|  |  |
| --- | --- |
| ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  Институт математики и информационных систем  Факультет автоматики и вычислительной техники  Кафедра систем автоматизации управления | |
| «ЗНАКОМСТВО С ЯЗЫКОМ ПРОГРАММИРОВАНИЯ JAVA»  Отчет по самостоятельной работе №1  по дисциплине  «Основы интернет и интернет-девелопмента»  Вариант 4 | |
|  | Выполнил: |
|  | студент гр. ИТб-2301 |
|  | Двоеглазов Н.А. |
|  | Проверил: |
|  | Земцов М.А. |
| Киров  2020 | |

**Цель работы:** изучение базовых основ программирования на языке Java и конфигурирование среды разработки NetBeans.

1. Установите OpenJDK или Oracle JDK.
2. Установите NetBeans.
3. Запустите NetBeans IDE: ярлык на рабочем столе или Пуск → Про-граммы → NetBeans → NetBeans IDE.

4. Выберите пункт меню Файл → Создать проект.

5. Выберите пункт Java (Категории) → Библиотека классов Java (Про-екты).

6. Укажите имя проекта и путь. Нажмите на кнопку «Готово».

7. Перед вами откроется среда визуальной разработки приложений

8. Создайте пакет для типов и назовите его, например, types.

9. Создайте класс, например, Tovar.

10. Определите необходимый набор полей.

11. Добавьте конструктор без параметров: правая кнопка мыши → «Вставка кода…» → Конструктор → Создать.

12. В полученном методе определите все значения полей в значения по умолчанию:

13. Добавить конструктор с параметрами: правая кнопка мыши → «Вставка кода…» → Конструктор → отметить все поля → Создать.

14. В результате сгенерируется программный код.

15. Добавьте методы получения и установки значений полей: Добавьте конструктор с параметрами: правая кнопка мыши → «Вставка кода…» → Методы получения и установки → отметить все поля → Создать.

16. Сформируется следующий программный код:

Результаты выполнения пунктов 9-16 соответствуют листингу 1.

Листинг 1:

package types;

public class Tovar {

private String name;

private int kol;

private double price;

public Tovar() {

name = "";

kol = 0;

price = 0;

}

public Tovar(String name, int kol, double price) {

this.name = name;

this.kol = kol;

this.price = price;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getKol() {

return kol;

}

public void setKol(int kol) {

this.kol = kol;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

}

17. Создайте пакеты: operation – для классов обработки, base – для класса формы.

Результат пункта 17 соответствует рисунку 1.

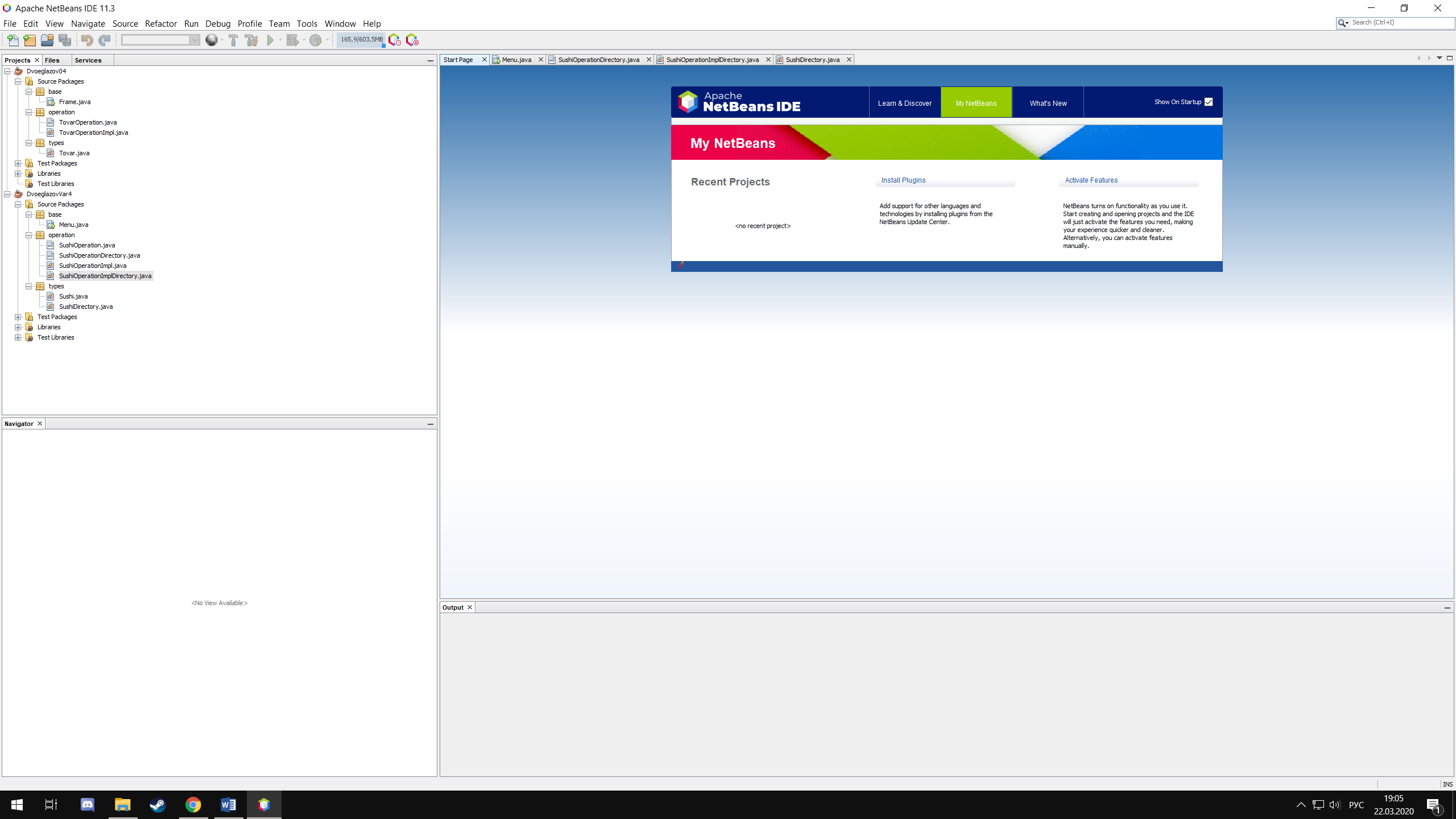


Рисунок 1

18. Добавьте в пакет operation interface TovarOperation.

Результат пункта 18 соответствует листингу 2.

Листинг 2:

package operation;

import java.util.List;

import types.Tovar;

public interface ITovarOperation {

List<Tovar> getListOfTovar();

List<Tovar> addNewTovar(Tovar item);

int getSumOfTovar();

}

19. Добавьте в пакет operation class TovarOperation.

Результат пункта 19 соответствует листингу 3.

Листинг 3:

package operation;

import java.util.ArrayList;

import java.util.List;

import types.Tovar;

public class TovarOperation implements ITovarOperation {

static List<Tovar> lstTovar = new ArrayList<>();

static{

lstTovar.add(new Tovar("Товар1",10,100));

lstTovar.add(new Tovar("Товар2",20,200));

lstTovar.add(new Tovar("Товар3",30,300));

lstTovar.add(new Tovar("Товар4",40,400));

}

@Override

public List<Tovar> getListOfTovar(){

return lstTovar;

}

@Override

public List<Tovar> addNewTovar(Tovar item){

lstTovar.add(item);

return lstTovar;

}

@Override

public int getSumOfTovar(){

int s = 0;

for(Tovar tmp : lstTovar){

s += tmp.getKol() \* tmp.getPrice();

}

return s;

}

}

20. Создайте форму (JFrame) в пакете base и поместите на форму объекты в соответствии с эскизом формы и таблицей свойств объектов.

Результат пункта 20 соответствует рисункам 4 и 5.

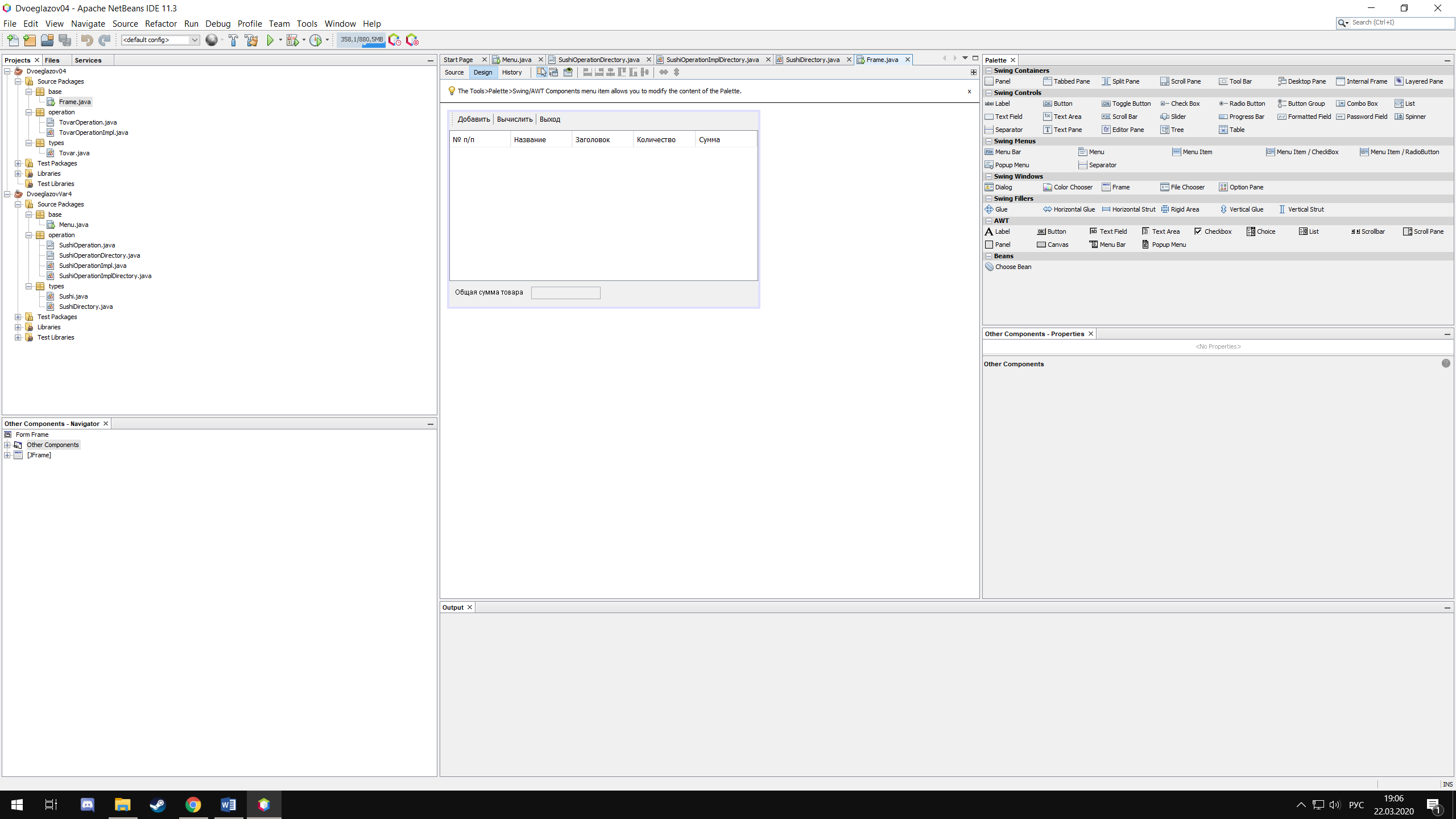


Рисунок 4

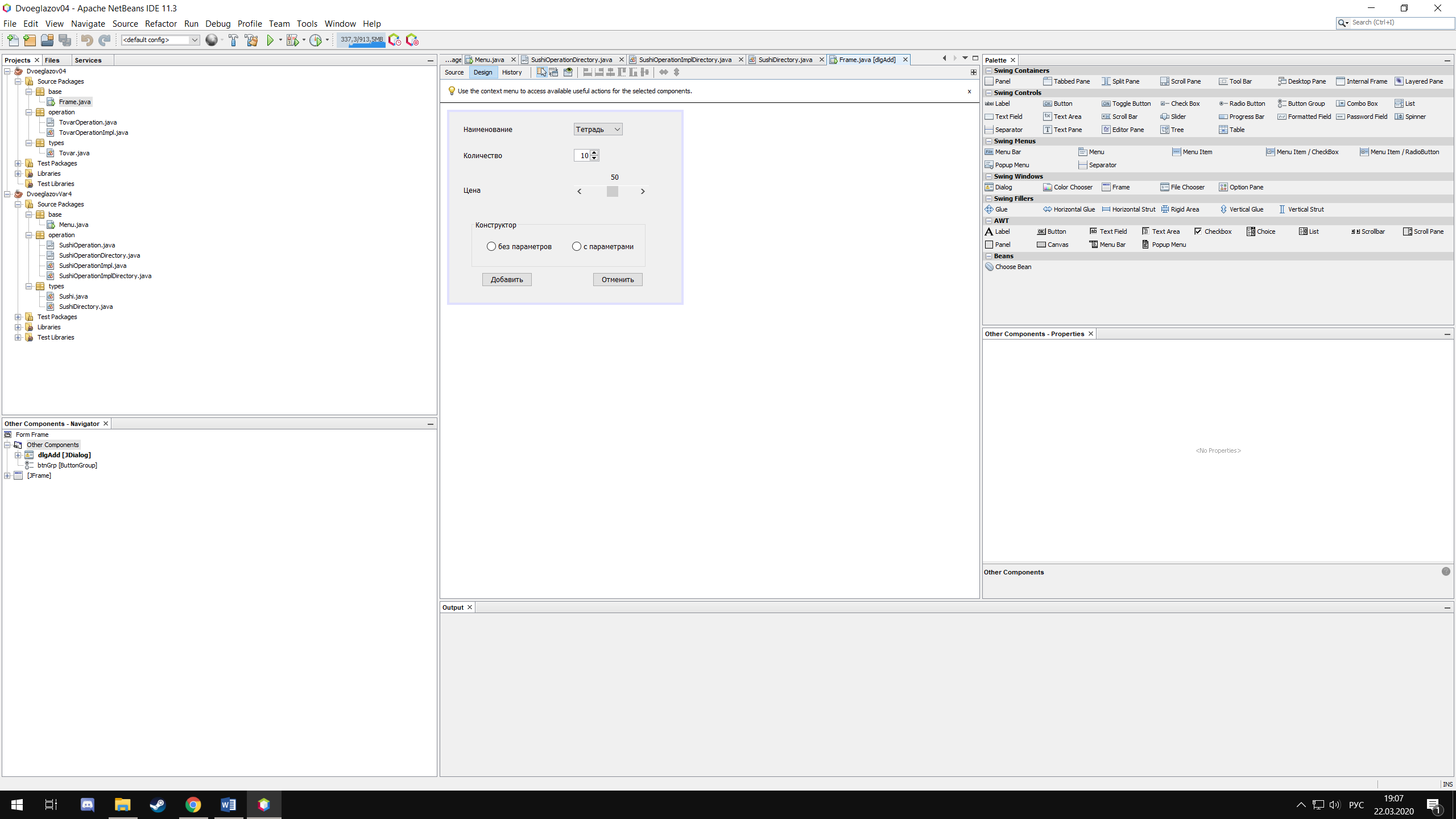


Рисунок 5

21. Структура проекта должна быть следующей.

Структура проекта соответствует рисунку 1.

22. Программный по файлам.

Программный код соответствует листингам 1 и 4.

Листинг 4 – класс form:

private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

dlgAdd.setSize(300, 320);

dlgAdd.setVisible(true);

}

private void btnDecideActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

txtTotalSumma.setText(Integer.toString(tovarOperation.getSumOfTovar()));

}

private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

setDefaultCloseOperation(this.EXIT\_ON\_CLOSE);

System.exit(0);

}

private void btnAddTovarActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

dlgAdd.setVisible(false);

// д.б. код для проверки полей на правильность ввода

Tovar el;

if (rdbYesParam.isSelected()) {

el = new Tovar(cmbName.getSelectedItem().toString(), (int) spnKol.getValue(), scrPrice.getValue());

} else {

el = new Tovar();

el.setName(cmbName.getSelectedItem().toString());

el.setKol((int) spnKol.getValue());

el.setPrice(scrPrice.getValue());

}

doVivod(tovarOperation.addNewTovar(el));

}

private void btnCancelActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

dlgAdd.setVisible(false);

}

private void scrPriceAdjustmentValueChanged(java.awt.event.AdjustmentEvent evt) {

// TODO add your handling code here:

lblPriceInfo.setText(Integer.toString((int) scrPrice.getValue()));

}

private void formWindowOpened(java.awt.event.WindowEvent evt) {

// TODO add your handling code here:

model = (DefaultTableModel)tbl.getModel();

doVivod(tovarOperation.getListOfTovar());

}

TovarOperation tovarOperation = new TovarOperation();

static DefaultTableModel model = new DefaultTableModel();

private void doVivod(List<Tovar> lstTovar){

doClearTable();

int i = 1;

for(Tovar tovar: lstTovar){

Object[] rowData = new Object[5];

rowData[0] = i++;

rowData[1] = tovar.getName();

rowData[2] = tovar.getPrice();

rowData[3] = tovar.getKol();

rowData[4] = tovar.getPrice() \* tovar.getKol();

model.addRow(rowData);

}

}

private void doClearTable(){

while (model.getRowCount()>0){

model.removeRow(0);

}

}

23. Протестируйте работу полученного приложения.

Результаты тестирования соответствуют рисункам 6, 7, 8.

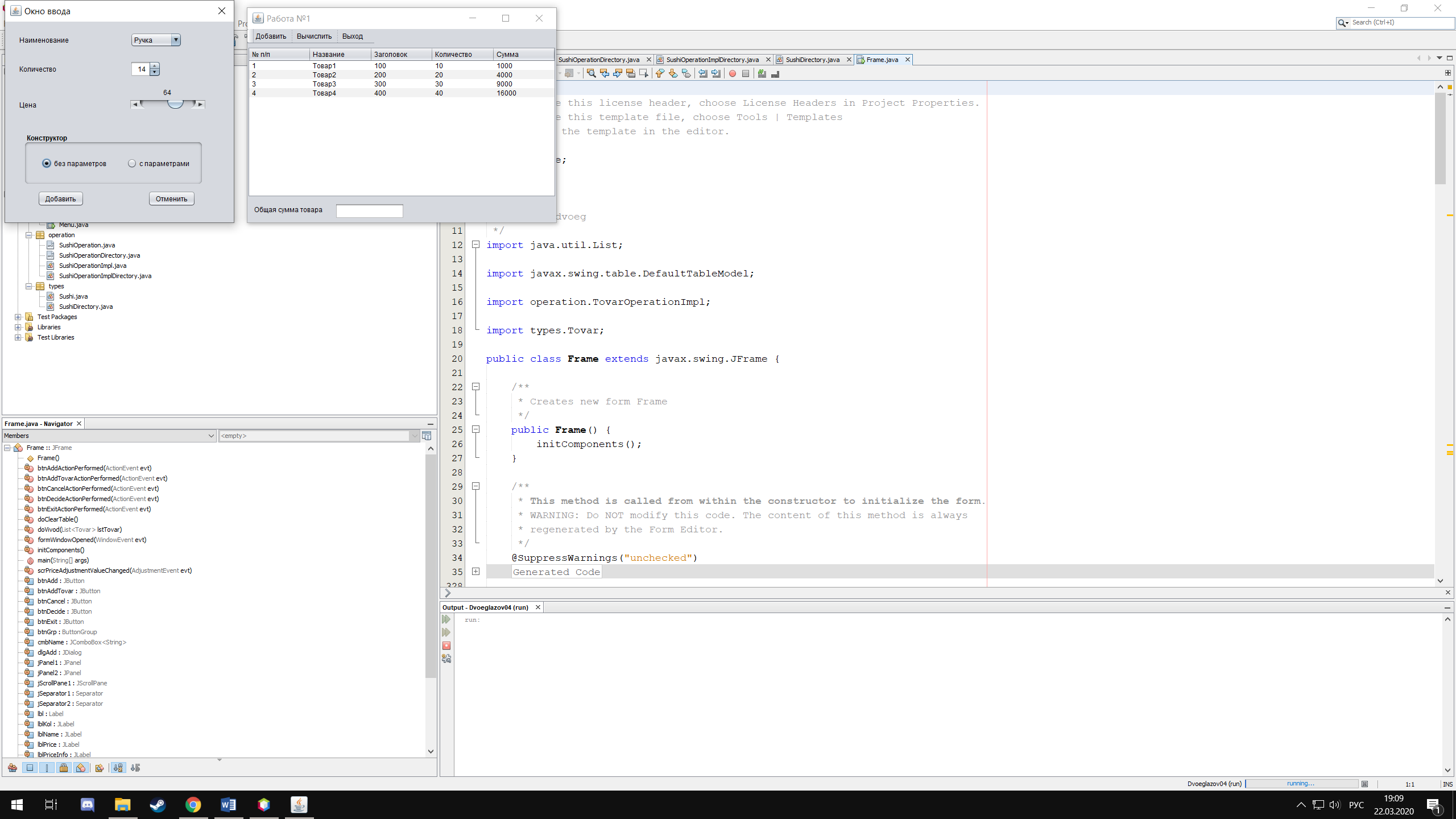


Рисунок 6

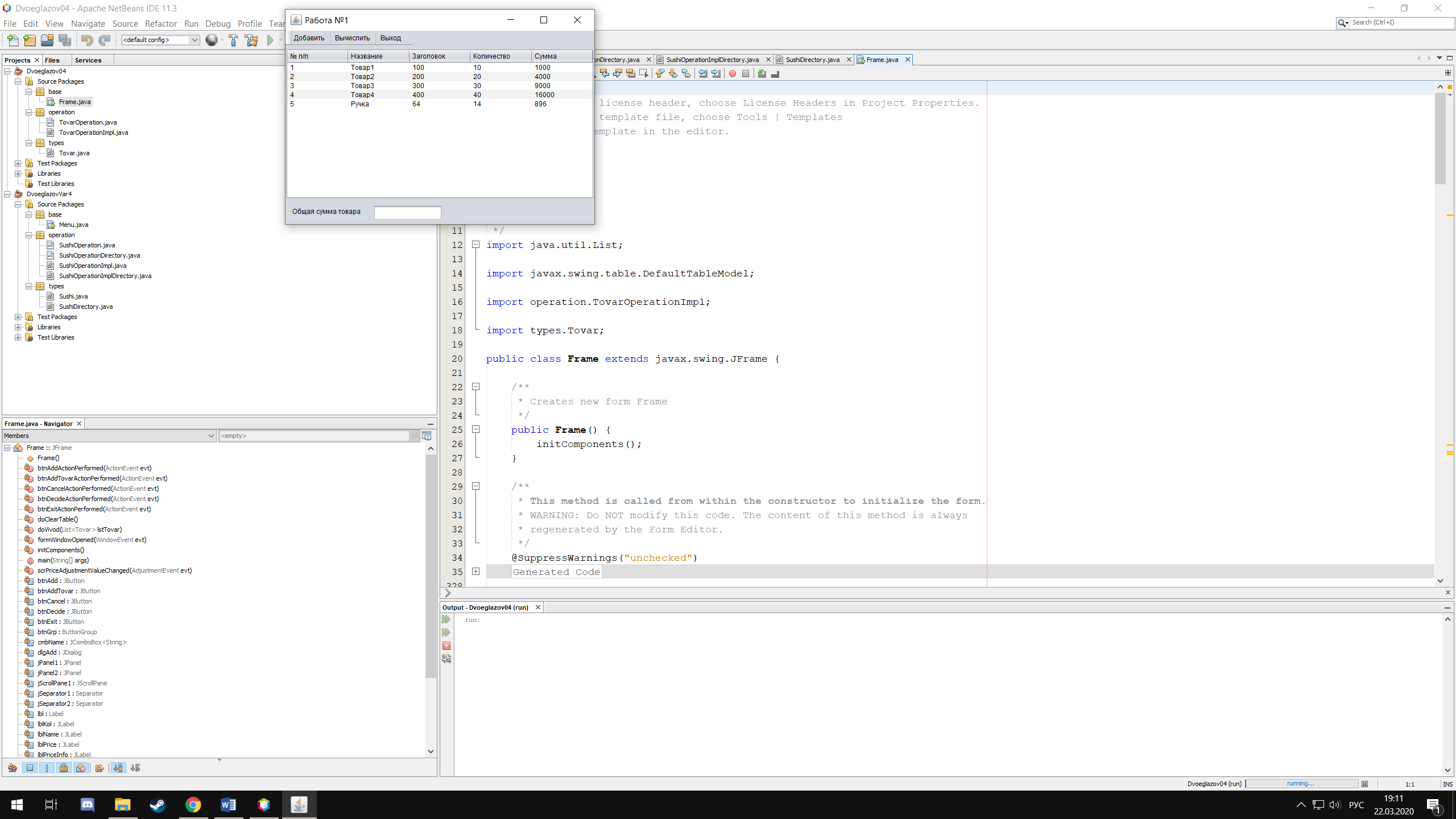


Рисунок 7

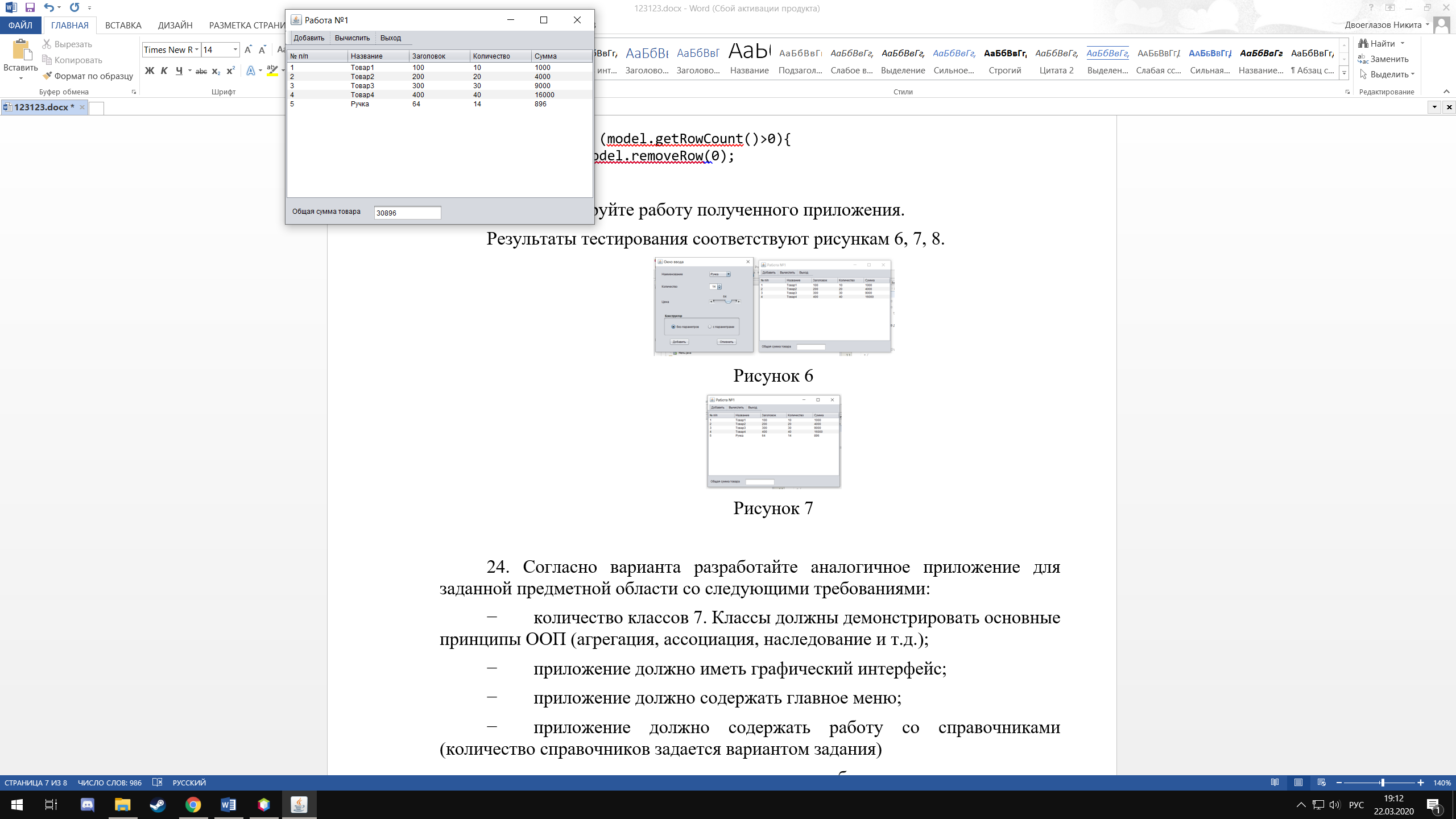


Рисунок 8

24. Согласно варианта разработайте аналогичное приложение для заданной предметной области со следующими требованиями:

− количество классов 7. Классы должны демонстрировать основные принципы ООП (агрегация, ассоциация, наследование и т.д.);

− приложение должно иметь графический интерфейс;

− приложение должно содержать главное меню;

− приложение должно содержать работу со справочниками (количество справочников задается вариантом задания)

− приложение должно реализовывать работу с основными дынными согласно варианта задания. Данные визуализируются в таблицу которая должна иметь поддержку следующих операций: добавление, удаление, получение всего списка, пересчет какого-нибудь вычисляемого поля.

|  |  |  |
| --- | --- | --- |
| 4 | Производство и доставка азиатской кухни (суши и роллы) | Меню, ингредиенты, сроки годности, рецепты, КЛАДР |

Структура приложения соответствует рисунку 8.

GUI соответствует рисункам 9-10.

Некоторые принципы ООП, реализованные в приложении, соответствуют листингам 5-7.

Листинг 5 – инкапсуляция:

public class Sushi {

private String name;

private int quantity;

private int weight;

private int price;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public int getWeight() {

return weight;

}

public void setWeight(int weight) {

this.weight = weight;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

Листинг 6 – полиморфизм:

@Override

public List<SushiDirectory> getListOfSushiDirectory() {

return lstSushiDirectory;

}

Листинг 7 – работа с интерфейсами:

public interface SushiOperationDirectory {

List<SushiDirectory> getListOfSushiDirectory();

List<SushiDirectory> addNewSushiDirectory(SushiDirectory item);

}

public class SushiOperationImplDirectory implements SushiOperationDirectory

**Вывод:** в ходе самостоятельной работы были получены базовые навыки программирования приложений с графическим пользовательским интерфейсом на языке Java с использованием интегрированный среды разработки NetBeans.