Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
Федерального государственного бюджетного образовательного учреждения высшего образования
«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет информационных технологий и радиоэлектроники
Кафедра информационных систем и технологий

КУРСОВАЯ РАБОТА

По курсу Теория принятий решений

На тему Разработка СППР при выборе медецинского учреждения

Руководитель
Ковалев Ю.А.
(фамилия, инициалы)

(оценка)

(подпись) (дата)

Студент ИСз - 120
(группа)

Члены комиссии

Малышева Е.В.
(фамилия, инициалы)

(подпись) (Ф.И.О.)

(подпись) (Ф.И.О.)

(подпись) (дата)

Муром 2023

В курсовом проекте выполнено проектирование и разработка диалоговой системы поддержки принятия решений при выборе медицинского учреждения. В ходе работы произведен анализ предметной области, сформированы требования к проектируемой системе, выбраны средства реализации. Математическая модель принятия решения реализована с помощью метода анализа иерархий.

Содержание

Введение

Системы поддержки принятия решений (СППР) – компьютерные автоматизированные системы, целью которых является помощь людям, принимающим решение в сложных условиях, для полного и объективного анализа предметной деятельности. СППР возникли в результате слияния управленческих информационных систем и систем управления базами данных.

В СППР используются разные методы: информационный поиск, интел лектуальный анализ данных, поиск знаний в базах данных, рассуждение на основе прецедентов, имитационное моделирование, эволюционные вычисления и генетические алгоритмы, нейронные сети, ситуационный анализ, когнитивное моделирование и др. Некоторые из этих методов были разработаны в рамках искусственного интеллекта. Близкие к СППР классы систем – это экспертные системы и автоматизированные системы управления.

Система поддержки принятия решений − это компьютерная система, которая путем сбора и анализа большого количества информации может влиять на процесс принятия решений организационного плана в бизнесе и предпринимательстве. Интерактивные системы позволяют руководителям получить полезную информацию из первоисточников, проанализировать ее, а также выявить существующие бизнес-модели для решения определенных задач. С помощью СППР можно проследить за всеми доступными информационными активами, получить сравнительные значения объемов продаж, спрогнозировать доход организации при гипотетическом внедрении новой технологии, а также рассмотреть все возможные альтернативные решения.

Система поддержки принятия решений − комплекс математических и эвристических методов и моделей, объединенных общей методикой формирования альтернатив управленческих решений в организационных системах, определения последствий реализации каждой альтернативы и обоснования выбора наиболее приемлемого управленческого решения.

СППР в большинстве случаев это интерактивная автоматизированная система, которая помогает ЛПР использовать данные и модели для идентификации и решения задач и принятия решений. Система должна обладать возможностью работать с интерактивными запросами с достаточно простым для изучения языком.

Появление в начале 80-х годов персональных компьютеров позволило автоматизировать ведение учета и обработку данных даже небольшим компаниям, не имеющим высококвалифицированного управленческого и технического персонала. Для этой категории потребителей программного обеспечения были созданы приложения нового, коммерческого типа, интегрирующие несколько разных функций и позволяющие нескольким частям приложения манипулировать единожды введенными данными.

1 Анализ технического задания

В данном курсовом проекте разрабатывается система поддержки принятия решений, реализующая минимально необходимый функционал для выбора медецинского учреждения по совокупности критериев. В ходе выполнения курсового проекта, будут решены следующие задачи:

- анализ предметной области;
- формирование требований к системе;
- разработка алгоритмов;
- программная реализация системы;
- тестирование программного продукта.

Результатом выполнения курсового проекта будет являться система поддержки принятия решений, которая по набору входных данных сможет рекомендовать пользователю медецинское учреждение.

Разработанная система поддержки принятия решений может быть полезна для любого человека заинтересованного в посещении медицинского учреждения.

Системы поддержки принятия решений DSS (Decision Support System) на базе аналитических данных подсказывают или помогают выбрать пользователю обоснованное решение. Они предназначены для:

- анализа аналитических данных для оценки сложившейся ситуации при выработке решения;
- выявления ограничений на принимаемое решение, противоречивых требований, формируемых внутренней и внешней средой;
- генерации списка возможных решений (альтернатив);
- оценки альтернатив с учетом ограничений и противоречивых требований для выбора решения;
- анализа последствий принимаемого решения;
- окончательного выбора решения.

Принятие решений предполагает последовательное прохождение следующих этапов: осмысление проблемы, диагностика, математическое или концептуальное моделирование, формулирование альтернатив и выбор из них наиболее удовлетворяющих поставленным целям, мониторинг осуществления решения.

В последнее время в СППР интегрируются системы, которые основаны на знаниях и это позволяет получать объяснения полученных решений. Также СППР эволюционируют и по уровню помощи, которую они оказывают лицу, принимающему решения, от пассивной поддержки к активной. Фактически, СППР с пассивным подходом к поддержке представляют собой интерактивные информационные системы с удобным интерфейсом. ЛПР выбирает альтернативы, оценивает их, при этом возможность анализировать даже простые альтернативы увеличивает продуктивность процесса принятия решений.

Система поддержки принятия решений — это диалоговая автоматизированная система, использующая правила принятия решений и соответствующие модели с базами данных, а также интерактивный компьютерный процесс моделирования.

Проектируемая СППР выбора медицинского учреждения предназначена для помощи пользователю в выборе учреждения на основании его оценки всех критериев.

СППР должна обеспечивать выбор учреждения на основании совокупности следующих критериев в любом их сочетании:

- Эффективное лечение;
- Отношение персонала;
- Качество оборудования;
- Быстрота обслуживания;
- Стоимость услуг;
- Озывы.

Каждая из альтернатив должна оцениваться пользователем по каждому из указанных выше критериев. Значение оценки для каждого из приведенных выше критериев должно быть выражены в качественном виде и оценивается в баллах.

В системе должен быть реализован автоматический алгоритм поддержки принятия решения. Входными данными алгоритма должны являться список альтернатив и оценка альтернатив по критериям. Выходными данными алгоритма должен быть список альтернатив с оценками (чем выше оценка, тем лучше альтернатива).

Работа пользователя с системой должна быть реализована в диалоговом режиме. Диалоговый режим должен быть реализован с помощью оконных форм.

Разрабатываться требуемая СППР будет в среде операционной системы Microsoft Windows. Это одна из самых распространенных операционных систем.

В качестве языка программирования для разработки СППР выбран язык Java в среде разработки IntelliJ IDEA. Для реализации графического интерфейса будет использована библиотека SWING.

Swing в Java является частью базового класса Java, который является независимым от платформы. Он используется для создания оконных приложений и включает в себя такие компоненты, как кнопка, полоса прокрутки, текстовое поле и т. д.

Swing в Java – это легкий инструментарий с графическим интерфейсом, который имеет широкий спектр виджетов для создания оптимизированных оконных приложений. Это часть JFC (Java Foundation Classes). Он построен на основе AWT API и полностью написан на Java. Он не зависит от платформы в отличие от AWT и имеет легкие компоненты.

2 Разработка алгоритма

2.1 Критерии выбор

На первом этапе определяется значимость каждого из критериев.

На основании анализа предметной области и сбора требований к системе на этапе анализа технического задания, выявлены следующие критерии, влияющие на выбор медицинского учреждения:

- Эффективное лечение;
- Отношение персонала;
- Качество оборудования;
- Быстрота обслуживания;
- Стоимость услуг;
- Озывы.

2.2. Дерево решений

В данной работе, принятие решения будет реализовано с помощью алгоритма анализа иерархий. Метод анализа иерархий является систематической процедурой для иерархического представления компонентов, определяющих суть любой проблемы. Метод состоит в декомпозиции проблемы на все более простые составляющие части и дальнейшей обработке последовательности суждений лица, принимающего решение (ЛПР), по парным сравнениям. В результате может быть выражена относительная степень взаимодействия элементов. Эти суждения затем выражаются численно. Метод анализа иерархии включает процедуры синтеза множественных суждений, выявления приоритетности критериев и нахождения альтернативных решений. Полученные таким образом значения являются оценками в шкале отношений и соответствуют некоторым численным оценкам.

Решение проблемы – это процедура поэтапного установления приоритетов. На первом этапе выявляются наиболее важные компоненты проблемы, на

втором – наилучший способ проверки наблюдений, испытания и оценка альтернатив; на следующем этапе вырабатывается решение и оценивается его качество. Процесс может быть проведен также над последовательностью иерархий: в этом случае результаты, полученные в одной из них, используются в качестве входных данных при изучении следующей. Метод многокритериального отбора систематизирует процесс решения такой многоступенчатой задачи.

Основные принципы метода анализа иерархий

1. Принцип идентичности и декомпозиции. Предусматривает структурирование проблем в виде иерархии или сети.

2. Принцип сравнительный суждений (парных сравнений). Предполагает, что элементы задачи (альтернативы и критерии) сравниваются попарно с позиции их воздействия на общую характеристику.

3. Принцип синтеза приоритетов. Предполагает формирование набора локальных приоритетов, которые выражают относительное влияние множества элементов на элемент примыкающего сверху уровня.

На рисунке 1 приведено дерево решений для выбора телевизора из трёх альтернатив по четырём указанным выше критериям.

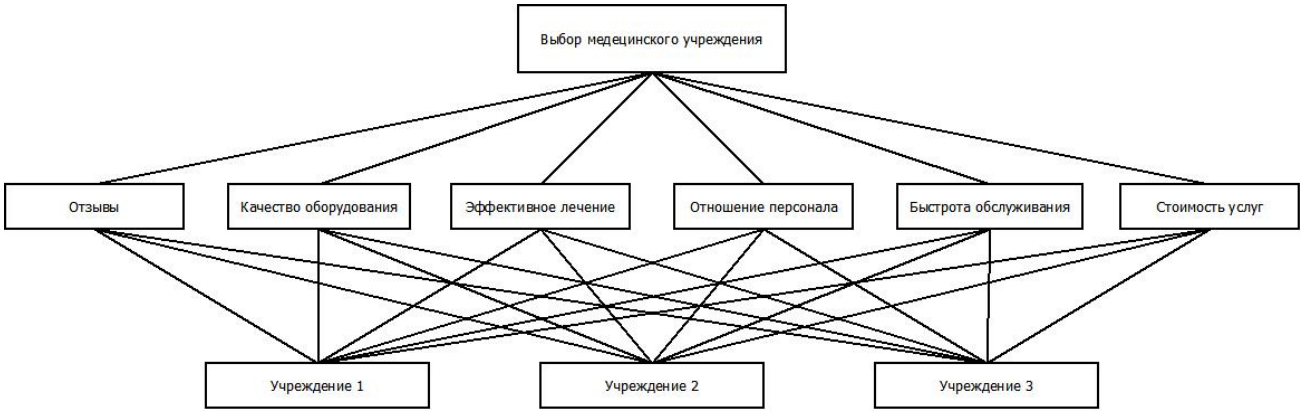

Рисунок 1

В качестве реализации алгоритма метода анализа иерархий в данном проекте используется «Метод анализа иерархий».

Обработка аналитических иерархий (Analytic Hierarchy Process, AHP) — структурированная техника принятия комплексных решений (en:MCDA). Она не дает ответа на вопрос, что правильно, а что нет, но позволяет человеку,

принимающему решение, оценить, какой из рассматриваемых им вариантов лучше всего удовлетворяет его нуждам и его пониманию проблемы (задачи).

2.3 Ранжирование критериев

Предположим, что у нас есть три альтернативы: Учреждение 1, Учреждение2 и Учреждение 3. Нам необходимо при помощи аналитического иерархического процесса выявить относительный приоритет каждого учреждения.

Итак, цель – выбор медецинского учреждения. Допустим, у нас есть шесть критериев, которые определяют выбор проекта: цена, качество, ассортимент и срок службы. Данный пример наглядно демонстрирует практическую применимость АНР: в зависимости от стратегии компании, упор может делаться на проекты с диаметрально противоположными характеристиками.

Сравним все критерии попарно. Для этого используем следующую шкалу:

1 - критерии равнозначны,

3 - один критерий имеет несколько большую значимость нежели другой,

5 - один критерий имеет существенно большую значимость нежели другой,

7 - один критерий имеет бесспорно большую значимость нежели другой, подтверждается не только экспертным путём, но и на практике,

9 - один критерий имеет абсолютно большую значимость нежели другой.

Стоит отметить, что если приоритет А над Б равен 7, то приоритет Б над А равен 1/7.

Допустим, что мы сравнили попарно три критерия и получили следующие результаты:

Таблица 1 Сравнение критериев

Критерии	Эффективное лечение	Отношение персонала	Качество оборудования	Быстрота обслуживания	Стоимость услуг	Озывы
Эффективное лечение	1	1	4	2	3	2
Отношение персонала	1	1	5	2	5	3
Качество оборудования	1/4	1/5	1	1	3	2
Быстрота обслуживания	1/2	1/2	1	1	3	4
Стоимость услуг	1/3	1/5	1/3	1/3	1	1/4
Озывы	1/2	1/3	1/2	1/4	4	1

Теперь посчитаем сумму в каждом столбце и разделим значение каждой ячейки на сумму значений соответствующего столбца (см. таблицу 2).

Таблица 2 Нормализированная матрица

Критерии	Эффективное лечение	Отношение персонала	Качество оборудования	Быстрота обслуживания	Стоимость услуг	Озывы
Эффективное лечение	0,2791	0,3093	0,3380	0,3038	0,1579	0,1633
Отношение персонала	0,2791	0,3093	0,4225	0,3038	0,2632	0,2449
Качество оборудования	0,0698	0,0619	0,0845	0,1519	0,1579	0,1633
Быстрота обслуживания	0,1395	0,1546	0,0845	0,1519	0,1579	0,3265
Стоимость услуг	0,0930	0,0619	0,0282	0,0506	0,0526	0,0204
Озывы	0,1395	0,1031	0,0423	0,0380	0,2105	0,0816

Посчитав средние значения по строкам, мы найдем удельный вес каждого из критериев (см. таблицу 3).

Таблица 3 удельный вес критериев

Эффективное лечение	Отношение персонала	Качество оборудования	Быстрота обслуживания	Стоимость услуг	Озывы
0,2586	0,3038	0,1149	0,1692	0,0511	0,1025

2.4 Ранжирование альтернатив по критериям

Ранжирование альтернатив производится отдельно по каждому из критериев. В нашем примере четыре критерия. Важно, чтобы шкала для каждого из них имела одинаковый диапазон значений. Для этого используется величина оценки каждого критерия из таблицы 1.

Допустим, что экспертным путём было выявлено, что каждая из фирм производителей телевизоров заслуживает следующих оценок:

Таблица 4. Оценки фирм по критериям

Эффективное лечение	Учреждение 1	Учреждение 2	Учреждение 3
Учреждение 1	1	4	3
Учреждение 2	1/4	1	1
Учреждение 3	1/3	1	1
Отношение персонала	Учреждение 1	Учреждение 2	Учреждение 3
Учреждение 1	1	2	5
Учреждение 2	1/2	1	4
Учреждение 3	1/5	1/4	1
Качество оборудования	Учреждение 1	Учреждение 2	Учреждение 3
Учреждение 1	1	2	3
Учреждение 2	1/2	1	2
Учреждение 3	1/3	1/2	1
Быстрота обслуживания	Учреждение 1	Учреждение 2	Учреждение 3
Учреждение 1	1	1	2

Учреждение 2	1	1	3
Учреждение 3	1/2	1/3	1
Стоимость услуг	Учреждение 1	Учреждение 2	Учреждение 3
Учреждение 1	1	3	4
Учреждение 2	1/3	1	2
Учреждение 3	1/4	1/2	1
Озывы	Учреждение 1	Учреждение 2	Учреждение 3
Учреждение 1	1	2	5
Учреждение 2	1/2	1	5
Учреждение 3	1/5	1/5	1

Взяв каждую из оценок с удельным весом критерия, найденным ранее, и сложив их по каждой альтернативе, получим:

Таблица 5 итоговая оценка учреждений

Учреждение 1	Учреждение 2	Учреждение 3
0,5524	0,3042	0,1434

Таким образом, будет выбрано медицинское учреждение 1, т.к. данная альтернатива имеет наибольший удельный вес.

3 Разработка программы

3.1 Реализация ввода вывода

Ввод исходных данных реализован в диалоговых окнах, реализованных в среде разработки IntelliJ IDEA. Для реализации графического интерфейса использована библиотека SWING.

Входными данными для диалогового окна является два одномерных массива String со списком названий критериев и альтернтив.

Выходными данными диалогового окна является вывод значений результаты расчетов в виде double массива где индекс массива альтернатив будет являться индексом результата его оценки.

3.2 Реализация алгоритма принятия решения

Алгоритм принятия решения реализован следующим образом:

Был создан класс MatrixPaired, который включает в себя методы для расчетов разных параметров и для правильного заполнения данных парной матрицы.

Присвоение данных матрице, где первый аргумент - строка, второй аргумент - ряд таблицы, и третий аргумент - оценка, после установления значения, значение первого и второго аргумента меняются автоматически и в качестве третьего аргумента выступает 1 деленный на этот аргумент:

```
M.setMatrixPaired(0, 1, 1);
M.setMatrixPaired(0, 2, 4);
M.setMatrixPaired(0, 3, 2);
M.setMatrixPaired(0, 4, 3);
M.setMatrixPaired(0, 5, 2);
M.setMatrixPaired(1, 2, 5);
M.setMatrixPaired(1, 3, 2);
M.setMatrixPaired(1, 4, 5);
M.setMatrixPaired(1, 5, 3);
```

Расчет итоговой оценки происходит через метод getWi(), который возвращает double массив удельных весов:

M.getWi();

Для оценок альтернатив был создан класс MatrixPairedAlternatives, хранящий в себе массивы альтернатив и массив критериев. Для получение общей оценки всех альтернатив, создан метод W(), который возрващает double массив оценок:

A.W();

4 Руководство по программному продукту

4.1 Руководство программиста

Программное обеспечение представляет собой windows-приложение, разработанное в среде разработки IntelliJ IDEA на языке программирования Java. Для реализации графического интерфейса была использована библиотека SWING.

IntelliJ IDEA — это IDE, интегрированная среда разработки (комплекс программных средств, который используется для написания, исполнения, отладки и оптимизации кода) для Java, JavaScript, Python и других языков программирования от компании JetBrains. Отличается обширным набором инструментов для рефакторинга (перепроектирования) и оптимизации кода.
Программы на Java транслируются в байт-код Java, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной.

Реализация всего алгоритма осуществлена в классе Main.

4.2 Руководство администратора

Для запуска программ Java на компьютере требуется JVM, зависящая от платформы.

Java Virtual Machine (сокращенно Java VM, JVM) — виртуальная машина Java — основная часть исполняющей системы Java, так называемой Java Runtime Environment (JRE). Виртуальная машина Java исполняет байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java (javac). JVM может также использоваться для выполнения программ, написанных на других языках программирования. Например, исходный код на языке Ada может быть скомпилирован в байт-код Java, который затем может выполниться с помощью JVM.

JVM является ключевым компонентом платформы Java. Так как виртуальные машины Java доступны для многих аппаратных и программных платформ, Java может рассматриваться и как связующее программное обеспечение, и как самостоятельная платформа. Использование одного байт-кода для многих платформ позволяет описать Java как «скомпилируй единожды, запускай везде» (compile once, run anywhere).

4.3 Руководство пользователя

Первым этапом идет ввод кол-ва альтернатив и их называний:

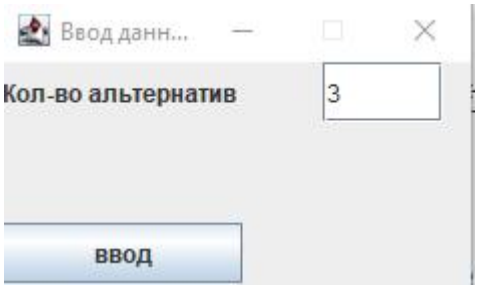

Рисунок 2 - ввод кол-ва альтернатир

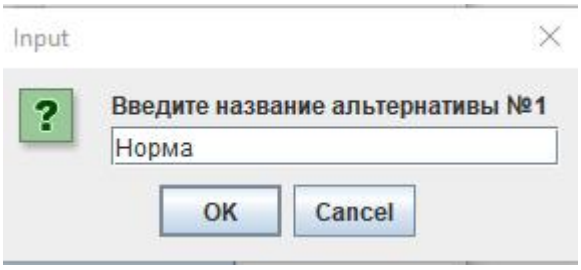

Рисунок 3 - ввод названия альтернатив

Вторым этапом будет заполнение парных матриц, основанных на личной оценке пользователя:

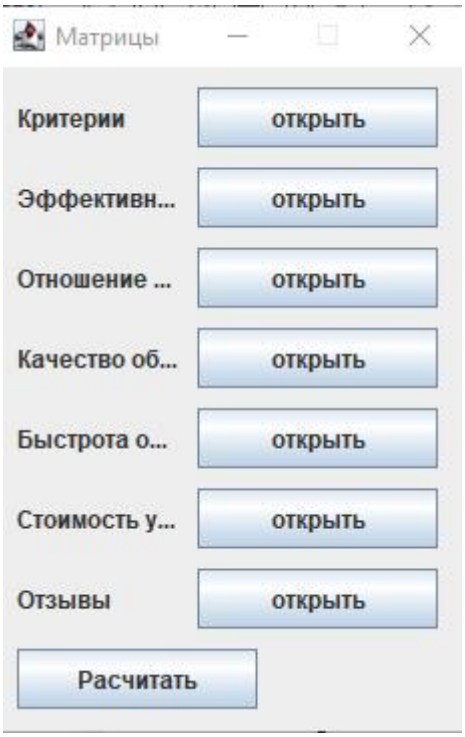

Рисунок 5 - выбор матриц для заполнения

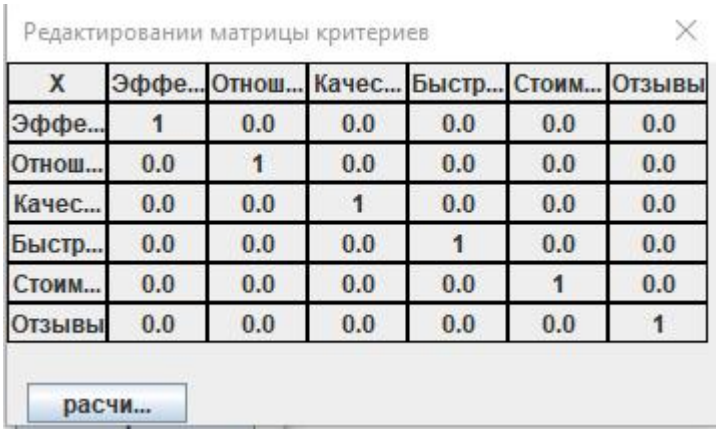

Рисунок 6 - матрица

Третим этапом будет ввод данных в матрицу, где пользователь может ввести целое число от 1 до 9, если он введет отрицательное число от 1 до 9, оно будет расценено как 1/n, где n - число от 1 до 9. В случае неправильного ввода, программа выдаст предупреждение:

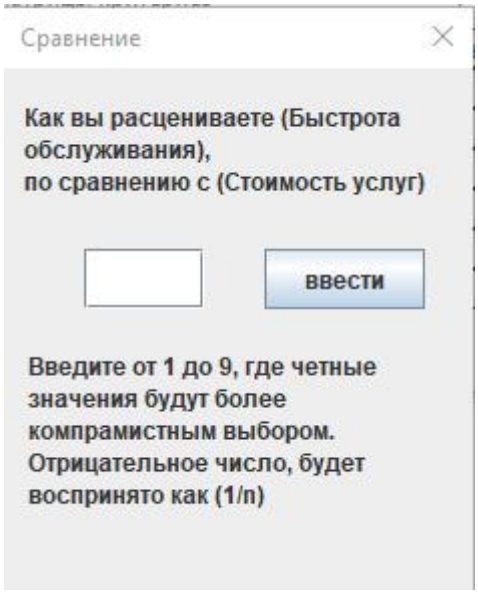

Рисунок 7 - ввод данных в матрицу

Четвертым этапом будет завершение редактирования матрицы, где нужно нажать расчет, для того чтобы узнать, является ли она согласованной или заполненой, в альтернативном случае пользователь получит предупреждение, но сможет дальше продолжить работу.

Пятым этапом будет полный расчет, если пользователь ввел или не ввел какие-то данные, (см. Рисунок 5), нужно будет нажать кнопку «расчитать», после чего появится окно с результатами оценок (см. Рисунок 8).

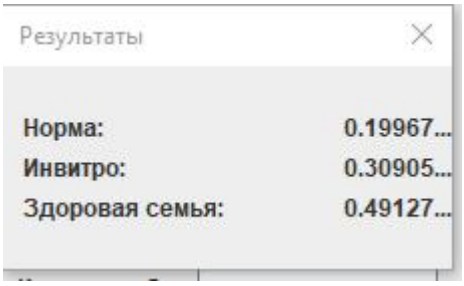

Рисунок 8 - результаты

5 Тестирование и отладка

В ходе разработки приложения, производилась консольная отладка вывода данных, в классе Main2, где вручную вводились данные и результаты расчетов выводились в консоль, для перерасчетов и перенастройки алгоритмов, если они требовались.

Вывод информации в консоль выводился следующим образом:

```
  static void print(String title, MatrixPaired matrix) {
    System.out.println(title);
    print("MatrixPaired", matrix.getMatrixPaired());
    print("MatrixNormalized", matrix.getMatrixNormalized());
    print("Wi", matrix.getWi());
    print("Nmax", matrix.getNmax());
    print("CI", matrix.getCI());
    print("RI", matrix.getRI());
    print("CR", matrix.getCR());
    System.out.println("\n");
  }
  static void print(String title, double[][] arr) {
    System.out.println(title);
    for(int i = 0; i < arr.length; i++) {
      for (int j = 0; j < arr[i].length; j++) {
        System.out.print(String.format("%.4f", arr[i][j]) + ' ');
      }
      System.out.println();
    }
  }
  static void print(String title, double[] arr) {
    System.out.print(title);
    for (int i = 0; i < arr.length; i++) {
      System.out.print(String.format(" %.4f", arr[i]));
    }
    System.out.println();
  }
  static void print(String title, double num) {
    System.out.print(title);
    System.out.println(String.format(" %.4f", num));
  }
}
```

По результатам тестирования, был созданн оконный интерфейс, для удобного восприятия, пользователями.

Заключение

В данном курсовом проекте разработана диалоговая система поддержки принятия решения при выборе медицинского учреждения.

В ходе выполнения курсового проекта, были решены следующие задачи:

- анализ предметной области;
- формирование требований к системе;
- разработка алгоритмов;
- программная реализация системы;
- тестирование программного продукта.

В результате выполнения курсового проекта, разработана система поддержки принятия решений, которая по набору входных данных сможет рекомендовать пользователю медицинское учреждение, наиболее подходящих его критериям оценки.

Разработанная система позволяет производить подбор наиболее подходящих медицинский учреждений по следующим критериям:

- Эффективное лечение;
- Отношение персонала;
- Качество оборудования;
- Быстрота обслуживания;
- Стоимость услуг;
- Озывы.

Разработанная система поддержки принятия решений может быть полезна для любого человека заинтересованного в тчательном выборе медисноского учреждения.

Список литературы

1. Берд, Барри Java для чайников / Барри Берд. - М.: Диалектика / Вильямс, 2013. - 521 с.;

2. Дробышев, А.В . Методы принятия решений. Методы Дельфи и ЭЛЕКТРА. - Методические указания к лабораторной работе по курсу "Системы поддержки принятий решений". - МГИЭМ. Сост.: И.Е.Сафонова,., К.Ю.Мишин, С.В.Цыганов: М., МГИЭМ, 2008. - 26 с.;

3. Евланов, А. Г. Теория и практика принятия решений. -- М.: Экономика, 2010. - 212 с.;

4. Ромащенко, В.Н. Принятие решений: ситуации и советы. - Киев, 2012. - 154 с.;

5. Шилдт, Герберт Java 8. Руководство для начинающих / Герберт Шилдт. - М.: Вильямс, 2015. - 720 с.

Приложение

Листинг программы:

```
public class Main
{
  public static void main(String[] args)
  {
    new WindowStart();
  }
}
import javax.swing.*;

public class WindowStart extends JFrame
{
  JTextField inputCountCriteria;
  JTextField inputCountAlternatives;
  public WindowStart()
  {
    setTitle("Ввод данных");
    /*inputCountCriteria = new JTextField();
    inputCountCriteria.setBounds(160,0,60,30);
    inputCountAlternatives = new JTextField();
    inputCountAlternatives.setBounds(160,40,60,30);*/
    /*JLabel text1 = new JLabel("Кол-во критериев");
    text1.setBounds(0, 0, 150, 30);
    JLabel text2 = new JLabel("Кол-во альтернатив");
    text2.setBounds(0, 40, 150, 30);*/
    inputCountAlternatives = new JTextField();
    inputCountAlternatives.setBounds(160,0,60,30);
    JLabel text2 = new JLabel("Кол-во альтернатив");
    text2.setBounds(0, 0, 150, 30);


    JButton button = new JButton("ввод");
    button.setBounds(0, 80, 120, 30);
    button.addActionListener(e -> { click(); });
    //add(inputCountCriteria);
    add(inputCountAlternatives);
    //add(text1);
    add(text2);
    add(button);
```

```
    setSize(250, 150);
    setResizable(false);
    setLayout(null);
    setVisible(true);
  }
  void click()
  {
    int Criteria = 6;
    int Alternatives = 0;
    try
    {
      //Criteria = Integer.parseInt(inputCountCriteria.getText());
      Alternatives = Integer.parseInt(inputCountAlternatives.getText());
    } catch (Exception e){}
    if(Criteria < 3 || Alternatives < 3)
    {
      JOptionPane.showMessageDialog(this,"Вы ввели неправельные значения!");
    }
    else
    {
      //String[] CriteriaT = new String[Criteria];
      String[] CriteriaT = new String[]
          {
              "Эффективность лечения",
              "Отношение персонала",
              "Качество оборудования",
              "Быстрота обслуживания",
              "Стоимость услуг",
              "Отзывы"
          };
      String[] AlternativesT = new String[Alternatives];
      /*for(int i = 0; i < Criteria; i++)
      {
        CriteriaT[i] = JOptionPane.showInputDialog(this, "Введите название критерия №" + (i+1));
      }*/
      for(int i = 0; i < Alternatives; i++)
      {
        AlternativesT[i] = JOptionPane.showInputDialog(this, "Введите название альтернативы №" + (i+1));
      }
      MatrixPairedAlternatives M = new MatrixPairedAlternatives(Alternatives, Criteria);
      M.getCriteria().setTitle(CriteriaT);
      for(int i = 0; i < Criteria; i++)
```

```
        {
          M.getAlternative(i).setTitle(AlternativesT);
        }
        new WindowMain(this, M);
        dispose();
      }
    }
}
import javax.swing.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class WindowMain extends JFrame
{
  MatrixPairedAlternatives matrix;
  public WindowMain(JFrame parent, MatrixPairedAlternatives Matrix)
  {
    matrix = Matrix;
    setTitle("Матрицы");
    setLocation(parent.getLocation().x, parent.getLocation().y);
    JButton button1 = new JButton("открыть");
    button1.addMouseListener(new MouseListenerA(-1));
    button1.setBounds(100, 10, 120, 30);
    add(button1);
    JLabel text1 = new JLabel("Критерии");
    text1.setBounds(10, 10, 80, 30);
    add(text1);
    for(int i = 0; i < matrix.getAlternativeLength(); i++)
    {
      JButton button = new JButton("открыть");
      button.setBounds(100, (i + 1) * 40 + 10, 120, 30);
      button.addMouseListener(new MouseListenerA(i));
      add(button);
      JLabel text = new JLabel(matrix.getCriteria().getTitle()[i]);
      text.setBounds(10, (i + 1) * 40 + 10, 80, 30);
      add(text);
    }
    setSize(250, (matrix.getAlternativeLength() + 3) * 40 + 10);
    addWindowListener(new WindowAdapter() { public void windowClosing(WindowEvent e) { close(); }});
    JButton button2 = new JButton("Расчитать");
```

```
    button2.setBounds(10, (matrix.getAlternativeLength() + 1) * 40 + 10, 120, 30);
    button2.addActionListener(e -> { clickCalc(); });
    add(button2);
    setResizable(false);
    setLayout(null);
    setVisible(true);
  }
  void clickCalc()
  {
    if(matrix.isFull())
    {
      if(matrix.isConsistent())
      {
        calc();
      }
      else
      {
        int confirmed = JOptionPane.showConfirmDialog(this, "У вас несоглассована какая-то матрица. Продолжить?", "Предупреждение", JOptionPane.YES_NO_OPTION);
        if (confirmed == JOptionPane.YES_OPTION)
        {
          calc();
        }
      }
    }
    else
    {
      int confirmed = JOptionPane.showConfirmDialog(this, "У вас незаполнена какая-то матрица. Продолжить?", "Предупреждение", JOptionPane.YES_NO_OPTION);
      if (confirmed == JOptionPane.YES_OPTION)
      {
        if(matrix.isConsistent())
        {
          calc();
        }
        else
        {
          int confirmed2 = JOptionPane.showConfirmDialog(this, "У вас так-же несоглассована какая-то матрица. Продолжить?", "Предупреждение", JOptionPane.YES_NO_OPTION);
          if (confirmed2 == JOptionPane.YES_OPTION)
          {
            calc();
```

```
            }
          }
        }
      }
    }
    void calc()
    {
      double[] r = matrix.W();
      new WindowResult(this, r, matrix.getAlternative(0).getTitle());
    }
    void close()
    {
      int confirmed = JOptionPane.showConfirmDialog(this, "Вы действительно хотите выйти?", "Выход", JOptionPane.YES_NO_OPTION);
      if (confirmed == JOptionPane.YES_OPTION)
      {
        dispose();
      }
      else
      {
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
      }
    }
    void click(int i)
    {
      if(i == -1)
      {
        new WindowMatrix(this, "Редактировании матрицы критериев", matrix.getCriteria(), null);
      }
      else
      {
        new WindowMatrix(this, "Редактировании матрицы алтернатив", matrix.getAlternative(i), matrix.getCriteria().getTitle()[i]);
      }
    }

    class MouseListenerA implements MouseListener
    {
      int i;
      public MouseListenerA(int i)
      {
        this.i = i;
```

```
    }
    @Override
    public void mouseClicked(MouseEvent e)
    {
      click(i);
    }
    @Override public void mousePressed(MouseEvent e) { }
    @Override public void mouseReleased(MouseEvent e) { }
    @Override public void mouseEntered(MouseEvent e) { }
    @Override public void mouseExited(MouseEvent e) { }
  }
}
import jdk.nashorn.internal.ir.Block;

import javax.swing.*;
import javax.swing.table.TableColumn;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class WindowMatrix extends JDialog
{
  static int WidthBlock = 50;
  static int HeightBlock = 20;
  WindowMatrix This = this;
  private JLabel[][] Table;
  private JLabel[] Wi;
  private JLabel nmax;
  private JLabel cr;
  MatrixPaired Matrix;
  String criteria;


  public WindowMatrix(JFrame parent, String title, MatrixPaired matrix, String criteria)
  {
    super(parent, ModalityType.APPLICATION_MODAL);
    if(parent != null)
    {
      setLocation(parent.getLocation().x + 20, parent.getLocation().y + 20);
    }
```

```
    this.criteria = criteria;
    Matrix = matrix;
    Table = new JLabel[Matrix.getN()][Matrix.getN()];
    setTitle(title);
    setSize((Matrix.getN()+1/*4*/)*(WidthBlock) + 20, (Matrix.getN()+1)*HeightBlock + 80);
    Creator();
    setResizable(false);
    setVisible(true);
  }
  public boolean set(int x, int y, double n)
  {
    boolean r = Matrix.setMatrixPaired(x, y, n);
    if(r)
      update();
    return r;
  }


  private void calc()
  {
    Matrix.setCached(true);
    nmax.setText(Matrix.getNmax() + "");
    cr.setText(Matrix.getCR() + "");
    if(Matrix.isConsistent())
      cr.setForeground(Color.GREEN);
    else
      cr.setForeground(Color.RED);
    for (int i = 0; i < Table.length; i++)
      this.Wi[i].setText(Matrix.getWi()[i] + "");
    Matrix.setCached(false);
  }
  private void update()
  {
    nmax.setText("0");
    cr.setText("0");
    cr.setForeground(Color.BLACK);
    for(int i = 0; i < Table.length; i++)
    {
      Wi[i].setText("0");
      for (int j = 0; j < Table[i].length; j++)
        if(i != j)
          Table[i][j].setText(Matrix.getMatrixPaired()[i][j] + "");
```

```
    }
  }
  private void Creator()
  {
    setLayout(null);
    JLabel table = new JLabel();
    table.setBounds(0, 0, (Matrix.getN()+1)*WidthBlock, (Matrix.getN()+1)*HeightBlock);
    table.add(CreatorBlock("X"));
    for(int i = 0; i < Matrix.getN(); i++)
    {
      for(int j = 0; j < 2; j++)
      {
        JLabel block = CreatorBlock(Matrix.getTitle()[i]);
        block.addMouseListener(new MouseListenerA(block));
        if(j == 0)
          block.setLocation(0, (i+1) * HeightBlock);
        else if(j == 1)
          block.setLocation((i+1) * WidthBlock, 0);
        table.add(block);
      }
    }
    for(int i = 0; i < Matrix.getN(); i++)
    {
      for (int j = 0; j < Matrix.getN(); j++)
      {
        JLabel block = CreatorBlock("");
        block.setLocation((i+1) * WidthBlock, (j+1) * HeightBlock);
        if(i != j)
          block.addMouseListener(new MouseListenerB(i, j));
        else
          block.setText("1");
        Table[i][j] = block;
        table.add(block);
      }
    }


    JLabel Wi = CreatorBlock("Wi");
    Wi.setLocation((Matrix.getN()+2) * WidthBlock + 20 , 0);
    add(Wi);
    this.Wi = new JLabel[Matrix.getN()];
    for(int i = 0; i < Matrix.getN(); i++)
```

```
{
  JLabel block = CreatorBlock("0");
  block.setLocation((Matrix.getN()+2)* WidthBlock + 20 , (i + 1) * HeightBlock);
  this.Wi[i] = block;
  add(block);
}
JButton buttonCalc = new JButton("расчитать");
buttonCalc.addMouseListener(new MouseListenerC());
buttonCalc.setSize(80, HeightBlock);
buttonCalc.setLocation(10, (Matrix.getN() + 2) * HeightBlock);
add(buttonCalc);

JLabel nmaxT = CreatorBlock("Nmax");
nmaxT.setLocation((Matrix.getN()+1) * WidthBlock + 20, (Matrix.getN() + 1) * HeightBlock);
add(nmaxT);
JLabel crT = CreatorBlock("CR");
crT.setLocation((Matrix.getN()+1) * WidthBlock + 20, (Matrix.getN() + 2) * HeightBlock);
add(crT);
nmax = CreatorBlock("0");
nmax.setLocation((Matrix.getN()+2) * WidthBlock + 20, (Matrix.getN() + 1) * HeightBlock);
add(nmax);
cr = CreatorBlock("0");
cr.setLocation((Matrix.getN()+2) * WidthBlock + 20, (Matrix.getN() + 2) * HeightBlock);
add(cr);

update();
add(table);



addWindowListener(new WindowAdapter() {
  public void windowClosing(WindowEvent e)
  {
    if(cr.getForeground() != Color.GREEN || !Matrix.isFullMatrixPaired())
    {
      String text;
      if(cr.getForeground() != Color.GREEN)
      {
        text = "У вас несогласованная матрица! Вы хотите выйти?";
      }
      else
      {
```

```
            text = "У вас не заполнена матрица! Вы хотите выйти?";;
          }
          int confirmed = JOptionPane.showConfirmDialog(This, text, "Выход", JOptionPane.YES_NO_OPTION);
          if (confirmed == JOptionPane.YES_OPTION)
          {
            dispose();
          }
          else
          {
            setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
          }
        }
        else
        {
          dispose();
        }
      }
    });
}
private JLabel CreatorBlock(String text)
{
  JLabel r = new JLabel(text);
  r.setSize(WidthBlock, HeightBlock);
  r.setBorder(BorderFactory.createLineBorder(Color.black));
  r.setHorizontalAlignment(SwingConstants.CENTER);
  r.setVerticalAlignment(SwingConstants.CENTER);
  return r;
}
private void onMessage(String text)
{
  JOptionPane.showMessageDialog(this,text);
}
class MouseListenerA implements MouseListener
{
  JLabel l;
  public MouseListenerA(JLabel l)
  {
    this.l = l;
  }
  @Override
  public void mouseClicked(MouseEvent e)
  {
```

```
        onMessage(l.getText());
      }
      @Override public void mousePressed(MouseEvent e) { }
      @Override public void mouseReleased(MouseEvent e) { }
      @Override public void mouseEntered(MouseEvent e) { }
      @Override public void mouseExited(MouseEvent e) { }
    }
    class MouseListenerB implements MouseListener
    {
      int x, y;
      public MouseListenerB(int x, int y)
      {
        this.y = y;
        this.x = x;
      }
      @Override
      public void mouseClicked(MouseEvent e)
      {
        JDialog d = new WindowAssessment(This, x, y, Matrix.getMatrixPaired()[x][y], Matrix.getTitle()[y],
Matrix.getTitle()[x], criteria);
      }
      @Override public void mousePressed(MouseEvent e) { }
      @Override public void mouseReleased(MouseEvent e) { }
      @Override public void mouseEntered(MouseEvent e) { }
      @Override public void mouseExited(MouseEvent e) { }
    }
    class MouseListenerC implements MouseListener
    {
      @Override
      public void mouseClicked(MouseEvent e)
      {
        if(Matrix.isFullMatrixPaired())
        {
          calc();
        }
        else
        {
          int dialogButton = JOptionPane.showConfirmDialog (This, "Матрица не заполнена! Продолжить?", "Матрица
не заполнена!", JOptionPane.YES_NO_OPTION);
          if(dialogButton == JOptionPane.YES_OPTION)
          {
            calc();
```

```
        }
      }
    }
    @Override public void mousePressed(MouseEvent e) { }
    @Override public void mouseReleased(MouseEvent e) { }
    @Override public void mouseEntered(MouseEvent e) { }
    @Override public void mouseExited(MouseEvent e) { }
  }
}
import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;

public class WindowAssessment extends JDialog
{
  int x, y;
  int n;
  WindowMatrix parent;
  JTextField input;
  public WindowAssessment(WindowMatrix parent, int x, int y, double n, String a, String b, String c)
  {
    super(parent, ModalityType.APPLICATION_MODAL);
    if(n < 1)
      n = -(1 / n);
    this.n = (int)n;
    this.x = x;
    this.y = y;
    this.parent = parent;
    setTitle("Сравнение");
    Creator(a, b, c);
    setVisible(true);
  }
  void Creator(String a, String b, String c)
  {
    setLocation(parent.getLocation().x + 20, parent.getLocation().y + 20);
    setResizable(false);
    setSize(250, 300);

    input = new JTextField();
    input.setSize(60, 30);
    JLabel text1;
```

```
    if(c != null)
      text1 = new JLabel("<html>Как вы расцениваете ("+a+"), по сравнению с (" + b + "), по критерию (" + c +
");</html>");
    else
      text1 = new JLabel("<html>Как вы расцениваете ("+a+"),<br>по сравнению с (" + b + ")</html>");
    text1.setSize(230, 80);
    JLabel text2 = new JLabel("<html>Введите от 1 до 9, где четные значения будут более компрамистным
выбором. Отрицательное число, будет воспринято как (1/n)</html>");
    text2.setSize(226, 100);
    JButton button = new JButton("ввести");
    button.setSize(80, 30);



    button.addActionListener(e -> { check(); });


    setLayout(null);
    text1.setLocation(10,0);
    input.setLocation(40,90);
    button.setLocation(130,90);
    text2.setLocation(12,130);
    add(text1);
    add(input);
    add(button);
    add(text2);
  }
  void check()
  {
    String a = input.getText();
    char num = ' ';
    int b = 0;
    if(a.length() == 1)
    {
      b = 1;
      num = a.charAt(0);
    } else if(a.length() == 2 && a.charAt(0) == '-')
    {
      b = -1;
      num = a.charAt(1);
    }
    try
    {
      b = b * Integer.parseInt(num + "");
```

```
    }
    catch (Exception E)
    {
      b = 0;
    }
    if(b <= -1 && b >= -9 || b >= 1 && b <= 9)
    {
      set(b);
    }
    else
    {
      JOptionPane.showMessageDialog(this,"Вы ввели неправельное значение!");
    }
  }
  void set(int a)
  {
    double n = a > 0 ? a : 1 / (double)(-a);
    if(parent.set(x, y, n))
    {
      dispose();
    }
    else
    {
      JOptionPane.showMessageDialog(this,"Произошла непредвиденная ошибка!");
    }
  }
}
import javax.swing.*;

public class WindowResult extends JDialog
{
  public WindowResult(JFrame parent, double[] result, String[] title)
  {
    super(parent, ModalityType.APPLICATION_MODAL);
    setTitle("Результаты");

    for(int i = 0; i < result.length; i++)
    {
      JLabel text1 = new JLabel(title[i] + ":");
      JLabel text2 = new JLabel(result[i] + "");
      text1.setBounds(10, (i-1) * 20, 30, 80);
      text2.setBounds(100, (i-1) * 20, 30, 80);
```

```
        add(text1);
        add(text2);
      }
      setSize(200, result.length * 20+60);
      setLocation(parent.getLocation().x + 20, parent.getLocation().y + 20);
      setLayout(null);
      setResizable(false);
      setVisible(true);
    }
  }
  /**
   * Объект для хранения и работы с Парной матрицей
   * @see #isCached()
   */
  public class MatrixPaired
  {
    private String[] Title;
    private double[][] MatrixPaired;
    private double[][] MatrixNormalized;
    private double[] Wi;
    private double Nmax = none;
    private boolean cache = false;
    private static short none = -1000;

    /**
     * Конструктор
     * @param n Размер матрицы
     * @param cached Должны ли кэшироваться данные
     * @see #isCached()
     */
    public MatrixPaired(int n, boolean cached)
    {
      cache = cached;
      MatrixPaired = new double[n][n];
      Title = new String[n];
      for(int i =0; i < n; i++)
      {
        Title[i] = ""+i;
        MatrixPaired[i][i] = 1;
      }
    }
```

```
  /**
   * Конструктор
   * @param n Размер матрицы
   */
  public MatrixPaired(int n)
  {
    this(n, false);
  }

  /**
   * Функция возвращающая информацию, о кэшированнии данных
   * @apiNote
   * Кэширование по умолчанию не устанавливается.
   * Если данное значение установлено на true, то используется минимум расчетов и все вычисленные данные сохраняются и не требуют перерасчетов,
   * но пользователь может изменить кэшированные данные через получение ссылки на них с помощью методов:
   * {@link #getMatrixNormalized()}, {@link #getWi()}.
   * В противном случае, тратится вычислительная мощность, на перерасчет данных объектов.
   * Так-же, метод {@link #getMatrixPaired()} дает ссылку на матрицу, которая всегда находится в кэше.
   * @see #ClaerCache()
   * @see #setCached(boolean)
   * @see #setMatrixPaired(int, int, double)
   * @see #MatrixPaired(int, boolean)
   */
  public boolean isCached()
  {
    return cache;
  }

  /**
   * Является ли матрица приемлемо согласованной
   */
  public boolean isConsistent()
  {
    return getCR() <= 0.1;
  }

  /**
   * Является ли матрица идеально согласованной
   */
  public boolean isPerfectlyConsistent()
  {
```

```
    return getNmax() == getN();
  }

  /**
   * Функция отчистки кэша
   * @see #isCached()
   */
  public void ClaerCache()
  {
    MatrixNormalized = null;
    Wi = null;
    Nmax = none;
  }

  /**
   * Функция изменения статуса кэширования
   * @param cached должны ли кэшироваться данные. В случае если не должны, то автоматически вызывается {@link #ClaerCache()}
   * @see #isCached()
   */
  public void setCached(boolean cached)
  {
    if(!cached)
    {
      ClaerCache();
    }
    this.cache = cached;
  }

  /**
   * Проверяет, является ли парная матрица, полностью заполненой
   * @return true если в ней нет значений с 0
   * @see #setMatrixPaired(int, int, double)
   */
  public boolean isFullMatrixPaired()
  {
    for(int i = 1; i < getN(); i++)
      for(int j = i; j < getN(); j++)
        if(MatrixPaired[i][j] == 0)
          return false;
    return true;
  }
```

```
/**
 * Функция по устанавления данных в матрицу MatrixPaired
 * @param x позиция по x, которое должно быть от 1 до N-1
 * @param y позиция по y, которое должно быть от 1 до N-1
 * @param number устанавливаемое значение, которое должно быть от 1 до N
 * @return установилось ли значение. Если установилось, то вызывается функция {@link #ClaerCache()}
 * @<code>
 *     MatrixPaired[x][y] = number;
 *     <br>
 *     MatrixPaired[y][x] = 1 / number;
 * </code>
 * @see #isCached()
 */
public boolean setMatrixPaired(int x, int y, double number)
{
  if(x == y || x < 0 || y < 0 || x >= getN() || y >= getN() || number <= 0 || number > 9)
  {
    return false;
  }
  MatrixPaired[x][y] = number;
  MatrixPaired[y][x] = 1 / number;
  ClaerCache();
  return true;
}

/**
 * Размер парной матрицы
 * @return значение больше 0;
 */
public int getN()
{
  return MatrixPaired.length;
}

/**
 * Парная матрица
 * @return ссылка на объект
 */
public double[][] getMatrixPaired()
{
  return MatrixPaired;
```

```
}

/**
 * Нормализованная матрица
 * @return ссылка на объект
 */
public double[][] getMatrixNormalized()
{
  if(MatrixNormalized != null)
  {
    return MatrixNormalized;
  }
  double[][] MatrixNormalized = MatrixNormalized(getN(), MatrixPaired);
  if(cache)
  {
    this.MatrixNormalized = MatrixNormalized;
  }
  return MatrixNormalized;
}

/**
 * Wi - Среднее значение строк нормализованной матрицы, или относительный вес
 * @return ссылка на объект
 */
public double[] getWi()
{
  if(Wi != null)
  {
    return Wi;
  }
  double[][] MatrixNormalized = this.MatrixNormalized != null ?
      this.MatrixNormalized :
      MatrixNormalized(getN(), MatrixPaired);
  double[] Wi = Wi(getN(), MatrixNormalized);
  if(cache)
  {
    this.MatrixNormalized = MatrixNormalized;
    this.Wi = Wi;
  }
  return Wi;
}
```

```
/**
 * Коэфициент Nmax
 * @return числовое значение
 */
public double getNmax()
{
  if(Nmax != none)
  {
    return Nmax;
  }
  double[][] MatrixNormalized = this.MatrixNormalized != null ?
      this.MatrixNormalized :
      MatrixNormalized(getN(), MatrixPaired);
  double[] Wi = this.Wi != null ?
      this.Wi :
      Wi(getN(), MatrixNormalized);
  double Nmax = nMax(getN(), MatrixPaired, Wi);
  if(cache)
  {
    this.MatrixNormalized = MatrixNormalized;
    this.Wi = Wi;
    this.Nmax = Nmax;
  }
  return Nmax;
}

/**
 * Коэфициент согласованности матрицы
 * @return числовое значение
 */
public double getRI()
{
  return RI(getN());
}

/**
 * Стохастический коэффициент согласованности матрицы
 * @return числовое значение
 */
public double getCI()
{
  return CI(getN(), getNmax());
```

```
}

/**
 * Уровень согласованности матрицы
 * @return числовое значение
 */
public double getCR()
{
  return CR(getCI(), getRI());
}

public String[] getTitle()
{
  return Title;
}
public boolean setTitle(String[] Title)
{
  if(this.Title.length != Title.length)
    return false;
  this.Title = Title;
  return true;
}
public boolean setTitle(int index, String Title)
{
  if(index < 0 || index >= this.Title.length)
    return false;
  this.Title[index] = Title;
  return true;
}
static double[][] MatrixNormalized(int n, double[][] MatrixPaired)
{
  double[][] MatrixNormalized = new double[n][n];
  double[] MatrixPairedDivider = new double[n];
  for(int i = 0; i < n; i++)
  {
    for(int j = 0; j < n; j++)
    {
      MatrixPairedDivider[j] += MatrixPaired[i][j];
    }
  }
  for(int i = 0; i < n; i++)
  {
```

```
      for(int j = 0; j < n; j++)
      {
        MatrixNormalized[i][j] = MatrixPaired[i][j] / MatrixPairedDivider[j];
      }
    }
    return MatrixNormalized;
  }
  static double[] Wi(int n, double[][] MatrixNormalized)
  {
    double[] Wi = new double[n];
    for(int i = 0; i < n; i++)
    {
      for(int j = 0; j < n; j++)
      {
        Wi[i] += MatrixNormalized[i][j];
      }
      Wi[i] /= n;
    }
    return Wi;
  }
  static double nMax(int n, double[][] MatrixPaired, double[] Wi)
  {
    double nMax = 0;
    double[] MatrixMultiply = new double[n];
    for(int i = 0; i < n; i++)
    {
      for(int j = 0; j < n; j++)
      {
        MatrixMultiply[i] += MatrixPaired[i][j] * Wi[j];
      }
    }
    for(int i = 0; i < n; i++)
    {
      nMax += MatrixMultiply[i];
    }
    return nMax;
  }
  static double CI(double n, double nMax)
  {
    return (nMax - n) / (n - 1);
  }
  static double RI(double n)
```

```
    {
      return 1.98 * (n - 2) / n;
    }
    static double CR(double CI, double RI)
    {
      return CI / RI;
    }
}
public class MatrixPairedAlternatives
{
    private MatrixPaired[] Alternatives;
    private MatrixPaired Criteria;

    /**
     * Конструткор
     * @param CountAlternatives количество альтернатив
     * @param CountCriteries количество критериев
     */
    public MatrixPairedAlternatives(int CountAlternatives, int CountCriteries)
    {
      Criteria = new MatrixPaired(CountCriteries);
      Alternatives = new MatrixPaired[CountCriteries];
      for(int i = 0; i < Alternatives.length; i++)
      {
        Alternatives[i] = new MatrixPaired(CountAlternatives);
      }
    }

    /**
     * Количество матриц альтернатив, которое ровняется количеству критериев
     */
    public int getAlternativeLength()
    {
      return Alternatives.length;
    }

    /**
     * Количество альтернатив
     */
    public int getAlternativeCount()
    {
      return Alternatives[0].getN();
```

```
}

/**
 * Матрица альтернатив по определенному критерию
 * @param index индекс критерия
 * @return ссылку на матрицу
 */
public MatrixPaired getAlternative(int index)
{
  if(index < 0 || index >= Alternatives.length)
    return null;
  return Alternatives[index];
}

/**
 * Матрица критериев
 * @return ссылку на матрицу
 */
public MatrixPaired getCriteria()
{
  return Criteria;
}

/**
 * Устанавливает значения в матрице критериев
 * @see MatrixPaired#setMatrixPaired(int, int, double)
 */
public boolean setCriteria(int x, int y, double number)
{
  return Criteria.setMatrixPaired(x, y, number);
}

/**
 * Устанавливает значение в матрице альтернатив
 * @param index индекс критерия
 * @see MatrixPaired#setMatrixPaired(int, int, double)
 */
public boolean setAlternatives(int index, int x, int y, double number)
{
  MatrixPaired a = getAlternative(index);
  if(a == null)
    return false;
```

```
    return a.setMatrixPaired(x, y, number);
  }

  /**
   * Весовые коэфециенты для каждой альтернативы, по которым делается выбор
   * @return массив
   */
  public double[] W()
  {
    double[][] Wa = new double[Alternatives.length][];
    for(int i = 0; i < Wa.length; i++) Wa[i] = Alternatives[i].getWi();

    double[] Wc = Criteria.getWi();
    double[] W = new double[Alternatives[0].getN()];
    for(int i = 0; i < Wa.length; i++)
      for(int j = 0; j < Wa[i].length; j++)
        W[j] += Wa[i][j] * Wc[i];
    return W;
  }

  /**
   * Являются ли все матрицы приемлемо согласованны
   */
  public boolean isConsistent()
  {
    if(!Criteria.isConsistent())
      return false;
    for(int i = 0; i < Alternatives.length; i++)
      if(!Alternatives[i].isConsistent())
        return false;
    return true;
  }

  /**
   * Являются ли все матрицы идеально согласованны
   * @return
   */
  public boolean isPerfectlyConsistent()
  {
    if(!Criteria.isPerfectlyConsistent())
      return false;
    for(int i = 0; i < Alternatives.length; i++)
```

```
        if(!Alternatives[i].isPerfectlyConsistent())
          return false;
      return true;
    }

    /**
     * Заполнены ли все матрицы
     * @return
     */
    public boolean isFull()
    {
      if(!Criteria.isFullMatrixPaired())
        return false;
      for(int i = 0; i < Alternatives.length; i++)
        if(!Alternatives[i].isFullMatrixPaired())
          return false;
      return true;
    }
}
```