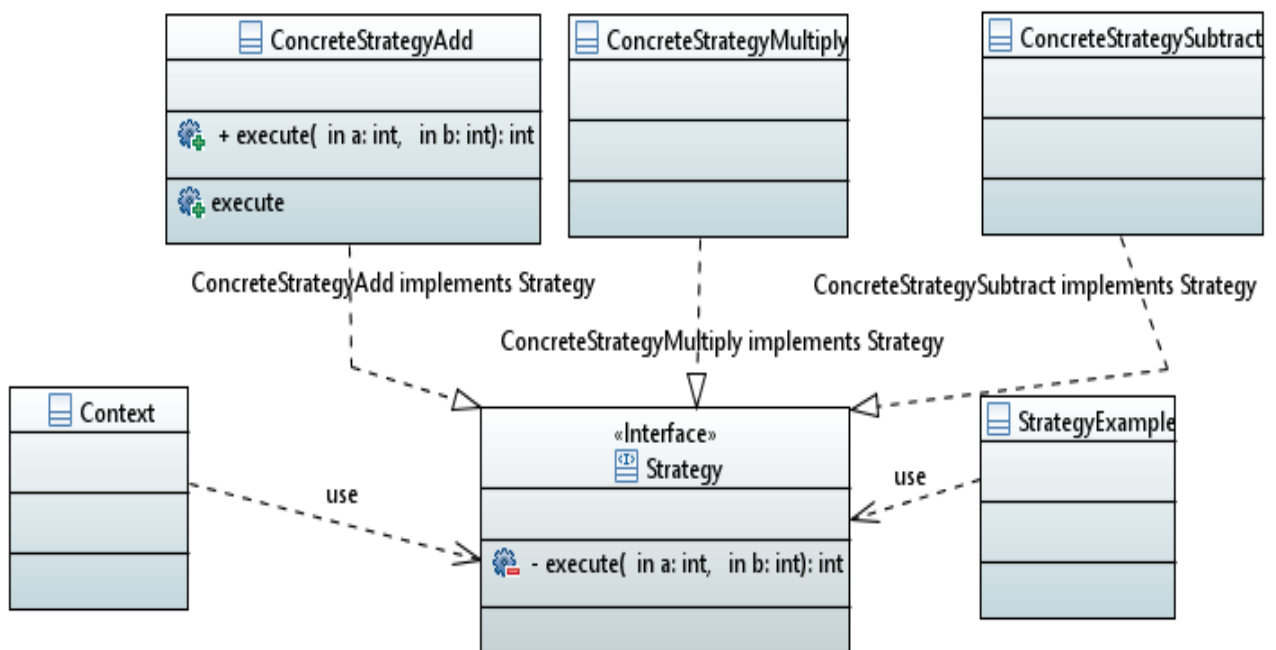


Рисунок 5

Класс будет иметь следующий вид (рисунок 6):

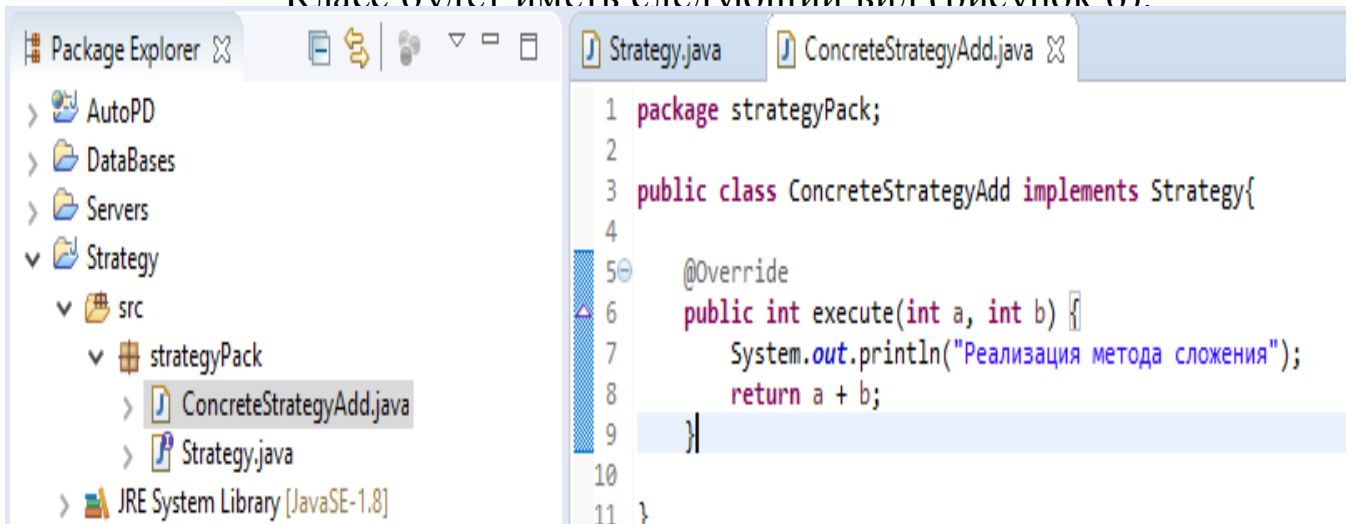


Рисунок 6

6. В классе ConcreteStrategySubtract нужно реализовать метод execute, который будет выводить результат вычитания двух целых чисел (рисунок 7).

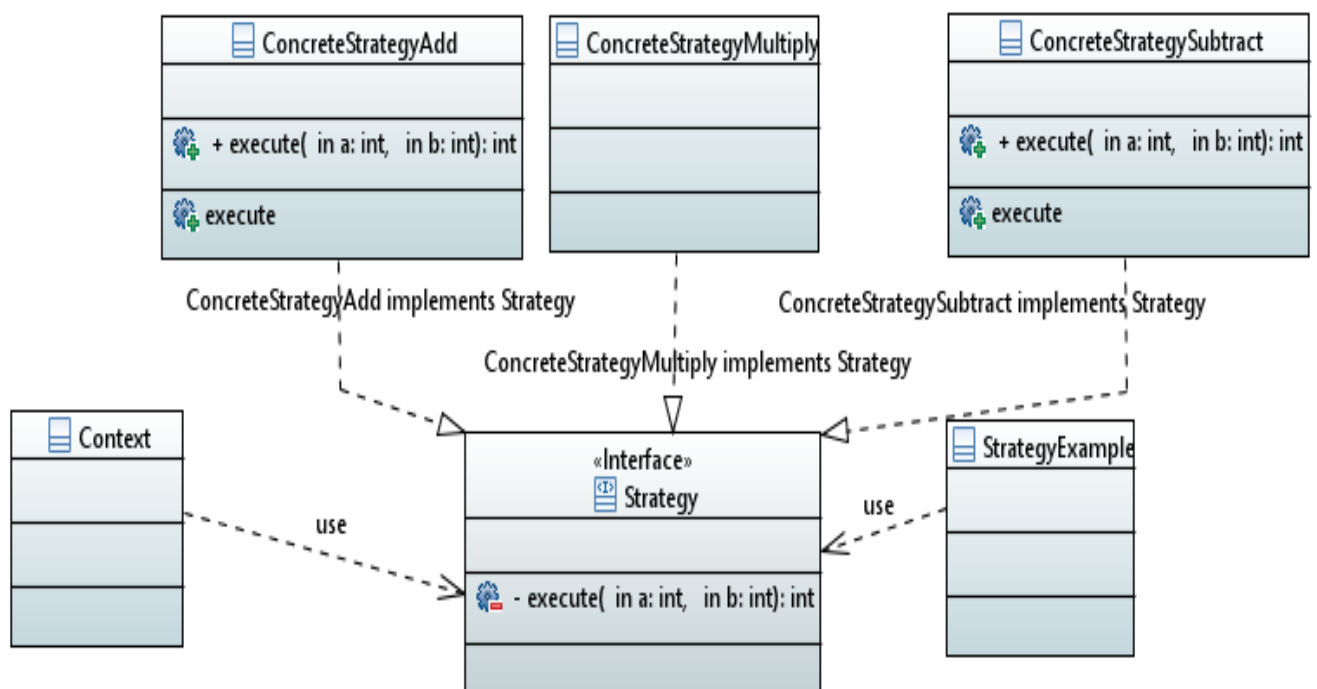


Рисунок 7

Класс будет иметь следующий вид (рисунок 8):

```

1 package strategyPack;
2
3 public class ConcreteStrategySubtract implements Strategy {
4
5     @Override
6     public int execute(int a, int b) {
7         System.out.println("Реализация метода вычитания");
8         return a - b;
9     }
10
11 }
12

```

Рисунок 8

7. В классе ConcreteStrategyMultiply нужно реализовать метод execute, который будет выводить результат умножения двух целых чисел (рисунок 9).

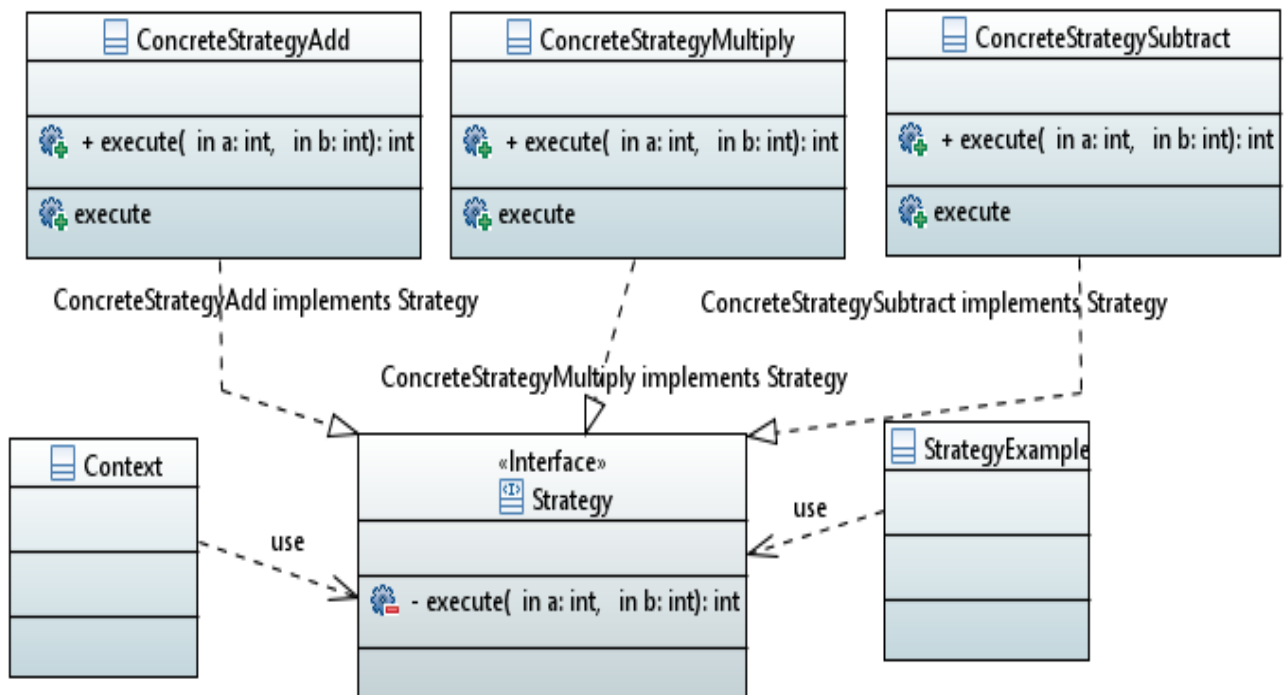


Рисунок 9

Класс будет иметь следующий вид (рисунок 10):

```

1 package strategyPack;
2
3 public class ConcreteStrategyMultiply implements Strategy {
4
5     @Override
6     public int execute(int a, int b) {
7         System.out.println("Реализация метода умножения");
8         return a * b;
9     }
10
11 }
12

```

Рисунок 10

8. В класс Context (рисунок 11), использующий интерфейс Strategy, добавить методы setStrategy (устанавливает конкретную стратегию) и executeStrategy (реализует установленную стратегию).

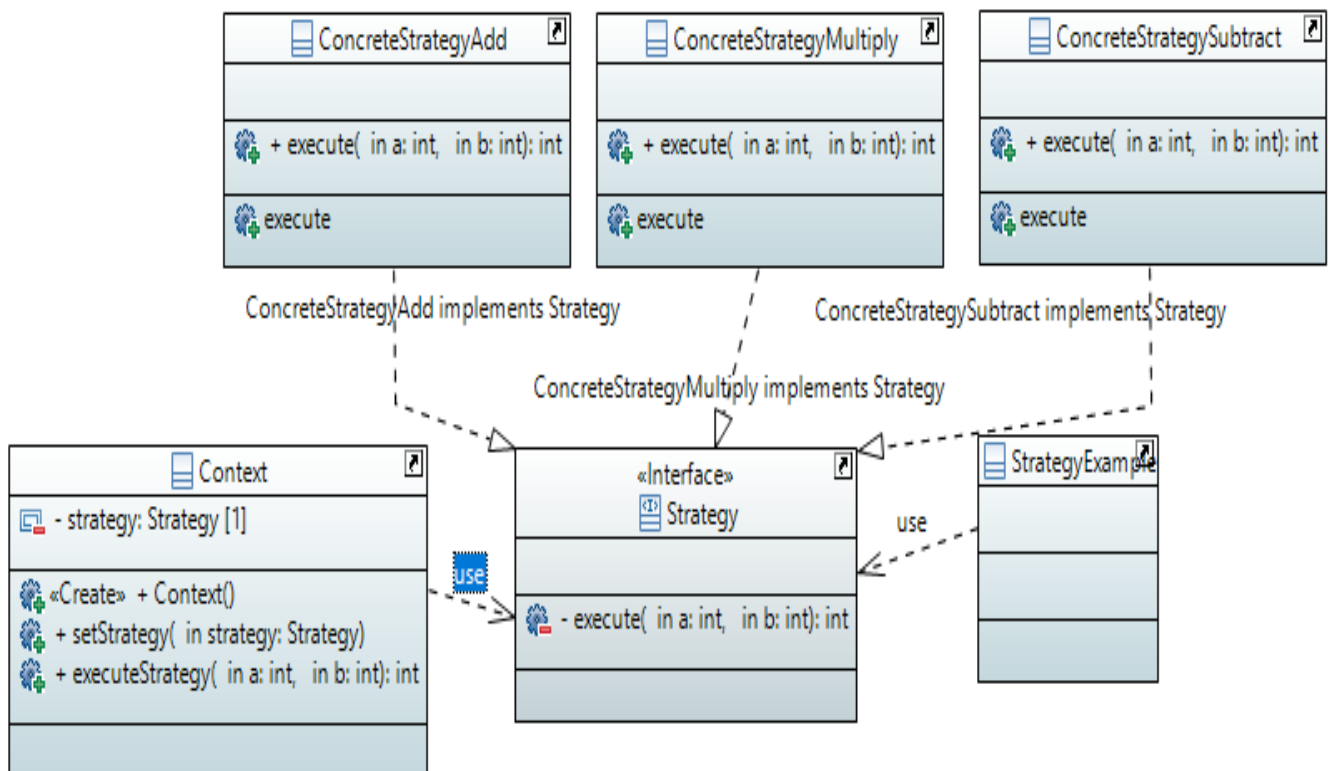


Рисунок 11

Класс будет иметь следующий вид (рисунок 12):

```

Context.java  Strategy.java  ConcreteStrategyAdd.java
1  package strategyPack;
2
3  class Context {
4      private Strategy strategy;
5
6      // КОНСТРУКТОР
7      public Context() {
8      }
9
10     // выбор стратегии
11     public void setStrategy(Strategy strategy) {
12         this.strategy = strategy;
13     }
14
15     public int executeStrategy(int a, int b) {
16         return strategy.execute(a, b);
17     }
18 }
19

```

Рисунок 12

9. В класс StrategyExample необходимо добавить метод main, в котором следует написать тестовое приложение для паттерна «Стратегия» (рисунок 13).

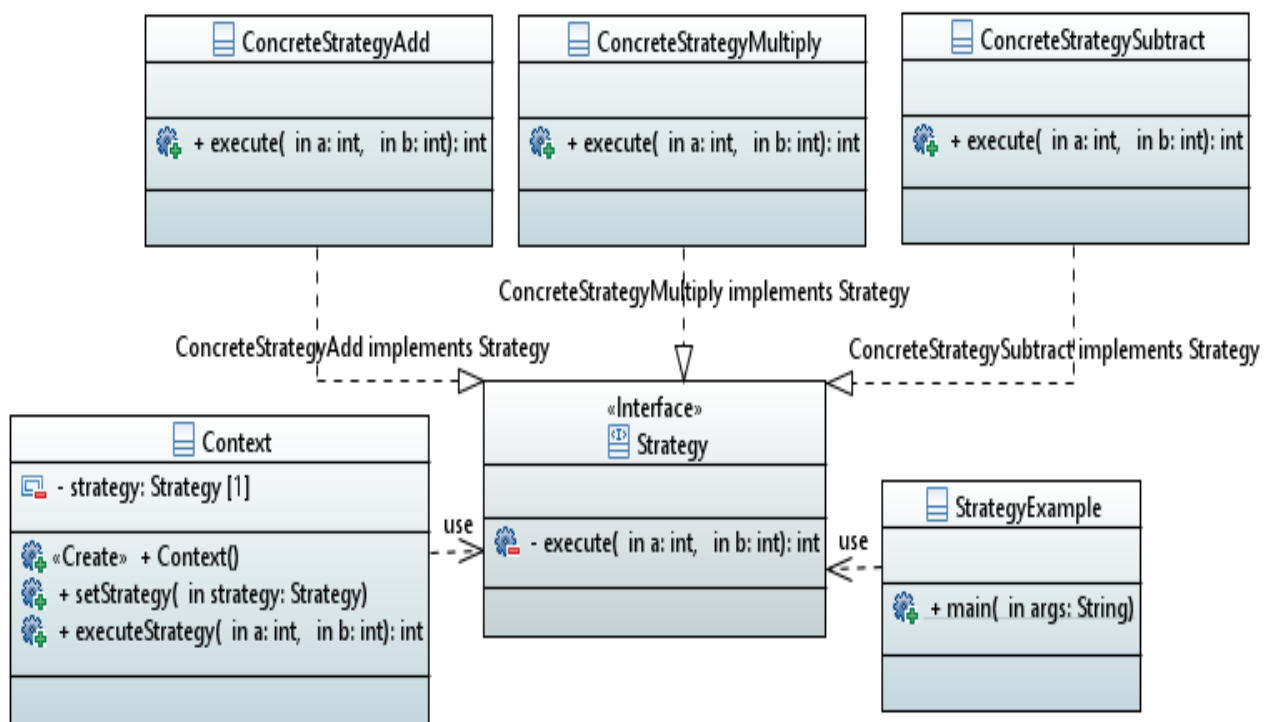
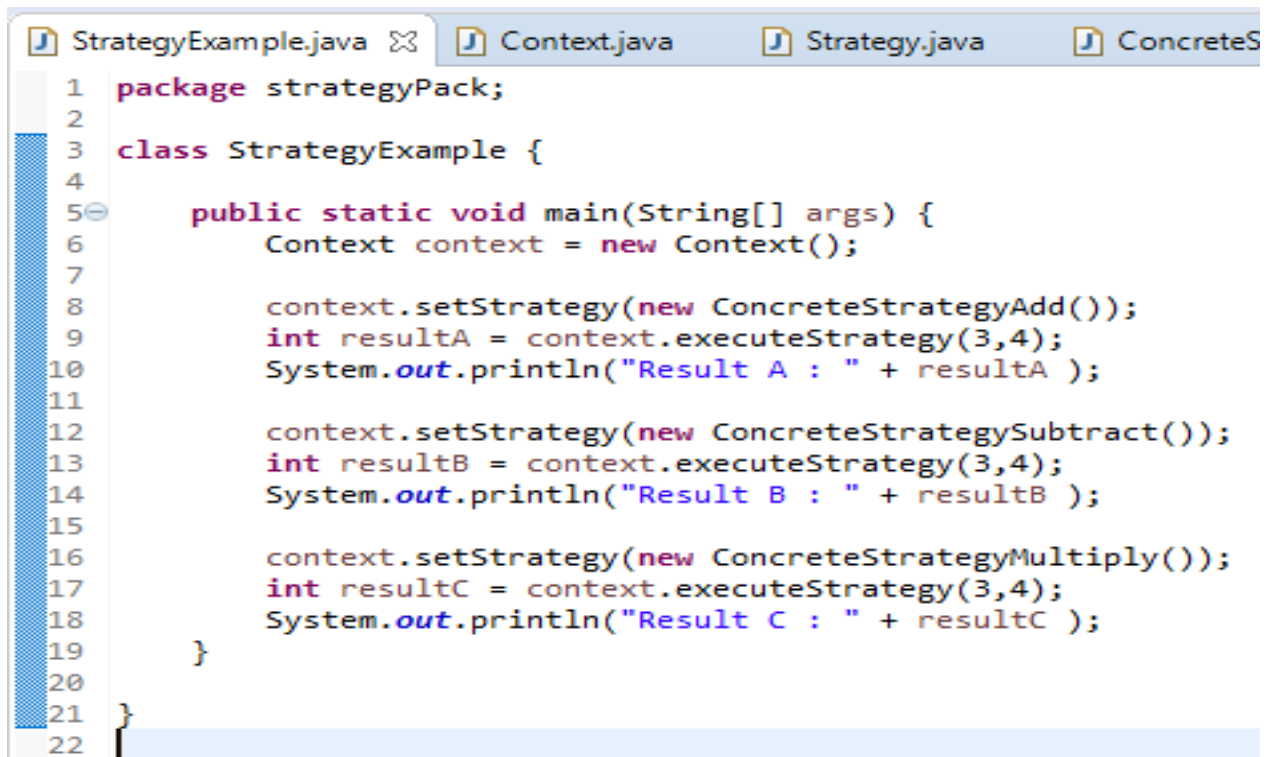


Рисунок 13

Класс тестового приложения будет иметь следующий вид (рисунок 14):



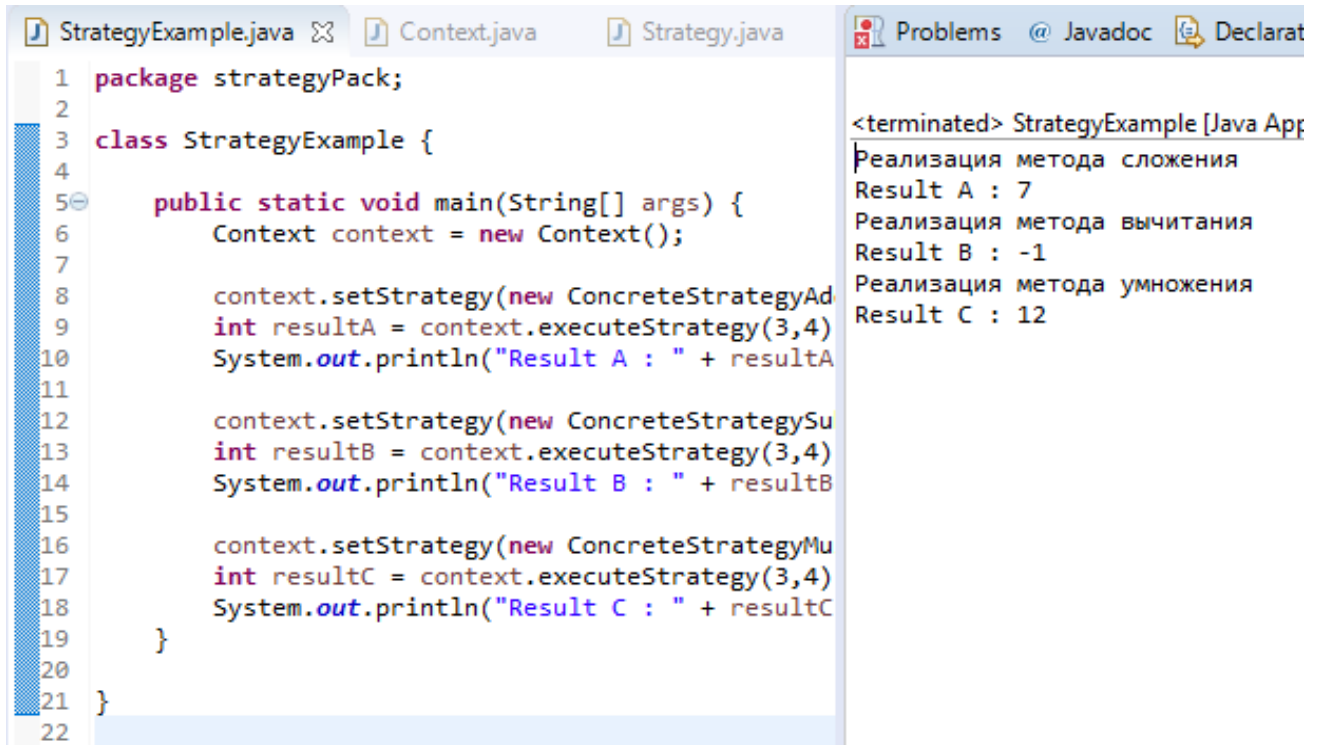
```

1 package strategyPack;
2
3 class StrategyExample {
4
5     public static void main(String[] args) {
6         Context context = new Context();
7
8         context.setStrategy(new ConcreteStrategyAdd());
9         int resultA = context.executeStrategy(3,4);
10        System.out.println("Result A : " + resultA );
11
12        context.setStrategy(new ConcreteStrategySubtract());
13        int resultB = context.executeStrategy(3,4);
14        System.out.println("Result B : " + resultB );
15
16        context.setStrategy(new ConcreteStrategyMultiply());
17        int resultC = context.executeStrategy(3,4);
18        System.out.println("Result C : " + resultC );
19    }
20 }
21
22

```

Рисунок 14

10. Результат выполнения тестового приложения (рисунок 15).



```

1 package strategyPack;
2
3 class StrategyExample {
4
5     public static void main(String[] args) {
6         Context context = new Context();
7
8         context.setStrategy(new ConcreteStrategyAdd());
9         int resultA = context.executeStrategy(3,4);
10        System.out.println("Result A : " + resultA);
11
12        context.setStrategy(new ConcreteStrategySubtract());
13        int resultB = context.executeStrategy(3,4);
14        System.out.println("Result B : " + resultB);
15
16        context.setStrategy(new ConcreteStrategyMultiply());
17        int resultC = context.executeStrategy(3,4);
18        System.out.println("Result C : " + resultC);
19    }
20 }
21
22

```

<terminated> StrategyExample [Java App  
 Реализация метода сложения  
 Result A : 7  
 Реализация метода вычитания  
 Result B : -1  
 Реализация метода умножения  
 Result C : 12

Рисунок 15

## 4 Содержание отчета по лабораторной работе

Отчет по лабораторной работе включает:

- титульный лист;
- условие задания;
- диаграммы классов для решения задачи;
- диаграммы последовательности для решения задачи;
- текст программы реализации паттерна проектирования «Стратегия» для индивидуального задания;
- результаты тестирования программы.

## 5 Вопросы к защите лабораторной работы

1. Что такое паттерн проектирования?
2. Для чего предназначен паттерн проектирования «Стратегия»?
3. К какому типу паттернов проектирования относится «Стратегия»?
4. Какие классы/интерфейсы являются участниками «Стратегии»?
5. Для чего необходим метод `setStrategy` в классе `Context`?
6. Назовите родственные для «Стратегии» паттерны проектирования.
7. Выделите достоинства и недостатки этого паттерна проектирования.

## 6 Индивидуальные задания

1. Реализовать программный продукт, выполняющий проверку введенной пользователем строки – что введено: число, строка в нижнем регистре или строка в верхнем регистре? При разработке использовать поведенческий паттерн проектирования «Стратегия».
2. Реализовать программный продукт, выполняющий проверку введенного пользователем числа – что введено: однозначное число, двузначное число или число с большим количеством знаков? При разработке использовать поведенческий паттерн проектирования «Стратегия».
3. Реализовать программный продукт, выполняющий подсчет заработной платы сотрудника за месяц в зависимости от типа его занятости – полная рабочая неделя (40 часов), сокращенная рабочая неделя для лиц от 16 до 18 лет (36 часов) и сокращенная рабочая неделя для лиц младше 16 лет (24 часа). Стоимость часа работы выбрать

самостоятельно. При разработке использовать поведенческий паттерн проектирования «Стратегия».

4. На сайте есть группа пользователей, каждый из них принадлежит к определённой группе, например, это администратор, редактор и гость. Администраторы могут добавлять материал, удалять материал, редактировать и читать. Редакторы могут только редактировать и читать. А гости только читать. Реализовать программный продукт, демонстрирующий возможности каждой из этих групп пользователей. При разработке использовать поведенческий паттерн проектирования «Стратегия».

5. Есть три типа героев – король, воин и маг. У каждого своя атака, тип оружия и наносимый урон. Реализовать программный продукт, демонстрирующий возможности каждого из этих типов героя. При разработке использовать поведенческий паттерн проектирования «Стратегия».

6. Существуют различные легковые машины, которые используют разные источники энергии: электричество, бензин, газ. Есть гибридные автомобили. Каждый из типов имеет свой расход топлива. Реализовать программный продукт, демонстрирующий возможности каждого из типов автомобиля. При разработке использовать поведенческий паттерн проектирования «Стратегия».

7. Реализовать программный продукт, демонстрирующий возможности компрессии файлов, для одного из доступных алгоритмов: zip, arj или rar. При разработке использовать поведенческий паттерн проектирования «Стратегия».

### **Список использованных источников**

1 Ларман, К. Применение UML 2.0 и шаблонов проектирования/ К. Ларман. - М.: Издательский дом «Вильямс», 2013. -736 с.

2 Османи, Э. Паттерны для масштабируемых JavaScript-приложений/ Э. Османи. - М.: Техносфера, 2015. -188 с.

3 Фримен Э. Паттерны проектирования/ Э. Фримен, Э. Фримен, К. Сьерра, Б. Бейтс. – СПб.: Питер, 2016. -653 с.