МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения информационных систем и технологий

**КУРСОВАЯ РАБОТА**

по дисциплине: “Программирование на языке Java”

на тему: **“Расчет подоходного налога”**

Выполнил**:** студент группы 10702121

Алюнина П. А.

Руководитель**:** доцент Сидорик В.В.

Минск 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по дисциплине: “Программирование на языке Java”

на тему: **“ Расчет подоходного налога”**

Выполнил**:** студент группы 10702121

Алюнина П. А.

Руководитель**:** доцент Сидорик В.В.

Минск 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc152928724)

[1. Постановка и описание задачи 6](#_Toc152928725)

[2. Теоретический раздел 7](#_Toc152928726)

[3. Практический раздел 8](#_Toc152928727)

[**3.1. Построение UML диаграммы классов** 8](#_Toc152928728)

[**3.2. Описание графических окон** 10](#_Toc152928729)

[Заключение 13](#_Toc152928730)

[Список использованной литературы 14](#_Toc152928731)

[Листинг программного кода 15](#_Toc152928732)

# **ВВЕДЕНИЕ**

Язык программирования Java был создан в рамках проекта корпорации Sun Microsystems по разработке компьютерных программно-аппаратных комплексов нового поколения. Первая версия языка была официально опубликована в 1995 году. С тех пор язык Java стал стандартом "де-факто", вытеснив за десять лет языки C и C++ из многих областей программирования.

В 1995 году они были абсолютными лидерами, но к 2006 году число программирующих на Java стало заметно превышать число программистов, использующих C и C++, и сейчас достигло более четырех с половиной миллионов человек. А число устройств, запрограммированных с помощью Java, превысило полтора миллиарда.

Программы, написанные на языке Java, представляют собой наборы классов (о классах будет рассказано несколько позже) и сохраняются в текстовых файлах с расширением java. При компиляции текст программы переводится (транслируется) в двоичные файлы с расширением class. Такие файлы содержат байт-код, представляющий собой совокупность инструкций для абстрактного Java-процессора в виде байтовых последовательностей команд этого процессора и данных к ним. Для того чтобы выполнить байт-код на каком-либо компьютере, его нужно перевести в инструкции для соответствующего процессора. Именно этим и занимается Java-машина.

Программы Java можно разделить на несколько основных категорий:

* Приложение (application) — аналог обычной прикладной программы.
* Апплет (applet) — специализированная программа с ограниченными возможностями, работающая в окне WWW-документа под управлением браузера.
* Мидлет (midlet) — специализированная программа с ограниченными возможностями, работающая на мобильном устройстве. В настоящее время мидлеты — один из наиболее популярных сегментов рынка программного обеспечения, пишущегося на языке Java.
* Сервлет (servlet) — специализированная программа с ограниченными возможностями, работающая в WWW на стороне сервера. Используется преимущественно в рамках технологии JSP (Java Server Pages — серверных страниц Java) для программирования WWW-документов со стороны сервера и обработки отсылаемой на сервер информации.
* Серверное приложение (Enterprise application) — предназначено для многократного выполнения на стороне сервера.
* Библиотека (Java Class Library — библиотека классов, либо NetBeans Module — модуль платформы NetBeans) — предназначена для многократного использования программами Java.

# **Постановка и описание задачи**

Цель программы: разработать приложение на языке программирования Java и реализовать функционал в соответствии с бизнес-логикой выбранной темы. Программный код документировать. Сформировать с помощью утилиты javadoc html-файл, документирующий проект. Приложение скомпоновать в единый jar-файл.

*Детализация*: написать программу, вычисляющая подоходный налог к уплате.

*Технологии, используемые в работе*:

1. Объектно-ориентированный язык программирования Java;
2. Платформа для создания приложений с графическим интерфейсом Swing
3. Операционная система Windows 11.
4. Среда разработки InteliJ IDEA 2023;

*Исходные данные*: характеристики физического лица, основная ставка подоходного налога.

*Выходные данные*: налог к уплате и налоговые вычеты.

# **Теоретический раздел**

Данное приложение написано на объектно-ориентированном языке программирования Java.

Преимущества данного языка программирования:

1. Обладает тесно связанными друг с другом и легко усваиваемыми языковыми средствами.
2. Java обеспечивает встроенную поддержку многопоточного программирования, что использовано в приложении при реализации архитектуры «клиент-сервер».
3. В Java воплощена современная философия объектно-ориентированного программирования.
4. Язык Java не привязан к конкретному типу вычислительной машины или архитектуре операционной системы, что делает данное приложение конкурентоспособным среди других.
5. Байт-код Java максимально оптимизируется для повышения производительности.
6. Мультифункциональность.

Для создания графического интерфейса была выбрана библиотека Swing. Swing предоставляет более гибкие интерфейсные компоненты, чем более ранняя библиотека AWT. В отличие от AWT, компоненты Swing разработаны для одинаковой кроссплатформенной работы, в то время как компоненты AWT повторяют интерфейс исполняемой платформы без изменений. AWT же использует только стандартные элементы ОС для отображения, то есть для каждого элемента создается отдельный объект ОС (окно), в связи с чем AWT не позволяет создавать элементы произвольной формы (возможно использовать только прямоугольные компоненты), элементы управления на основе AWT всегда отображаются поверх Swing-элементов (так как все Swing-компоненты отображаются на поверхности контейнера).

Компоненты Swing поддерживают специфические динамически подключаемые виды и поведения, благодаря которому возможна адаптация к графическому интерфейсу платформы (то есть к компоненту можно динамически подключить другой, специфический для операционной системы, в том числе и созданный программистом вид и поведение). Таким образом, приложения, использующие Swing, могут выглядеть как родные приложения для данной операционной системы. Основным минусом таких «легковесных» компонентов является относительно медленная работа. Положительная сторона — универсальность интерфейса созданных приложений на всех платформах.

# **Практический раздел**

## **3.1. Построение UML диаграммы классов**

**Унифицированный язык моделирования** (UML) – это язык для специфицирования, визуализации, конструирования и документирования программных систем, а также бизнес-моделей и прочих не программных систем. UML представляет собой объединение инженерных приемов, которые ранее успешно использовались при моделировании больших и сложных систем.

При решении сложных задач заблаговременное планирование и моделирование значительно упрощают программирование. Кроме того, вносить изменения в диаграммы классов легче, чем в исходный код.

Таким образом, первым шагом к реализации курсового проекта были построены UML диаграммы (см. рисунок 3.1.1).

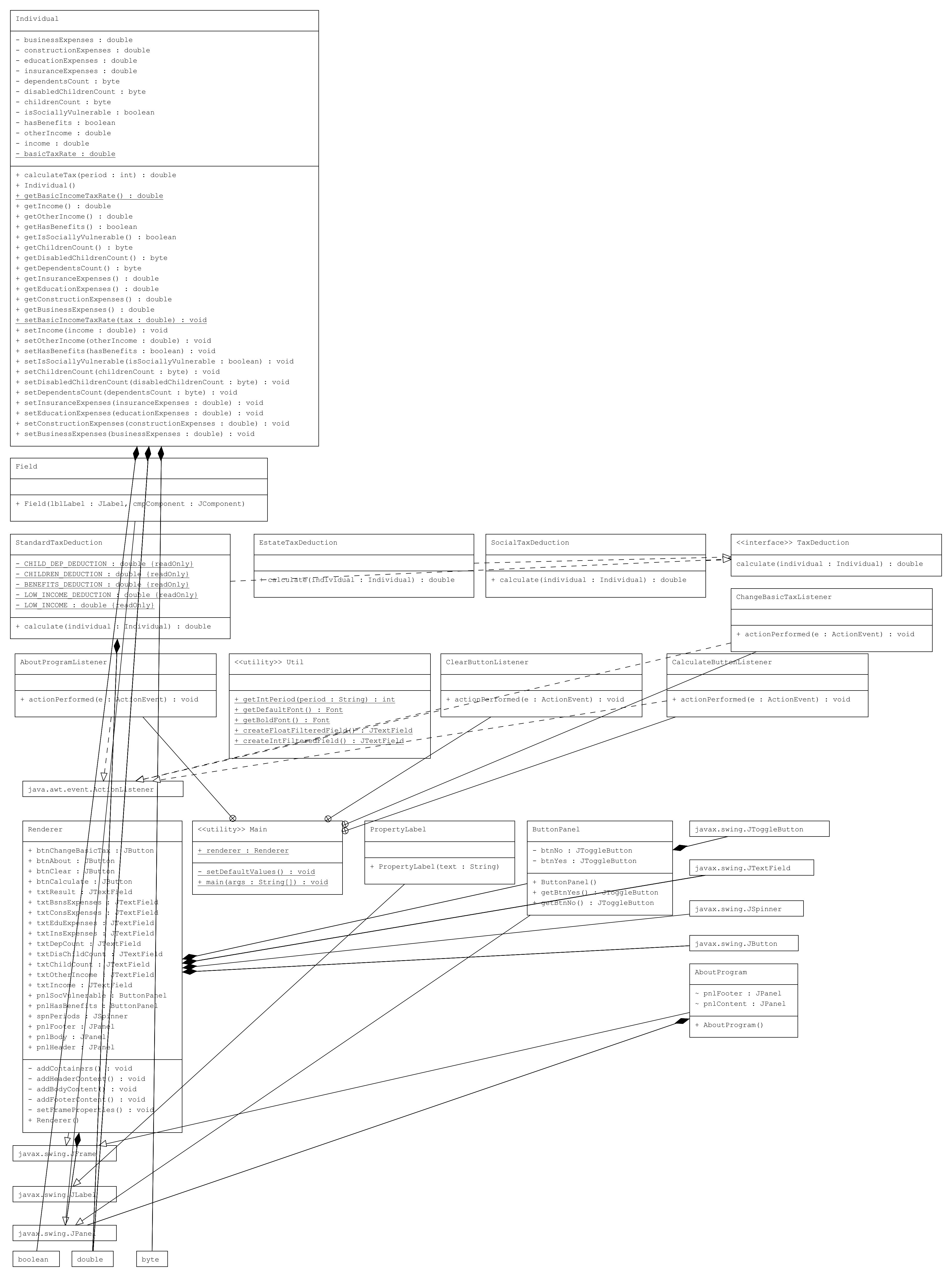


Рисунок 3.1.1 – UML диаграмма окон и контроллера

## **3.2. Описание графических окон**

**Главное окно**

Данное окно содержит основную логику приложения (Рис 3.2.1).

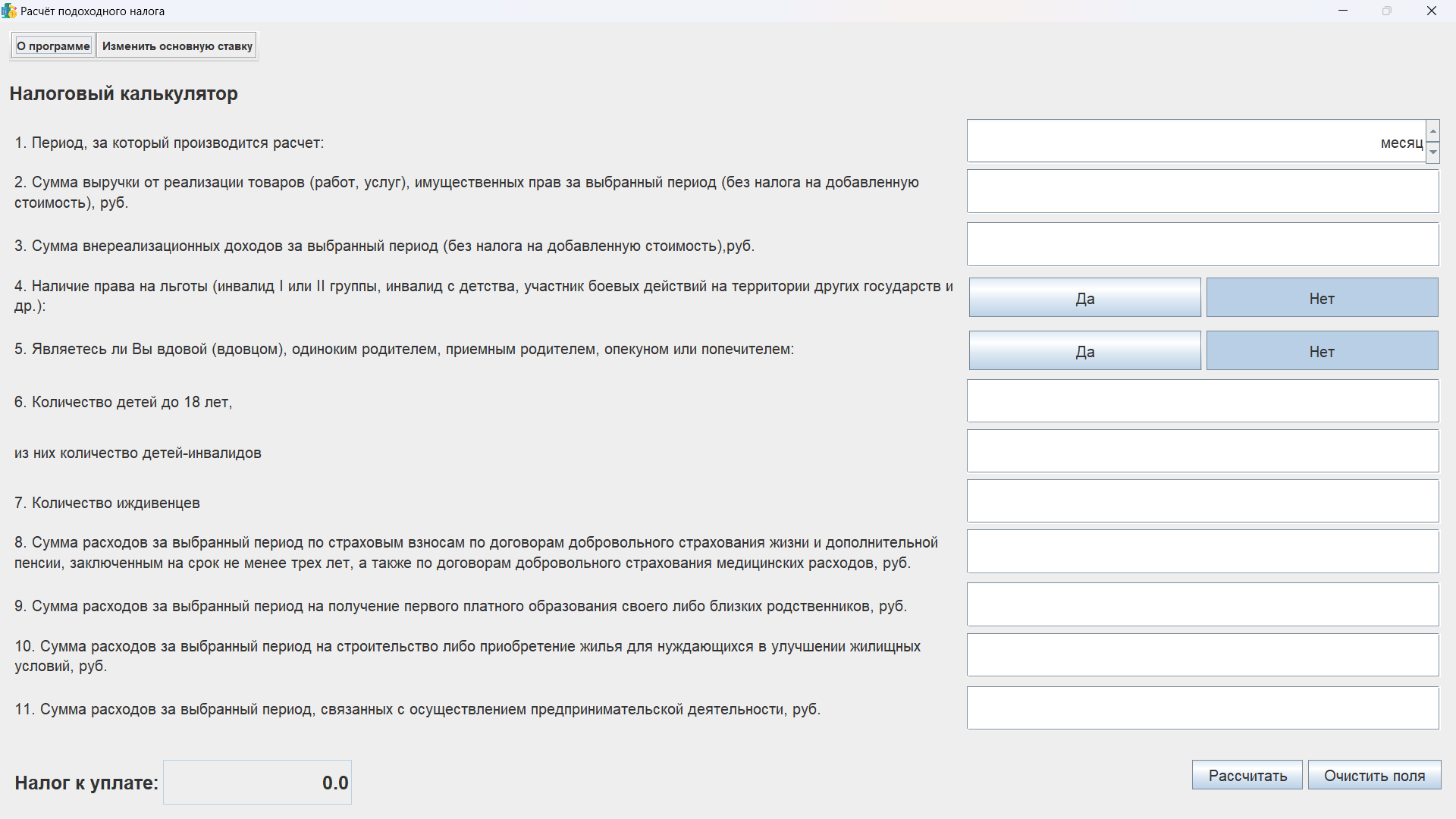


Рисунок 3.2.1 – Вид главного окна

Это окно содержит следующие ключевые элементы управления:

1. Перейти к окну “О программе”;
2. Открыть диалоговое окно “Изменить основную ставку”;
3. Рассчитать;
4. Очистить

**Окно «О программе»**

Данное окно содержит всю информацию о программе (Рис 3.2.2:

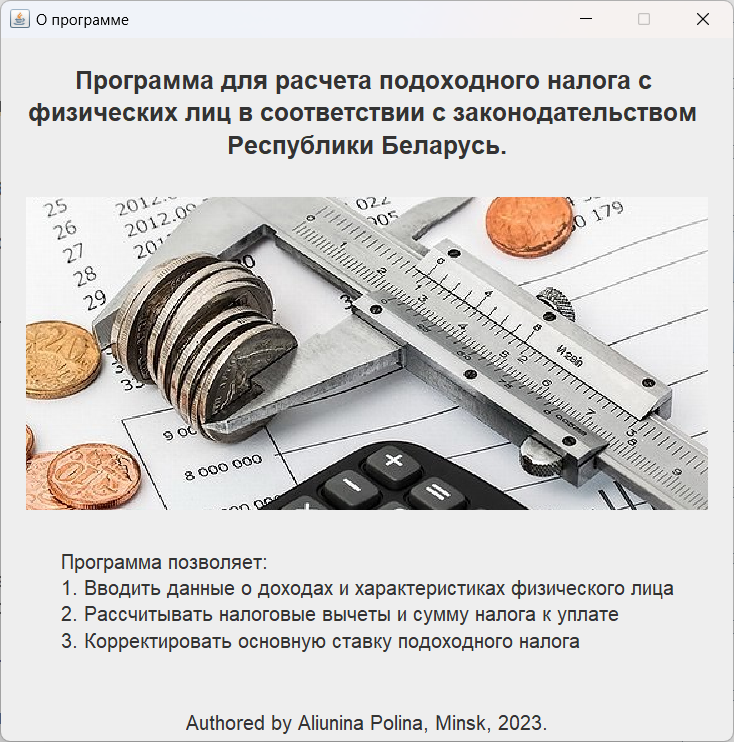


Рисунок 3.2.2 – Вид окна «О программе»

**Диалоговое окно «Изменить основную ставку»**

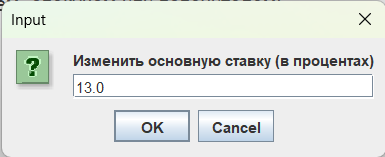


Рисунок 3.3.3 – Вид диалогового окна «Изменить основную ставку»

**Описание структуры проекта**

Структура проекта была спроектирована в соответствии с требованиями к курсовому проекту (см. Рис. 4.1):

1. javadoc – папка с файлами Javadoc;
2. src – папка с файлами источниками .java;
3. out – папка, в которой находится исполняемый файл.
4. doc – папка с запиской
5. images – папка с картинками

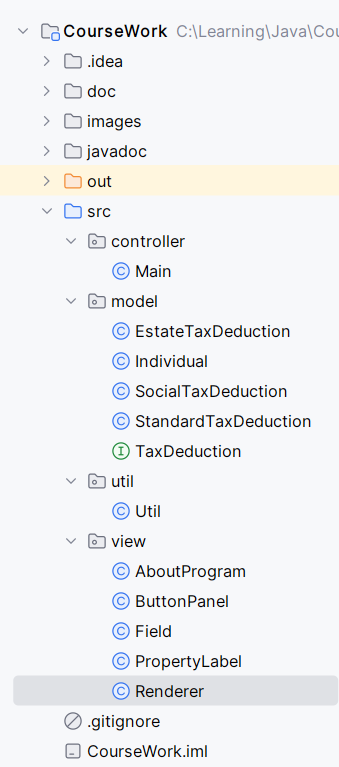


Рисунок 4.1 – Структура проекта в папке CourseWork

# **Заключение**

В данном курсовом проекте была реализована программа, вычисляющая подоходный налог с учетом налоговых вычетов.

При изучении данной темы были полученные новые теоретические и практические знания при реализации курсового проекта.

В процессе реализации были соблюдены принципы объектно-ориентированного программирования и архитектуры MVC.

# **Список использованной литературы**

***Книги***

1. Java. Промышленное программирование : практ. пособие / И.Н. Блинов, В.С. Романчик. – Минск : УниверсалПресс, 2007. – 704 с.

***Электронные издания***

1. Как начать пользоваться Swing GUE-визардом Intellij IDEA. Подробная инструкция [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/305974/
2. Пакет Java Swing [Электронный ресурс] – Режим доступа: https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/javax/swing/package-summary.html

# **Листинг программного кода**

controller.Main

package controller;  
  
import model.\*;  
import util.Util;  
import view.\*;  
import view.Renderer;  
  
import javax.swing.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
*/\* Класс реализует основную логику программы  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class Main {  
  
 */\*  
 \* Указатель на главный фрейм  
 \*/* public static Renderer *renderer*;  
  
 */\*  
 \* Задаёт начальные значения всем полям ввода  
 \*/* private static void setDefaultValues() {  
 *renderer*.spnPeriods.setValue("месяц");  
 *renderer*.txtIncome.setText("");  
 *renderer*.txtOtherIncome.setText("");  
 *renderer*.pnlHasBenefits.getBtnYes().setSelected(false);  
 *renderer*.pnlHasBenefits.getBtnNo().setSelected(true);  
 *renderer*.pnlSocVulnerable.getBtnYes().setSelected(false);  
 *renderer*.pnlSocVulnerable.getBtnNo().setSelected(true);  
 *renderer*.txtChildCount.setText("");  
 *renderer*.txtDisChildCount.setText("");  
 *renderer*.txtDepCount.setText("");  
 *renderer*.txtInsExpenses.setText("");  
 *renderer*.txtEduExpenses.setText("");  
 *renderer*.txtConsExpenses.setText("");  
 *renderer*.txtBsnsExpenses.setText("");  
 *renderer*.txtResult.setText("0.0");  
 }  
  
 */\*  
 \* Класс-слушатель кнопки "Очистить"  
 \* Устанавливает всем полям ввода начальные значения  
 \* @see view.Renderer#btnClear  
 \*/* public static class ClearButtonListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 *setDefaultValues*();  
 }  
 }  
  
 */\*  
 \* Класс-слушатель кнопки "Рассчитать"  
 \* Рассчитывает налог к уплате и выводит окно с налоговыми вычетами  
 \* @see view.Renderer#btnCalculate  
 \*/* public static class CalculateButtonListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 try {  
 int period = Util.*getIntPeriod*(*renderer*.spnPeriods.getValue().toString());  
 Individual individual = new Individual();  
 double resultTax = individual.calculateTax(period);  
 *renderer*.txtResult.setText(String.*format*("%4.2f", resultTax));  
 JOptionPane.*showMessageDialog*(null,  
 String.*format*("Стандартные: %4.2f\nСоциальные: %4.2f\nНа строительство: %4.2f",  
 new StandardTaxDeduction().calculate(individual),  
 new SocialTaxDeduction().calculate(individual),  
 new EstateTaxDeduction().calculate(individual)),  
 "Налоговые вычеты",  
 JOptionPane.*INFORMATION\_MESSAGE*);  
 } catch (Exception exception) {  
 *renderer*.txtResult.setText("0.00");  
 JOptionPane.*showMessageDialog*(null, exception.getMessage(), "",  
 JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 }  
  
 */\*  
 \* Класс-слушатель кнопки "О программе"  
 \* Выводит окно с информацией о программе  
 \* @see view.Renderer#btnAbout  
 \* @see view.AboutProgram  
 \*/* public static class AboutProgramListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 new AboutProgram();  
 }  
 }  
  
 */\*  
 \* Класс-слушатель кнопки "Изменить основную ставку"  
 \* Выводит окно с полем ввода для изменения основной ставки подоходного налога.  
 \* @see view.Renderer#btnChangeBasicTax  
 \* @see model.Individual#basicTaxRate  
 \*/* public static class ChangeBasicTaxListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String result = JOptionPane.*showInputDialog*(  
 *renderer*,  
 "Изменить основную ставку (в процентах)",  
 Individual.*getBasicIncomeTaxRate*() \* 100  
 );  
 try {  
 if (result != null && result.matches("^[0-9]{0,2}[.]?[0-9]{0,2}$")  
 && Integer.*parseInt*(result) < 100) {  
 Individual.*setBasicIncomeTaxRate*(Double.*parseDouble*(result) / 100);  
 } else if (result != null) {  
 throw new Exception();  
 }  
 } catch (Exception exception) {  
 JOptionPane.*showMessageDialog*(null, "Проверьте правильность ввода.");  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 *renderer* = new Renderer();  
 *setDefaultValues*();  
 }  
}

model.EstateTaxDeduction

package model;  
  
*/\*  
 \* Класс рассчитывает налоговый вычет на строительство  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class EstateTaxDeduction implements TaxDeduction {  
  
 */\*  
 \* Рассчитывает налоговый вычет на строительство  
 \* @param individual - физическое лицо  
 \*/* public double calculate(Individual individual) {  
 double deduction = 0.0;  
 deduction += individual.getConstructionExpenses();  
 return deduction;  
 }  
}

model.Individual

package model;  
  
import controller.\*;  
import javax.swing.\*;  
  
*/\*  
 \* Класс "Физическое лицо"  
 \* Содержит поле основной ставки подоходного налога и методы для ее изменения  
 \* Может рассчитывать подоходный налог с учетом налоговых вычетов  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class Individual {  
 */\*  
 \* Основная ставка подоходного налога  
 \*/* private static double basicTaxRate = 0.13;  
  
 */\*  
 \* Основной доход  
 \*/* private double income = 0.0;  
  
 */\*  
 \* Другие доходы (без НДС)  
 \*/* private double otherIncome = 0.0;  
  
 */\*  
 \* Имеет ли льготы (инвалид, участник ВОВ и т. д.)  
 \*/* private boolean hasBenefits = false;  
  
 */\*  
 \* Является ли вдовой (вдовцом), опекуном и т. д.  
 \*/* private boolean isSociallyVulnerable = false;  
  
 */\*  
 \* Количество детей до 18 лет  
 \*/* private byte childrenCount = 0;  
  
 */\*  
 \* Количество детей-инвалидов  
 \*/* private byte disabledChildrenCount = 0;  
  
 */\*  
 \* Количество иждивенцев  
 \*/* private byte dependentsCount = 0;  
  
 */\*  
 \* Расходы на страхование  
 \*/* private double insuranceExpenses = 0.0;  
  
 */\*  
 \* Расходы на образование  
 \*/* private double educationExpenses = 0.0;  
  
 */\*  
 \* Расходы на строительство  
 \*/* private double constructionExpenses = 0.0;  
  
 */\*  
 \* Расходы на предпринимательскую деятельность  
 \*/* private double businessExpenses = 0.0;  
  
 */\*  
 \* Рассчитывает налог к уплате с учетом налоговых вычетов  
 \* @param period - Период, за который производится расчет  
 \* @throws ArithmeticException  
 \* @return tax - налог к уплате  
 \*/* public double calculateTax(int period) throws Exception{  
 double tax = getIncome() + getOtherIncome();  
 if (tax <= 0) {  
 throw new ArithmeticException("Ошибка ввода данных: Доход должен быть больше нуля.");  
 }  
 tax -= new StandardTaxDeduction().calculate(this);  
 tax -= new SocialTaxDeduction().calculate(this);  
 tax -= new EstateTaxDeduction().calculate(this);  
 tax \*= basicTaxRate;  
 return tax \* period;  
 }  
  
 */\*  
 \* Конструктор  
 \* Создаёт объект класса, заполняя поля значениями из полей ввода  
 \* @throws NumberFormatException  
 \*/* public Individual() {  
 try {  
 setIncome(Double.parseDouble(Main.renderer.txtIncome.getText().isEmpty()? "0" :  
 Main.renderer.txtIncome.getText()));  
 setOtherIncome(Double.parseDouble(Main.renderer.txtOtherIncome.getText().isEmpty()? "0" :  
 Main.renderer.txtOtherIncome.getText()));  
 setHasBenefits(Main.renderer.pnlHasBenefits.getBtnYes().isSelected());  
 setIsSociallyVulnerable(Main.renderer.pnlSocVulnerable.getBtnYes().isSelected());  
 setChildrenCount(Byte.parseByte(Main.renderer.txtChildCount.getText().isEmpty()? "0" :  
 Main.renderer.txtChildCount.getText()));  
 setDisabledChildrenCount(Byte.parseByte(Main.renderer.txtDisChildCount.getText().isEmpty()? "0" :  
 Main.renderer.txtDisChildCount.getText()));  
 setDependentsCount(Byte.parseByte(Main.renderer.txtDepCount.getText().isEmpty()? "0" :  
 Main.renderer.txtDepCount.getText()));  
 setInsuranceExpenses(Double.parseDouble(Main.renderer.txtInsExpenses.getText().isEmpty()? "0" :  
 Main.renderer.txtInsExpenses.getText()));  
 setEducationExpenses(Double.parseDouble(Main.renderer.txtEduExpenses.getText().isEmpty()? "0" :  
 Main.renderer.txtEduExpenses.getText()));  
 setConstructionExpenses(Double.parseDouble(Main.renderer.txtConsExpenses.getText().isEmpty()? "0" :  
 Main.renderer.txtConsExpenses.getText()));  
 setBusinessExpenses(Double.parseDouble(Main.renderer.txtBsnsExpenses.getText().isEmpty() ? "0" :  
 Main.*renderer*.txtBsnsExpenses.getText()));  
 } catch (NumberFormatException exception){  
 JOptionPane.showMessageDialog(null, "Проверьте правильность ввода.");  
 }  
 }  
  
 */\*  
 \* Возвращает значение basicTaxRate  
 \*/* public static double getBasicIncomeTaxRate() {  
 return basicTaxRate;  
 }  
  
 */\*  
 \* Возвращает значение income  
 \* @return income  
 \*/* public double getIncome() {  
 return this.income;  
 }  
  
 */\*  
 \* Возвращает значение otherIncome  
 \* @return otherIncome  
 \*/* public double getOtherIncome() {  
 return this.otherIncome;  
 }  
  
 */\*  
 \* Возвращает значение hasBenefits  
 \* @return hasBenefits  
 \*/* public boolean getHasBenefits() {  
 return this.hasBenefits;  
 }  
  
 */\*  
 \* Возвращает значение isSociallyVulnerable  
 \* @return isSociallyVulnerable  
 \*/* public boolean getIsSociallyVulnerable() {  
 return this.isSociallyVulnerable;  
 }  
  
 */\*  
 \* Возвращает значение childrenCount  
 \* @return childrenCount  
 \*/* public byte getChildrenCount() {  
 return this.childrenCount;  
 }  
  
 */\*  
 \* Возвращает значение disabledChildrenCount  
 \* @return disabledChildrenCount  
 \*/* public byte getDisabledChildrenCount() {  
 return this.disabledChildrenCount;  
 }  
  
 */\*  
 \* Возвращает значение dependentsCount  
 \* @return dependentsCount  
 \*/* public byte getDependentsCount() {  
 return this.dependentsCount;  
 }  
  
 */\*  
 \* Возвращает значение insuranceExpenses  
 \* @return insuranceExpenses  
 \*/* public double getInsuranceExpenses() {  
 return this.insuranceExpenses;  
 }  
  
 */\*  
 \* Возвращает значение educationExpenses  
 \* @return educationExpenses  
 \*/* public double getEducationExpenses() {  
 return this.educationExpenses;  
 }  
  
 */\*  
 \* Возвращает значение constructionExpenses  
 \* @return constructionExpenses  
 \*/* public double getConstructionExpenses() {  
 return this.constructionExpenses;  
 }  
  
 */\*  
 \* Возвращает значение businessExpenses  
 \* @return businessExpenses  
 \*/* public double getBusinessExpenses() {  
 return this.businessExpenses;  
 }  
  
 */\*  
 \* Устанавливает значение basicTaxRate  
 \* @param tax - новое значение основной ставки подоходного налога  
 \*/* public static void setBasicIncomeTaxRate(double tax) {  
 basicTaxRate = tax;  
 }  
  
 */\*  
 \* Устанавливает значение income  
 \* @param income - новое значение основного дохода  
 \*/* public void setIncome(double income) {  
 this.income = income;  
 }  
  
 */\*  
 \* Устанавливает значение otherIncome  
 \* @param otherIncome - новое значение других доходов  
 \*/* public void setOtherIncome(double otherIncome) {  
 this.otherIncome = otherIncome;  
 }  
  
 */\*  
 \* Устанавливает значение hasBenefits  
 \* @param hasBenefits - смена значения "Имеет ли льготы"  
 \*/* public void setHasBenefits(boolean hasBenefits) {  
 this.hasBenefits = hasBenefits;  
 }  
  
 */\*  
 \* Устанавливает значение isSociallyVulnerable  
 \* @param isSociallyVulnerable - смена значения "Является ли вдовой(вдовцом) и т. д."  
 \*/* public void setIsSociallyVulnerable(boolean isSociallyVulnerable) {  
 this.isSociallyVulnerable = isSociallyVulnerable;  
 }  
  
 */\*  
 \* Устанавливает значение childrenCount  
 \* @param childrenCount - новое значение количества детей  
 \*/* public void setChildrenCount(byte childrenCount) {  
 this.childrenCount = childrenCount;  
 }  
  
 */\*  
 \* Устанавливает значение disabledChildrenCount  
 \* @param disabledChildrenCount - новое значение количества летей инвалидов  
 \*/* public void setDisabledChildrenCount(byte disabledChildrenCount) {  
 this.disabledChildrenCount = disabledChildrenCount;  
 }  
  
 */\*  
 \* Устанавливает значение dependentsCount  
 \* @param dependentsCount - количество иждивенцев  
 \*/* public void setDependentsCount(byte dependentsCount) {  
 this.dependentsCount = dependentsCount;  
 }  
  
 */\*  
 \* Устанавливает значение insuranceExpenses  
 \* @param insuranceExpenses - новое значение расходов на страхование  
 \*/* public void setInsuranceExpenses(double insuranceExpenses) {  
 this.insuranceExpenses = insuranceExpenses;  
 }  
  
 */\*  
 \* Устанавливает значение educationExpenses  
 \* @param educationExpenses - новое значение расходов на образование  
 \*/* public void setEducationExpenses(double educationExpenses) {  
 this.educationExpenses = educationExpenses;  
 }  
  
 */\*  
 \* Устанавливает значение constructionExpenses  
 \* @param constructionExpenses - новое значение расходов на строительство  
 \*/* public void setConstructionExpenses(double constructionExpenses) {  
 this.constructionExpenses = constructionExpenses;  
 }  
  
 */\*  
 \* Устанавливает значение businessExpenses  
 \* @param businessExpenses - новое значение расходов на предпринимательство  
 \*/* public void setBusinessExpenses(double businessExpenses) {  
 this.businessExpenses = businessExpenses;  
 }  
}

model.SocialTaxDeduction

package model;  
  
*/\*  
 \* Класс рассчитывает социальный налоговый вычет  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class SocialTaxDeduction implements TaxDeduction {  
  
 */\*  
 \* Рассчитывает социальный налоговый вычет  
 \* @param individual - физическое лицо  
 \*/* public double calculate(Individual individual) {  
 double deduction = 0.0;  
 deduction += individual.getInsuranceExpenses() + individual.getEducationExpenses();  
 return deduction;  
 }  
}

model.StandardTaxDeduction

package model;  
  
*/\*  
 \* Класс рассчитывает стандартный налоговый вычет  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class StandardTaxDeduction implements TaxDeduction {  
  
 */\*  
 \* Минимальный доход  
 \*/* private static final double *LOW\_INCOME* = 944;  
  
 */\*  
 \* Вычет при условии дохода меньше минимального  
 \*/* private static final double *LOW\_INCOME\_DEDUCTION* = 156;  
  
 */\*  
 \* Вычет при наличии льгот  
 \*/* private static final double *BENEFITS\_DEDUCTION* = 220;  
  
 */\*  
 \* Вычет на ребёнка  
 \*/* private static final double *CHILDREN\_DEDUCTION* = 46;  
  
 */\*  
 \* Вычет на ребёнка/иждивенца  
 \*/* private static final double *CHILD\_DEP\_DEDUCTION* = 87;  
  
 */\*  
 \* Рассчитывает стандартный налоговый вычет  
 \* @param individual - физическое лицо  
 \* @throws ArithmeticException  
 \* @return deduction - стандартный налоговый вычет  
 \*/* public double calculate(Individual individual) throws Exception{  
 double deduction = 0.0, allIncome = individual.getIncome() + individual.getOtherIncome();  
 if (allIncome < individual.getBusinessExpenses()) {  
 throw new ArithmeticException("Ошибка ввода данных: Расход не может превышать доходы.");  
 }  
 if (individual.getBusinessExpenses() > 0) {  
 deduction += (allIncome - individual.getBusinessExpenses()) \* 0.16;  
 }  
 if (allIncome <= LOW\_INCOME) {  
 deduction += LOW\_INCOME\_DEDUCTION;  
 }  
 if (individual.getHasBenefits()) {  
 deduction += BENEFITS\_DEDUCTION;  
 }  
 if (individual.getChildrenCount() == 1) {  
 deduction += CHILDREN\_DEDUCTION;  
 }  
 if (individual.getDependentsCount() > 0)  
 deduction += individual.getDependentsCount() \* CHILDREN\_DEDUCTION;  
 if (individual.getChildrenCount() >= 2 || individual.getDisabledChildrenCount() > 0) {  
 deduction += CHILD\_DEP\_DEDUCTION;  
 }  
 if (individual.getIsSociallyVulnerable()) {  
 deduction += individual.getChildrenCount() \* CHILD\_DEP\_DEDUCTION  
 + individual.getDependentsCount() \* CHILD\_DEP\_DEDUCTION;  
 }  
 return deduction;  
 }  
}

model.TaxDeduction

package model;  
  
*/\*  
 \* Интерфейс для налоговых вычетов  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public interface TaxDeduction {  
  
 */\*  
 \* Метод для расчета налогового вычета  
 \* @param individual - физическое лицо  
 \* @throws Exception  
 \*/* public double calculate(Individual individual) throws Exception;  
}

util.Util

package util;  
  
import javax.swing.\*;  
import javax.swing.text.AbstractDocument;  
import javax.swing.text.AttributeSet;  
import javax.swing.text.BadLocationException;  
import javax.swing.text.DocumentFilter;  
import java.awt.\*;  
  
*/\* Вспомогательный класс  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class Util {  
  
 */\* Возвращает числовое значение периода  
 \* @param period - строковое значение периода  
 \* @return result - числовое значение периода  
 \*/* public static int getIntPeriod(String period) {  
 int result = 1;  
 switch (period) {  
 case "квартал": {  
 result = 3;  
 break;  
 }  
 case "полугодие": {  
 result = 6;  
 break;  
 }  
 case "9 месяцев": {  
 result = 9;  
 break;  
 }  
 case "год": {  
 result = 12;  
 break;  
 }  
 }  
 return result;  
 }  
  
 */\* Возвращает стандартный шрифт  
 \* @return Font - стандартный шрифт  
 \*/* public static Font getDefaultFont() {  
 return new Font("Courier", Font.PLAIN, 16);  
 }  
  
 */\* Возвращает жирный, большой шрифт  
 \* @return Font - жирный, большой шрифт  
 \*/* public static Font getBoldFont() {  
 return new Font("Courier", Font.BOLD, 20);  
 }  
  
 */\* Создает текстовое поле для дробных чисел  
 \* @return field - текстовое поле для дробных чисел  
 \*/* public static JTextField createFloatFilteredField() {  
 JTextField field = new JTextField();  
 AbstractDocument document = (AbstractDocument) field.getDocument();  
 final int maxCharacters = 15;  
 document.setDocumentFilter(new DocumentFilter() {  
 public void replace(FilterBypass fb, int offs, int length, String str, AttributeSet a)  
 throws BadLocationException {  
 String text = fb.getDocument().getText(0, fb.getDocument().getLength());  
 text += str;  
 if ((fb.getDocument().getLength() + str.length() - length) <= maxCharacters  
 && text.matches("^[0-9]+[.]?[0-9]{0,2}$")) {  
 super.replace(fb, offs, length, str, a);  
 }  
 }  
  
 public void insertString(FilterBypass fb, int offs, String str, AttributeSet a)  
 throws BadLocationException {  
 String text = fb.getDocument().getText(0, fb.getDocument().getLength());  
 text += str;  
 if ((fb.getDocument().getLength() + str.length()) <= maxCharacters  
 && text.matches("^[0-9]+[.]?[0-9]{0,2}$")) {  
 super.insertString(fb, offs, str, a);  
 }  
 }  
 });  
 return field;  
 }  
  
 */\* Создает текстовое поле для целых чисел  
 \* @return field - текстовое поле для целых чисел  
 \*/* public static JTextField createIntFilteredField() {  
 JTextField field = new JTextField();  
 AbstractDocument document = (AbstractDocument) field.getDocument();  
 final int maxCharacters = 2;  
 document.setDocumentFilter(new DocumentFilter() {  
 public void replace(FilterBypass fb, int offs, int length, String str, AttributeSet a)  
 throws BadLocationException {  
 String text = fb.getDocument().getText(0, fb.getDocument().getLength());  
 text += str;  
 if ((fb.getDocument().getLength() + str.length() - length) <= maxCharacters  
 && text.matches("^[0-9]+$")) {  
 super.replace(fb, offs, length, str, a);  
 }  
 }  
  
 public void insertString(FilterBypass fb, int offs, String str, AttributeSet a)  
 throws BadLocationException {  
 String text = fb.getDocument().getText(0, fb.getDocument().getLength());  
 text += str;  
 if ((fb.getDocument().getLength() + str.length()) <= maxCharacters  
 && text.matches("^[0-9]+$")) {  
 super.insertString(fb, offs, str, a);  
 }  
 }  
 });  
 return field;  
 }  
}

view.AboutProgram

package view;  
  
import util.Util;  
  
import javax.imageio.ImageIO;  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.io.IOException;  
  
*/\* Класс для создания окна "О программе"  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class AboutProgram extends JFrame {  
  
 */\*  
 \* Панели окна  
 \*/* JPanel pnlContent, pnlFooter;  
  
 */\*  
 \* Конструктор  
 \* Создаёт окно "О программе" и добавляет в него элементы"  
 \*/* public AboutProgram() {  
 super("О программе");  
  
 setLayout(new BorderLayout());  
 pnlContent = new JPanel();  
 pnlFooter = new JPanel();  
 pnlContent.setLayout(new BoxLayout(pnlContent, BoxLayout.Y\_AXIS));  
 JLabel lblTheme = new JLabel("<html><div style='text-align: center;'>Программа для расчета подоходного налога с физических лиц в " +  
 "соответствии с законодательством Республики Беларусь.</div></html>");  
 lblTheme.setBorder(BorderFactory.createEmptyBorder(0, 0, 30, 0));  
 lblTheme.setFont(Util.getBoldFont());  
 lblTheme.setHorizontalAlignment(SwingConstants.CENTER);  
 pnlContent.add(lblTheme);  
  
 BufferedImage imgTaxes;  
 try {  
 imgTaxes = ImageIO.read(new File("images/taxes.jpg"));  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 JLabel lblImageAbout = new JLabel(new ImageIcon(imgTaxes));  
 lblImageAbout.setPreferredSize(new Dimension(100, 250));  
  
 pnlContent.add(lblImageAbout);  
  
 JLabel lblFeatures = new JLabel("<html>Программа позволяет:<br/>" +  
 "1. Вводить данные о доходах и характеристиках физического лица<br/>" +  
 "2. Рассчитывать налоговые вычеты и сумму налога к уплате<br/>" +  
 "3. Корректировать основную ставку подоходного налога</html>");  
 lblFeatures.setFont(Util.getDefaultFont());  
 lblFeatures.setHorizontalAlignment(SwingConstants.CENTER);  
 lblFeatures.setBorder(BorderFactory.createEmptyBorder(30, 0, 0, 0));  
 pnlContent.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));  
  
 JLabel lblFooter = new JLabel("<html><div style='text-align: center;'>Authored by Aliunina Polina, Minsk, 2023.</div></html>");  
 lblFooter.setFont(Util.getDefaultFont());  
 pnlFooter.add(lblFooter);  
 pnlContent.add(lblFeatures);  
  
 add(pnlContent, BorderLayout.NORTH);  
 add(pnlFooter, BorderLayout.SOUTH);  
 setSize(600, 600);  
 setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);  
 setLocationRelativeTo(null);  
 setResizable(false);  
 setVisible(true);  
  
 }  
}

view.ButtonPanel

package view;  
  
import util.Util;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
*/\* Класс для создания панели с кнопками "Да" и "Нет"  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class ButtonPanel extends JPanel {  
  
 */\*  
 \* Кнопка "Да"  
 \*/* private JToggleButton btnYes;  
  
 */\*  
 \* Кнопка "Нет"  
 \*/* private JToggleButton btnNo;  
  
 */\*  
 \* Конструктор  
 \* Создает панель с кнопками "Да" и "Нет"  
 \*/* public ButtonPanel() {  
 super();  
 this.btnYes = new JToggleButton("Да");  
 this.btnNo = new JToggleButton("Нет");  
 this.btnYes.setFont(Util.getDefaultFont());  
 this.btnNo.setFont(Util.getDefaultFont());  
 this.btnYes.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 btnNo.setSelected(false);  
 }  
 });  
 btnNo.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 btnYes.setSelected(false);  
 }  
 });  
 this.btnYes.setPreferredSize(new Dimension(245, 42));  
 this.btnNo.setPreferredSize(new Dimension(245, 42));  
 this.btnNo.setSelected(true);  
 this.add(btnYes);  
 this.add(btnNo);  
 }  
  
 */\*  
 \* Возвращает кнопку "Да"  
 \* @return btnYes  
 \*/* public JToggleButton getBtnYes() {  
 return this.btnYes;  
 }  
  
 */\*  
 \* Возвращает кнопку "Нет"  
 \* @return btnNo  
 \*/* public JToggleButton getBtnNo() {  
 return this.btnNo;  
 }  
}

view.Field

package view;  
  
import util.Util;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
*/\* Класс для создания панели "лейбл + поле для ввода"  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class Field extends JPanel {  
  
 */\*  
 \* Конструктор  
 \* Создает панель "лейбл + поле для ввода"  
 \* Устанавливает шрифт и размеры  
 \* @param lblLabel - лейбл  
 \* @param cmpComponent - поле для ввода  
 \*/* public Field(JLabel lblLabel, JComponent cmpComponent) {  
 super();  
 this.setLayout(new FlowLayout(FlowLayout.LEFT));  
 this.add(lblLabel);  
  
 cmpComponent.setFont(Util.getDefaultFont());  
 cmpComponent.setPreferredSize(new Dimension(500, 47));  
 this.add(cmpComponent);  
 }  
}

view.PropertyLabel

package view;  
  
import util.Util;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
*/\* Класс для создания лейбла  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class PropertyLabel extends JLabel {  
  
 */\*  
 \* Конструктор  
 \* Создает лейбл определенного размера с переносом текста  
 \* Задает шрифт лейбла  
 \* @param text - текст лейбла  
 \*/* public PropertyLabel(String text) {  
 super("<html>" + text + "</html>");  
 setPreferredSize(new Dimension(1000, 47));  
 setFont(Util.getDefaultFont());  
 }  
}

view.Renderer

package view;  
  
import javax.swing.\*;  
import java.awt.\*;  
import controller.\*;  
import util.Util;  
  
*/\* Основной класс, отрисовывающий компоненты интерфейса  
 \* @author Aliunina P.A.  
 \* @version 1.0  
 \*/*public class Renderer extends JFrame {  
  
 */\*  
 \* Основные панели окна  
 \*/* public JPanel pnlHeader, pnlBody, pnlFooter;  
  
 */\*  
 \* Список с выбором периодов  
 \*/* public JSpinner spnPeriods;  
  
 */\*  
 \* Панели кнопок "Да" и "Нет"  
 \*/* public ButtonPanel pnlHasBenefits, pnlSocVulnerable;  
  
 */\*  
 \* Поля для ввода  
 \*/* public JTextField txtIncome, txtOtherIncome, txtChildCount, txtDisChildCount, txtDepCount;  
  
 */\*  
 \* Поля для ввода  
 \*/* public JTextField txtInsExpenses, txtEduExpenses, txtConsExpenses, txtBsnsExpenses;  
  
 */\*  
 \* Поле для вывода результата  
 \*/* public JTextField txtResult;  
  
 */\*  
 \* Основные кнопки окна: "Рассчитать", "Очистить", "О программе", "Изменить основную ставку"  
 \*/* public JButton btnCalculate, btnClear, btnAbout, btnChangeBasicTax;  
  
 */\*  
 \* Создает основные контейнеры окна  
 \*/* private void addContainers() {  
 pnlHeader = new JPanel();  
 pnlBody = new JPanel();  
 pnlFooter = new JPanel();  
 add(pnlHeader, BorderLayout.*NORTH*);  
 add(pnlBody, BorderLayout.*CENTER*);  
 add(pnlFooter, BorderLayout.*SOUTH*);  
 pnlHeader.setLayout(new BoxLayout(pnlHeader, BoxLayout.*Y\_AXIS*));  
 pnlBody.setLayout(new BoxLayout(pnlBody, BoxLayout.*Y\_AXIS*));  
 pnlFooter.setLayout(new BorderLayout());  
 }  
  
 */\*  
 \* Добавляет и заполняет header-панель  
 \*/* private void addHeaderContent() {  
 JToolBar tbToolbar = new JToolBar();  
 tbToolbar.setFloatable(false);  
 tbToolbar.setAlignmentX(JComponent.*LEFT\_ALIGNMENT*);  
 tbToolbar.setAlignmentY(JComponent.*TOP\_ALIGNMENT*);  
 btnAbout = new JButton("О программе");  
 btnAbout.addActionListener(new Main.AboutProgramListener());  
 btnChangeBasicTax = new JButton("Изменить основную ставку");  
 btnChangeBasicTax.addActionListener(new Main.ChangeBasicTaxListener());  
 tbToolbar.add(btnAbout);  
 tbToolbar.add(btnChangeBasicTax);  
 pnlHeader.add(tbToolbar);  
  
 JLabel lblWelcome = new JLabel("Налоговый калькулятор");  
 lblWelcome.setFont(Util.*getBoldFont*());  
 lblWelcome.setBorder(BorderFactory.*createEmptyBorder*(20, 0, 0, 0));  
 pnlHeader.add(lblWelcome);  
 pnlHeader.setBorder(BorderFactory.*createEmptyBorder*(10, 10, 0, 10));  
 }  
  
 */\*  
 \* Добавляет и заполняет body-панель  
 \*/* private void addBodyContent() {  
 spnPeriods = new JSpinner(new SpinnerListModel(new String[] {"год", "9 месяцев", "полугодие", "квартал", "месяц"}));  
 pnlBody.add(new Field(new PropertyLabel("1. Период, за который производится расчет:"), spnPeriods));  
  
 txtIncome = Util.*createFloatFilteredField*();  
 txtOtherIncome = Util.*createFloatFilteredField*();  
 txtIncome.setHorizontalAlignment(SwingConstants.*RIGHT*);  
 txtOtherIncome.setHorizontalAlignment(SwingConstants.*RIGHT*);  
 pnlBody.add(new Field(new PropertyLabel("2. Сумма выручки от реализации товаров (работ, услуг), " +  
 "имущественных прав за выбранный период (без налога на добавленную стоимость), руб."),  
 txtIncome));  
 pnlBody.add(new Field(new PropertyLabel("3. Сумма внереализационных доходов за выбранный период " +  
 "(без налога на добавленную стоимость),руб."),  
 txtOtherIncome));  
  
 pnlHasBenefits = new ButtonPanel();  
 pnlSocVulnerable = new ButtonPanel();  
 pnlBody.add(new Field(new PropertyLabel("4. Наличие права на льготы (инвалид I или II группы, " +  
 "инвалид с детства, участник боевых действий на территории других государств и др.):"),  
 pnlHasBenefits));  
 pnlBody.add(new Field(new PropertyLabel("5. Являетесь ли Вы вдовой (вдовцом), одиноким родителем, " +  
 "приемным родителем, опекуном или попечителем:"),  
 pnlSocVulnerable));  
  
 txtChildCount = Util.*createIntFilteredField*();  
 txtDisChildCount = Util.*createIntFilteredField*();  
 txtDepCount = Util.*createIntFilteredField*();  
 txtChildCount.setHorizontalAlignment(SwingConstants.*RIGHT*);  
 txtDisChildCount.setHorizontalAlignment(SwingConstants.*RIGHT*);  
 txtDepCount.setHorizontalAlignment(SwingConstants.*RIGHT*);  
  
 pnlBody.add(new Field(new PropertyLabel("6. Количество детей до 18 лет,"),  
 txtChildCount));  
 pnlBody.add(new Field(new PropertyLabel("из них количество детей-инвалидов"),  
 txtDisChildCount));  
 pnlBody.add(new Field(new PropertyLabel("7. Количество иждивенцев"),  
 txtDepCount));  
  
 txtInsExpenses = Util.*createFloatFilteredField*();  
 txtEduExpenses = Util.*createFloatFilteredField*();  
 txtConsExpenses = Util.*createFloatFilteredField*();  
 txtBsnsExpenses = Util.*createFloatFilteredField*();  
 txtInsExpenses.setHorizontalAlignment(SwingConstants.*RIGHT*);  
 txtEduExpenses.setHorizontalAlignment(SwingConstants.*RIGHT*);  
 txtConsExpenses.setHorizontalAlignment(SwingConstants.*RIGHT*);  
 txtBsnsExpenses.setHorizontalAlignment(SwingConstants.*RIGHT*);  
 pnlBody.add(new Field(new PropertyLabel("8. Сумма расходов за выбранный период по страховым " +  
 "взносам по договорам добровольного страхования жизни и дополнительной пенсии, заключенным на " +  
 "срок не менее трех лет, а также по договорам добровольного страхования медицинских расходов, руб."),  
 txtInsExpenses));  
 pnlBody.add(new Field(new PropertyLabel("9. Сумма расходов за выбранный период на получение " +  
 "первого платного образования своего либо близких родственников, руб."),  
 txtEduExpenses));  
 pnlBody.add(new Field(new PropertyLabel("10. Сумма расходов за выбранный период на строительство " +  
 "либо приобретение жилья для нуждающихся в улучшении жилищных условий, руб."),  
 txtConsExpenses));  
 pnlBody.add(new Field(new PropertyLabel("11. Сумма расходов за выбранный период, связанных с " +  
 "осуществлением предпринимательской деятельности, руб."),  
 txtBsnsExpenses));  
 pnlBody.setBorder(BorderFactory.*createEmptyBorder*(10, 10, 10, 10));  
 }  
  
 */\*  
 \* Добавляет и заполняет footer-панель  
 \*/* private void addFooterContent() {  
 JPanel pnlResult = new JPanel();  
 JLabel lblResult = new JLabel("Налог к уплате:");  
 txtResult = new JTextField();  
 txtResult.setEditable(false);  
 txtResult.setPreferredSize(new Dimension(200, 47));  
 txtResult.setFont(Util.*getBoldFont*());  
 lblResult.setFont(Util.*getBoldFont*());  
 txtResult.setHorizontalAlignment(SwingConstants.*RIGHT*);  
 pnlResult.add(lblResult);  
 pnlResult.add(txtResult);  
 pnlFooter.add(pnlResult, BorderLayout.*WEST*);  
  
 JPanel pnlActions = new JPanel();  
 btnCalculate = new JButton("Рассчитать");  
 btnClear = new JButton("Очистить поля");  
 btnCalculate.setFont(Util.*getDefaultFont*());  
 btnClear.setFont(Util.*getDefaultFont*());  
 btnCalculate.addActionListener(new Main.CalculateButtonListener());  
 btnClear.addActionListener(new Main.ClearButtonListener());  
 pnlActions.add(btnCalculate);  
 pnlActions.add(btnClear);  
 pnlFooter.add(pnlActions, BorderLayout.*EAST*);  
 pnlFooter.setBorder(BorderFactory.*createEmptyBorder*(10, 10, 10, 10));  
 }  
  
 */\*  
 \* Задает основные свойства окна  
 \*/* private void setFrameProperties() {  
 setIconImage(new ImageIcon("images/logo.png").getImage());  
 setExtendedState(Frame.*MAXIMIZED\_BOTH*);  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 setLocationRelativeTo(null);  
 setLayout(new BorderLayout());  
 setResizable(false);  
 }  
  
 */\*  
 \* Конструктор  
 \* Создает основное окно приложения  
 \*/* public Renderer() {  
 super("Расчёт подоходного налога");  
 setFrameProperties();  
 addContainers();  
 addHeaderContent();  
 addBodyContent();  
 addFooterContent();  
 setVisible(true);  
 }  
}