­­­Санкт-Петербургский государственный политехнический университет

Факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

­­­

# Курсовая работа по курсу «Основы программирования на Java»

Тема: «Разработка приложения на языке Java».

Студентка группы 1081/3 Никитина Анна

Преподаватель Пышкин Е.В.

Санкт-Петербург 2012

# Образ и границы проекта.

Целью данного проекта является проведение сравнительной характеристики команд-стран на Олимпийских играх.

На входе – несколько документов (а именно столько, сколько дней длилась Олимпиада), каждый из которых будет содержать информацию о конкретном дне. Данную информацию следует считать и посчитать на ее основе количество медалей у определенной команды в каждый день Олимпиады.

На выходе для каждой страны должен быть представлен график, показывающий набранное количество медалей в каждый день. Причем график должен строится по нарастающей, то есть от меньшего количества медалей к большему. Также на выходе должны быть представлены таблицы, отображающие то, что записано во входных файлах.

Тем самым будет проведена сравнительная характеристика результатов в выступлениях стран на Олимпийских играх, графики будут наглядно показывать разницу в набранном количестве медалей.

# Техническое задание.

Входные данные представлены в нескольких документах, так как для каждого дня существует отдельный документ. Он представляет собой некоторое количество строк, каждая строка состоит из следующего:

* Дисциплина
* 1
* Страна, занявшая первое место по данной дисциплине
* 2
* Страна, занявшая второе место по данной дисциплине
* 3
* Страна, занявшая третее место по данной дисциплине

Например, одна из строк будет выглядеть следующим образом:

volleyball 1 Russia 2 China 3 USA .

Входные данные обрабатываются следующим образом: создаются классы с контейнерами, куда записываются данные, считанные из файлов. На их основе считается общее количество набранных медалей в каждый из дней, после чего для любой страны возможно построение графика, показывающего прирост количества медалей за все дни Олимпиады.

Так же следует заметить, что программа разделена на три части: первая отвечает за считывание входных данным, она работает только с этим типом данных. Вторая часть программы реализует обработку данных и создание собственных классов. Третяя часть отвечает за графическое представление обработанных данных: создание главного окна, построение таблиц, графиков.

Построив программу таким образом мы добились того, что при изменении формата входных данных, можно легко деформировать программу, изменив лишь небольшую ее часть.

## Основной сценарий функционирования приложения.

Программа представлена в виде цикла, продолжающегося до тех пор, пока есть документы с соответствующими названиями.

Первоначально открывается документ под названием “1.txt”, то есть документ, соответствующий первому дню.

1. Считывается дисциплина. Если такая дисциплина еще не встречалась, вызывается конструктор, который создает нужную дисциплину, после чего она записывается в общую коллекцию дисциплин.

Цикл, продолжающийся до того, пока строка не закончится.

1. Считывается цифра(1,2 или3)
2. Считывается команда. Если такая команда еще не встречалась, вызывается конструктор, который создает нужную команду по названию, после чего она записывается в общую коллекцию команд. Далее в зависимости от того, после какой цифры записана команда, в ее количество золотых, серебряных или бронзовых медалей добавляется еще одна.

Далее к названию документа прибавляется 1, то есть следующий документ будет называться “2.txt” и цикл повторяется вновь.

## Требования к графическому пользовательскому интерфейсу.

Экран разделен на 3 панели.

1. Заголовочная панель-это просто панель с заголовком «Олимпийские игры».
2. Панель опций, данная панель предназначена для выбора пользователем того, что он хочет сделать. А именно, имеется опция «Построить таблицу» с выпадающим списком, в котором указаны все дни, представленные во входных файлах. Следующая опция - построить график, ниже кнопки выбора стран, страны представлены так же только те, информация о которых имеется во входных файлах. Кнопка «Выход», предназначенная для выхода из приложения.
3. Основная панель. Она разделена на две области, в верхней ее части строится таблица по дню, который выбрал пользователь. В таблице указаны дисциплины, а так же страны, занявшие первое, второе и третее места по данной дисциплине. Нижняя область первоначально содержит только координатные оси, когда пользователь выбрал какую-либо страну на панели опции, строится график для данной страны, который показывает количество набранных медалей данной страной в порядке их увеличения. Цвет линии генерируется автоматически, справа сверху указывается страна с соответствующим ей цветом. Если пользователь выберет еще страны, на панель добавятся соответствующие им графики, что позволяет сравнивать то, как выступают страны на соревнованиях.

## Системная архитектура.

Все классы, используемые в программе можно разделить на три группы.

**Классы, отвечающие за обработку информации.**

1. Класс “Disciplines” содержит в себе конструктор, который по строке создает дисциплину с соответствующим названием, имеется метод, сравнивающий две дисциплины.
2. Класс “LineOfDiscipline” содержит HashSet из дисциплин, метод, добавляющий дисциплину в HashSet, если там ее еще нет.
3. Класс “ TeamForOneDay” содержит номер дня и три ArrayList, в первом из них указаны дисциплины, по которым команда «взяла» золото, во втором-серебро, в третем-бронзу.

public class TeamForOneDay {

int Day;

public ArrayList<Disciplines> GoldMedals=new ArrayList<Disciplines>();

public ArrayList<Disciplines> SilverMedals=new ArrayList<Disciplines>();

public ArrayList<Disciplines> CopperMedals=new ArrayList<Disciplines>();

/\*\*Добавление новой дисциплины в контейнер с нужным видом медали\*/

public void addMedalForOneDay(Disciplines d,Team.Price Price){

if(Price==Team.Price.GOLD)

this.GoldMedals.add(d);

if(Price==Team.Price.SILVER)

this.SilverMedals.add(d);

if(Price==Team.Price.COPPER)

this.CopperMedals.add(d);

}

/\*\*Функция, выводящая количество ВСЕХ видов медалей за один день по всем видам дисциплин\*/

public int getMedalForOneDay(){

int value=getMedalForOneDay(Team.Price.GOLD)+

getMedalForOneDay(Team.Price.SILVER)+

getMedalForOneDay(Team.Price.COPPER);

return value;

}

/\*\*Функция, выводящая количество медалей за один день по определенной дициплине

\*аргументы-дисциплина,номинал медали\*/

public int getMedalForOneDisciplineForOneDay(Disciplines d,Team.Price Price){

int value=0;

switch(Price){

case GOLD:

Iterator<Disciplines> it1=GoldMedals.iterator();

while(it1.hasNext()){

Disciplines discipline=it1.next();

if(discipline.equals(d)==true)

value++;

}

break;

case SILVER:

Iterator<Disciplines> it2=SilverMedals.iterator();

while(it2.hasNext()){

Disciplines discipline=it2.next();

if(discipline.equals(d)==true)

value++;

}

break;

case COPPER:

Iterator<Disciplines> it3=CopperMedals.iterator();

while(it3.hasNext()){

Disciplines discipline=it3.next();

if(discipline.equals(d)==true)

value++;

}

break;

}

return value;

}

/\*\*Функция, выводящая количество ВСЕХ видов медалей за один день по определенной дициплине

\*аргумент-дисциплина\*/

public int getMedalForOneDisciplineForOneDay(Disciplines d){

int value=getMedalForOneDisciplineForOneDay(d,Team.Price.GOLD)+

getMedalForOneDisciplineForOneDay(d,Team.Price.SILVER)+

getMedalForOneDisciplineForOneDay(d,Team.Price.COPPER);

return value;

}

}

1. Класс “ Team” содержит имя команды, цвет, которым она будет обозначаться на графике, HashSet для всех дней из TeamForOneDay, перечисление Price(GOLD,SILVER, COPPER,ALL), что означает номинал медали- золотые, серебряные , бронзовые медали или их сумма. Содержит метод, автоматически генерирующий цвет, методы, выводящие количество медалей за определенные дни по определенным дисциплинам.

**Классы, отвечающие за считывание информации.**

1.Класс “Reader” считывает информацию из отдельного документа и записывает в готовые имеющиеся контейнеры. Основной метод:

public void readerFile(int Day) throws IOException,NoSuchElementException,NullPointerException{

String[] tblheader;

Scanner in1;Scanner in;Object[][] tbldata;

Team team1=new Team();Team team2=new Team();Team team3=new Team();

TeamForOneDay t=new TeamForOneDay(Day);Disciplines d=new Disciplines();

try { this.fileOpen(Day);

}catch(FileNotFoundException ex){

}

try{in1=new Scanner(this.file);

while(in1.hasNextLine()){in1.nextLine();size++;}

tbldata =new Object[size][];

in=new Scanner(this.file);

int value=0;

while(in.hasNextLine()){

if(in.hasNext()){ d=new Disciplines(in.next(),in.next(),in.next());

while(in.hasNextInt()){if(!in.hasNextInt()){throw new NoSuchElementException();}

boolean coincidence=false;

switch(in.nextInt()){

case 1:

team1=new Team(in.next());

t.addMedalForOneDay(d, Team.Price.GOLD);

for(Team team:this.LineOfTeam){if(team1.compare(team)){

team.add(t);coincidence=true;team.setNumberOfMedals();

}}if(coincidence==false){team1.add(t); this.LineOfTeam.add(team1);team1.setNumberOfMedals();}

break;

case 2:

team2=new Team(in.next());

t.addMedalForOneDay(d, Team.Price.SILVER);

for(Team team:this.LineOfTeam){if(team2.compare(team)){

team.add(t); coincidence=true;

}}if(coincidence==false){team2.add(t); this.LineOfTeam.add(team2);}

break;

case 3:

team3=new Team(in.next());

t.addMedalForOneDay(d, Team.Price.COPPER);

for(Team team:this.LineOfTeam){if(team3.compare(team)){

team.add(t);coincidence=true;

}}if(coincidence==false){team3.add(t); this.LineOfTeam.add(team3);}

break;

}

}

if(value<size){tbldata[value]=new Object[] {d.Name, team1.NameOfTeam, team2.NameOfTeam, team3.NameOfTeam};

value++;}

}

in.nextLine();

}

if(value<size){throw new NullPointerException();}

tblheader = new String[] { "Discipline", "Gold", "Silver", "Copper" };

tbl = new JTable(tbldata, tblheader);

}catch(FileNotFoundException ex){

}}

public void readerFileForGrafic(int Day) throws IOException,NoSuchElementException,NullPointerException{

Scanner in;

Team team1=new Team();Disciplines d=new Disciplines();

try { this.fileOpen(Day);

}catch(FileNotFoundException ex){

}

in=new Scanner(this.file);

while(in.hasNextLine()){

if(in.hasNext()){ d=new Disciplines(in.next(),in.next(),in.next());

while(in.hasNextInt()){if(!in.hasNextInt()){throw new NoSuchElementException();}

boolean coincidence=false;in.nextInt();

team1=new Team(in.next());

for(Team team:this.LineOfTeam){if(team1.compare(team)){

coincidence=true;team.setNumberOfMedals();

}}if(coincidence==false){this.LineOfTeam.add(team1);team1.setNumberOfMedals();}

}

in.nextLine();

}}}

2.Класс “ReaderAllFile” отвечает за открытие и считывание всех входных файлов, содержит контейнеры в которых хранятся списки всех дисциплин и стран, встречающихся во входных файлах

**Классы, отвечающие за графическое представление.**

1.Класс “GPanel” является наследником JPanel, отвечает за отрисовку графиков и координатных осей. Основная функция отрисовки:

public void paintGrafic(Graphics g){Graphics2D g2 = (Graphics2D)g;

BasicStroke pen1 ;

pen1=new BasicStroke(3);//толщина линии 3

g2.setStroke(pen1);

for(Team team1:LineOfChooseTeam){

g2.setColor(team1.Color);

int PreviousCoordY=getHeight()-CoordY();

r.SortNumberOfMedals(team1);

for(int i=1;i<r.GetNumberOfDay();i++){

if(r.LineNumberOfMedals.size()!=r.GetNumberOfDay()-i){

g2.drawLine(CoordX()+(i-1)\*getSpaceForCutoffX(),getHeight()-CoordY(),

CoordX()+i\*getSpaceForCutoffX(),getHeight()-CoordY());

}

else{

Integer number=r.MinNumberOfMedals();

g2.drawLine(CoordX()+(i-1)\*getSpaceForCutoffX(),PreviousCoordY,

CoordX()+(i-1)\*getSpaceForCutoffX(),getHeight()-CoordY()-number\*getSpaceForCutoffY());

g2.drawLine(CoordX()+(i-1)\*getSpaceForCutoffX(),getHeight()-CoordY()-number\*getSpaceForCutoffY(),

CoordX()+i\*getSpaceForCutoffX(),getHeight()-CoordY()-number\*getSpaceForCutoffY());

PreviousCoordY=getHeight()-CoordY()-number\*getSpaceForCutoffY();r.LineNumberOfMedals.remove(number);

r.LineNumberOfMedals.remove(number);

}}}}

public void paintOsi(Graphics g){

Graphics2D g2 = (Graphics2D)g;

g2.setColor(Color.BLACK);

BasicStroke pen1 ;

pen1=new BasicStroke(3);//толщина линии 3

g2.setStroke(pen1);

Font font = new Font ("Serif", Font.BOLD, 18);g2.setFont(font);

g2.drawString("Дни",getWidth()-CoordX()-30,getHeight()-CoordY()-10);

g2.drawString("Медали",CoordX()+10,CoordY()+5);

//Рисуем две прямые линии

g2.drawLine(CoordX(),CoordY(),CoordX(), getHeight()-CoordY());

g2.drawLine(CoordX(), getHeight()-CoordY(),getWidth()-CoordX(),getHeight()-CoordY());

//Рисуем две стрелки

g2.drawLine(CoordX(),CoordY(),

CoordX()+CoordOfMarkX(),CoordY()+CoordOfMarkY());

g2.drawLine(CoordX(),CoordY(),

CoordX()-CoordOfMarkX(),CoordY()+CoordOfMarkY());

g2.drawLine(getWidth()-CoordX(),getHeight()-CoordY(),

getWidth()-CoordX()-CoordOfMarkY(),getHeight()-CoordY()+CoordOfMarkX());

g2.drawLine(getWidth()-CoordX(),getHeight()-CoordY(),

getWidth()-CoordX()-CoordOfMarkY(),getHeight()-CoordY()-CoordOfMarkX());

//Меняем толщину линии и рисуем отсечки по оси X и по оси Y

pen1=new BasicStroke(2);//толщина линии 2

g2.setStroke(pen1);

for(int i=1;i<MaxNumberOfMedals;i++){g2.drawLine(CoordX()-SizeOfCutoff/2,getHeight()-CoordY()-i\*getSpaceForCutoffY(),

this.CoordX()+SizeOfCutoff/2,getHeight()-CoordY()-i\*getSpaceForCutoffY());}

for(int i=1;i<r.GetNumberOfDay();i++){g2.drawLine(CoordX()+i\*getSpaceForCutoffX(),getHeight()-CoordY()-SizeOfCutoff/2,

this.CoordX()+i\*getSpaceForCutoffX(),getHeight()-CoordY()+SizeOfCutoff/2);

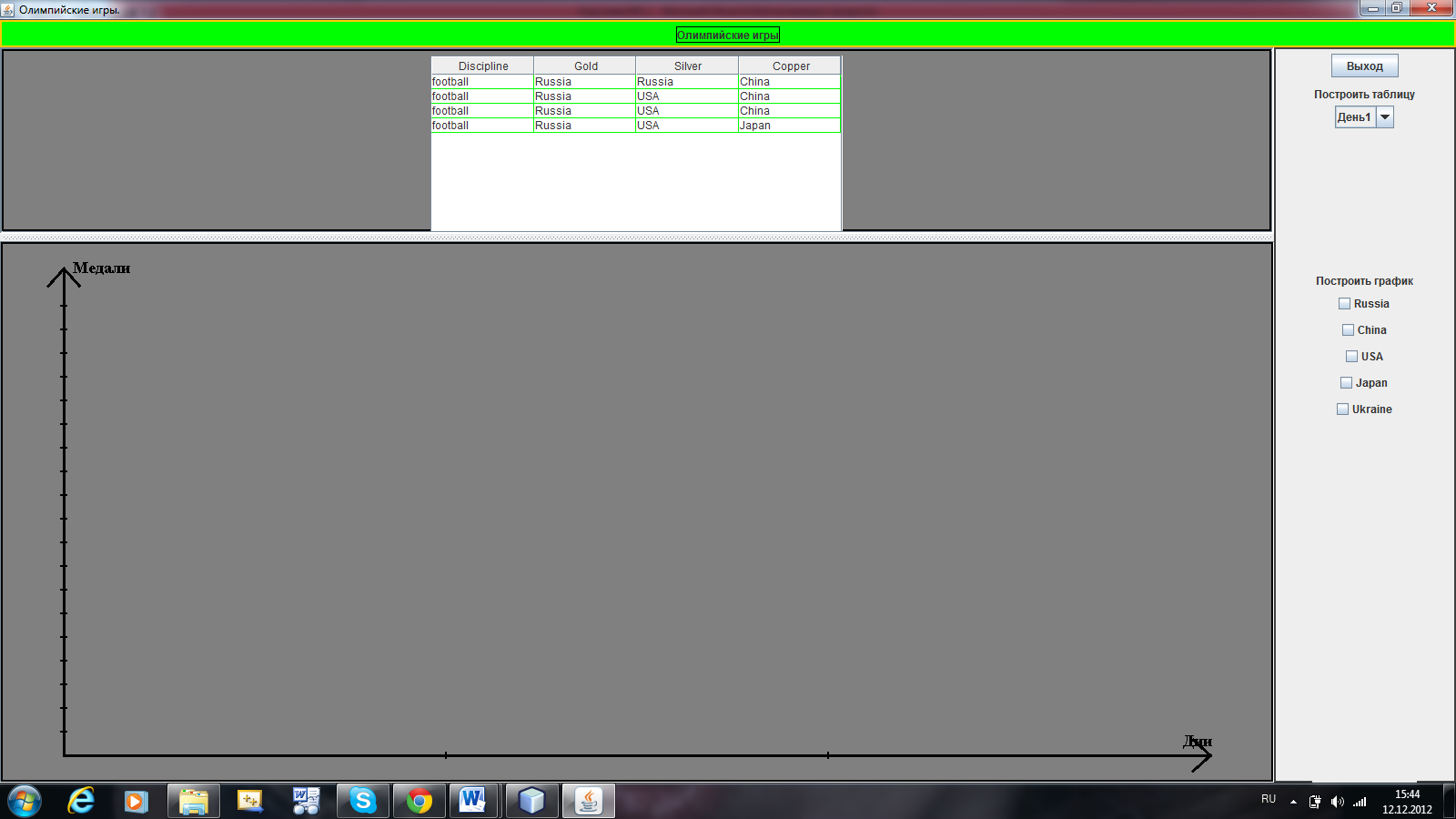
} }

2. Класс “ MainPanel ” является наследником JPanel, отвечает за отрисовку таблицы.

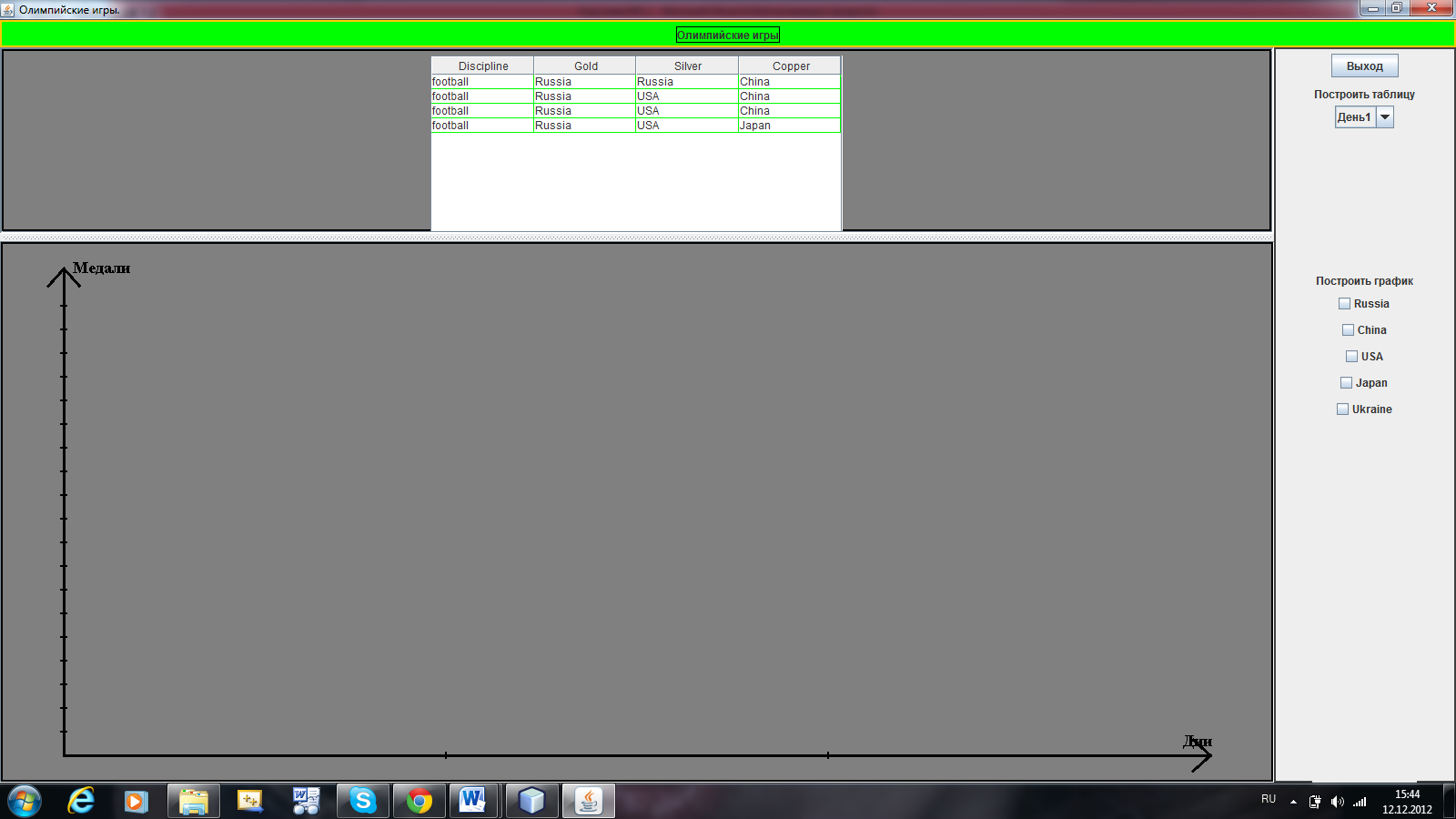
3.Класс “MainFrame”, именно он непосредственно создает вышеперечисленные панели, является главным классом программы.

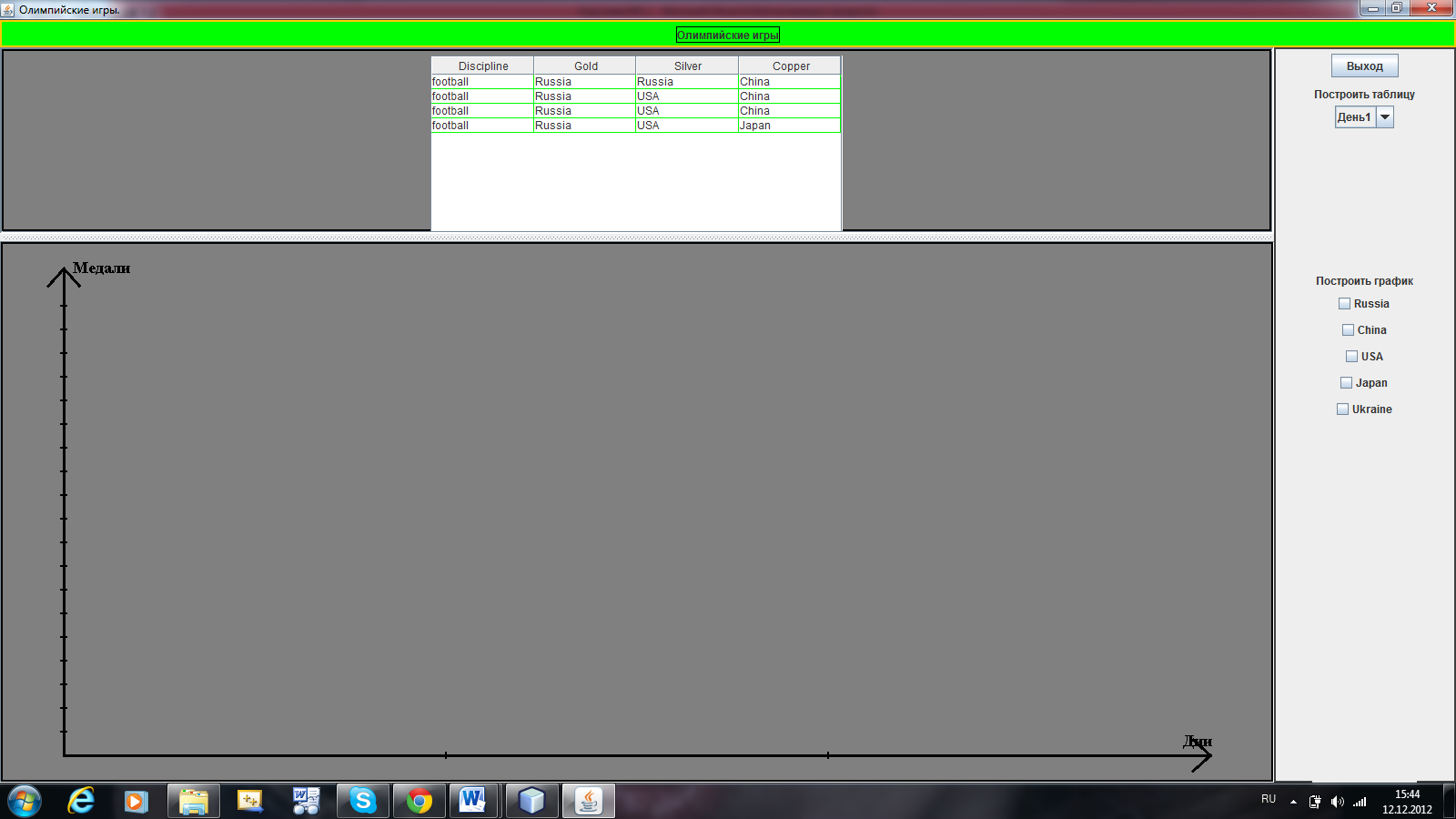
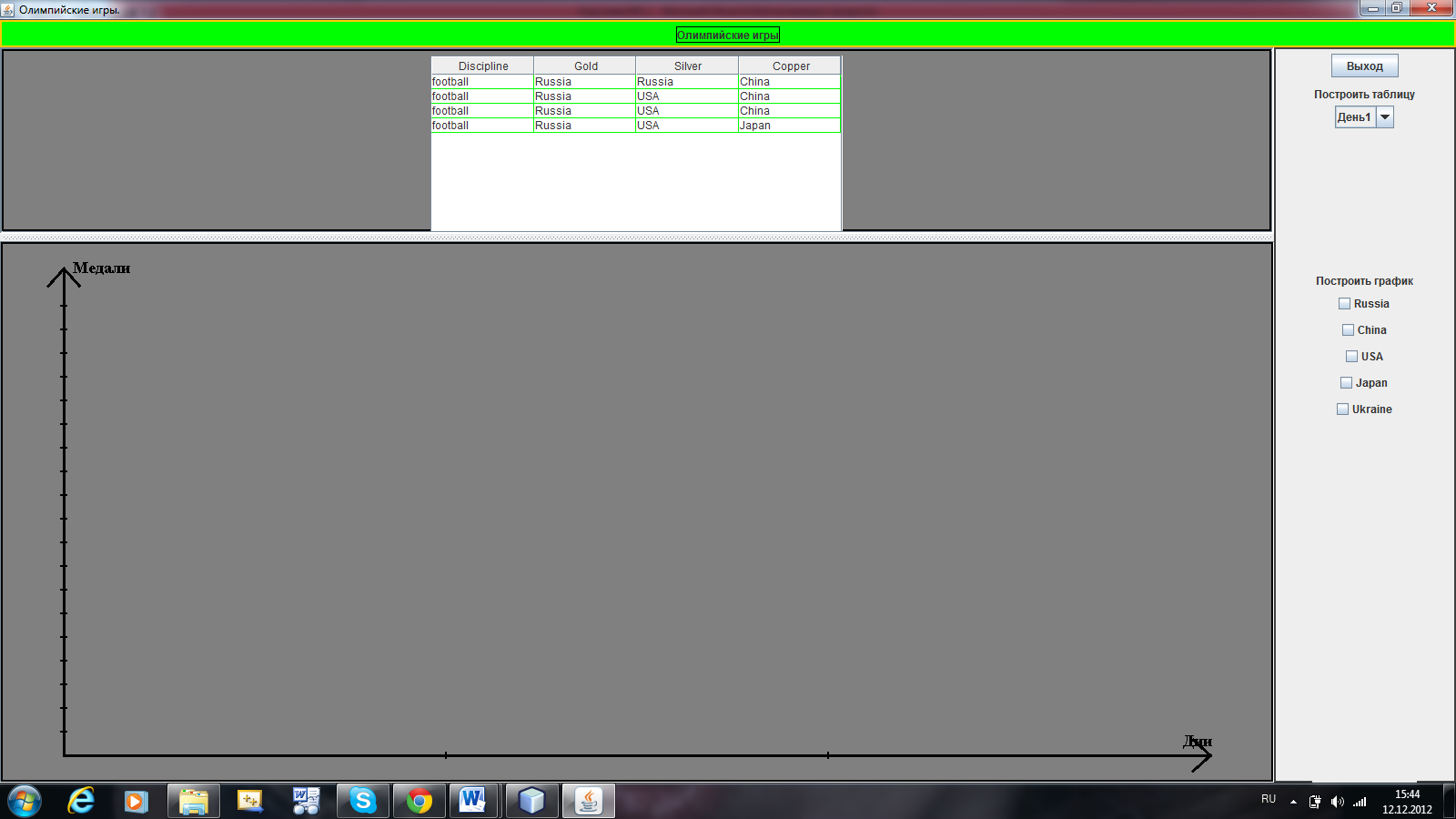
## Графический интерфейс.

Взаимодействие программы с пользователем происходит при помощи графического интерфейса. Первоначально внешний вид программы:

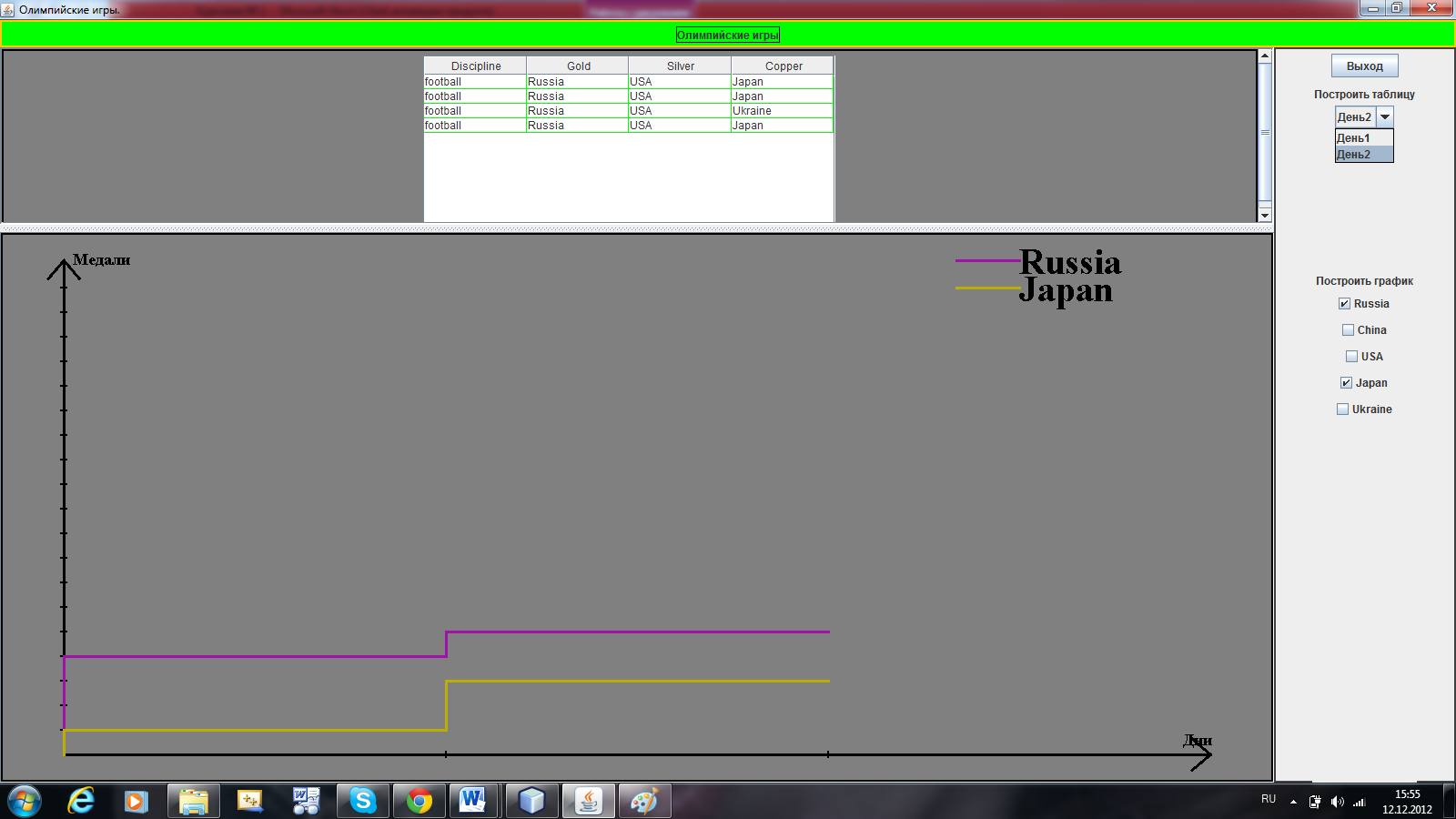


Заголовочная панель:

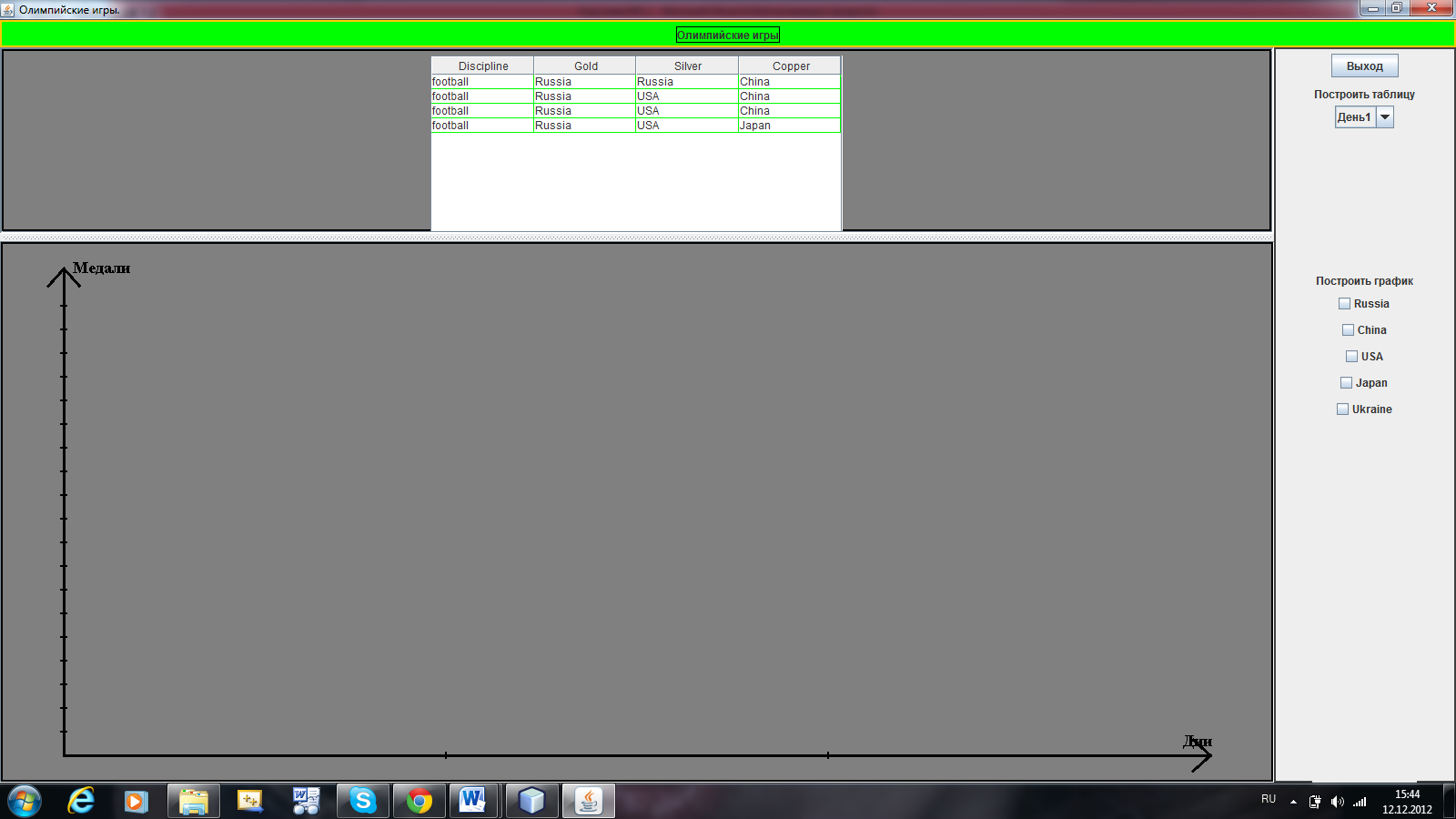
Панель опций: Верхняя область основной панели (таблица):

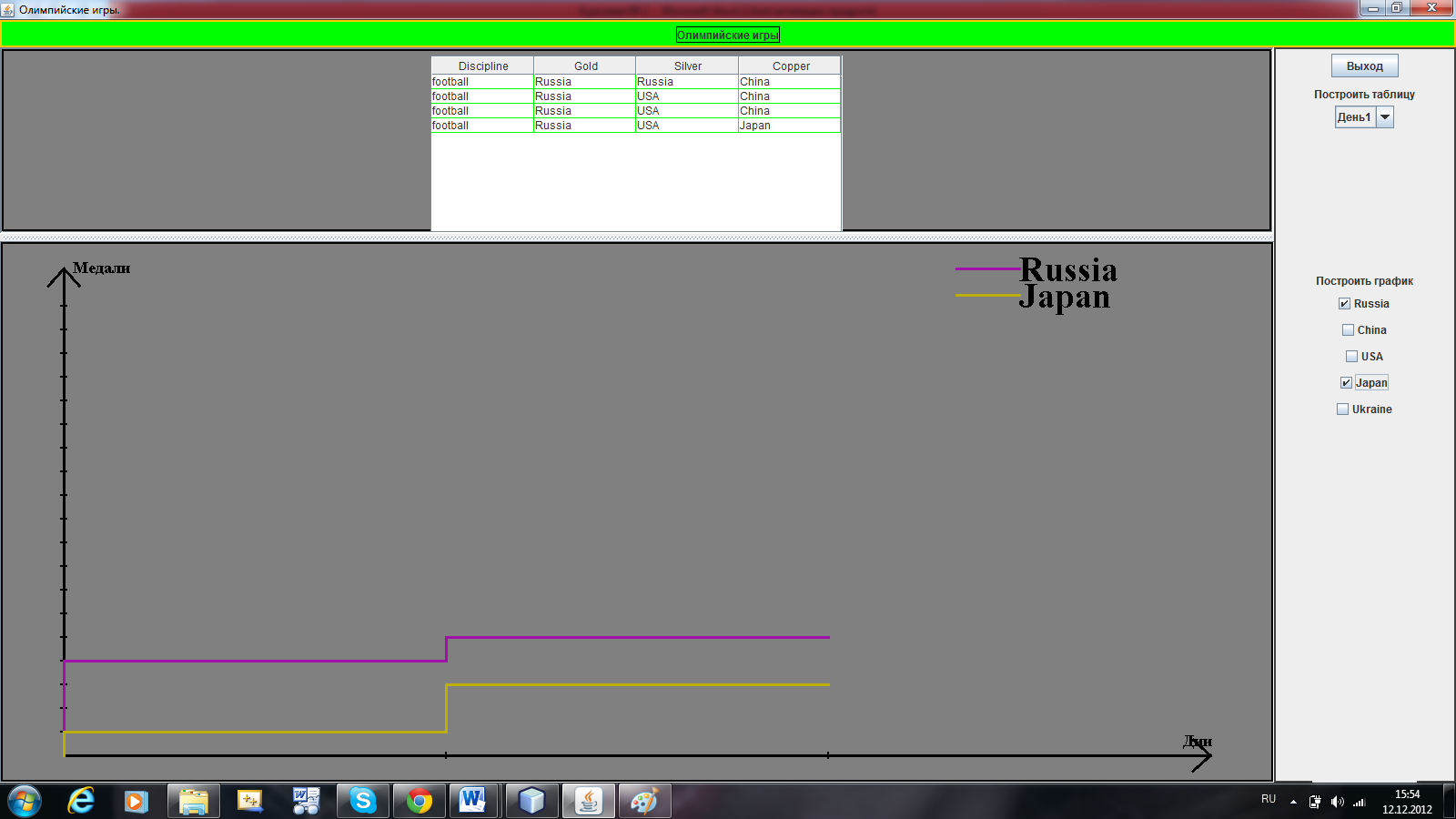
Первоначально таблица строится по первому дню. Если в разделе «Построить таблицу» выбрать другой день, используя Combobox, таблица будет построена по выбранному дню.



Нижняя область основной панели(первоначально координатная ось):



Если пользователь выберет из имеющихся стран несколько, то эта область будет выглядеть следующим образом:



Для выхода из приложения пользователю следуют нажить “крестик” или кнопку 