Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных Центр программной инженерии

ОТЧЕТ

к лабораторной работе №1 по дисциплине:

|  |
| --- |
| «Технология разработки программных комплексов» |
| Шаблоны проектирования |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | | | | |
| группы: | ИСТб-19-2 |  |  |  | Комогорцева Ю.В. |
| Проверил: | шифр группы  доцент |  | подпись  15 баллов |  | Фамилия ИО  Бахвалова З.А. |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск. 2022 г.

**Цель работы**

Ознакомиться с основными шаблонами проектирования, научиться применять их при проектировании и разработке ПО.

**Общее задание:**

Разработать каркас системы в соответствии с выданным вариантом задания.

При проектировании архитектуры **необходимо** использовать обозначенный в задании шаблон проектирования, иные варианты реализации задания **не принимаются**.

Программа должна функционировать в соответствии с приведённым в варианте задания примером работы.

Выбранный язык программирования должен поддерживать объектно-ориентированную парадигму. Выбор остального окружения производится студентом самостоятельно.

**Индивидуальный вариант**

8 Вариант. Паттерн «Фасад».

Вы принимаете участие в создании пакета для статистического анализа данных. В ваши задачи входит создание архитектуры модуля проведения описательного анализа. В ходе анализа производится расчёт среднего выборочного, медианы и моды (отдельный класс «средние величины»), расчет значений квартилей (отдельный класс «процентили распределения»), а также расчет дисперсии и среднеквадратического отклонения (отдельный класс «показатели рассеивания»).

Необходимо реализовать каркас модуля. Для обеспечения удобного интерфейса доступа ко всему описательному анализу главный класс модуля должен иметь метод «Произвести описательный анализ», который производит все вышеперечисленные операции. При запуске программа должна вызывать этот метод главного класса модуля и выводить названия операций, которые были произведены. Программа должна быть закрыта для изменения, но открыта для расширения. **Вычислений вышеприведенных характеристик производить не требуется, выводятся только названия операций.**

Пример работы программы:

Было рассчитано среднее выборочное

Была рассчитана медиана

Была рассчитана мода

Были рассчитаны значения квартилей

Были рассчитаны дисперсия и среднеквадратическое отклонение.

**1 Краткая характеристика паттерна «Фасад»**

Шаблон «Фасад» структурный шаблон проектирования, позволяющий упростить работу с подсистемой путем сведения всех возможных внешних вызовов к одному объекту, делегирующему их соответствующим объектам системы.

При использовании паттерна Фасад мы создаем класс, который упрощает и унифицирует набор более сложных классов, образующих некую подсистему. В отличие от многих других паттернов, Фасад относительно прост; в нем нет умопомрачительных абстракций, в которых приходится подолгу разбираться. Но от этого он не становится менее полезным; паттерн Фасад предотвращает появление сильных связей между клиентом и подсистемой. Это хорошо видно из диаграммы классов паттерна:

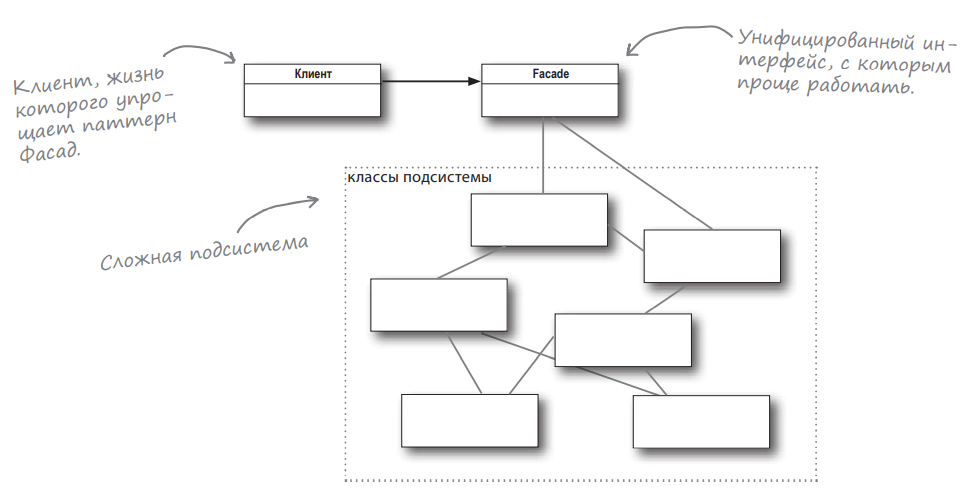


Рисунок 1 – Диаграмма классов паттерна Фасад

Структура.

Клиент – использует фасад вместо прямой работы с объектами сложной подсистемы.

Фасад – предоставляет быстрый доступ к определённой функциональности подсистемы. Он «знает», каким классам нужно переадресовать запрос, и какие данные для этого нужны.

Сложная подсистема – состоит из множества разнообразных классов. Для того, чтобы заставить их что-то делать, нужно знать подробности устройства подсистемы, порядок инициализации объектов и так далее.

Применение:

* Когда имеется сложная система, и необходимо упростить с ней работу. Фасад позволит определить одну точку взаимодействия между клиентом и системой.
* Когда надо уменьшить количество зависимостей между клиентом и сложной системой. Фасадные объекты позволяют отделить, изолировать компоненты системы от клиента и развивать и работать с ними независимо.
* Когда нужно определить подсистемы компонентов в сложной системе. Создание фасадов для компонентов каждой отдельной подсистемы позволит упростить взаимодействие между ними и повысить их независимость друг от друга.

Достоинства: изолирует клиентов от компонентов сложной подсистемы.

Недостатки: фасад рискует стать божественным объектом, привязанным ко всем классам программы.

**2 Шаги реализации**

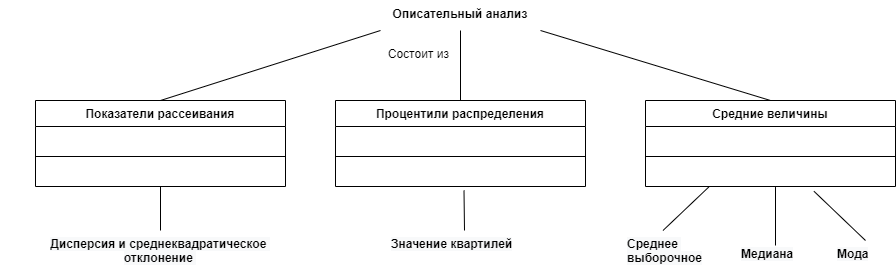
Даны классы и сущности показателей описательного анализа:

Рисунок 2 – Сущности описательного анализа

С первого взгляда, можно было бы реализовать этот модуль через наследование классов. Например, класс «Значения квартилей» наследовался бы от класса «Процентили распределения». Однако в таком случае архитектура модуля проведения описательного анализа данных не будет соответствовать принципу открытости/закрытости: программа должна быть закрыта для изменения, но открыта для расширения. В связи с этим необходимо добавить интерфейс IValues, в котором будет абстрагироваться расчет величин методом Calculation(). А конкретные способы расчета будут определены в реализациях этого интерфейса. Тем самым, при необходимости мы сможем добавить новые классы, которые будут реализовывать интерфейс IValue, но не сможем внести изменения в уже существующие методы основных классов.

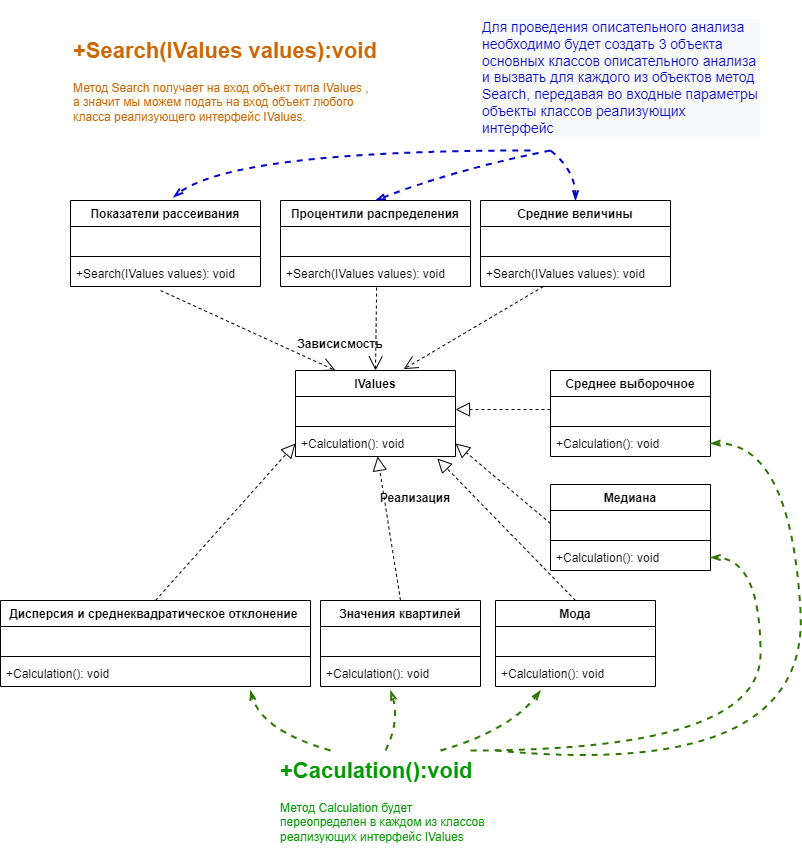


Рисунок 3 – Подсистема классов описательного анализа

На этом этапе архитектура модуля проведения описательного анализа выглядит завершенной, но это сложная подсистема которой предстоит взаимодействовать с другими подсистемами, и нам необходимо упростить с ней работу. Фасад позволит упростить взаимодействие между другими подсистемами.

Класс фасада будет содержать 3 приватных поля объектов основных классов подсистемы описательного анализа. А также нам необходим метод MakeDescriptiveAnalysis() , который будет вызывать выполнение описательного анализа. Конечно, можно код этого метода напрямую прописывать в месте использования описательного анализа, но если это делать много раз, то код будет не эффективен. Поэтому что бы избежать повторения кода в месте использования описательного анализа мы будем создавать объект класса фасада и вызывать метод MakeDescriptiveAnalysis().

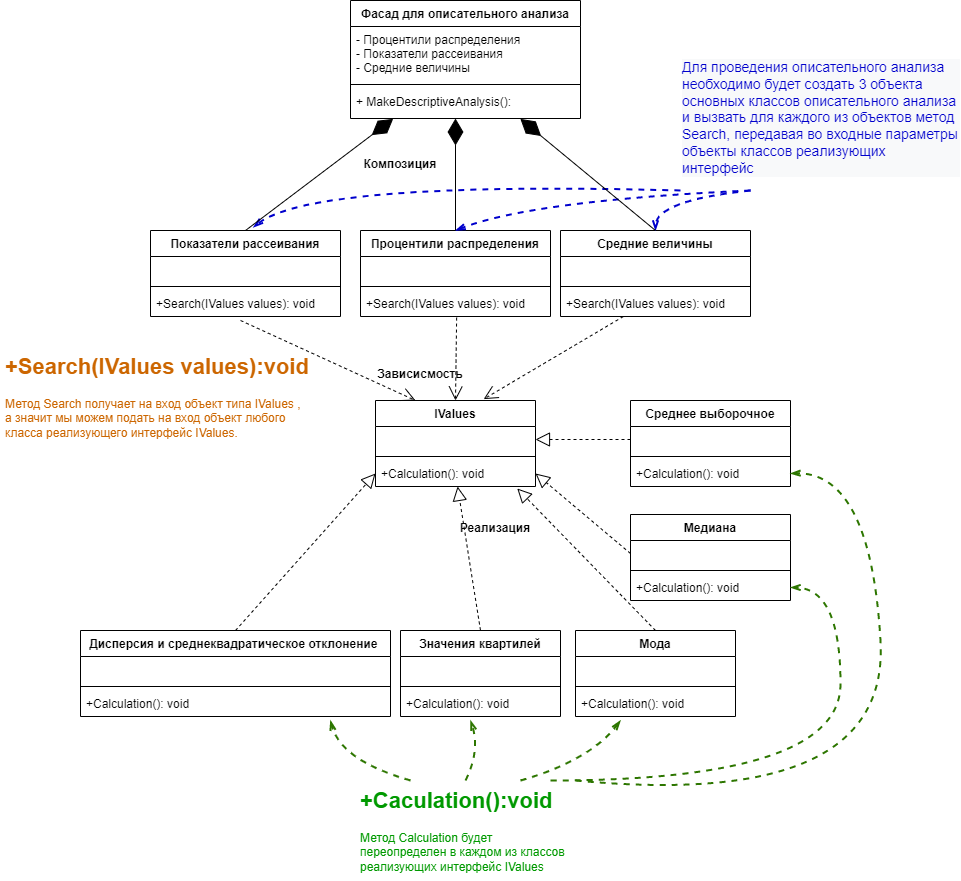


Рисунок 4 – Разработанные классы сущностей и фасада

**3 UML-диаграмма классов**

Клиент – точка входа в программу, не отображена на диаграмме.

Фасад – класс DescriptiveAnalysisFacade.

Сложная подсистема – состоит из множества разнообразных классов, показана на диаграмме.

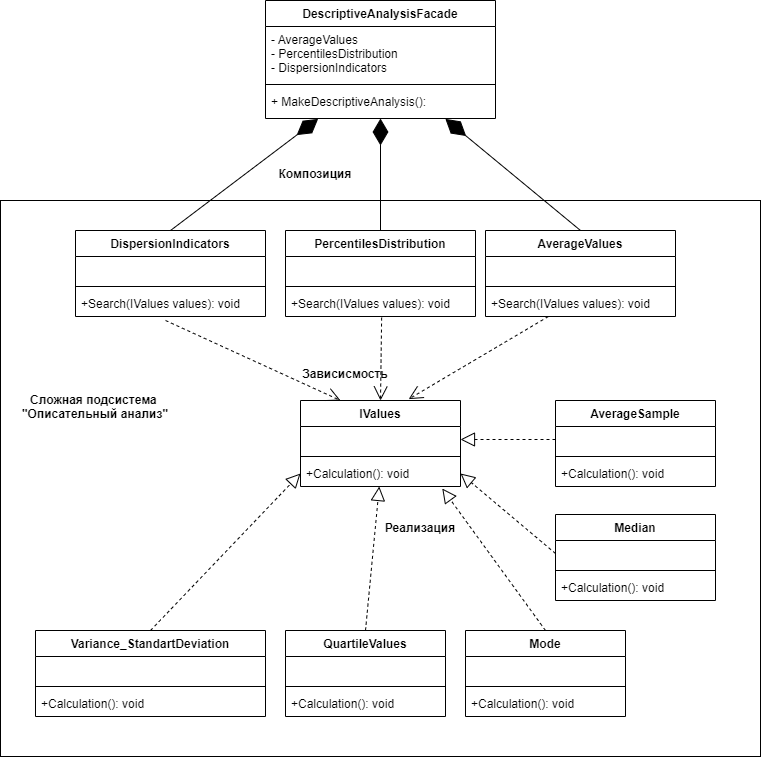


Рисунок 5 – UML диаграмма классов

**4 Результаты работы программы**

При запуске программа должна вызывать метод класса фасада модуля и выводить названия операций, которые были произведены. Вычислений вышеприведенных характеристик производить не требуется, выводятся только названия операций.

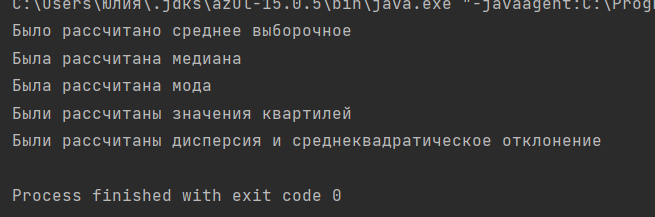


Рисунок 6 – Результат работы программы

**5 Исходный код**

Класс Main, содержащий точку входа в программу, создание объекта класса DescriptiveAnalysisFacade и вызов метода MakeDescriptiveAnalysis() – выполнение описательного анализа.

package com.company;  
  
public class Main {  
  
 public static void main(String[] args) {  
 DescriptiveAnalysisFacade descriptiveAnalysis =  
 new DescriptiveAnalysisFacade();  
 descriptiveAnalysis.MakeDescriptiveAnalysis();  
 }  
}

Класс DescriptiveAnalysisFacade, представляющий фасад подсистемы описательного анализа.

package com.company;  
  
public class DescriptiveAnalysisFacade {  
  
 private AverageValues averageValues;  
 private PercentilesDistribution percentilesDistribution;  
 private DispersionIndicators dispersionIndicators;  
 {  
 averageValues=new AverageValues();  
 percentilesDistribution=new PercentilesDistribution();  
 dispersionIndicators=new DispersionIndicators();  
 }  
  
 public void MakeDescriptiveAnalysis() {  
 averageValues.Search(new AverageSample());  
 averageValues.Search(new Median());  
 averageValues.Search(new Mode());  
  
 percentilesDistribution.Search(new QuartileValues());  
  
 dispersionIndicators.Search(new Variance\_StandardDeviation());  
 }  
}

Класс PercentilesDistribution

package com.company;  
  
public class PercentilesDistribution {  
 public void Search(IValues values){  
 values.Calculation();  
 }  
}

Класс DispersionIndicators

package com.company;  
  
public class DispersionIndicators {  
 public void Search(IValues values){  
 values.Calculation();  
 }  
}

Класс AverageValues

package com.company;  
  
public class AverageValues {  
  
 public void Search(IValues values){  
 values.Calculation();  
 }  
}

Интерфейс IValues

package com.company;  
  
public interface IValues {  
 void Calculation();  
}

Класс Median, реализует интерфейс IValues

package com.company;  
  
class Median implements IValues {  
  
 @Override  
 public void Calculation() {  
 System.out.println("Была рассчитана медиана");  
 }  
}

Класс Mode, реализует интерфейс IValues

package com.company;  
  
public class Mode implements IValues {  
  
 @Override  
 public void Calculation() {  
 System.*out*.println("Была рассчитана мода");  
 }  
}

Класс AverageSample, реализует интерфейс IValues

package com.company;  
  
public class AverageSample implements IValues {  
  
 @Override  
 public void Calculation() {  
 System.*out*.println("Было рассчитано среднее выборочное");  
 }  
}

Класс QuartileValues, реализует интерфейс IValues

package com.company;  
  
public class QuartileValues implements IValues {  
  
 @Override  
 public void Calculation() {  
 System.*out*.println("Были рассчитаны значения квартилей");  
 }  
}

Класс Variance\_StandardDeviation, реализует интерфейс IValues

package com.company;  
  
public class Variance\_StandardDeviation implements IValues {  
  
 @Override  
 public void Calculation() {  
 System.*out*.println("Были рассчитаны дисперсия и среднеквадратическое отклонение");  
 }  
}

# Литература

* 1. Эрик Фримен, Элизабет Робсон Head First Паттерны проектирования обновленное юбилейное издание, 2020. 656 с. ISBN: 978-5-4461-1034-6 Серия: Head First O’Reilly (дата обращения: 17.02.2022)
  2. Репозиторий с исходным кодом проекта // GitHub URL: https://github.com/KomogortsevaYulia/TRPK/tree/main/Pattern (дата обращения: 12.02.2022)
  3. Рефакторинг и паттерны проектирования. Фасад // Refactoring Guru URL: https://refactoring.guru/ru/design-patterns/facade (дата обращения: 17.02.2022)