Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

Кафедра Автоматизированных систем

ОТЧЕТ

Вариант 13

к лабораторной работе по дисциплине:

|  |
| --- |
| Объектно–ориентированное программирование |
| Разработка классов с использованием механизмов наследования, |
| полиморфизма и инкапсуляции. |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | АСУб-19-1 |  |  |  | Вилачёв А.Е. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Маланова Т.В, |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск 2020 г.

**Содержание**

[1 Постановка задачи 3](#_Toc26866821)

[2 Проектирование классов 4](#_Toc26866822)

[3 Описание структуры меню 7](#_Toc26866823)

[4 Таблица тестов 9](#_Toc26866824)

[5 Результаты тестирования 10](#_Toc26866825)

[6 Исходный код 13](#_Toc26866827)

[Список использованных источников 26](#_Toc26866828)

1 Постановка задачи

1. Изучение механизмов наследования, полиморфизма и инкапсуляции на практике.

2. Получение практических навыков описания иерархии классов и организации работы с объектами-членами классов-«групп».

3. Получение практических навыков использования абстрактных классов и переопределения унаследованных методов.

В соответствии с индивидуальным заданием описать иерархию классов, для каждого класса описать поля и соответствующие методы доступа к ним. В зависимости от задания некоторые из этих классов (как минимум один) являются абстрактными и служат для выделения общих данных и поведения для других классов. Абстрактный класс должен содержать как минимум один абстрактный метод, реализация которого у его наследников должна различаться. Помимо этого, в общую часть задания входит разработка класса, группирующего объекты описанных в соответствии с заданием классов. Для первого варианта индивидуального задания это может быть класс «кафедра». Для выполнения задания необходимо создать некоторое количество объектов, добавить их в группу используя предусмотренные методы класса-«группы» и для каждого из них вызвать унаследованный метод.

республика, монархия, королевство, государство.

2 Проектирование классов

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 – Проектирование классов | | | |
| **№** | **Название** | **Назначение** | **Поля (атрибуты)** |
| 1 | public class Main | Класс для запуска программы | public static Scanner in = new Scanner(System.in); |
| 2 | public class Interface | Класс для работы пользователя с программой | -- -- |
| 3 | public class Work\_with\_States | Класс для работы со списком стран с их атрибутами | private static ArrayList<State> states = new ArrayList<>(); |
| 4 | abstract public class State | Абстрактный класс | protected String name;  protected String form\_of\_government;  protected int population;  protected int army;  protected float happiness\_index; |
| 5 | public class Republic extends State | Класс-наследник от State | private String president; |
| 6 | abstract public class Monarchy extends State | Класс-наследник от State | -- -- |
| 7 | public class Kingdom extends Monarchy | Класс-наследник от Monarchy | private String king; |
| 8 | public class Sultanate extends Monarchy | Класс-наследник от Monarchy | private String sultan; |

**Main:**

Методы:

public static void main(String[] args) – начало программы, вызывает интерфейс

public static int StrtoInt() – контролирует ввод чисел типа Int

public static float StrtoFloat() - контролирует ввод чисел типа Float

public static boolean StrtoBoolean() - контролирует ввод чисел типа Boolean

**Interface:**

Методы:

public static void Menu() – Создаёт среду для работы пользователся с программой

**Work\_with\_States:**

Методы:

public static void AddState(State state) – добавляет в список новое поле со Страной и её атрибутами

public static void PrintStatesList() – Выводит на экран список стран с атрибутами

public static void DeleatState(String name) – Удаляет выбранную страну из списка покупал

public static void SortName() – Сортирует список стран по названию

public static void SortPopulation() – Сортирует список стран по количеству населения

public static void Sortarmy() - Сортирует список стран по количеству людей в армии

public static void Sorthappiness\_index() - Сортирует список стран по индексу счастья

public static void Sorthappiness\_index(boolean q) Сортирует список стран по индексу счастья, не выводя очарованный мною сиг

public static int SumPopulation() – Суммарное население в странах из списка

public static float ArmyPercentage(String name) – Процент, который занимает армия от общего населения в выбранной стране

public static float Difference\_in\_Happines\_index() – Считает разницу между максимальным и минимальным индексом счастья среди стран из списка

public static void Predominance\_getform\_of\_government() – Выясняет, каких государств больше: республик или монархических государств

public static void Predominance\_gettype\_of\_monarchy() – Выясняет, каких монархических государств больше – королевств или султанатов

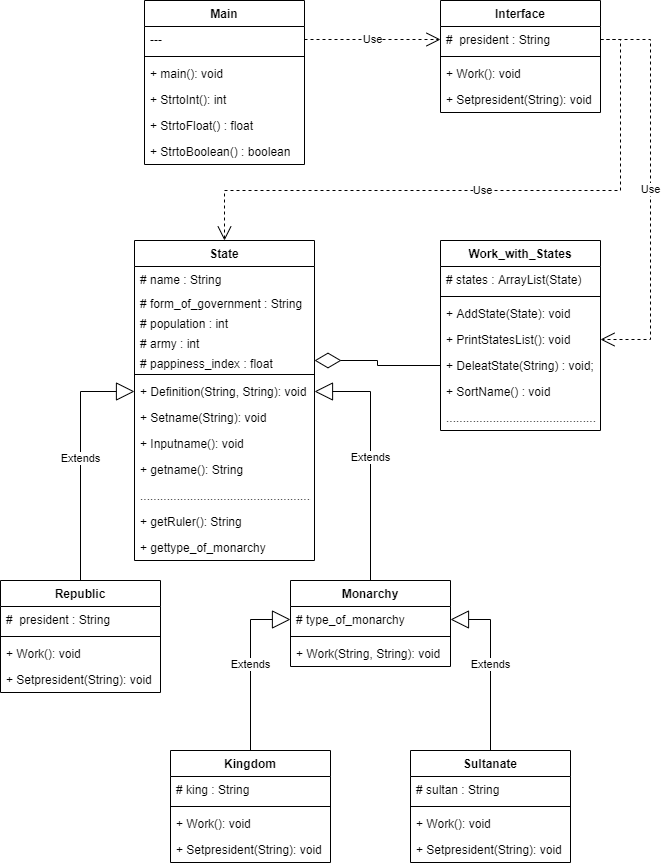


Рисунок 1 - Схема классов

3 Описание структуры меню

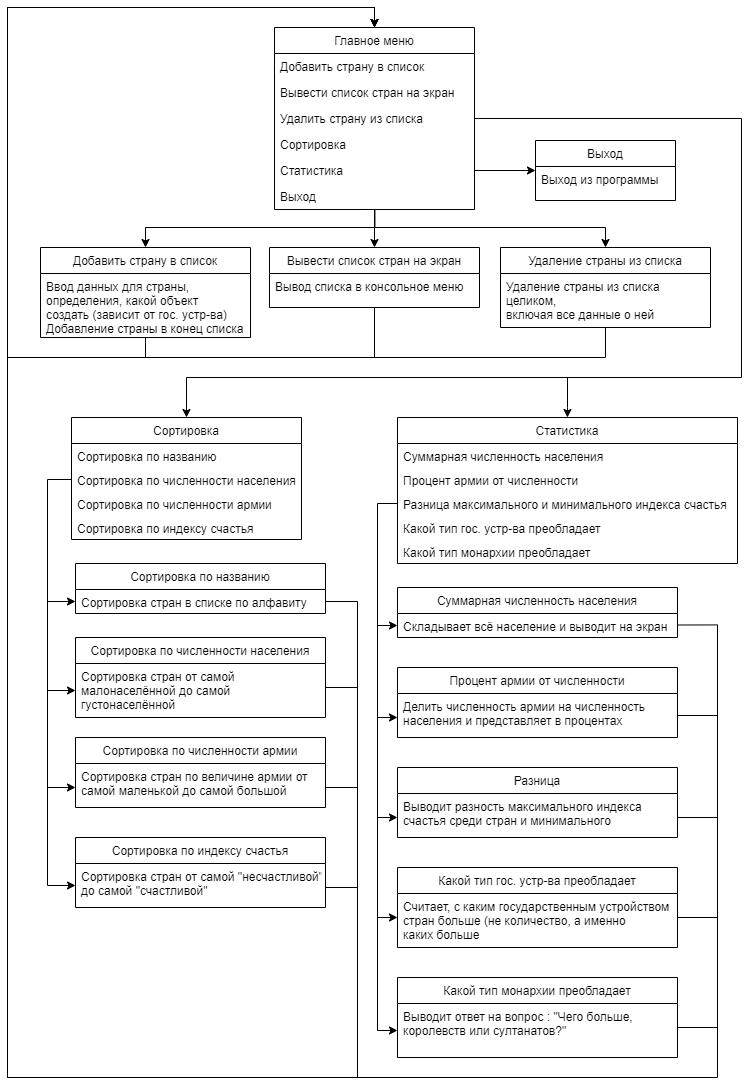
****

Рисунок 2 – Структура пользовательского меню

4 Таблица тестов

Таблица 2 – Таблица тестов.

|  |  |  |
| --- | --- | --- |
| **№** | **Словесное описание теста** | **Что должна сделать программа** |
| 1 | Меню | Вывести меню |
| 2 | Выбор в меню пункт 1 | Переход к добавлению человека |
| 3 | Выбор в меню пункт 2 | Вывод списка |
| 4 | Выбор в меню пункт 3 | Переход к удалению человека |
| 5 | Выбор в меню пункт 4 | Меню сортировки |
| 6 | Выбор в пункте 4 пункта 1 | Сортировка стран по названию и вывод |
| 7 | Выбор в пункте 4 пункта 2 | Сортировка стран по численности населения и вывод |
| 8 | Выбор в пункте 4 пункта 3 | Сортировка стран по численности армии и вывод |
| 9 | Выбор в пункте 4 пункта 4 | Сортировка стран по индексу счастья и вывод |
| 10 | Выбор в меню пункт 5 | Вывод статистики |
| 11 | Выбор в пункте 5 пункта 1 | Вывод суммарной численности |
| 12 | Выбор в пункте 5 пункта 2 | Вывод процента части армии от населения |
| 13 | Выбор в пункте 5 пункта 3 | Вывод разницы максимального и минимального индекса счастья |
| 14 | Выбор в пункте 5 пункта 4 | Вывод, какой тип государственного строя преобладает |
| 15 | Выбор в пункте 5 пункта 5 | Вывод, какой тип монархии преобладает |
| 16 | Выбор в меню пункт 0 | Завершение программы |

5 Результаты тестирования

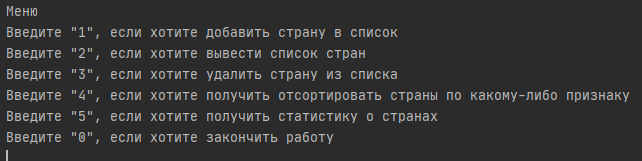


Рисунок 3 – Тест 1

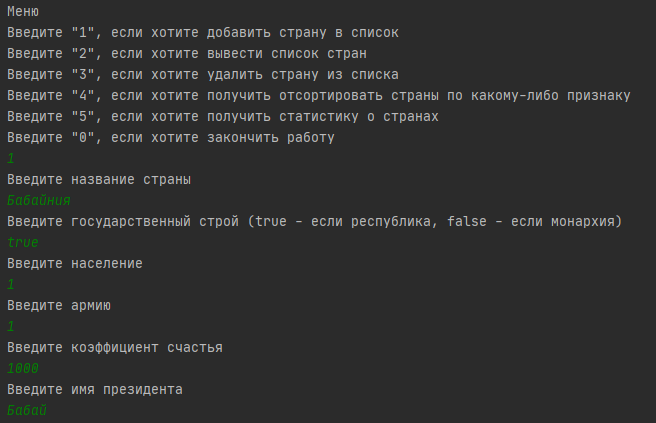


Рисунок 4 – Тест 2

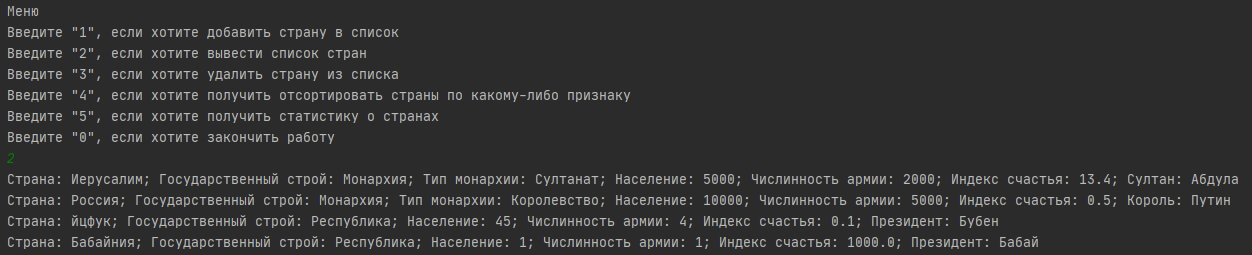


Рисунок 5 – Тест 3

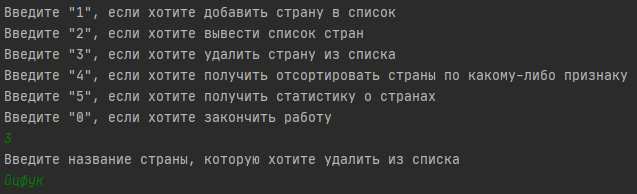


Рисунок 6 – Тест 4

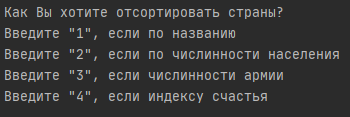


Рисунок 7 – Тест 5

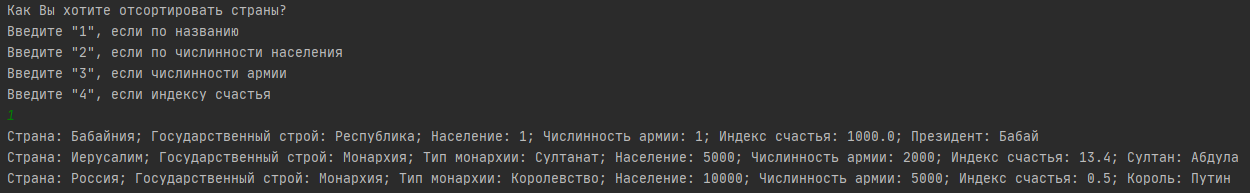


Рисунок 8 – Тест 6

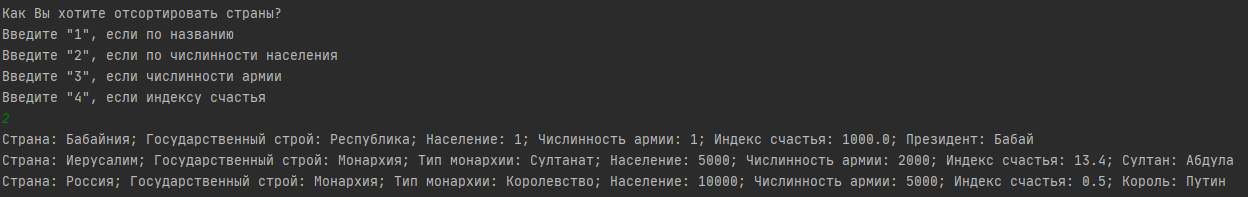


Рисунок 9 – Тест 7

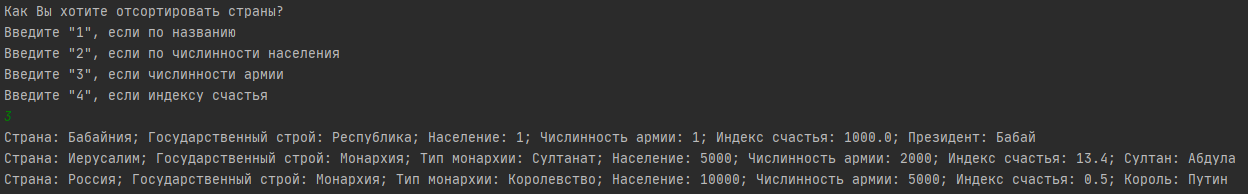


Рисунок 10 – Тест 8

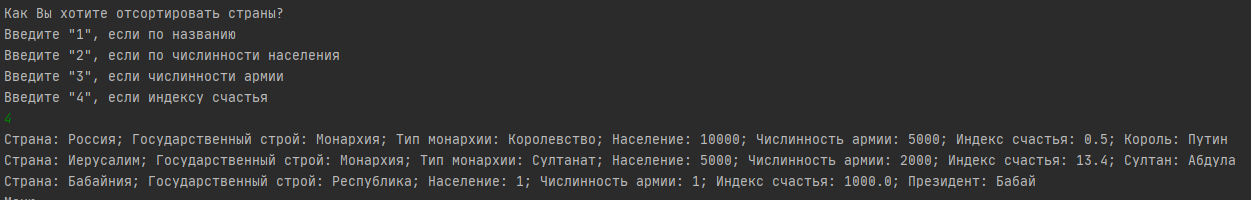


Рисунок 11 – Тест 9

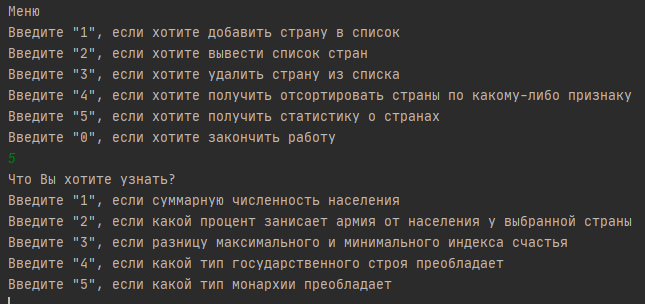


Рисунок 12 – Тест 10

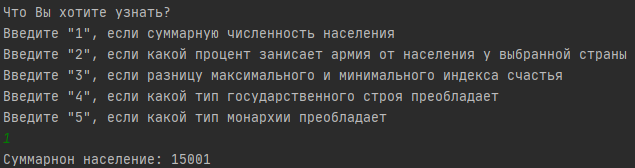


Рисунок 13 – Тест 11

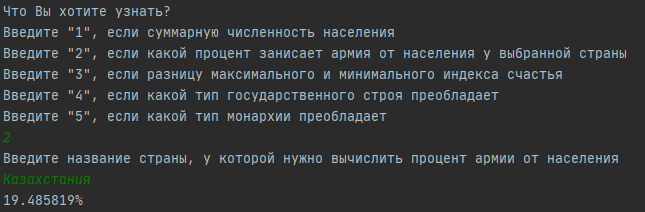


Рисунок 14 – Тест 12

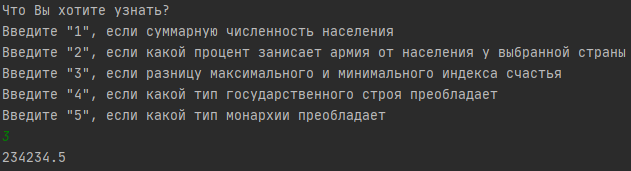


Рисунок 15 – Тест 13

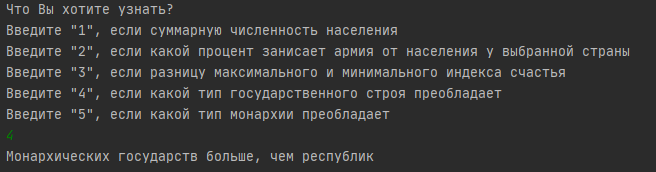


Рисунок 16 – Тест 14

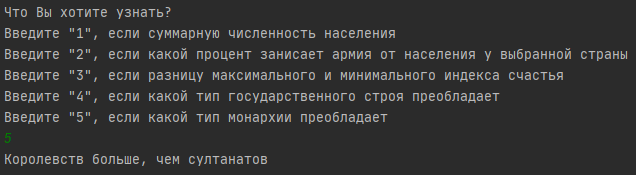


Рисунок 17 – Тест 15

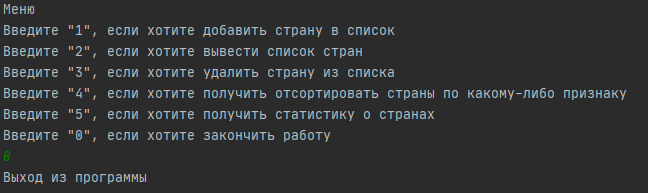


Рисунок 18 – Тест 16

6 Исходный код

Main:

package com.company;  
import java.util.Scanner;  
public class Main {  
 public static Scanner *in* = new Scanner(System.*in*);  
 public static void main(String[] args)  
 {  
 Interface.*Menu*();  
 }  
  
 public static int StrtoInt()  
 {  
 String Str = *in*.nextLine();  
 int exit;  
 try {  
 exit = Integer.*valueOf*(Str);  
 return exit;  
 }catch (NumberFormatException e) {  
 System.*err*.println("Неправильный формат строки!");  
 return *StrtoInt*();  
 }  
 }  
 public static float StrtoFloat()  
 {  
 String Str = *in*.nextLine();  
 float exit;  
 try {  
 exit = Float.*valueOf*(Str);  
 return exit;  
 }catch (NumberFormatException e) {  
 System.*err*.println("Неправильный формат строки!");  
 return *StrtoInt*();  
 }  
 }  
 public static boolean StrtoBoolean()  
 {  
 String Str = *in*.nextLine();  
 boolean help;  
 if((Str.equals("true")) | (Str.equals("false")))  
 {  
 help = Boolean.*parseBoolean*(Str);  
 return help;  
 }  
 else  
 {  
 System.*err*.println("Неправильный формат строки!");  
 return *StrtoBoolean*();  
 }  
 }  
}

State:

package com.company;  
  
abstract public class State {  
 protected String name;  
 protected String form\_of\_government;  
 protected int population;  
 protected int army;  
 protected float happiness\_index;  
 public static void Definition(String name, String form\_of\_government)  
 {  
 if (form\_of\_government == "Республика")  
 {  
 Republic republic = new Republic(name, form\_of\_government);  
 Work\_with\_States.*AddState*(republic);  
 }  
 else  
 {  
 Monarchy.*Work*(name, form\_of\_government);  
 }  
 }  
  
 private void Setname(String name)  
 {  
 this.name = name;  
 }  
 private void Inputname()  
 {  
 Setname(Main.*in*.nextLine());  
 }  
 public String getname()  
 {  
 return name;  
 }  
  
 private void Setform\_of\_government(String form\_of\_government)  
 {  
 this.form\_of\_government = form\_of\_government;  
 }  
 private void Inputform\_of\_government()  
 {  
 Setform\_of\_government(Main.*in*.nextLine());  
 }  
 public String getform\_of\_government()  
 {  
 return form\_of\_government;  
 }  
  
 private void Setpopulation(int population)  
 {  
 this.population = population;  
 }  
 protected void Inputpopulation()  
 {  
 Setpopulation(Main.*StrtoInt*());  
 }  
 protected int getpopulation()  
 {  
 return population;  
 }  
  
 private void Setarmy(int army)  
 {  
 this.army = army;  
 }  
 protected void Inputarmy()  
 {  
 Setarmy(Main.*StrtoInt*());  
 }  
 protected int getarmy()  
 {  
 return army;  
 }  
  
 private void Sethappiness\_index(float happiness\_index)  
 {  
 this.happiness\_index = happiness\_index;  
 }  
 protected void Inputhappiness\_index()  
 {  
 Sethappiness\_index(Main.*StrtoFloat*());  
 }  
 protected float gethappiness\_index()  
 {  
 return happiness\_index;  
 }  
  
 protected abstract String getRuler();  
 protected abstract String gettype\_of\_monarchy();  
}

Republic:

package com.company;  
  
public class Republic extends State  
{  
 private String president;  
 protected Republic(String name, String form\_of\_government)  
 {  
 this.name = name;  
 this.form\_of\_government = form\_of\_government;  
 /\*this.population = population;  
 this.army = army;  
 this.happiness\_index = happiness\_index;\*/  
 Work();  
 }  
  
 private void Work()  
 {  
 System.*out*.println("Введите население");  
 super.Inputpopulation();  
 System.*out*.println("Введите армию");  
 super.Inputarmy();  
 System.*out*.println("Введите коэффициент счастья");  
 super.Inputhappiness\_index();  
 System.*out*.println("Введите имя президента");  
 Setpresident(Main.*in*.nextLine());  
 }  
  
  
  
 private void Setpresident(String president)  
 {  
 this.president = president;  
 }  
  
 @Override  
 protected String getRuler() {  
 return "Президент: " + president;  
 }  
 @Override  
 protected String gettype\_of\_monarchy() {  
 return "";  
 }  
  
}

Monarchy:

package com.company;  
  
abstract public class Monarchy extends State  
{  
 protected String type\_of\_monarchy;  
 protected static void Work(String name, String form\_of\_government)  
 {  
 boolean check;  
 System.*out*.println("Какой тип монархии в этом государстве? (Введите \"true\", если королевство и \"false\", если султанат");  
 check = Main.*StrtoBoolean*();  
 if(check)  
 {  
 Kingdom kingdom = new Kingdom(name, form\_of\_government, "Королевство");  
 Work\_with\_States.*AddState*(kingdom);  
 }  
 else  
 {  
 Sultanate sultanate = new Sultanate(name, form\_of\_government, "Султанат");  
 Work\_with\_States.*AddState*(sultanate);  
 }  
 }  
  
}

Kingdom:

package com.company;  
  
public class Kingdom extends Monarchy  
{  
 private String king;  
 protected Kingdom(String name, String form\_of\_government, String type\_of\_monarchy)  
 {  
 this.name = name;  
 this.form\_of\_government = form\_of\_government;  
 this.type\_of\_monarchy = type\_of\_monarchy;  
 Work();  
 }  
  
 private void Work()  
 {  
 System.*out*.println("Введите население");  
 super.Inputpopulation();  
 System.*out*.println("Введите армию");  
 super.Inputarmy();  
 System.*out*.println("Введите коэффициент счастья");  
 super.Inputhappiness\_index();  
 System.*out*.println("Введите имя короля");  
 Setpresident(Main.*in*.nextLine());  
 }  
  
 private void Setpresident(String president)  
 {  
 this.king = president;  
 }  
  
  
 @Override  
 protected String getRuler() {  
 return "Король: " + king;  
 }  
  
 @Override  
 protected String gettype\_of\_monarchy() {  
 return "; Тип монархии: " + super.type\_of\_monarchy;  
 }  
}

Sultanate:

package com.company;  
  
public class Sultanate extends Monarchy  
{  
 private String sultan;  
 protected Sultanate(String name, String form\_of\_government, String type\_of\_monarchy)  
 {  
 this.name = name;  
 this.form\_of\_government = form\_of\_government;  
 this.type\_of\_monarchy = type\_of\_monarchy;  
 Work();  
 }  
  
 private void Work()  
 {  
 System.*out*.println("Введите население");  
 super.Inputpopulation();  
 System.*out*.println("Введите армию");  
 super.Inputarmy();  
 System.*out*.println("Введите индекс счастья");  
 super.Inputhappiness\_index();  
 System.*out*.println("Введите имя султана");  
 Setpresident(Main.*in*.nextLine());  
 }  
  
 private void Setpresident(String president)  
 {  
 this.sultan = president;  
 }  
  
  
 @Override  
 protected String getRuler() {  
 return "Султан: " + sultan;  
 }  
  
 @Override  
 protected String gettype\_of\_monarchy() {  
 return "; Тип монархии: " + super.type\_of\_monarchy;  
 }  
}

Interface:

System.*out*.println("Введите \"5\", если хотите получить статистику о странах");  
 System.*out*.println("Введите \"0\", если хотите закончить работу");  
 exit = Main.*StrtoInt*();  
 switch (exit)  
 {  
 case 0:  
 {  
 System.*out*.println("Выход из программы");  
 break;  
 }  
 case 1:  
 {  
 String name;  
 String form\_of\_government = null;  
 boolean check;  
 System.*out*.println("Введите название страны");  
 name = Main.*in*.nextLine();  
 System.*out*.println("Введите государственный строй (true - если республика, false - если монархия)");  
 check = Main.*StrtoBoolean*();  
 if(check)  
 {  
 form\_of\_government = "Республика";  
 }  
 else  
 {  
 form\_of\_government = "Монархия";  
 }  
 State.*Definition*(name, form\_of\_government);  
 break;  
 }  
 case 2:  
 {  
 Work\_with\_States.*PrintStatesList*();  
 break;  
 }  
 case 3:  
 {  
 System.*out*.println("Введите название страны, которую хотите удалить из списка");  
 Work\_with\_States.*DeleatState*(Main.*in*.nextLine());  
 break;  
 }  
 case 4:  
 {  
 int check;  
 System.*out*.println("Как Вы хотите отсортировать страны?");  
 System.*out*.println("Введите \"1\", если по названию");  
 System.*out*.println("Введите \"2\", если по числинности населения");  
 System.*out*.println("Введите \"3\", если числинности армии");  
 System.*out*.println("Введите \"4\", если индексу счастья");  
 check = Main.*StrtoInt*();  
 switch (check)  
 {  
 case 1:  
 {  
 Work\_with\_States.*SortName*();  
 break;  
 }  
 case 2:  
 {  
 Work\_with\_States.*SortPopulation*();  
 break;  
 }  
 case 3:  
 {  
 Work\_with\_States.*Sortarmy*();  
 break;  
 }  
 case 4:  
 {  
 Work\_with\_States.*Sorthappiness\_index*();  
 break;  
 }  
 default:  
 {  
 System.*out*.println("Неверный ввод");  
 break;  
 }  
 }  
 break;  
 }  
 case 5:  
 {  
 int check;  
 System.*out*.println("Что Вы хотите узнать?");  
 System.*out*.println("Введите \"1\", если суммарную численность населения");  
 System.*out*.println("Введите \"2\", если какой процент занисает армия от населения у выбранной страны");  
 System.*out*.println("Введите \"3\", если разницу максимального и минимального индекса счастья");  
 System.*out*.println("Введите \"4\", если какой тип государственного строя преобладает");  
 System.*out*.println("Введите \"5\", если какой тип монархии преобладает");  
 check = Main.*StrtoInt*();  
 switch (check)  
 {  
 case 1:  
 {  
 System.*out*.println("Суммарнон население: " + Work\_with\_States.*SumPopulation*());  
 break;  
 }  
 case 2:  
 {  
 System.*out*.println("Введите название страны, у которой нужно вычислить процент армии от населения");  
 System.*out*.println(Work\_with\_States.*ArmyPercentage*(Main.*in*.nextLine()) + "%");  
 break;  
 }  
 case 3:  
 {  
 System.*out*.println(Work\_with\_States.*Difference\_in\_Happines\_index*());  
 break;  
 }  
 case 4:  
 {  
 Work\_with\_States.*Predominance\_getform\_of\_government*();  
 break;  
 }  
 case 5:  
 {  
 Work\_with\_States.*Predominance\_gettype\_of\_monarchy*();  
 break;  
 }  
 default:  
 {  
 System.*out*.println("Неверный ввод");  
 }  
 }  
 break;  
 }  
 default:  
 {  
 System.*out*.println("Неверный ввод");  
 break;  
 }  
 }  
 }  
 }  
}

Work\_with\_States:

package com.company;  
  
import java.util.ArrayList;  
import java.util.Collections;  
  
public class Work\_with\_States  
{  
 private static ArrayList<State> *states* = new ArrayList<>();  
  
 public static void AddState(State state)  
 {  
 *states*.add(state);  
 System.*out*.println(*states*.size());  
 }  
  
 public static void PrintStatesList()  
 {  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 System.*out*.print("Страна: " + *states*.get(i).getname() + "; Государственный строй: " + *states*.get(i).getform\_of\_government() + *states*.get(i).gettype\_of\_monarchy() + "; Население: " + *states*.get(i).getpopulation() + "; ");  
 System.*out*.println("Числинность армии: " + *states*.get(i).getarmy() + "; Индекс счастья: " + *states*.get(i).gethappiness\_index() + "; " + *states*.get(i).getRuler());  
 }  
 //System.out.println(states.toString());  
 }  
  
 public static void DeleatState(String name)  
 {  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 if(*states*.get(i).getname().equals(name))  
 {  
 *states*.remove(i);  
 break;  
 }  
 }  
 }  
  
 public static void SortName()  
 {  
 ArrayList<String> names = new ArrayList<>();  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 names.add(*states*.get(i).getname());  
 }  
  
 Collections.*sort*(names);  
 for(int i = 0; i < names.size(); i++)  
 {  
 for(int j = 0; j < *states*.size() - i; j++)  
 {  
 if(names.get(i).equals(*states*.get(j).getname()))  
 {  
 System.*out*.print("Страна: " + *states*.get(j).getname() + "; Государственный строй: " + *states*.get(j).getform\_of\_government() + *states*.get(j).gettype\_of\_monarchy() + "; Население: " + *states*.get(j).getpopulation() + "; ");  
 System.*out*.println("Числинность армии: " + *states*.get(j).getarmy() + "; Индекс счастья: " + *states*.get(j).gethappiness\_index() + "; " + *states*.get(j).getRuler());  
 *states*.add(*states*.get(j));  
 *states*.remove(j);  
 break;  
 }  
 }  
 }  
 }  
  
 public static void SortPopulation()  
 {  
 ArrayList<Integer> populations = new ArrayList<>();  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 populations.add(*states*.get(i).getpopulation());  
 }  
 Collections.*sort*(populations);  
  
 for(int i = 0; i < populations.size(); i++)  
 {  
 for(int j = 0; j < *states*.size() - i; j++)  
 {  
 if(populations.get(i) == *states*.get(j).getpopulation())  
 {  
 System.*out*.print("Страна: " + *states*.get(j).getname() + "; Государственный строй: " + *states*.get(j).getform\_of\_government() + *states*.get(j).gettype\_of\_monarchy() + "; Население: " + *states*.get(j).getpopulation() + "; ");  
 System.*out*.println("Числинность армии: " + *states*.get(j).getarmy() + "; Индекс счастья: " + *states*.get(j).gethappiness\_index() + "; " + *states*.get(j).getRuler());  
 *states*.add(*states*.get(j));  
 *states*.remove(j);  
 break;  
 }  
 }  
 }  
 }  
  
 public static void Sortarmy()  
 {  
 ArrayList<Integer> armys = new ArrayList<>();  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 armys.add(*states*.get(i).getarmy());  
 }  
 Collections.*sort*(armys);  
 for(int i = 0; i < armys.size(); i++)  
 {  
 for(int j = 0; j < *states*.size(); j++)  
 {  
 if(armys.get(i) == *states*.get(j).getarmy())  
 {  
 System.*out*.print("Страна: " + *states*.get(j).getname() + "; Государственный строй: " + *states*.get(j).getform\_of\_government() + *states*.get(j).gettype\_of\_monarchy() + "; Население: " + *states*.get(j).getpopulation() + "; ");  
 System.*out*.println("Числинность армии: " + *states*.get(j).getarmy() + "; Индекс счастья: " + *states*.get(j).gethappiness\_index() + "; " + *states*.get(j).getRuler());  
 *states*.add(*states*.get(j));  
 *states*.remove(j);  
 break;  
 }  
 }  
 }  
 }  
  
 public static void Sorthappiness\_index()  
 {  
 ArrayList<Float> happiness\_indexes = new ArrayList<>();  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 happiness\_indexes.add(*states*.get(i).gethappiness\_index());  
 }  
 Collections.*sort*(happiness\_indexes);  
 for(int i = 0; i < happiness\_indexes.size(); i++)  
 {  
 for(int j = 0; j < *states*.size(); j++)  
 {  
 if(happiness\_indexes.get(i) == *states*.get(j).gethappiness\_index())  
 {  
 System.*out*.print("Страна: " + *states*.get(j).getname() + "; Государственный строй: " + *states*.get(j).getform\_of\_government() + *states*.get(j).gettype\_of\_monarchy() + "; Население: " + *states*.get(j).getpopulation() + "; ");  
 System.*out*.println("Числинность армии: " + *states*.get(j).getarmy() + "; Индекс счастья: " + *states*.get(j).gethappiness\_index() + "; " + *states*.get(j).getRuler());  
 *states*.add(*states*.get(j));  
 *states*.remove(j);  
 break;  
 }  
 }  
 }  
 }  
 public static void Sorthappiness\_index(boolean q)  
 {  
 ArrayList<Float> happiness\_indexes = new ArrayList<>();  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 happiness\_indexes.add(*states*.get(i).gethappiness\_index());  
 }  
 Collections.*sort*(happiness\_indexes);  
 for(int i = 0; i < happiness\_indexes.size(); i++)  
 {  
 for(int j = 0; j < *states*.size(); j++)  
 {  
 if(happiness\_indexes.get(i) == *states*.get(j).gethappiness\_index())  
 {  
 *states*.add(*states*.get(j));  
 *states*.remove(j);  
 break;  
 }  
 }  
 }  
 }  
  
 public static int SumPopulation()  
 {  
 int sum = 0;  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 sum += *states*.get(i).getpopulation();  
 }  
 return sum;  
 }  
  
 public static float ArmyPercentage(String name)  
 {  
 float percent = 1;  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 if(*states*.get(i).getname().equals(name))  
 {   
 percent = ((float) *states*.get(i).getarmy() / *states*.get(i).getpopulation());  
 }  
 }  
 return percent\*100;  
 }  
 public static float Difference\_in\_Happines\_index()  
 {  
 *Sorthappiness\_index*(true);  
 return *states*.get(*states*.size() - 1).gethappiness\_index() - *states*.get(0).gethappiness\_index();  
 }  
 public static void Predominance\_getform\_of\_government()  
 {  
 int count = 0;  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 if(*states*.get(i).getform\_of\_government().equals("Республика"))  
 {  
 count += 1;  
 }  
 else  
 {  
 count -= 1;  
 }  
 }  
 if(count > 0) System.*out*.println("Республик больше, чем монархических государств");  
 else if(count < 0) System.*out*.println("Монархических государств больше, чем республик");  
 else System.*out*.println("Республик и монархических государств поровну");  
 }  
 public static void Predominance\_gettype\_of\_monarchy()  
 {  
 int count = 0;  
 for(int i = 0; i < *states*.size(); i++)  
 {  
 if(*states*.get(i).gettype\_of\_monarchy().equals("; Тип монархии: Королевство"))  
 {  
 count += 1;  
 }  
 else  
 if(*states*.get(i).gettype\_of\_monarchy().equals("; Тип монархии: Султанат"))  
 {  
 count -= 1;  
 }  
 }  
 if(count > 0) System.*out*.println("Королевств больше, чем султанатов");  
 else if(count < 0) System.*out*.println("Cултанатов больше, чем Королевств");  
 else System.*out*.println("Королевств и султанатов поровну");  
 }  
  
}

Список использованных источников

1. Аршинский В.Л. Объектно-ориентированное программирование. Метод. указания по выполнению лабораторных работ / В.Л. Аршинский. – Иркутск: Изд-во ИРНИТУ, 2015. – 23 c.