МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
УО «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра экономической информатики

**ОТЧЁТ**

по дисциплине: **Системы поддержки принятия решений и аудит информационных технологий**на тему: **Создание системы поддержки принятия решений на основе метода анализа иерархий**

Студенты  
ФМ, 3-й курс, 15ДКИ-1 А.И. Дмитриев   
 М.С. Карпук

Проверил  
ассистент Г.Н. Подгорная

МИНСК 2017

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc501450013)

[1 Основные характеристики систем поддержки принятия решений (СППР) 4](#_Toc501450014)

[1.1 Понятие и функции СППР 4](#_Toc501450015)

[1.2 Классификация СППР 5](#_Toc501450016)

[2 Алгоритм работы метода анализа иерархий Т. Саати 7](#_Toc501450017)

[3 Описание процесса работы созданной СППР 9](#_Toc501450018)

[Заключение 15](#_Toc501450019)

[Список использованных источников 16](#_Toc501450020)

[Приложение A 17](#_Toc501450021)

# ВВЕДЕНИЕ

На протяжении всей жизни человек склонен делать выбор между определёнными альтернативами, стараясь поступить рационально. Из этих альтернатив он выбирает наилучший вариант действий в соответствии с некоторыми критериями – это и есть процесс принятия решения.

Их эффективность существенно зависит от степени использования постоянно возрастающего объёма информации, имеющейся в распоряжении принимающего решение лица. Часто привлекаются знания специалистов того или иного вида деятельности и определённой области знаний, с целью получения экспертной оценки.

Развитие информационных технологий и фундаментальность математических методов в своей совокупности создали средство, способное удовлетворить современные требования к качеству принимаемых решений, которое воплощено в системах поддержки принятия решений (СППР).

Данные системы используются руководителями организаций для решения стратегических задач нерегулируемого характера, например, формирование направления развития предприятия, а также их стратегических целей, поиск источников финансирования и иных задач.

Целью данной работы будет являться создание собственной СППР на основе метода анализа иерархий Т. Саати на базе языка программирования Java.

# 1 Основные характеристики систем поддержки принятия решений (СППР)

# 1.1 Понятие и функции СППР

Система поддержки принятия решений (СППР) (англ. Decision Support System, DSS) — компьютерная автоматизированная система, целью которой является помощь людям, принимающим решение в сложных условиях для полного и объективного анализа предметной деятельности [1].

Данная система позволяет значительно облегчить работу руководителям организаций, а также повысить её эффективность. С их помощью также оперативно решаются поставленные задачи и возникающие проблемы в бизнесе.

Система поддержки принятия решений применяется с целью поддержки многокритериальных решений в сложной информационной среде. Стоит заметить, что под многокритериальностью понимается тот факт, что результаты принимаемых решений оцениваются не по отдельности, а по совокупности многих показателей, которые рассматриваются одноременно.

СППР возникли в результате слияния управленческих информационных систем и систем управления базами данных (СУБД) [2].

Основными функциями СППР являются:

* Оказание помощи лицу, принимающему решение (ЛПР) при анализе исходной информации (оценке сложившейся обстановки и ограничений, накладываемых внешней средой).
* Выявление и ранжирование приоритетов, учет неопределенности в оценках ЛПР и формирование его предпочтений.
* Генерация возможных решений (формирование списка альтернатив).
* Оценка возможных альтернатив, исходя из предпочтений ЛПР, и ограничение, накладываемое внешней средой.
* Анализ возможных последствий принимаемых решений.
* Выбор лучшего, с точки зрения ЛПР, возможного варианта [3].

В той или иной степени, системы поддержки принятия решений присутствуют в любой информационно-управляющей системе. По мере развития предприятия, упорядочения структуры организации и налаживания межкорпоративных связей, проблема разработки и внедрения систем поддержки принятия решений становится особенно актуальной.

СППР решают две основные задачи:

* выбор наилучшего решения из множества возможных (оптимизация);
* упорядочение возможных решений по предпочтительности (ранжирование).

Выбор определённой совокупности критериев – главное правило задач. На их основе будут сопоставляться и оцениваться возможные альтернативы.

СППР обладает следующими четырьмя основными характеристиками:

* СППР использует и данные, и модели;
* СППР предназначены для помощи менеджерам в принятии решений для слабоструктурированных и неструктурированных задач;
* СППР поддерживают, а не заменяют выработку решений менеджерами;
* Основная цель СППР – улучшение эффективности решений.

# 1.2 Классификация СППР

По взаимодействию с пользователем выделяют три вида:

1. пассивные, помогают в процессе принятия решений, но не могут выдвинуть конкретного предложения;
2. активные, непосредственно участвуют в разработке правильного решения;
3. кооперативные, предполагают взаимодействие СППР с пользователем.

Выдвинутое системой предложение пользователь может доработать,

усовершенствовать, а затем отправить обратно в систему для проверки. После этого предложение вновь представляется пользователю, и так до тех пор, пока он не одобрит решение [3].

По способу поддержки различают:

1. модельно-ориентированные СППР - используют в работе доступ к статистическим, финансовым или иным моделям;
2. СППР, основанные на коммуникациях - поддерживают работу двух и более пользователей, занимающихся общей задачей;
3. СППР, ориентированные на данные - использующие в работе не только внутренние, но и внешние данные;
4. СППР, ориентированные на документы - манипулируют неструктурированной информацией, заключенной в различных электронных форматах;
5. СППР, ориентированные на знания – предоставляют специализированные решения проблем, основанные на фактах [2].

По сфере использования выделяют:

1. Общесистемные - работают с большими системами хранения данных и применяются многими пользователями.
2. Настольные - являются небольшими системами и подходят для управления с персонального компьютера одного пользователя [3].

Существующие СППР ограничиваются оценкой текущего уровня эффективности и выдачей рекомендуемых значений входных и выходных показателей системы. Также они не позволяют, проведя исследование эффективности всех подсистем, интегрировать полученные оценки в единую оценку (или группу оценок) эффективности сложной системы.

При работе сложного оборудования, функционирующего в условиях неопределенности и неполноты информации, оператор не имеет возможности заранее предвидеть ситуацию о необходимости каких-либо действий и выполняет их уже после возникновения проблемной ситуации.

Структура СППР включает в себя четыре основных компонента:

1. Информационные хранилища данных (Data Warehouse);
2. многомерная база данных, средства анализа и оперативная аналитическая обработка (OLAP);
3. средства интеллектуального анализа данных (Data Mining);
4. многокритериальный анализ (МКА, Multi Criteria Analysis) [2,4].

# 2 Алгоритм работы метода анализа иерархий Т. Саати

В начале 1970 года американский математик Томас Саати разработал процедуру поддержки принятия решений, которую назвал «Analityc hierarchy process» (AHP).  Авторы русского издания перевели это название как «Метод анализа иерархий». Этот метод относится к классу критериальных и занимает особое место, благодаря тому, что он получил исключительно широкое распространение и активно применяется по сей день. Использование данного метода предполагает представление экспертами исследуемой пролемы в виде иерархии критериев и подчинённых им альтернатив. Анализ проблемы основан на экспертном моделировании знаний в форме оценок важности одной альтернативы по отношению к другой.

В данной работе студентами будет использован механизм применения метода анализа иерархий Саати (МАИС) программного модуля «Expert Choice».

Итак, допустим, что у нас имеется проблема с несколькими альтернативами её решения. Каждая альтернатива может быть оценена по ряду определённых критериев. В свою очередь ряд критериев также имеет составные части – подкритерии.

Уровень важности критериев, подкритериев, заполнение списка альтернатив осуществляется непосредственно пользователем СППР. Расчёт итоговых весов альтернатив производится автоматически.

Принцип работы МАИС заключается в следующих этапах:

1. формируется иерархия критериев и подкритериев для оценки альтернатив;
2. внутри каждого уровня иерархии производится сравнительный анализ заданных критериев и подкритериев. Пользователем в матрице выставляются оценки важности, причём главная диагональ этой матрицы будет автоматически заполнена единицами, а остальные оценки, находящиеся на одной побочной диагонали, будут обратнопропроциональны друг друту (одно значение вводит пользователель, второе рассчитывается и вставляется в матрицу автоматически):

**Таблица 1 – Визуализация алгоритма работы матрицы оценок**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии | K1 | K2 | … | Kj |
| K1 | 1 | X12 | … | X1j |
| K2 | 1/X12 | 1 | … | X2j |
| … | … | … | 1 | … |
| Ki | 1/X1j | 1/X2j | … | 1 |

1. по каждому из предварительно заданных пользователем критериев или подкритериев на основе аналогичного вида матрицы происходит сравнение альтернатив;
2. осуществляется нормализация выставленных оценок по определённым создателем СППР алгоритмам:
3. находится среднее геометрическое полученных оценок определённой строки путём нахождения произведения элементов и вознесения под корень степени, обратнопропорциональной количеству критериев (подкритериев);
4. производится расчёт суммы средних геометрических по всем строкам матрицы;
5. путём деления среднего геометрического i-й строки на сумму средних геометрических всей матрицы, рассчитывается веса критериев;
6. аналогичным образом составляется матрица для подкритериев системы с итогом в виде получения весов подкритериев;
7. путём перемножения элементов матриц, состоящих из предварительно полученных весов критериев и подкритериев, получаем промежуточные веса альтернатив;
8. конечным этапом работы с СППР является построение итогового вектора, который будет находиться путём перемножения матриц, состоящих из промежуточных весов альтернатив и весов критериев, полученных в первой матрице;
9. Альтернативы выводятся на экран вместе со значениями их глобальных приоритетов. Наилучшей будет признаваться та альтернатива, которая имеет наибольший по значению коэффициент глобального приоритета.

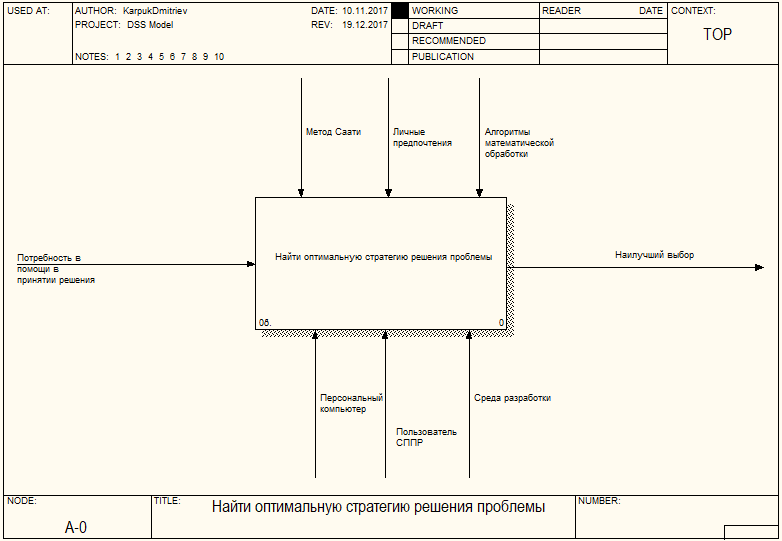
СППР была реализована с помощью языка программирования Java с использованием Intellij IDEA как среды разработки.

# 3 Описание процесса работы созданной СППР

Перед непосредственным описанием технической реализации, необходимо отразить процесс работы СППР. Для отражения процесса работы СППР была выбрана программа BPwin – мощный инструмент моделирования, разработанный фирмой Computer Associates Technologies, который используется для анализа, документирования и реорганизации сложных бизнес-процессов на основе стандарта IDEF0.

На рисунке 10 можно увидеть диаграмму процесса выбора оптимальной стратегии, на которой процесс представлен одним функциональным блоком, который описывает общее название процесса.

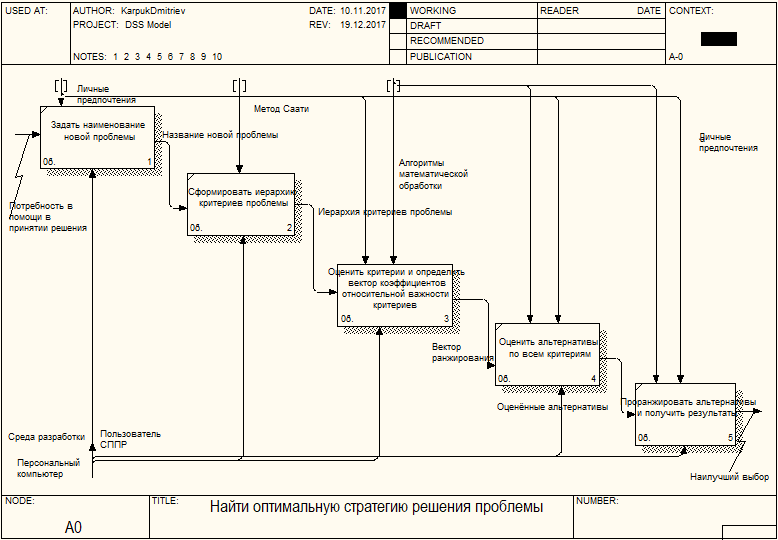
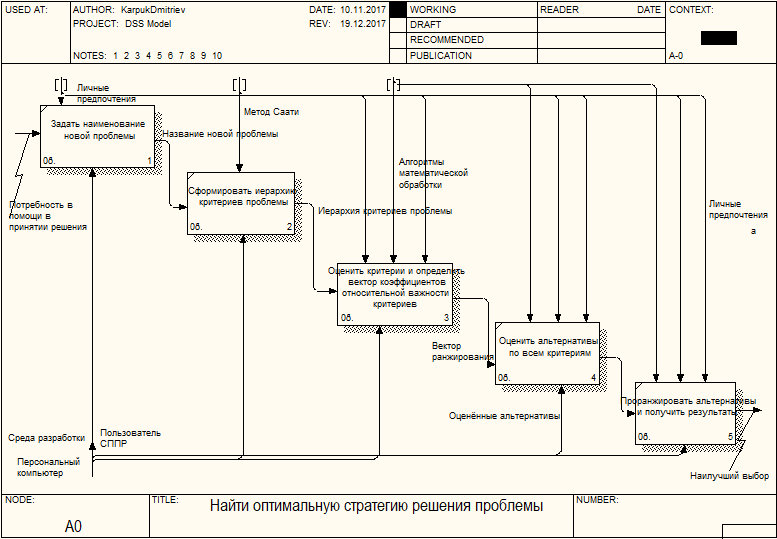
Уровень детализации А1 схемы работы СППР представлена на рисунке 1.



**Рисунок 1 – IDEF0-диаграмма процесса нахождения оптимиальной  
 стратегии решения проблемы**

Основной задачей системы является нахождение оптимальной стратегии решения проблемы с помощью СППР. Входящими данные будут являться алгоритмы математической обработки, метод Саати и личные предпочтения пользователя. Механизмы данной модели – персональный компьютер, пользователь СППР и среда разработки. Выходные данные – наилучший выбор альтернативы вместе с ранжированием всех альтернатив.

На рисунке 2 представлен следующий уровень иерархии, где выполнятся декомпозиция процесса нахождения оптимальной стратегии решения проблемы с помощью СППР.

****

**Рисунок 2 – Декомпозиция первого уровня IDEF-диаграмм**

Опишем каждый блок декомпозиции. На выходе мы получаем проблему, задаём ей наименование, а далее выбираем критерии, подкритерии и альтернативы.

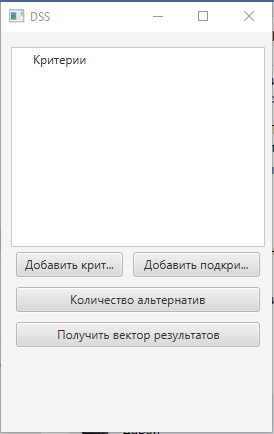
Во втором блоке определяем критерии и их подкритерии. Эти данные вводятся пользователем самой СППР с учётом его личных предпочтений.

Третий блок отвечает за оценку критериев и составление вектора коэффициентов относительной важности критериев и действует в соответствии со всеми тремя входящими данными модели.

Оценка альтернатив по всем вышеуказанным критериям осуществляется в четвёртом блоке, основываясь на векторе ранжирования критериев и подкритериев

Заключительной стадией модели станет конечное ранжирование альтернатив и получение наилучшего выбора системы. Здесь вычисляются коэффициенты глобальных приоритетов альтернатив на основе предварительно рассчитанных приоритетов критериев, подкритериев и альтернатив. Далее находится вариант с наивысшей оценкой, который и будет подходящим вариантом. На основании полученных результатов создаётся отчёт с рассчитанными коэффициентами приоритетов и указанием наилучшей альтернативы.

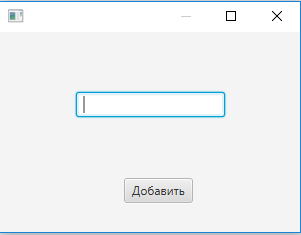
С помощью языка программирования Java была осуществлена реализация данной модели в виде приложения, которое принимает введённые пользователем данные и самостоятельно осуществляет математические расчёты согласно МАИС.



**Рисунок 3 - Стартовое окно приложения**

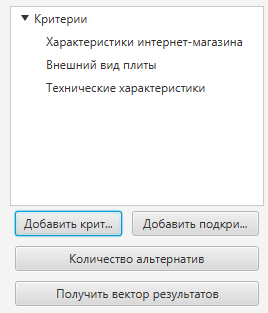
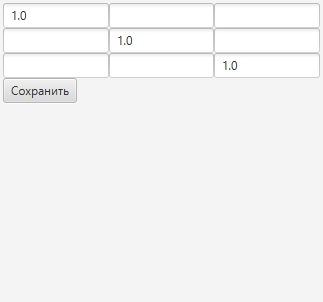
Приведём пример работы с приложение на конкретной проблеме: покупка газовой плиты в интернет-магазине.

Для начала пользователем вводятся критерии, путём нажатия на кнопку «Добавить критерии». Пользователь вводит в окно, изображённое на рисунке 4, определённый критерий.



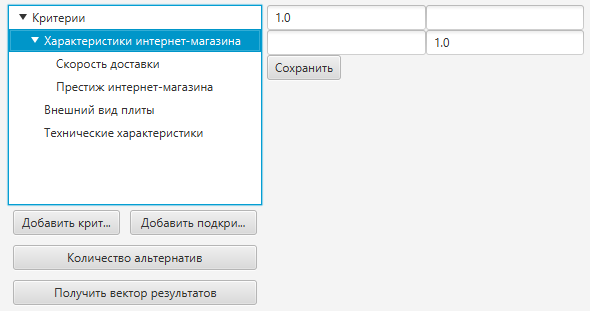
**Рисунок 4 – Окно ввода критериев**

По окончанию ввода критериев имеем ситуацию, аналогичную той, что изображена на рисунке 5. Помимо введённых критериев, пользователю отображается матрица с количеством столбцов и строк, равному количеству критериев, а также автоматически заполняется единицами главная диагональ матрицы.

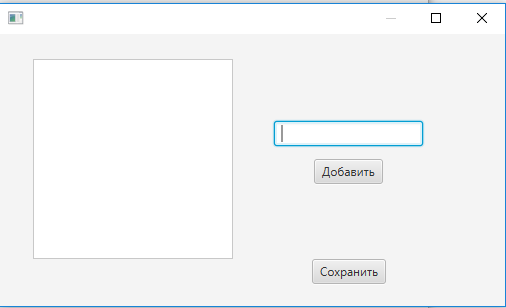
**Рисунок 5 – Введённые критерии и матрица оценки критериев**

Выделяя определённый критерий и нажимая кнопку «Добавить подкритерии» приложение получает подкритерии для каждого критерия, а также матрицу для их оценки.



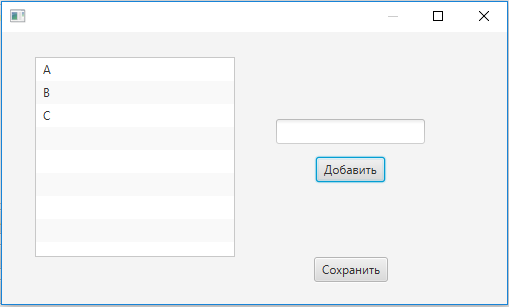
**Рисунок 6 – Ввод подкритериев и матрица оценки подкритериев**

Следующим этапом является ввод альтернатив решения проблемы путём нажатия на кнопку «Количество альтернатив» в главном окне программы.



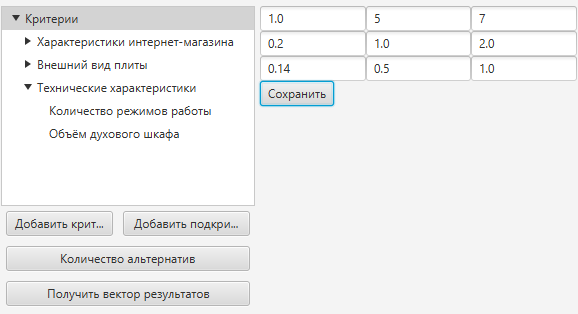
**Рисунок 7 – Окно ввода альтернатив**

По итогу ввода альтернатив мы получаем ситуацию, аналогичную с рисунком 8. При нажатии кнопки «Сохранить» данные записываются в память приложения и происходит возврат к главному окну приложения.



**Рисунок 8 – Введённые альтернативы**

Далее, согласно личным предпочтениям пользователя, в раздел критериев, подкритериев и альтернатив вводятся оценки. Ввод оценок для критериев нашей проблемы изображён на рисунке 9.



**Рисунок 9 – Ввод оценок для критериев**

Обязательным образом пользователю необходимо нажимать кнопку «Сохранить» для записи данных в память приложения. Аналогичным образом осуществляется ввод оценок и для подкритериев, и для альтернатив.

После последовательного осуществления всех вышеперечисленных действий, на главном окне приложения пользователь должен нажать кнопку «Получить вектор результатов» для представления глобальных оценок альтернатив. Пример для нашего случая представлен на рисунке 10.

****

**Рисунок 10 – Вектор результатов**

Альтернативы выводятся на экран в случайном порядке и с соответствующими им глобальными оценками. Таким образом СППР определила, что наиболее подходящей альтернативой, согласно указанным критериям и подкритериям является альтернатива С.

# ЗАКЛЮЧЕНИЕ

Таким образом, созданная СППР на базе языка программирования Java, использующая модифицированный метод анализа иерархий Саати (МАИС), является простым и удобным средством, которое является подспорьем для постановки проблемы, построения дерева критериев, выделить характеризующие их факторы, задать их значимость, оценить альтернативы по каждому из факторов и вычислить коэффициенты глобальных приоритетов альтернатив.

Созданная СППР, как и любая другая система, хоть и дает представление об оптимальном решении проблемы, однако не стоит исключать человеческого фактора, ведь окончательное решение все равно принимает пользователь, поскольку он способен принимать решения не только на основе результатов данной системы, но также и на свой опыт и знания.

В качестве результатов МАИС можно назвать установление иерархии целей, факторов, критериев, альтернатив и сценариев по обсуждаемой проблеме, а также выявление приоритетов элементов каждого уровня иерархии.

Написанная СППР на базе языка программирования Java обладает дружственным и относительно несложным к понимаю пользователя интерфейсом.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Железко, Б.А. Информационно-аналитические системы поддержки принятия решений / Б.А. Железко, А.Н. Морозевич. – Минск: НИУ, 1999. – 140 с.
2. Асанович, В.Я. Математические основы теории принятия решений в экономике: Учебное пособие / В.Я. Асанович. – Минск: БГЭУ, 2012. – 158 с.
3. Мелихова О.А. Методы построения интеллектуальных систем на основе нечеткой логики: Научное издание. – О.А. Мелихова. – Таганрог: ТРТУ, 2007. – 92с.
4. Системы поддержки принятия решений [Электронный ресурс] / – Режим доступа: <http://bourabai.kz/tpoi/dss.htm> – Дата доступа: 17.12.2017.

# ПРИЛОЖЕНИЕ А

**Код созданной СППР на языке программирования Java**

**package** main.java.com.decisionSupportSystem.logic;  
  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.Map;  
  
**public class** AnalyticHierarchyProcess {  
 **private** DataModel **dataModel**;  
  
 **public** AnalyticHierarchyProcess(DataModel dataModel) {  
 **this**.**dataModel** = dataModel;  
 }  
  
 **public** HashMap<Alternative, Double> getVector() {  
 printData();  
  
 *// rates of criterias F33:F35* HashMap<Criteria, Double> criteriasRates = **new** HashMap<>();  
 ArrayList<Criteria> cl = **dataModel**.getCriterias();  
 **double**[] cr = getWeights(**dataModel**.getCriteriasRates());  
 **for**(**int** i = 0; i < cl.size(); i++) {  
 criteriasRates.put(cl.get(i), cr[i]);  
 }  
  
 System.***out***.println(**"--- Веса критериев: "**);  
 **for**(Map.Entry<Criteria, Double> entry : criteriasRates.entrySet()) {  
 System.***out***.println(entry.getKey() + **" "** + entry.getValue());  
 }  
  
 *// rates of subcriterias like E39:E40, Criteria is an actual critera* HashMap<Criteria, **double**[]> subcriteriasRates = **new** HashMap<>();  
 HashMap<Criteria, **double**[][]> sr = **dataModel**.getSubCriteriasRates();  
 **for**(Map.Entry<Criteria, **double**[][]> entry : sr.entrySet()) {  
 subcriteriasRates.put(entry.getKey(), getWeights(entry.getValue()));  
 }  
  
 System.***out***.println(**"--- Веса подкритериев для каждого критерия: "**);  
 **for**(Map.Entry<Criteria, **double**[]> entry : subcriteriasRates.entrySet()) {  
 System.***out***.println(**"Критерий: "** + entry.getKey());  
 **for**(**int** i = 0; i < entry.getValue().**length**; i++) {  
 System.***out***.println(**dataModel**.getSubCriterias().get(entry.getKey()).get(i) + **" "** + entry.getValue()[i]);  
  
 }  
 }  
  
 *// rates of alternatives like F44:F46, Criteria = subcritera* HashMap<Criteria, **double**[]> alternativesRates = **new** HashMap<>();  
 HashMap<Criteria, **double**[][]> ar = **dataModel**.getAlternativesRates();  
 **for**(Map.Entry<Criteria, **double**[][]> entry : ar.entrySet()) {  
 alternativesRates.put(entry.getKey(), getWeights(entry.getValue()));  
 }  
  
 System.***out***.println(**"--- Веса альтернатив для каждого подкритерия: "**);  
 **for**(Map.Entry<Criteria, **double**[]> entry : alternativesRates.entrySet()) {  
 System.***out***.println(**"Подкритерий: "** + entry.getKey());  
 **for**(**int** i = 0; i < entry.getValue().**length**; i++) {  
 System.***out***.println(**dataModel**.getAlternatives().get(i) + **" "** + entry.getValue()[i]);  
 }  
 }  
  
 System.***out***.println(**"---- Веса альтернатив \* веса подкритериев ----"**);  
 *// alternatives rates \* subcriterias weights A71:F84* HashMap<Criteria, **double**[]> subcriteriasAlternativesVector = **new** HashMap<>();  
 System.***out***.println(**"--- Свертываем альтернативы и подкритерии: "**);  
 **for**(Criteria c : **dataModel**.getCriterias()) {  
 **int** columnsAmount = 0;  
 **for**(Map.Entry<Criteria, **double**[]> entry : alternativesRates.entrySet()) {  
 **if**(entry.getKey().getParent().equals(c)) {  
 columnsAmount++;  
 }  
 }  
  
 **int** rowsAmount = 0;  
 **for**(Map.Entry<Criteria, **double**[]> entry : alternativesRates.entrySet()) {  
 **if**(entry.getKey().getParent().equals(c)) {  
 rowsAmount = entry.getValue().**length**;  
 }  
 }  
 *// A72:B74* System.***out***.println(**"Критерий: "** + c + **" Таблица на "** + columnsAmount + **" колонок и "** + rowsAmount + **" строк."**);  
 **double** alternativesVector[][] = **new double**[rowsAmount][columnsAmount];  
  
 **int** x = 0;  
  
 ArrayList<Criteria> orderOfSubcriterias = **dataModel**.getSubCriterias().get(c);  
 **for**(Criteria crit : orderOfSubcriterias) {  
 **for**(Map.Entry<Criteria, **double**[]> entry : alternativesRates.entrySet()) {  
 **if**(entry.getKey().equals(crit)) {  
 **for**(**int** i = 0; i < rowsAmount; i++) {  
 alternativesVector[i][x] = entry.getValue()[i];  
 }  
 x++;  
 }  
 }  
 }  
  
 **double** matrixSubcriteria[] = subcriteriasRates.get(c);  
  
  
  
 System.***out***.println(**"Альтернативы: "**);  
 **for**(**int** i = 0; i < alternativesVector.**length**; i++) {  
 **for**(**int** j = 0; j < alternativesVector[i].**length**; j++) {  
 System.***out***.print(alternativesVector[i][j] + **" "**);  
 }  
 System.***out***.println();  
 }  
  
 System.***out***.println(**"Подкритерии"**);  
 **for**(**int** i = 0; i < matrixSubcriteria.**length**; i++) {  
 System.***out***.println(matrixSubcriteria[i]);  
 }  
  
 **double**[] resultVector = multiplyMatrices(alternativesVector, matrixSubcriteria);  
  
 **for**(**int** i = 0; i < resultVector.**length**; i++) {  
 System.***out***.println(resultVector[i]);  
 }  
 subcriteriasAlternativesVector.put(c, resultVector);  
 }  
  
 **for**(Map.Entry<Criteria, **double**[]> entry : subcriteriasAlternativesVector.entrySet()) {  
 System.***out***.println(entry.getKey());  
 **for**(**int** i = 0; i < entry.getValue().**length**; i++) {  
 System.***out***.println(entry.getValue()[i] + **" "**);  
 }  
 }  
  
 **int** noc = subcriteriasAlternativesVector.size();  
 **int** nor = 0;  
 **for**(Map.Entry<Criteria, **double**[]> entry : subcriteriasAlternativesVector.entrySet()) {  
 nor = entry.getValue().**length**;  
 }  
  
 **double** subcriteriasMatrix[][] = **new double**[nor][noc];  
 **int** col = 0;  
 **for**(**int** i = **dataModel**.getCriterias().size() - 1; i >= 0; i--) {  
 **for**(Map.Entry<Criteria, **double**[]> entry : subcriteriasAlternativesVector.entrySet()) {  
 **if**(entry.getKey().equals(**dataModel**.getCriterias().get(i))) {  
 **for**(**int** x = 0; x < entry.getValue().**length**; x++) {  
 subcriteriasMatrix[x][col] = entry.getValue()[x];  
 }  
 col++;  
 }  
 }  
 }  
  
 **double**[] criteriasWeights = **new double**[**dataModel**.getCriterias().size()];  
 **for**(**int** i = 0; i < **dataModel**.getCriterias().size(); i++) {  
 **for**(Map.Entry<Criteria, Double> crit : criteriasRates.entrySet()) {  
 **if**(crit.getKey().equals(**dataModel**.getCriterias().get(i))) {  
 criteriasWeights[i] = crit.getValue();  
 }  
 }  
 }  
  
 **double** finalVector[] = multiplyMatrices(subcriteriasMatrix, criteriasWeights);  
 **for**(**int** i = 0; i < finalVector.**length**; i++) {  
 System.***out***.println(**dataModel**.getAlternatives().get(i) + **" "** + finalVector[i]);  
 }  
  
 System.***out***.println(**"---- Final matrix ----"**);  
 **for**(**int** i = 0; i < subcriteriasMatrix.**length**; i++) {  
 **for**(**int** j = 0; j < subcriteriasMatrix[i].**length**; j++) {  
 System.***out***.print(subcriteriasMatrix[i][j] + **" "**);  
 }  
 System.***out***.println();  
 }  
  
 System.***out***.println(**"---- Criterias weights vector ----"**);  
 **for**(**int** i = 0; i < criteriasWeights.**length**; i++) {  
 System.***out***.print(criteriasWeights[i] + **" "**);  
 }  
  
 HashMap<Alternative, Double> result = **new** HashMap<>();  
 **for**(**int** i = 0; i < **dataModel**.getAlternatives().size(); i++) {  
 result.put(**dataModel**.getAlternatives().get(i), finalVector[i]);  
 }  
 **return** result;  
 }  
  
 **private double**[] getWeights(**double**[][] matrix) {  
 **double**[] avgGeom = avgGeometric(matrix);  
 **double** sumCol = sumColumn(avgGeom);  
 **return** weightsVector(avgGeom, sumCol);  
 }  
  
 */\*\*  
 \* Finds geometric mean for every row of the matrix.  
 \* \*/* **public double**[] avgGeometric(**double**[][] matrix) {  
 **double**[] multline = **new double**[matrix.**length**];  
 **double**[] result = **new double**[matrix.**length**];  
 **double** mult = 1;  
 **for** (**int** i = 0; i < matrix.**length**; i++) {  
 **for** (**int** j = 0; j < matrix[i].**length**; j++) {  
 mult = mult \* matrix[i][j];  
 }  
 multline[i] = mult;  
 mult = 1;  
  
 }  
 **for** (**int** i = 0; i < matrix.**length**; i++) {  
 **double** avgGeom = Math.*pow*(multline[i], 1.0/matrix.**length**);  
 result[i] = avgGeom;  
 }  
 **return** result;  
 }  
  
 */\*\*  
 \* Finds the sum of all rows in column.  
 \* \*/* **public double** sumColumn(**double**[] column) {  
 **double** result = 0;  
 **for** (**int** i = 0; i < column.**length**; i++) {  
 result = result + column[i];  
 }  
 **return** result;  
 }  
  
 */\*\*  
 \* Finds weights for each criteria/subcriteria  
 \* (avgGeometric[i]/sumColumn)  
 \* \*/* **public double**[] weightsVector(**double**[] column, **double** sumOfColumn) {  
 **double** finalResult[] = **new double**[column.**length**];  
 **for** (**int** i = 0; i < column.**length**; i++) {  
 finalResult[i] = column[i]/sumOfColumn;  
 }  
 **return** finalResult;  
 }  
  
 */\*\*  
 \* Multiplies two matrices.  
 \*/* **public double**[] multiplyMatrices(**double**[][] matrix, **double**[] array) {  
 **double** result[] = **new double** [matrix.**length**];  
 **for** (**int** i = 0; i < matrix.**length**; i++) {  
 **double** sum = 0;  
 **for** (**int** j = 0; j < matrix[i].**length**; j++) {  
 sum = sum + matrix[i][j] \* array[j];  
 }  
 result[i] = sum;  
 }  
 **return** result;  
 }  
  
 **private void** printData() {  
 System.***out***.println(**"Alternatives: "**);  
 **for**(**int** i = 0; i < **dataModel**.getAlternatives().size(); i++) {  
 System.***out***.println(**dataModel**.getAlternatives().get(i));  
 }  
  
 System.***out***.println(**"Criterias: "**);  
 **for**(**int** i = 0; i < **dataModel**.getCriterias().size(); i++) {  
 System.***out***.println(**dataModel**.getCriterias().get(i));  
 }  
  
  
 System.***out***.println(**"Subcriterias: "**);  
 **for**(Map.Entry<Criteria, ArrayList<Criteria>> entry : **dataModel**.getSubCriterias().entrySet()) {  
 System.***out***.println(**"Parent: "** + entry.getKey() + **" --- "** + entry.getValue());  
 }  
  
 System.***out***.println(**"Оценки критериев: "**);  
 **for**(**int** i = 0; i < **dataModel**.getCriteriasRates().**length**; i++) {  
 **for**(**int** j = 0; j < **dataModel**.getCriteriasRates()[i].**length**; j++) {  
 System.***out***.print(**dataModel**.getCriteriasRates()[i][j] + **" "**);  
 }  
 System.***out***.println();  
 }  
  
 System.***out***.println(**"Оценки подкритериев: "**);  
 **for**(Map.Entry<Criteria, **double**[][]> entry : **dataModel**.getSubCriteriasRates().entrySet()) {  
 System.***out***.println(entry.getKey());  
 **for**(**int** i = 0; i < entry.getValue().**length**; i++) {  
 **for**(**int** j = 0; j < entry.getValue()[i].**length**; j++) {  
 System.***out***.print(entry.getValue()[i][j] + **" "**);  
 }  
 System.***out***.println();  
 }  
 }  
  
 System.***out***.println(**"Оценки альтернатив: "**);  
 **for**(Map.Entry<Criteria, **double**[][]> entry : **dataModel**.getAlternativesRates().entrySet()) {  
 System.***out***.println(entry.getKey());  
 **for**(**int** i = 0; i < entry.getValue().**length**; i++) {  
 **for**(**int** j = 0; j < entry.getValue()[i].**length**; j++) {  
 System.***out***.print(entry.getValue()[i][j] + **" "**);  
 }  
 System.***out***.println();  
 }  
 }  
 }  
}