Національний Технічний Університет України “КПІ”

Навчально-науковий комплекс

«Інститут прикладного системного аналізу»

ЛАБОРАТОРНА РОБОТА № 5

З дисципліни: Основи системного аналізу

ЗАВДАННЯ ВИБОРУ СТРУКТУРИ СКЛАДНОЇ СИСТЕМИ

# Виконали:

# Глузман Марк

Череда Григорій

# група КА-11

Київ 2015

**Задание**

В процессе выполнения работы необходимо:

1. Построить иерархическую структуру сложной системы (СС) любой природы с альтернативными вариантами каждого типа ФЭ на каждом иерархическом уровне;

2. Построить множество Парето создаваемой СС;

3. Предложить СС на основании приоритетных требований, предъявляемых к СС в целом.

**Реализация**

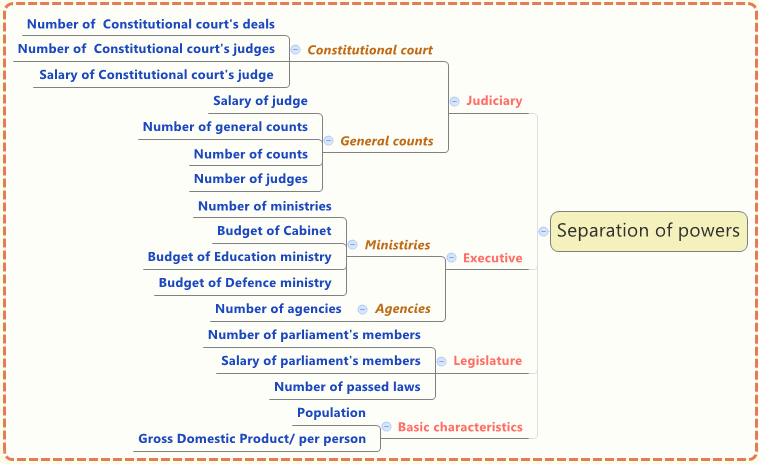
В лаболаторной работе мы взяли для исследования сложную систему «***Ветки власти в государстве***».

В процессе выполнения лаболаторной работы нашей целью было построить структуру государственной власти в демократических странах, то есть странах, где есть распределение всасти между разными независимыми ветвями: законодательной, судебной и исполнительной.

В соответствие с принципом Монтескье, эти три ветви власти являються независимыми. Более того, мы предпологаем, что также некоторые «подветки» (структурные элементы) в модели нашей структуры также будут независимыми. Например: параметры конституционого суда и судой общей инстанции не коррелируют.

Также, мы добавили в структуру к верхнему иерархическому уровню (уровню веток власти) еше один элемент, который служит для определения основных параметров страны.

Для выбраной модели мы построили следующую иерархическую структуру:

****

Таким образом, первый иерархический уровень состоит из следующих функциональных элементов: судебной власти, исполнительной власти, законодательной власти, основный характеристики государства.

Второй иерархический уровень есть у судебной и исполнительной власти. У судебной власти есть подуровни; конституционный суд и суды общей инстанции. У Исполнительной – министерства и государственные агенства (департаменты, службы, комитеты, ведомства).

Каждый фунциональный элемент на последнем уровне иерархии имеет свои параметры.

Законодательная власть: количество депутатов в парламенте (во всех палатах), зарплата депутата, количество принятых законов за год

Министерства: количество министерств, общий бюджет кабинета министров, бюджет министерства обороны, бюджет министерства образования.

Службы: количество общегосударственных служб

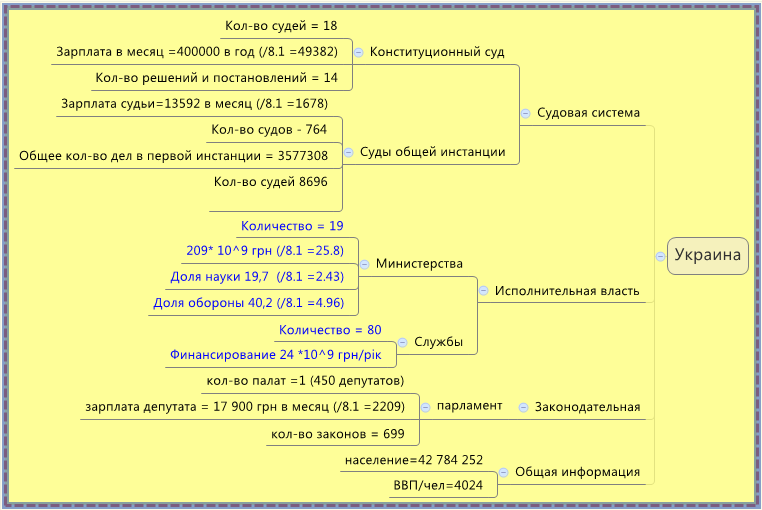
Конституционный суд: количество судей, зарплата судьи, количество принятых решения за год

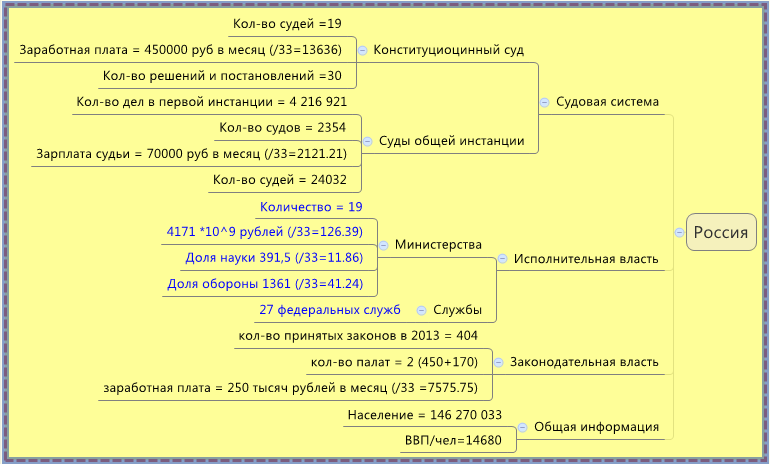
Суды общей инстанции: количество судов первой инстанции, общее количество судей в стране, зарплата судьи, количество рассмотреных дел за год по стране.

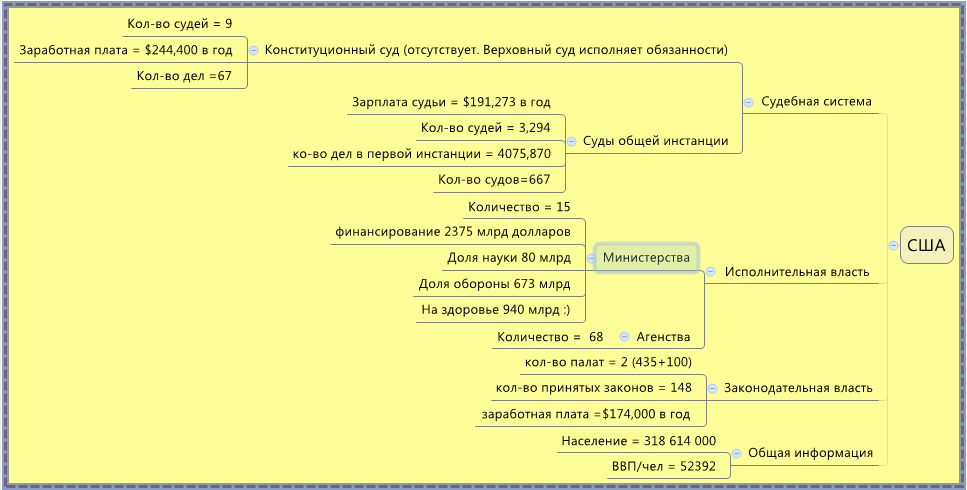
Основные характерстики государства: ВВП на человека, население.

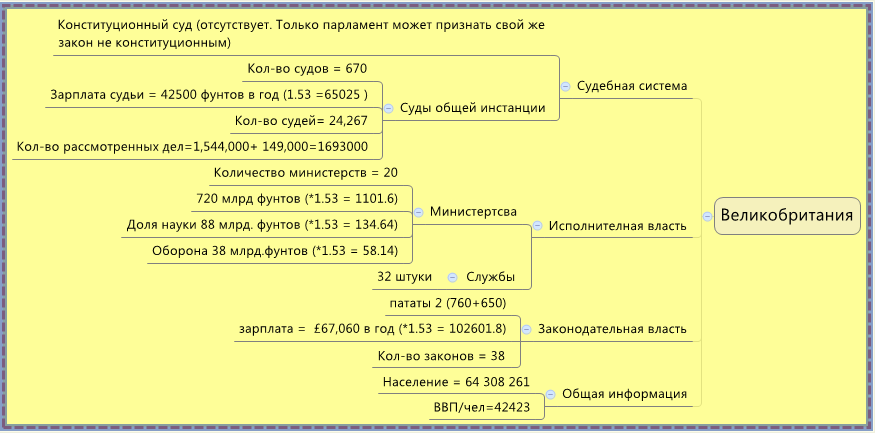
За альтернативы функциональных элементов мы взяли реальные показатели перечисленных функцианальных элементов следующих стран: США, Великобратании, Индии, России, Украины.

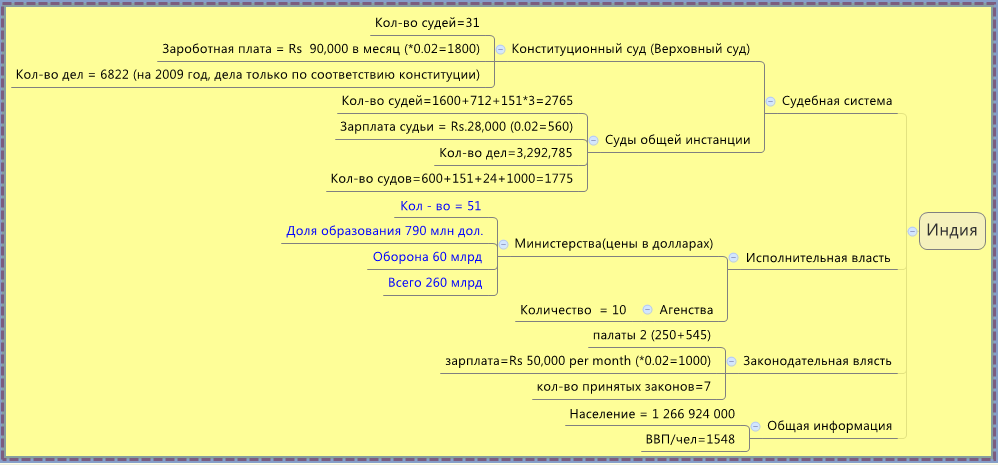
Данные приведены на 2013 год (если не указано иного).











Ссылки на источники приведены в Дополнении.

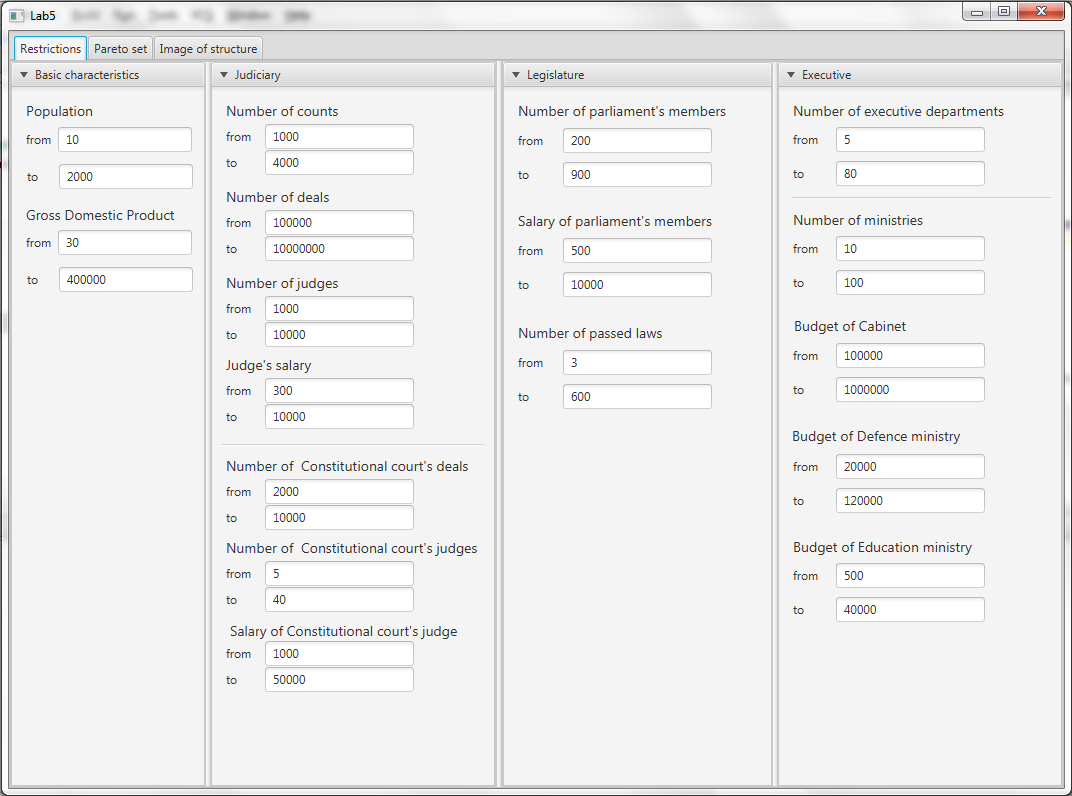
Таким образом, каждый функциональный элемент имеет 5 альтернатив – соответствующий элемент в какой-то из 5 стран.

Целью лабораторной работы являеться построение сложной структуры государственной власти на основании приоритетных требований, предъявляемых к СС в целом.

**Программная реализация**

На основе алгоритма, предложенного для реализации по нахождения множества Парето структуры системы, была организована программная реализация.

На первой вкладке пользователю предоставляеться возможность задать параметры (ограничения) на параметры функциональных элементов.

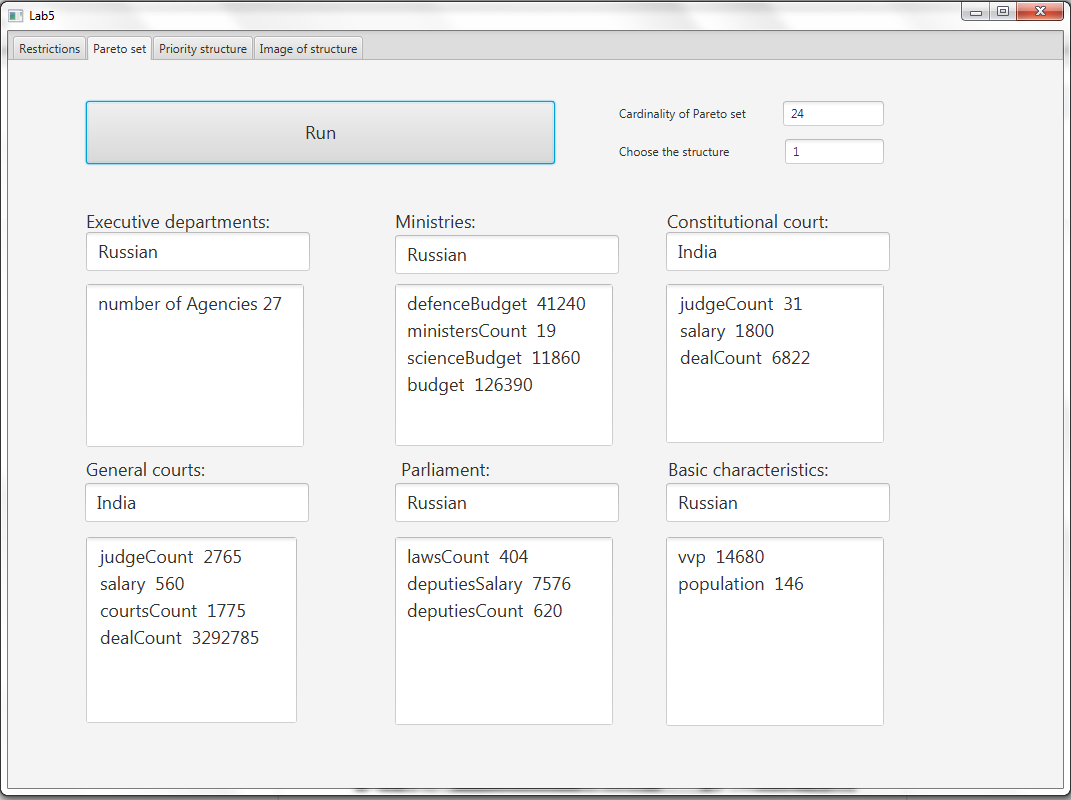


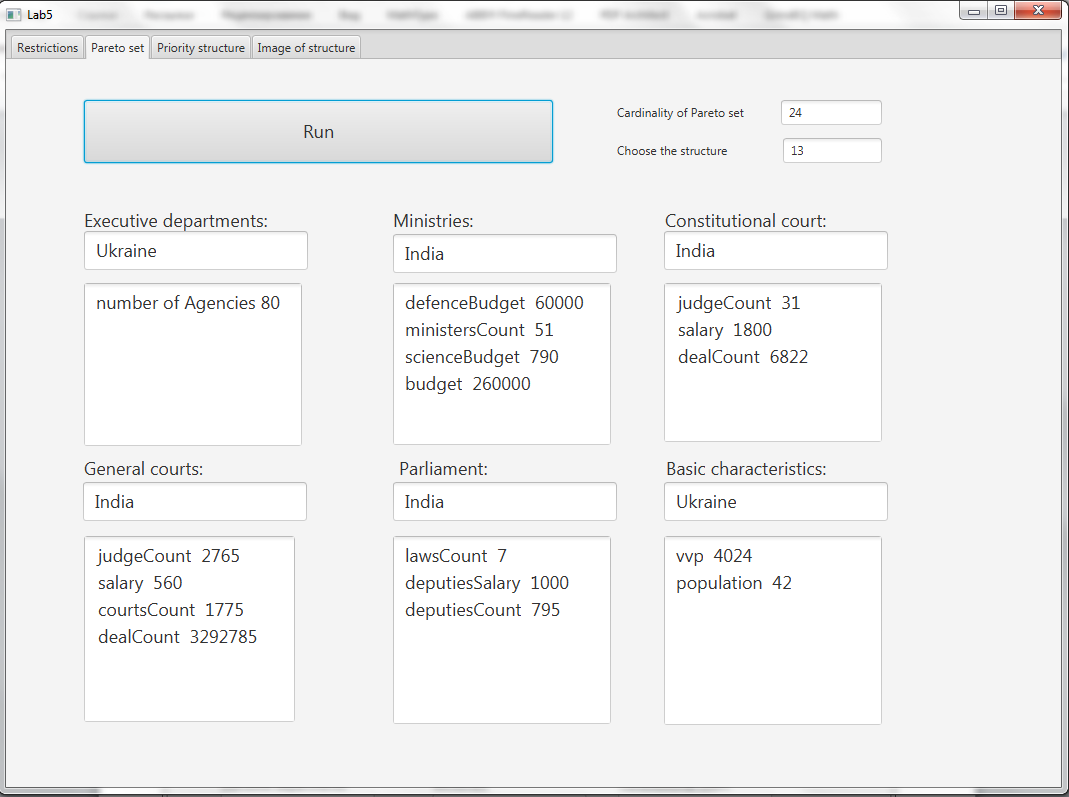
Для каждого параметра пользователь задает требуемый интервал.

На следующей вкладке происходит собственно нахождения подходящей структуры из множества Парето. Также выдается мощность этого множества (количество элементов в множестве) и задать номер и просмотреть структуру в этом множестве.

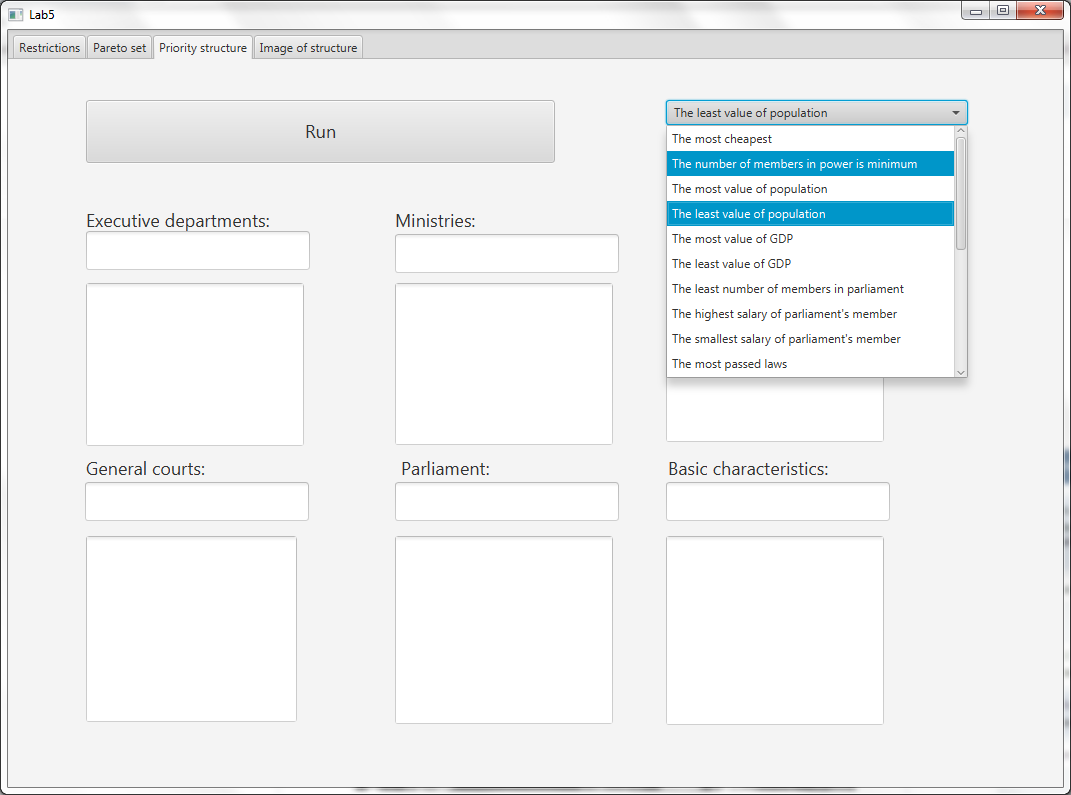
На форме отображается страна, функциональный элемент которой добходит под ограничения, и все параметры этого элемента.

При повторном нажатии на кпопку Run и выбере другого номера в окошке “Choose the structure” программа находит следующую подходящую структуру сложной системы.



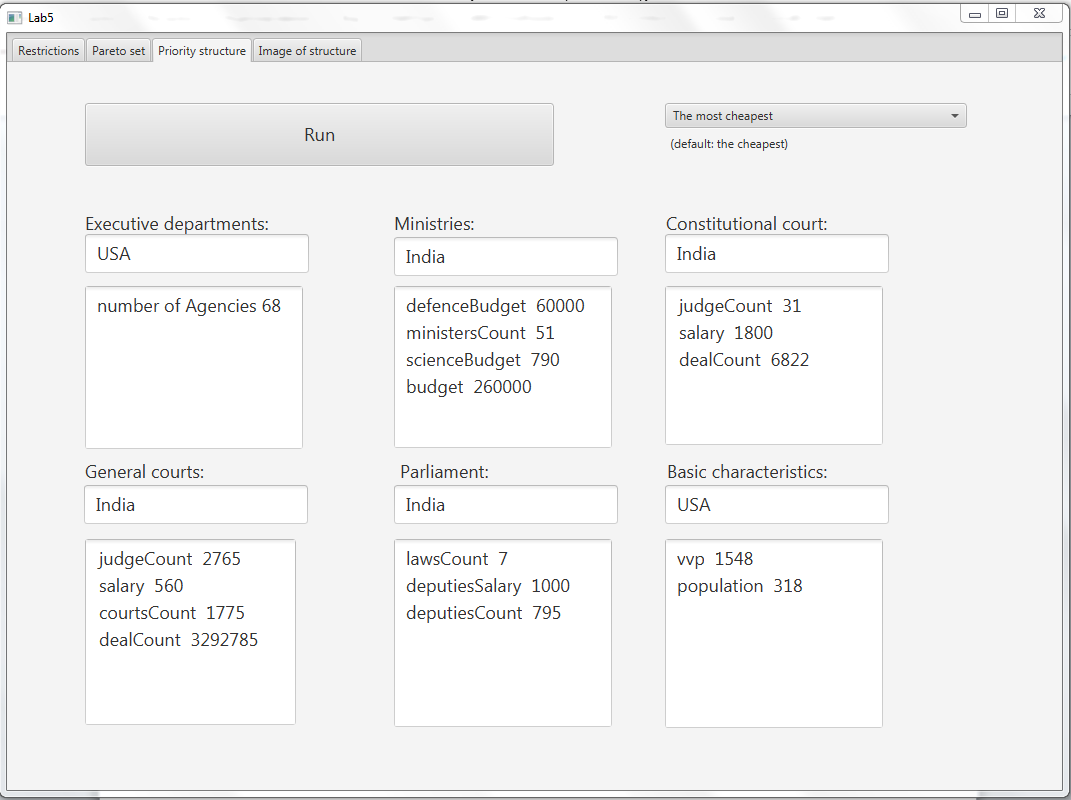


В следующей вкладке «Priority structure» мы выбираем самый важный критерий, в соответствии с которым программа подбирает единственную структуру.



После нажатия на кнопку “Run” программа подбирает нужную структуру.

Если под критерий подходят несколько структур, то программа выдает первую подходящую.



**Дополнение (источники статистической информации)**

бюджет Британии

http://en.wikipedia.org/wiki/United\_Kingdom\_budget

Великобритании службы и количество министерств

https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BF%D0%B0%D1%80%D1%82%D0%B0%D0%BC%D0%B5%D0%BD%D1%82%D1%8B\_%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%B0\_%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D0%BE%D0%B1%D1%80%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D0%B8

Украина министерства

http://biz.liga.net/ekonomika/all/stati/2914721-byudzhetnye-peregony-kto-iz-ministrov-slishkom-dorog-dlya-strany-.htm

Украина службы

https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%9C%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D0%B5%D1%80%D1%81%D1%82%D0%B2%D0%B0\_%D0%A3%D0%BA%D1%80%D0%B0%D0%B8%D0%BD%D1%8B

Индия бюджет

http://en.wikipedia.org/wiki/2013\_Union\_budget\_of\_India

Индия министерства и службы

https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B0%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE\_%D0%98%D0%BD%D0%B4%D0%B8%D0%B8

Сша бюджет

http://en.wikipedia.org/wiki/2013\_United\_States\_federal\_budget

Сша агенства

https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA\_%D1%84%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D1%85\_%D0%B0%D0%B3%D0%B5%D0%BD%D1%82%D1%81%D1%82%D0%B2\_%D0%A1%D0%A8%D0%90

Россия министерства

http://ria.ru/infografika/20121126/912241929.html

Россия службы

https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%A4%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5\_%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D1%8B\_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8

Количество рассматриваемых дел в Индии

<http://www.supremecourt.gov/orders/journal/jnl13.pdf>

Количество судов в Индии

<http://www.newindianexpress.com/nation/Despite-1000-Fast-Track-Courts-32-Millions-Cases-Still-Pending/2013/12/23/article1961278.ece>

Количество судов, судей, рассмотреных дел в России

<https://rospravosudie.com/jurisdiction-fed/date_from-2013-01-01/date_to-2013-12-31/>

Зарплата судьи в США

http://www.fjc.gov/history/home.nsf/page/js\_2.html

<http://en.wikipedia.org/wiki/Federal_judge_salaries_in_the_United_States#Supreme_Court>

Информация о судебной власти в Укране

[http://court.gov.ua/sudova\_statystyka/#](http://court.gov.ua/sudova_statystyka/)

Информация о зарплатах судей в Украине

<http://www.epravda.com.ua/rus/columns/2014/06/23/469686/view_print/>

Количество рассмотреных дел в Великобритании

<https://www.gov.uk/government/collections/court-statistics-quarterly>

<https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/217494/judicial-court-stats-2011.pdf>

Зарплата судьи в Великобритании

<https://www.gov.uk/government/publications/judicial-salaries-and-fees-2013-14>

Зарплата судьи в Индии

<http://en.wikipedia.org/wiki/Supreme_Court_of_India>

<http://www.prsindia.org/billtrack/the-high-court-and-supreme-court-judges-salaries-and-conditions-of-service-amendment-bill-2008-173/>

Данные по ВВП

<https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D1%81%D1%82%D1%80%D0%B0%D0%BD_%D0%BF%D0%BE_%D0%92%D0%92%D0%9F_(%D0%9F%D0%9F%D0%A1)_%D0%BD%D0%B0_%D0%B4%D1%83%D1%88%D1%83_%D0%BD%D0%B0%D1%81%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F>

Население

<https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D1%81%D1%82%D1%80%D0%B0%D0%BD_%D0%BF%D0%BE_%D0%BD%D0%B0%D1%81%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8E>

Количество депутатов в парламентах

<https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B3%D1%80%D0%B5%D1%81%D1%81_%D0%A1%D0%A8%D0%90>

<https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D1%85%D0%BE%D0%B2%D0%BD%D0%B0%D1%8F_%D1%80%D0%B0%D0%B4%D0%B0_%D0%A3%D0%BA%D1%80%D0%B0%D0%B8%D0%BD%D1%8B>

<https://ru.wikipedia.org/wiki/%D0%A4%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%81%D0%BE%D0%B1%D1%80%D0%B0%D0%BD%D0%B8%D0%B5_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B9%D1%81%D0%BA%D0%BE%D0%B9_%D0%A4%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8>

<https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B5%D0%BD%D1%82_%D0%98%D0%BD%D0%B4%D0%B8%D0%B8>

https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%BB%D0%B0%D0%BC%D0%B5%D0%BD%D1%82\_%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D0%BE%D0%B1%D1%80%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D0%B8

**Выводы:**

В процессе работы над лаболаторной работой был выполнен анализ и построена и проанализирована структура СС «***Структура веток власти в государстве***».

Был реализован алгоритм целенаправленного выбора иерархической структуры, в данном случае – структура власти государства.

Результаты работы имеют практическое применение т.к. все расчеты были произведены на основе реальных данных.

На последнем рисунке приведена полученная рациональная структура ветвей власти государства, при использовании метода целенаправленного выбора функциональных элементов, для построения которой потребовалось 21 попытка. В то время как при использовании метода случайного поиска для обеспечения гарантированного выбора структуры, удовлетворяющей требованиям заданного уровня качества, требуется выполнить 56 попыток.

**Література по даному питанню:**

1. Kevin J. Dooley. Начало формы

Конец формы

1. A Complex Adaptive Systems Model of Organization Change// [Nonlinear Dynamics, Psychology, and Life Sciences](http://link.springer.com/journal/11054) January 1997, Volume 1, [Issue 1](http://link.springer.com/journal/11054/1/1/page/1), pp 69-97

http://link.springer.com/article/10.1023/A:1022375910940#page-1

1. Melvin F. Shakun. Modeling and supporting task-oriented group processes: Purposeful complex adaptive systems and Evolutionary Systems Design. // [Group Decision and Negotiation](http://link.springer.com/journal/10726). September–December 1996, Volume 5, [Issue 4-6](http://link.springer.com/journal/10726/5/4/page/1), pp 305-317

http://link.springer.com/article/10.1007/BF00553905

1. [Russell L. Ackoff](http://www.google.com.ua/search?hl=uk&tbo=p&tbm=bks&q=inauthor:%22Russell+L.+Ackoff%22), [Russell Lincoln Ackoff](http://www.google.com.ua/search?hl=uk&tbo=p&tbm=bks&q=inauthor:%22Russell+Lincoln+Ackoff%22), [Fred E. Emery](http://www.google.com.ua/search?hl=uk&tbo=p&tbm=bks&q=inauthor:%22Fred+E.+Emery%22). On Purposeful Systems: An Interdisciplinary Analysis of Individual And Social Behavior As a System of Purposeful Events. // Transaction Publishers, 30 жовт. 2005 р. - 288.

<http://www.google.com.ua/books?hl=uk&lr=&id=R-RSHfnS7VcC&oi=fnd&pg=PA4&dq=Purposeful+choice+of+the+structure+of+a+complex+system&ots=1WktZfAIH_&sig=i1nsmQgJVz5v0jt5_Oq4AoWRQ0Y&redir_esc=y#v=onepage&q&f=false>

# [Michael L. Tushman](http://amr.aom.org/search?author1=Michael+L.+Tushman&sortspec=date&submit=Submit), [David A. Nadler](http://amr.aom.org/search?author1=David+A.+Nadler&sortspec=date&submit=Submit)1. Information Processing as an Integrating Concept in Organizational Design[1](http://amr.aom.org/content/3/3/613.short#fn-2). ACAD MANAGE REV July 1, 1978 vol. 3 no. 3 613-624

http://amr.aom.org/content/3/3/613.short

1. [Michael C. Jensen, FOUNDATIONS OF ORGANIZATIONAL STRATEGY, Harvard University Press, 1998](http://papers.ssrn.com/sol3/papers.cfm?abstract_id=94036##) [Accounting Review, Vol. LVIII, No. 2, pp. 319-339, April 1983](http://papers.ssrn.com/sol3/papers.cfm?abstract_id=94036##)

http://papers.ssrn.com/sol3/papers.cfm?abstract\_id=94036

Лістинг програми:

package Executive;

import AlternativeSelection.AlternativeChoosingAlgorithm;

import CountriesAndCountryDictionaries.Country;

import javafx.event.ActionEvent;

import javafx.scene.control.Button;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* Created by Марк on 21.02.2015.

\*/

public class Controller {

public TextField Text\_population\_from;

public TextField Text\_population\_to;

public Button Button\_run;

public TextField Text\_gdp\_from;

public TextField Text\_gdp\_to;

public TextField Text\_NumDealsConst\_from;

public TextField Text\_NumDealsConst\_to;

public TextField Text\_NumJudgesConst\_from;

public TextField Text\_JudgeSalaryConst\_from;

public TextField Text\_JudgeSalaryConst\_to;

public TextField Text\_NumJudges\_to;

public TextField Text\_NumJudges\_from;

public TextField Text\_JudgeSalary\_from;

public TextField Text\_JudgeSalary\_to;

public TextField Text\_NumDeals\_from;

public TextField Text\_NumDeals\_to;

public TextField Text\_NumCounts\_to;

public TextField Text\_NumCounts\_from;

public TextField Text\_NumJudgesConst\_to;

public TextField Text\_NumberDepartments\_from;

public TextField Text\_NumberDepartments\_to;

public TextField Text\_NumberMinistries\_to;

public TextField Text\_NumberMinistries\_from;

public TextField Text\_NumberPassedLaws\_to;

public TextField Text\_NumberPassedLaws\_from;

public TextField Text\_SalaryParliamentMember\_to;

public TextField Text\_SalaryParliamentMember\_from;

public TextField Text\_NumberParliamentMembers\_from;

public TextField Text\_NumberParliamentMembers\_to;

public TextField Text\_BudgetCabinet\_from;

public TextField Text\_BudgetCabinet\_to;

public TextField Text\_BudgetDefence\_to;

public TextField Text\_BudgetDefence\_from;

public TextField Text\_BudgetEducation\_from;

public TextField Text\_BudgetEducation\_to;

public TextField Text\_departments;

public TextField Text\_Ministries;

public TextField Text\_GeneralCourts;

public TextField Text\_Parliament;

public TextField Text\_ConstCount;

public TextField Text\_Several\_Info;

public TextArea Text\_Details;

public TextArea TextArea\_Ministries;

public TextArea TextArea\_Executive;

public TextArea TextArea\_ConstCount;

public TextArea TextArea\_BasicCharact;

public TextArea TextAreaParliament;

public TextArea TextArea\_GenetalCounts;

public void Button\_run\_on(ActionEvent actionEvent) {

final Integer[] population = {Integer.parseInt(Text\_population\_from.getText()),Integer.parseInt(Text\_population\_to.getText())};

final Integer[] vvp = {Integer.parseInt(Text\_gdp\_from.getText()),Integer.parseInt(Text\_gdp\_to.getText())};

Map<String, Integer[]> severalInfoConstraints = new HashMap<String, Integer[]>(){{

put("population", population); // millions;

put("vvp", vvp);

}

};

final Integer[] deputiesCount = {Integer.parseInt(Text\_NumberParliamentMembers\_from.getText()),Integer.parseInt(Text\_NumberParliamentMembers\_to.getText())};

final Integer[] deputiesSalary = {Integer.parseInt(Text\_SalaryParliamentMember\_from.getText()),Integer.parseInt(Text\_SalaryParliamentMember\_to.getText())};

final Integer[] lawsCount = {Integer.parseInt(Text\_NumberPassedLaws\_from.getText()),Integer.parseInt(Text\_NumberPassedLaws\_to.getText())};

Map<String, Integer[]> legisletureInfoConstraints = new HashMap<String, Integer[]>(){{

put("deputiesCount",deputiesCount);

put("deputiesSalary", deputiesSalary);

put("lawsCount", lawsCount);

}

};

final Integer[] ministersCount = {Integer.parseInt(Text\_NumberMinistries\_from.getText()),Integer.parseInt(Text\_NumberMinistries\_to.getText())};

final Integer[] scienceBudget = {Integer.parseInt(Text\_BudgetEducation\_from.getText()),Integer.parseInt(Text\_BudgetEducation\_to.getText())};// millions

final Integer[] defenceBudget = {Integer.parseInt(Text\_BudgetDefence\_from.getText()),Integer.parseInt(Text\_BudgetDefence\_to.getText())};// millions

final Integer[] budget = {Integer.parseInt(Text\_BudgetCabinet\_from.getText()),Integer.parseInt(Text\_BudgetCabinet\_to.getText())};// millions

Map<String, Integer[]> ministersInfoConstraints = new HashMap<String, Integer[]>(){{

put("ministersCount",ministersCount);

put("scienceBudget", scienceBudget);

put("defenceBudget", defenceBudget);

put("budget", budget);

}

};

final Integer [] courtsCount = {Integer.parseInt(Text\_NumCounts\_from.getText()),Integer.parseInt(Text\_NumCounts\_to.getText())};

final Integer[] dealCount = {Integer.parseInt(Text\_NumDeals\_from.getText()),Integer.parseInt(Text\_NumDeals\_to.getText())};

final Integer[] judgeCount = {Integer.parseInt(Text\_NumJudges\_from.getText()),Integer.parseInt(Text\_NumJudges\_to.getText())};

final Integer[] salary = {Integer.parseInt(Text\_JudgeSalary\_from.getText()),Integer.parseInt(Text\_JudgeSalary\_to.getText())};

Map<String, Integer[]> generalJurisdictionCourtConstraints = new HashMap<String, Integer[]>(){{

put("courtsCount", courtsCount);

put("dealCount", dealCount);

put("judgeCount", judgeCount);

put("salary", salary);

}

};

final Integer[] constitDealCount = {Integer.parseInt(Text\_NumDealsConst\_from.getText()),Integer.parseInt(Text\_NumDealsConst\_to.getText())};

final Integer[] constitJudgeCount = {Integer.parseInt(Text\_NumJudgesConst\_from.getText()),Integer.parseInt(Text\_NumJudgesConst\_to.getText())};

final Integer[] constitSalary = {Integer.parseInt(Text\_JudgeSalaryConst\_from.getText()),Integer.parseInt(Text\_JudgeSalaryConst\_to.getText())};

Map<String, Integer[]> constitJurisdictionCourtConstraints = new HashMap<String, Integer[]>(){{

put("dealCount", constitDealCount);

put("judgeCount", constitJudgeCount);

put("salary", constitSalary);

}

};

Integer[] agenciesCountConstraints = {Integer.parseInt(Text\_NumberDepartments\_from.getText()),Integer.parseInt(Text\_NumberDepartments\_to.getText())};

Text\_Details.setText("");

// contry chosing

AlternativeChoosingAlgorithm alternativeChoosingAlgorithm = new AlternativeChoosingAlgorithm(legisletureInfoConstraints,

ministersInfoConstraints, agenciesCountConstraints, constitJurisdictionCourtConstraints,

generalJurisdictionCourtConstraints, severalInfoConstraints,Text\_Details );

alternativeChoosingAlgorithm.makeAllCalculationsInOneStep();

Country expectedCountry = alternativeChoosingAlgorithm.getChosenCountry();

Text\_Several\_Info.setText(expectedCountry.severalInfo.getIdentifier());

Text\_departments.setText(expectedCountry.executivePower.getNumberOfAgencies().getIdentifier());

Text\_ConstCount.setText(expectedCountry.judiciary.getConstitCourt().getIdentifier());

Text\_GeneralCourts.setText(expectedCountry.judiciary.getGeneralJurisdictionCourt().getIdentifier());

Text\_Parliament.setText(expectedCountry.legisleture.getIdentifier());

Text\_Ministries.setText(expectedCountry.executivePower.getMinisters().getIdentifier());

/\* System.out.println("SeveralInfo " + expectedCountry.severalInfo.getIdentifier());

System.out.println("NumberOfAgencies " +expectedCountry.executivePower.getNumberOfAgencies().getIdentifier());

System.out.println("Ministers " + expectedCountry.executivePower.getMinisters().getIdentifier());

System.out.println("ConstitCourt " + expectedCountry.judiciary.getConstitCourt().getIdentifier());

System.out.println("GeneralCourt " + expectedCountry.judiciary.getGeneralJurisdictionCourt().getIdentifier());

System.out.println("Legisleture " + expectedCountry.legisleture.getIdentifier());

\*/

}

}

package AlternativeSelection;

import CountriesAndCountryDictionaries.Country;

import CountriesAndCountryDictionaries.IntegersForDictionaries;

import Courts.ConstitCourt;

import Courts.GeneralJurisdictionCourt;

import Executive.Ministers;

import Executive.NumberOfAgencies;

import LegisletureAndSeveralInfo.Legisleture;

import LegisletureAndSeveralInfo.SeveralInfo;

import javafx.scene.control.TextArea;

import java.util.ArrayList;

import java.util.Map;

/\*\*

\* Created by Gregory on 17.02.2015.

\*/

public class AlternativeChoosingAlgorithm {

SeveralInfo chosenSeveralInfo;

Legisleture chosenLegisleture;

ConstitCourt chosenConstitCourt;

GeneralJurisdictionCourt chosenGeneralJurisdictionCourt;

Ministers chosenMinisters;

NumberOfAgencies chosenNumberOfAgencies;

TextArea Text\_Details;

// public Country(String identifier, Map<String, Integer> countryLegisleture, Map<String, Integer> countryMinisters,

// Integer countryNumberOfAgencies,

// Map<String, Integer> countryConstitCourt, Map<String, Integer> countryGeneralJurisdictionCourt,

// Map<String, Integer> countrySeveralInfo){

// all this constraints are dictionaries with identical for Map, which above, keys. Value for certain key is Integer[2] array.

// Integer [0] smaller value

// Integer [1] bigger value

Map<String, Integer[]> legisletureConstraints;

Map<String, Integer[]> severalInfoConstraints;

Map<String, Integer[]> constitCourtConstraints;

Map<String, Integer[]> generalJurisdictionConstraints;

Map<String, Integer[]> ministersConstraints;

Integer[] numberOfAgenciesConstraints;

private ArrayList<Country> dictionaryOfCountries;

private Country chosenCountry;

public boolean isContainedInRange(Integer number, Integer[] range){

if (range.length > 1){

if (range[0] >= range[1])

Text\_Details.setText(Text\_Details.getText()+"\n"+" Inverted range" + range[0] +" "+ range[1]);

// System.out.println(" Inverted range" + range[0] +" "+ range[1]);

if (range[0] <= number && range[1] >= number) return true;

}

return false;

}

void createChosenCountry(){

chosenCountry = new Country(chosenLegisleture, chosenMinisters, chosenNumberOfAgencies,

chosenConstitCourt,chosenGeneralJurisdictionCourt,

chosenSeveralInfo);

}

Integer[] randomIndexes(Integer size){

Integer[] randArray = new Integer[size];

int j;

randArray[0] = (int)(Math.random() \* size);

for(int i = 1; i<size; ++i) {

boolean isRepeat = true;

while (isRepeat) {

isRepeat = false;

randArray[i] = (int) (Math.random() \* size);

for (j = 0; j < i; ++j) {

if (randArray[j] == randArray[i]) {

isRepeat = true;

break;

}

}

}

}

return randArray;

}

void makeChosenBranchesOfCountry(){

//int rnd = 1 + (int)(Math.random() \* 6);// delete one of chosen branches

Integer countriesCount = dictionaryOfCountries.size();

//---------------------------SeveralInfo-------------------------------

Integer[] randIndexes = randomIndexes(countriesCount);

for (int i = 0; i < countriesCount; ++i){

Country country = dictionaryOfCountries.get(randIndexes[i]);

Map <String, Integer> currentSeveralInfoDictionary = country.severalInfo.getSeveralInfoDictionary();

if(doesFirstDictionaryInRangeOfSecond(currentSeveralInfoDictionary,severalInfoConstraints)) {

chosenSeveralInfo = new SeveralInfo(country.severalInfo.getIdentifier(),currentSeveralInfoDictionary);

break;

}

}

//---------------------------Legisleture-------------------------------

for (int i = 0; i < countriesCount; ++i){

Country country = dictionaryOfCountries.get(randIndexes[i]);

Map <String, Integer> currentLegisletureDictionary = country.legisleture.getLegisletureDictionary();

if(doesFirstDictionaryInRangeOfSecond(currentLegisletureDictionary,legisletureConstraints)) {

chosenLegisleture = new Legisleture(country.legisleture.getIdentifier(),currentLegisletureDictionary);

break;

}

}

//---------------------------executivePower.Ministers-------------------------------

for (int i = 0; i < countriesCount; ++i){

Country country = dictionaryOfCountries.get(randIndexes[i]);

Map <String, Integer> currentMinistersDictionary = country.executivePower.getMinisters().getMinistersDictionary();

if(doesFirstDictionaryInRangeOfSecond(currentMinistersDictionary,ministersConstraints)) {

chosenMinisters = new Ministers(country.executivePower.getMinisters().getIdentifier(),currentMinistersDictionary);

break;

}

}

//---------------------------executivePower.NumberOfAgencies-------------------------------

for (int i = 0; i < countriesCount; ++i){

Country country = dictionaryOfCountries.get(randIndexes[i]);

Integer currentNumberOfAgencies = country.executivePower.getNumberOfAgencies().getNumberOfAgencies();

if(isContainedInRange(currentNumberOfAgencies,numberOfAgenciesConstraints)) {

chosenNumberOfAgencies = new NumberOfAgencies(country.executivePower.getNumberOfAgencies().getIdentifier(),currentNumberOfAgencies);

break;

}

}

//---------------------------Judiciary.ConstitCourt-------------------------------

for (int i = 0; i < countriesCount; ++i){

Country country = dictionaryOfCountries.get(randIndexes[i]);

Map <String, Integer> currentConstitJurisdictionDictionary = country.judiciary.getConstitCourt().getConstitJurisdictionDictionary();

if(doesFirstDictionaryInRangeOfSecond(currentConstitJurisdictionDictionary,constitCourtConstraints)) {

chosenConstitCourt = new ConstitCourt(country.judiciary.getConstitCourt().getIdentifier(),currentConstitJurisdictionDictionary);

break;

}

}

//---------------------------Judiciary.GeneralJurisdictionCourt-------------------------------

for (int i = 0; i < countriesCount; ++i){

Country country = dictionaryOfCountries.get(randIndexes[i]);

Map <String, Integer> currentGeneralJurisdictionDictionary = country.judiciary.getGeneralJurisdictionCourt().getGeneralJurisdictionDictionary();

if(doesFirstDictionaryInRangeOfSecond(currentGeneralJurisdictionDictionary,generalJurisdictionConstraints)) {

chosenGeneralJurisdictionCourt = new GeneralJurisdictionCourt(country.judiciary.getGeneralJurisdictionCourt().getIdentifier(),currentGeneralJurisdictionDictionary);

break;

}

}

}

boolean doesFirstDictionaryInRangeOfSecond(Map<String, Integer> info, Map<String, Integer[]> infoConstraints){

if (info == null) {

Text\_Details.setText(Text\_Details.getText()+"\n"+" doesFirstDictionaryInRangeOfSecond: Info is null");

//System.out.println(" doesFirstDictionaryInRangeOfSecond: Info is null");

return false;

}

for (Map.Entry<String, Integer> entry: info.entrySet()) {

String key = entry.getKey();

Integer value = entry.getValue();

Text\_Details.setText(Text\_Details.getText()+"\n"+key +" "+value);

// System.out.println(key +" "+value);

Integer []range = infoConstraints.get(key);

if (range == null )

Text\_Details.setText(Text\_Details.getText()+"\n"+" in constraints does not exist");

//System.out.println(key + "in constraints does not exist");

if(!isContainedInRange(value,range)) return false;

}

return true;

}

public ArrayList<Country> createDictOfCountriesFromDictionaries(){

ArrayList<Country> countriesList = new ArrayList<Country>(5);

countriesList.add(new Country("India",IntegersForDictionaries.IndiaLegisletureInfo,

IntegersForDictionaries.IndiaMinistersInfo,

IntegersForDictionaries.IndiaAgenciesCount,IntegersForDictionaries.IndiaConstitJurisdictionCourt,

IntegersForDictionaries.IndiaGeneralJurisdictionCourt,IntegersForDictionaries.IndiaSeveralInfo));

countriesList.add(new Country("USA",IntegersForDictionaries.USALegisletureInfo,

IntegersForDictionaries.USAMinistersInfo,

IntegersForDictionaries.USAAgenciesCount,IntegersForDictionaries.USAConstitJurisdictionCourt,

IntegersForDictionaries.USAGeneralJurisdictionCourt,IntegersForDictionaries.USASeveralInfo));

countriesList.add(new Country("Ukraine",IntegersForDictionaries.UkraineLegisletureInfo,

IntegersForDictionaries.UkraineMinistersInfo,

IntegersForDictionaries.UkraineAgenciesCount,IntegersForDictionaries.UkraineConstitJurisdictionCourt,

IntegersForDictionaries.UkraineGeneralJurisdictionCourt,IntegersForDictionaries.UkraineSeveralInfo));

countriesList.add(new Country("GreatBritain",IntegersForDictionaries.GreatBritainLegisletureInfo,

IntegersForDictionaries.GreatBritainMinistersInfo,

IntegersForDictionaries.GreatBritainAgenciesCount,IntegersForDictionaries.GreatBritainConstitJurisdictionCourt,

IntegersForDictionaries.GreatBritainGeneralJurisdictionCourt,IntegersForDictionaries.GreatBritainSeveralInfo));

countriesList.add(new Country("Russian",IntegersForDictionaries.RussianLegisletureInfo,

IntegersForDictionaries.RussianMinistersInfo,

IntegersForDictionaries.RussianAgenciesCount,IntegersForDictionaries.GreatBritainConstitJurisdictionCourt,

IntegersForDictionaries.RussianGeneralJurisdictionCourt,IntegersForDictionaries.RussianSeveralInfo));

dictionaryOfCountries = countriesList;

return countriesList;

}

//

public AlternativeChoosingAlgorithm(Map<String, Integer[]> legisletureConstraints,

Map<String, Integer[]> ministersConstraints,Integer[] numberOfAgenciesConstraints,

Map<String, Integer[]> constitCourtConstraints,

Map<String, Integer[]> generalJurisdictionConstraints,

Map<String, Integer[]> severalInfoConstraints,

TextArea Text\_Details

)

{

this.legisletureConstraints = legisletureConstraints;

this.ministersConstraints = ministersConstraints;

this.numberOfAgenciesConstraints = numberOfAgenciesConstraints;

this.constitCourtConstraints = constitCourtConstraints;

this.generalJurisdictionConstraints = generalJurisdictionConstraints;

this.severalInfoConstraints = severalInfoConstraints;

this.Text\_Details=Text\_Details;

createDictOfCountriesFromDictionaries();

}

public Country getChosenCountry() {

return chosenCountry;

}

public void makeAllCalculationsInOneStep(){

makeChosenBranchesOfCountry();

createChosenCountry();

}

}

package CountriesAndCountryDictionaries;

import Courts.ConstitCourt;

import Courts.GeneralJurisdictionCourt;

import Courts.Judiciary;

import Executive.ExecutivePower;

import Executive.Ministers;

import Executive.NumberOfAgencies;

import LegisletureAndSeveralInfo.Legisleture;

import LegisletureAndSeveralInfo.SeveralInfo;

import java.util.Map;

/\*\*

\* Created by Gregory on 16.02.2015.

\*/

public class Country {

public ExecutivePower executivePower;

public Judiciary judiciary;

public Legisleture legisleture;

public SeveralInfo severalInfo;

public String identifier;

public Country(Legisleture legisleture,

Ministers ministers, NumberOfAgencies numberOfAgencies,

ConstitCourt constitCourt, GeneralJurisdictionCourt generalJurisdictionCourt,

SeveralInfo severalInfo)

{

this.legisleture = legisleture;

this.severalInfo = severalInfo;

this.judiciary = new Judiciary(constitCourt, generalJurisdictionCourt);

this.executivePower = new ExecutivePower(ministers, numberOfAgencies);

}

public Country(String identifier, Map<String, Integer> countryLegisleture, Map<String, Integer> countryMinisters,

Integer countryNumberOfAgencies,Map<String, Integer> countryConstitCourt,

Map<String, Integer> countryGeneralJurisdictionCourt,

Map<String, Integer> countrySeveralInfo)

{

this.executivePower = new ExecutivePower(identifier,countryMinisters, countryNumberOfAgencies);

this.severalInfo = new SeveralInfo(identifier,countrySeveralInfo);

this.judiciary = new Judiciary(identifier,countryConstitCourt, countryGeneralJurisdictionCourt);

this.legisleture = new Legisleture(identifier,countryLegisleture);

}

}

package Executive;

import java.util.Map;

/\*\*

\* Created by Gregory on 16.02.2015.

\*/

public class ExecutivePower {

private NumberOfAgencies numberOfAgencies;

private Ministers ministers;

public ExecutivePower( Ministers ministers, NumberOfAgencies numberOfAgencies){

this.ministers = ministers;

this.numberOfAgencies = numberOfAgencies;

}

public ExecutivePower(String identifier, Map<String, Integer> ministersDictionary, Integer numberOfAgencies){

this.ministers = new Ministers(identifier, ministersDictionary);

this.numberOfAgencies = new NumberOfAgencies(identifier, numberOfAgencies);

}

public NumberOfAgencies getNumberOfAgencies() {

return numberOfAgencies;

}

public Ministers getMinisters() {

return ministers;

}

}

package LegisletureAndSeveralInfo;

import java.util.Map;

/\*\*

\* Created by Gregory on 16.02.2015.

\*/

public class SeveralInfo {

private Map<String, Integer> severalInfoDictionary; // includes population, vvp;

private String identifier;

public SeveralInfo(String identifier, Map<String, Integer> severalInfoDictionary){

this.severalInfoDictionary = severalInfoDictionary;

this.identifier = identifier;

}

static String[] getKeysForSeveralInfoDictionary(){

String[] keysForSeveralInfoDictionary = {"population", "vvp"};

return keysForSeveralInfoDictionary;

}

public Map<String, Integer> getSeveralInfoDictionary() {

return severalInfoDictionary;

}

public String getIdentifier() {

return identifier;

}

}

package LegisletureAndSeveralInfo;

import java.util.Map;

/\*\*

\* Created by Gregory on 16.02.2015.

\*/

public class Legisleture {

private Map<String, Integer> legisletureDictionary; // includes palatesCount, deputiesCount, deputiseSalary

private String identifier;

public Legisleture(String identifier, Map <String, Integer> legisletureDictionary){

this.identifier = identifier;

this.legisletureDictionary = legisletureDictionary;

}

static String[] getKeysForLegisletureDictionary(){

String[] keysForLegisletureDictionary = {"lawsCount", "deputiesCount", "deputiesSalary"};

return keysForLegisletureDictionary;

}

public Map<String, Integer> getLegisletureDictionary() {

return legisletureDictionary;

}

public String getIdentifier() {

return identifier;

}

}

package Executive;

public class NumberOfAgencies{

private Integer numberOfAgencies;

private String identifier;

public NumberOfAgencies(String identifier, Integer numberOfAgencies){

this.numberOfAgencies = numberOfAgencies;

this.identifier = identifier;

}

public Integer getNumberOfAgencies() {

return numberOfAgencies;

}

public String getIdentifier() {

return identifier;

}

}

package CountriesAndCountryDictionaries;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* Created by Gregory on 16.02.2015.

\*/

public class IntegersForDictionaries {

// ------------------------------------India Info------------------------------------------

public static Map<String, Integer> IndiaSeveralInfo = new HashMap<String, Integer>(){{

put("population", 1266); // millions;

put("vvp", 1548);

}

};

public static Map<String, Integer> IndiaLegisletureInfo = new HashMap<String, Integer>(){{

put("deputiesCount",250+545);

put("deputiesSalary", 1000);

put("lawsCount", 7);

}

};

public static Map<String, Integer> IndiaMinistersInfo = new HashMap<String, Integer>(){{

put("ministersCount",51);

put("scienceBudget", 790);

put("defenceBudget", 60000);

put("budget", 260000);

}

};

public static Map<String, Integer> IndiaGeneralJurisdictionCourt = new HashMap<String, Integer>(){{

put("courtsCount", 1775);

put("dealCount", 3292785);

put("judgeCount", 2765);

put("salary", 560);

}

};

public static Map<String, Integer> IndiaConstitJurisdictionCourt = new HashMap<String, Integer>(){{

put("dealCount", 6822);

put("judgeCount", 31);

put("salary", 1800);

}

};

public static Integer IndiaAgenciesCount = 10;

// ------------------------------------USA Info------------------------------------------

public static Map<String, Integer> USASeveralInfo = new HashMap<String, Integer>(){{

put("population", 318); // millions;

put("vvp", 1548);

}

};

public static Map<String, Integer> USALegisletureInfo = new HashMap<String, Integer>(){{

put("deputiesCount",435 + 100);

put("deputiesSalary", 14500);

put("lawsCount", 148);

}

};

public static Map<String, Integer> USAMinistersInfo = new HashMap<String, Integer>(){{

put("ministersCount",15);

put("scienceBudget", 80000); //millions

put("defenceBudget", 673000); //millions

put("budget", 2375000); // millions

}

};

public static Integer USAAgenciesCount = 68;

public static Map<String, Integer> USAGeneralJurisdictionCourt = new HashMap<String, Integer>(){{

put("courtsCount", 667);

put("dealCount", 4075870);

put("judgeCount", 2765);

put("salary", 16000);

}

};

public static Map<String, Integer> USAConstitJurisdictionCourt = new HashMap<String, Integer>(){{

put("dealCount", 67);

put("judgeCount", 9);

put("salary", 20400);

}

};

// ------------------------------------Ukraine Info------------------------------------------

public static Map<String, Integer> UkraineSeveralInfo = new HashMap<String, Integer>(){{

put("population", 42); // millions;

put("vvp", 4024);

}

};

public static Map<String, Integer> UkraineLegisletureInfo = new HashMap<String, Integer>(){{

put("deputiesCount",450);

put("deputiesSalary", 2200);

put("lawsCount", 699);

}

};

public static Map<String, Integer> UkraineMinistersInfo = new HashMap<String, Integer>(){{

put("ministersCount",19);

put("scienceBudget", 1200); //millions

put("defenceBudget", 2225); //millions

put("budget", 12900); // millions

}

};

public static Integer UkraineAgenciesCount = 80;

public static Map<String, Integer> UkraineGeneralJurisdictionCourt = new HashMap<String, Integer>(){{

put("courtsCount", 764);

put("dealCount", 3577307);

put("judgeCount", 8696);

put("salary", 1678);

}

};

public static Map<String, Integer> UkraineConstitJurisdictionCourt = new HashMap<String, Integer>(){{

put("dealCount", 14);

put("judgeCount", 18);

put("salary", 4100);

}

};

// ------------------------------------GreatBritain Info------------------------------------------

public static Map<String, Integer> GreatBritainSeveralInfo = new HashMap<String, Integer>(){{

put("population", 64); // millions;

put("vvp", 42423);

}

};

public static Map<String, Integer> GreatBritainLegisletureInfo = new HashMap<String, Integer>(){{

put("deputiesCount",1410);

put("deputiesSalary", 8500);

put("lawsCount", 38);

}

};

public static Map<String, Integer> GreatBritainMinistersInfo = new HashMap<String, Integer>(){{

put("ministersCount", 20);

put("scienceBudget", 134640); //millions

put("defenceBudget", 58140); //millions

put("budget", 1101600); // millions

}

};

public static Integer GreatBritainAgenciesCount = 32;

public static Map<String, Integer> GreatBritainGeneralJurisdictionCourt = new HashMap<String, Integer>(){{

put("courtsCount", 670);

put("dealCount", 1693000);

put("judgeCount", 24267);

put("salary", 5500);

}

};

public static Map<String, Integer> GreatBritainConstitJurisdictionCourt = new HashMap<String, Integer>(){{

put("dealCount", 0);

put("judgeCount", 0);

put("salary", 0);

}

};

// ------------------------------------Russian Info------------------------------------------

public static Map<String, Integer> RussianSeveralInfo = new HashMap<String, Integer>(){{

put("population", 146); // millions;

put("vvp", 14680);

}

};

public static Map<String, Integer> RussianLegisletureInfo = new HashMap<String, Integer>(){{

put("deputiesCount",450 + 170);

put("deputiesSalary", 7576);

put("lawsCount", 404);

}

};

public static Map<String, Integer> RussianMinistersInfo = new HashMap<String, Integer>(){{

put("ministersCount",19);

put("scienceBudget", 11860); //millions

put("defenceBudget", 41240); //millions

put("budget", 126390); // millions

}

};

public static Integer RussianAgenciesCount = 27;

public static Map<String, Integer> RussianGeneralJurisdictionCourt = new HashMap<String, Integer>(){{

put("courtsCount", 2354);

put("dealCount", 4216921);

put("judgeCount", 24032);

put("salary", 2121);

}

};

public static Map<String, Integer> RussianConstitJurisdictionCourt = new HashMap<String, Integer>(){{

put("dealCount", 30);

put("judgeCount", 19);

put("salary", 13636);

}

};

}

package Executive;

import java.util.Map;

public class Ministers{ // this class about ministers

private Map<String, Integer> ministersDictionary; // includes fields ministersCount,

// budget, scienceBudget, defenceBudget;

private String identifier;

public Ministers(String identifier, Map<String, Integer> ministersDictionary){

this.identifier = identifier;

this.ministersDictionary = ministersDictionary;

}

static String[] getKeysForMinistersDictionary(){

String[] keysForMinistersDictionary = {"ministersCount", "budget", "scienceBudget", "defenceBudget"};

return keysForMinistersDictionary;

}

public Map<String, Integer> getMinistersDictionary() {

return ministersDictionary;

}

public String getIdentifier() {

return identifier;

}

package Executive;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

/\*\*

\* Created by Gregory on 18.02.2015.

\*/

public class TestingRun extends Application {

@Override

public void start(Stage primaryStage) throws Exception{

Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));

primaryStage.setTitle("Lab5");

primaryStage.setScene(new Scene(root));

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}