Министерство науки и высшего образования РФ

Федеральное государственное образовательное учреждение

высшего образования

«УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ»

Кафедра АСУ

Курсовая работа

по дисциплине

«Объектно-ориентированное моделирование

и программирование»

«Совместное написание программного обеспечения»

Выполнили:

студент гр. ПИ-101-Бзу

Макуха Т.Б.

Проверил:

преподаватель

Казанцев А.В.

**Министерство науки и высшего образования РФ**

Федеральное государственное образовательное учреждение

высшего образования

«УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ»

Кафедра автоматизированных систем управления

ЗАДАНИЕ

на курсовую работу (КР) по дисциплине «Объектно-ориентированное моделирование и программирование»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | Макуха Т.Б | Группа | ПИ-101-Бзу | Консультант | *Казанцев А.В.* |
| Фамилия И.О. | | номер группы | | Фамилия И.О. | |

1. Тема КР: калькулятор строительства дома
2. Основное содержание:
3. Пояснительная записка с необходимыми материалами
4. Репозиторий системы контроля версий содержащий программный код с комментариями
5. Требования к оформлению:

3.1 Пояснительная записка должна быть оформлена в текстовом процессоре LibreOffice Writer в соответствии с требованиями ГОСТ. В бумажном виде оформляются: титульный лист, бланк задания, также прикладывается аннотация, где указывается ссылка на репозиторий, содержащий исходный проект, работающую программу и пояснительную записку.

4. Графическая часть должна включать:

- экранные формы инструментальных средств

- экранные формы разрабатываемого программного продукта

Дата выдачи \_\_\_02.12.2023\_\_\_\_\_\_ Дата окончания\_\_\_\_28.03.2024\_\_\_\_

Руководитель\_\_\_\_Казанцев А.В. Студент\_\_\_\_\_Макуха Т.Б.

**Содержание**

[**Раздел 1. Описание предметной области** 4](#_Toc162671885)

[**Раздел 2. Описание прототипа программного продукта (ПП): Калькулятор строительства дома** 5](#_Toc162671886)

[**Раздел 3. Математическая модель работы ПП** 7](#_Toc162671887)

[**Раздел 4. Написание программного кода по зонам ответственности** 9](#_Toc162671888)

[**Раздел 5. Тестирование ПП** 11](#_Toc162671889)

[**Заключение** 12](#_Toc162671890)

[**Приложение 1 (Main.java)** 13](#_Toc162671891)

[**Приложение 2 (Controllers)** 14](#_Toc162671892)

[**Приложение 3 (UnitTest)** 22](#_Toc162671893)

[**Приложение 4 (Calculator)** 23](#_Toc162671894)

[**Приложение 5 (View)** 25](#_Toc162671895)

[**Список использованной литературы** 29](#_Toc162671896)

# **Раздел 1. Описание предметной области**

Курсовая работа посвящена разработке калькулятора стоимости строительства дома. Цель работы заключается в создании программного инструмента, который позволяет пользователям оценить приблизительную стоимость строительства дома в зависимости от различных параметров проектирования и выбора материалов.

В предметной области строительства дома важным является учет различных факторов, таких как площадь участка, размеры дома, количество этажей, тип крыши, материалы стен и фундамента. Данные параметры оказывают существенное влияние на общую стоимость строительства. Поэтому разработка калькулятора, способного учитывать все эти факторы, имеет высокую практическую значимость для застройщиков, девелоперов, архитекторов и частных лиц, интересующихся строительством дома.

Курсовая работа включает в себя анализ требований пользователей к функциональности калькулятора, проектирование архитектуры приложения, разработку интерфейса пользователя с использованием технологии JavaFX, а также реализацию логики расчета стоимости строительства на основе введенных пользователем данных.

Основные функции калькулятора включают в себя:

* Ввод пользователем основных параметров дома, таких как размеры, количество этажей, тип крыши, материалы стен и фундамента.
* Расчет приблизительной стоимости строительства на основе введенных данных.
* Вывод результатов расчета на экран для пользователя.

Предметная область курсовой работы охватывает ключевые аспекты процесса строительства дома и предлагает решение в виде программного инструмента для удобного и быстрого определения стоимости проекта.

# **Раздел 2. Описание прототипа программного продукта (ПП): Калькулятор строительства дома**

Название: Калькулятор строительства дома (прототип)

Цель: Целью данного программного продукта является создание прототипа калькулятора строительства дома, который позволит пользователям оценить примерную стоимость строительства жилого здания на основе заданных параметров.

Основные функции:

* + - 1. Ввод данных:
* Пользователь может ввести основные параметры дома, такие как длина, ширина, высота потолков.
* Выбрать количество этажей из предоставленных вариантов.
* Выбрать материалы кровли, стен и тип фундамента из предоставленных вариантов.
  + - 1. Расчет стоимости:
* Производится расчет общей стоимости строительства дома на основе введенных данных.
* Расчет включает в себя учет выбранных материалов и размеров дома.
  + - 1. Визуализация результатов:
* Пользователю отображается расчетная стоимость строительства.
* Показывается детализация выбранных параметров, таких как размеры дома, количество этажей, материалы и тип фундамента.

1. Управление данными:

* Пользователь может корректировать введенные параметры и повторно производить расчет стоимости.

1. Удобство использования:

* Интерфейс калькулятора предоставляет простой и интуитивно понятный способ ввода данных и получения результатов.

1. Технические особенности прототипа:

* Программный продукт разработан на языке Java с использованием JavaFX для создания графического интерфейса.
* Прототип обладает минимальным набором функций, не включая в себя расширенные опции и дополнительные возможности, присущие полноценным приложениям.
* Для простоты и удобства тестирования и демонстрации прототип представляет собой основную функциональность без дополнительных деталей.
* Интерфейс прототипа ориентирован на простоту использования и ясность представления информации.

# **Раздел 3. Математическая модель работы ПП**

Калькулятор стоимости строительства дома представляет собой программный инструмент, предназначенный для расчета приблизительной стоимости строительства дома на основе введенных пользователем параметров. Математическая модель данного программного продукта основана на следующих основных принципах:

* + - 1. Вычисление площади дома:
* Площадь дома вычисляется как произведение его длины на ширину.
  + - 1. Вычисление периметра стен:
* Периметр стен определяется как сумма периметров всех этажей дома, где периметр каждого этажа равен сумме длины и ширины, умноженной на 2, умноженную на количество этажей.
  + - 1. Вычисление стоимости фундамента:
* Стоимость фундамента зависит от выбранного типа фундамента. Для свайного фундамента стоимость рассчитывается как произведение площади дома на стоимость одной сваи, а для ленточного фундамента - как произведение площади дома на стоимость одного метра ленты.
  + - 1. Вычисление стоимости стен:
* Стоимость стен зависит от выбранного материала стен. Для каждого материала (сруб, газоблок, кирпич) рассчитывается стоимость на основе периметра стен, умноженного на стоимость одного метра погонного.
  + - 1. Вычисление стоимости крыши:
* Стоимость крыши зависит от выбранного типа крыши. Для каждого типа крыши (односкатная, двускатная, прямая) рассчитывается стоимость на основе площади крыши, умноженной на стоимость одного квадратного метра.
  + - 1. Вычисление общей стоимости:
* Общая стоимость строительства дома определяется как сумма стоимости фундамента, стен и крыши. Эта математическая модель обеспечивает точные и надежные расчеты стоимости строительства дома на основе введенных пользователем параметров, что делает калькулятор удобным и полезным инструментом при планировании строительства.

# **Раздел 4. Написание программного кода по зонам ответственности**

Зона ответственности определяет, кто из участников является исполнителем той или иной задачи при написании программного кода и тестировании.

Разработка ПО ведется на языке высокого уровня Java с использованием инструментальной среды Eclipse.

Таблица 1 — Зоны ответственности

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **ФИО азработчика/ модератора** | **Зона ответственности** | **№ приложения** |
| 1 | Макуха Тимур Борисович | Полная разработка программного кода | см. Приложение П-1-5 |

Программный код расположен в приложении

Инструкция по подключению к git:

1. Открыть в терминале папку .ssh
2. Создать пару ключей через команду ssh-keygen -t rsa -b 4096 -C «[email@mail.ru](mailto:email@mail.ru)», далее следовать инструкции в терминале
3. Открыть настройки профиля github
4. Открыть вкладку ssh and gpg keys
5. Нажать «New SSH key»
6. Вставить сгенерированный из шага 2 публичный ключ (находится в файле .pub)
7. Сохранить ключ
8. Скопировать ssh link на репозиторий
9. Открыть в терминале папку в которой будет храниться проекта
10. Ввести команду git clone «ssh link»

# **Раздел 5. Тестирование ПП**

При создании программного обеспечения основной целью является реализация заявленных функций и обеспечение высокого качества продукта. Качество ПО определяется набором его характеристик, направленных на удовлетворение потребностей пользователей. Тестирование ПО представляет собой процесс проверки соответствия реального поведения программы ожидаемому, осуществляемый с помощью определенного набора тестов. В рамках курсовой работы тестирование включает разработку и использование UNIT-тестов, каждый из участников группы отвечает за разработку одного UNIT-теста для своей части программы с использованием фреймворка Junit.

Таблица 2 – Описание Unit-тестов

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **ФИО азработчика/ модератора** | **Описание Unit-теста** | **№ приложения** |
| 1 | Макуха Тимур Борисович | Проверяет работу метода расчета стоимости | см. Приложение П-4 |

Программный код тестов расположен в приложении.

# **Заключение**

В ходе выполнения данной курсовой работы был разработан калькулятор стоимости строительства дома, представляющий собой программный инструмент для оценки приблизительной стоимости строительства на основе введенных пользователем параметров. Реализация данного калькулятора позволяет пользователям быстро и удобно определить общую стоимость строительства дома, учитывая различные факторы, такие как размеры дома, количество этажей, тип крыши и выбранные материалы стен и фундамента.

Математическая модель калькулятора была разработана на основе принципов арифметических вычислений, учитывающих основные параметры строительства. Путем использования этой модели пользователь может получить быструю и точную оценку стоимости строительства, что делает калькулятор полезным инструментом для застройщиков, девелоперов, архитекторов и частных лиц, занимающихся планированием строительства дома.

В ходе работы был проведен анализ требований пользователей к функциональности калькулятора, спроектирована архитектура приложения, разработан пользовательский интерфейс с использованием технологии JavaFX, а также реализована логика расчета стоимости строительства.

Основываясь на результатах работы, можно сделать вывод, что разработанный калькулятор стоимости строительства дома является эффективным инструментом для оценки затрат на строительство и может быть широко использован в практических целях. Дальнейшее развитие проекта может включать в себя расширение функциональности калькулятора, добавление новых параметров и улучшение пользовательского интерфейса для повышения удобства использования.

# **Приложение 1 (Main.java)**

package application;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

public class Main extends Application {

@Override

public void start(Stage primaryStage) throws Exception {

FXMLLoader loader = new FXMLLoader(getClass().getResource("/view/HomePage.fxml"));

Parent root = loader.load();

primaryStage.setTitle("Калькулятор стоимости строительства дома (примерный)");

primaryStage.setScene(new Scene(root, 600, 400));

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

# **Приложение 2 (Controllers)**

package controllers;

import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.AnchorPane;

import javafx.stage.Stage;

import java.io.IOException;

public abstract class BaseController {

public void initializeExitButton(Button exitBtn) {

exitBtn.setOnAction(event -> System.exit(0));

}

protected void initializeGoBackButton(Button goBackBtn, String fxmlPath, Button sourceButton) {

goBackBtn.setOnAction(event -> {

FXMLLoader loader = new FXMLLoader(getClass().getResource(fxmlPath));

try {

AnchorPane previousPage = (AnchorPane) loader.load();

Scene scene = new Scene(previousPage);

Stage stage = (Stage) sourceButton.getScene().getWindow();

stage.setScene(scene);

stage.show();

} catch (IOException e) {

e.printStackTrace();

}

});

}

}

package controllers;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

public class CalcPageController extends BaseController{

@FXML

private Button exitBtn;

@FXML

private Button goBackBtn;

@FXML

private Label totalPriceLabel;

@FXML

private Label lengthLabel;

@FXML

private Label widthLabel;

@FXML

private Label floorsCountLabel;

@FXML

private Label squareLabel;

@FXML

private Label roofLabel;

@FXML

private Label wallLabel;

@FXML

private Label foundationLabel;

@FXML

private void initialize() {

// listener exit button

initializeExitButton(exitBtn);

// listener go back button

initializeGoBackButton(goBackBtn, "/view/HomePage.fxml", goBackBtn);

}

public void updateLengthOutput(double length) {

lengthLabel.setText(String.format("Длина: %.2f", length));

}

public void updateWidthOutput(double width) {

widthLabel.setText(String.format("Ширина: %.2f", width));

}

public void updateFloorsCountOutput(int floorsCount) {

floorsCountLabel.setText(String.format("Количество этажей: %d", floorsCount));

}

public void updateSquareOutput(double square) {

squareLabel.setText(String.format("Площадь фундамента: %.2f", square));

}

public void updateRoofOutput(String roofType) {

roofLabel.setText(String.format("Тип крыши: %s", roofType));

}

public void updateWallOutput(String wallType) {

wallLabel.setText(String.format("Тип стен: %s", wallType));

}

public void updateFoundationOutput(String foundationType) {

foundationLabel.setText(String.format("Тип фундамента: %s", foundationType));

}

// set total price from home page calculate method

public void updateTotalPrice(double totalPrice) {

totalPriceLabel.setText(String.format("Итоговая сумма: %.2f", totalPrice));

}

}

package controllers;

import java.io.IOException;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.ChoiceBox;

import javafx.scene.control.RadioButton;

import javafx.scene.control.TextField;

import javafx.scene.layout.AnchorPane;

import utils.HouseCalculator;

public class HomePageController extends BaseController {

@FXML

private Button exitBtn;

@FXML

private Button calculationBtn;

@FXML

private TextField houseLenghtInput;

@FXML

private TextField houseWidthInput;

@FXML

private TextField ceilingHeightInput;

@FXML

private RadioButton oneFloorRadioBtn;

@FXML

private RadioButton twoFloorsRadioBtn;

@FXML

private ChoiceBox<String> roofMaterialChoiceBox;

@FXML

private ChoiceBox<String> wallMaterialChoiceBox;

@FXML

private ChoiceBox<String> foundationTypeChoiceBox;

@FXML

private void initialize() {

// choice boxes data

roofMaterialChoiceBox.getItems().addAll("прямая", "односкатная", "двускатная");

wallMaterialChoiceBox.getItems().addAll("сруб", "кирпич", "газоблок");

foundationTypeChoiceBox.getItems().addAll("свайный", "ленточный");

// setup default house data

houseLenghtInput.setText("0");

houseWidthInput.setText("0");

ceilingHeightInput.setText("0");

// exit button listener

initializeExitButton(exitBtn);

// radio buttons listener

oneFloorRadioBtn.setOnAction(this::handleOneFloorSelected);

twoFloorsRadioBtn.setOnAction(this::handleTwoFloorsSelected);

}

/\*\*

\* Calculation home price

\* @param event

\*/

@FXML

private void handleCalculation(ActionEvent event) {

// Check all fields filled

if (houseLenghtInput.getText().isEmpty() || houseWidthInput.getText().isEmpty() || ceilingHeightInput.getText().isEmpty() ||

roofMaterialChoiceBox.getValue() == null || wallMaterialChoiceBox.getValue() == null || foundationTypeChoiceBox.getValue() == null) {

// Throw error if any field is not filled

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Ошибка");

alert.setHeaderText(null);

alert.setContentText("Заполните все обязательные поля!");

alert.showAndWait();

return;

}

// if all fields filled go calculation

double length = Double.parseDouble(houseLenghtInput.getText());

double width = Double.parseDouble(houseWidthInput.getText());

int floorCount = oneFloorRadioBtn.isSelected() ? 1 : 2;

String roofMaterial = roofMaterialChoiceBox.getValue();

String wallMaterial = wallMaterialChoiceBox.getValue();

String foundationType = foundationTypeChoiceBox.getValue();

double totalPrice = HouseCalculator.calculateTotalPrice(length, width, floorCount, roofMaterial, wallMaterial, foundationType);

try {

FXMLLoader loader = new FXMLLoader(getClass().getResource("/view/CalcPage.fxml"));

Parent calcPageParent = loader.load();

CalcPageController calcPageController = loader.getController();

calcPageController.updateLengthOutput(length);

calcPageController.updateWidthOutput(width);

calcPageController.updateFloorsCountOutput(floorCount);

calcPageController.updateSquareOutput(length \* width);

calcPageController.updateRoofOutput(roofMaterial);

calcPageController.updateWallOutput(wallMaterial);

calcPageController.updateFoundationOutput(foundationType);

calcPageController.updateTotalPrice(totalPrice);

AnchorPane root = (AnchorPane) calculationBtn.getScene().getRoot();

root.getChildren().setAll(calcPageParent.getChildrenUnmodifiable());

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* unselect other radiobutton when this was selected

\* @param event

\*/

private void handleOneFloorSelected(ActionEvent event) {

twoFloorsRadioBtn.setSelected(false);

}

/\*\*

\* unselect other radiobutton when this was selected

\* @param event

\*/

private void handleTwoFloorsSelected(ActionEvent event) {

oneFloorRadioBtn.setSelected(false);

}

}

# **Приложение 3 (UnitTest)**

package test;

import static org.junit.Assert.\*;

import org.junit.Test;

import utils.HouseCalculator;

public class UnitTest {

@Test

public void testCalculateTotalPrice() {

double length = 10.0;

double width = 8.0;

int floorCount = 2;

String roofMaterial = "односкатная";

String wallMaterial = "сруб";

String foundationType = "Свайный";

double expectedTotalPrice = 216000 + 240000 + 160000;

double actualTotalPrice = HouseCalculator.calculateTotalPrice(length, width, floorCount, roofMaterial, wallMaterial, foundationType);

assertEquals(expectedTotalPrice, actualTotalPrice, 0.01);

}

}

# **Приложение 4 (Calculator)**

package utils;

public class HouseCalculator {

public static double calculateTotalPrice(double length, double width, int floorCount, String roofMaterial, String wallMaterial, String foundationType) {

// Вычисление площади дома

double area = length \* width;

// Вычисление периметра стен

double wallPerimeter = (length + width) \* 2 \* floorCount;

// Вычисление стоимости фундамента

double foundationPrice = foundationType.equals("Свайный") ? 2000 \* area : 1600 \* area;

// Вычисление стоимости стен

double wallPrice = calculateWallPrice(wallMaterial, wallPerimeter);

// Вычисление стоимости крыши

double roofPrice = calculateRoofPrice(roofMaterial, area);

// Вычисление общей стоимости

return roofPrice + wallPrice + foundationPrice;

}

private static double calculateWallPrice(String wallMaterial, double wallPerimeter) {

switch (wallMaterial) {

case "сруб":

return wallPerimeter \* 3000;

case "газоблок":

return wallPerimeter \* 700;

case "кирпич":

return wallPerimeter \* 5000;

default:

return 0;

}

}

private static double calculateRoofPrice(String roofMaterial, double area) {

switch (roofMaterial) {

case "односкатная":

return area \* 3000;

case "двускатная":

return area \* 5000;

case "прямая":

return area \* 7000;

default:

return 0;

}

}

}

# **Приложение 5 (View)**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controllers.CalcPageController">

<children>

<Button fx:id="exitBtn" layoutX="14.0" layoutY="362.0" mnemonicParsing="false" text="Выход" />

<Button fx:id="goBackBtn" layoutX="14.0" layoutY="14.0" mnemonicParsing="false" text="&lt;" />

<Label layoutX="115.0" layoutY="56.0" text="Расчет стоимости строительства дома">

<font>

<Font size="18.0" />

</font>

</Label>

<Label fx:id="totalPriceLabel" layoutX="134.0" layoutY="298.0" />

<Label layoutX="116.0" layoutY="107.0" text="Параметры дома:" />

<Label layoutX="352.0" layoutY="107.0" text="Типы конструкции:" />

<Label fx:id="lengthLabel" layoutX="116.0" layoutY="140.0" />

<Label fx:id="widthLabel" layoutX="116.0" layoutY="167.0" />

<Label fx:id="floorsCountLabel" layoutX="116.0" layoutY="192.0" />

<Label fx:id="squareLabel" layoutX="116.0" layoutY="218.0" />

<Label fx:id="roofLabel" layoutX="352.0" layoutY="140.0" />

<Label fx:id="wallLabel" layoutX="352.0" layoutY="167.0" />

<Label fx:id="foundationLabel" layoutX="352.0" layoutY="192.0" />

</children>

</AnchorPane>

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ChoiceBox?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.RadioButton?>

<?import javafx.scene.control.Tab?>

<?import javafx.scene.control.TabPane?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controllers.HomePageController">

<children>

<TabPane prefHeight="400.0" prefWidth="600.0" side="RIGHT" tabClosingPolicy="UNAVAILABLE">

<tabs>

<Tab fx:id="homeTab" text="Дом">

<content>

<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0">

<children>

<Button fx:id="exitBtn" layoutX="14.0" layoutY="362.0" mnemonicParsing="false" text="Выход" />

<Button fx:id="calculationBtn" layoutX="459.0" layoutY="362.0" mnemonicParsing="false" onAction="#handleCalculation" text="Рассчитать" />

<Label layoutX="76.0" layoutY="120.0" text="Длина м." />

<Label layoutX="76.0" layoutY="79.0" text="Параметры дома" />

<Label layoutX="76.0" layoutY="155.0" text="Ширина м." />

<Label layoutX="76.0" layoutY="193.0" text="Высота потолков" />

<Label layoutX="71.0" layoutY="228.0" text="Количество этажей" />

<TextField fx:id="houseLenghtInput" layoutX="218.0" layoutY="116.0" />

<TextField fx:id="houseWidthInput" layoutX="218.0" layoutY="151.0" />

<TextField fx:id="ceilingHeightInput" layoutX="218.0" layoutY="189.0" />

<RadioButton fx:id="oneFloorRadioBtn" contentDisplay="TOP" layoutX="239.0" layoutY="227.0" mnemonicParsing="false" prefHeight="11.0" prefWidth="18.0" selected="true" />

<RadioButton fx:id="twoFloorsRadioBtn" contentDisplay="TOP" layoutX="295.0" layoutY="227.0" mnemonicParsing="false" prefHeight="11.0" prefWidth="18.0" />

<Label layoutX="244.0" layoutY="251.0" text="1" />

<Label layoutX="300.0" layoutY="251.0" text="2" />

</children>

</AnchorPane>

</content>

</Tab>

<Tab fx:id="materialTab" text="Материалы">

<content>

<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0">

<children>

<Button fx:id="exitBtn" layoutX="14.0" layoutY="362.0" mnemonicParsing="false" text="Выход" />

<Button fx:id="calculationBtn" layoutX="459.0" layoutY="362.0" mnemonicParsing="false" onAction="#handleCalculation" text="Рассчитать" />

<Label layoutX="76.0" layoutY="96.0" text="Крыша" />

<Label alignment="CENTER" layoutX="246.0" layoutY="23.0" text="Материалы" />

<Label layoutX="78.0" layoutY="142.0" text="Стены" />

<Label layoutX="76.0" layoutY="192.0" text="Фундамент" />

<ChoiceBox fx:id="wallMaterialChoiceBox" layoutX="210.0" layoutY="138.0" prefWidth="150.0" />

<ChoiceBox fx:id="roofMaterialChoiceBox" layoutX="210.0" layoutY="92.0" prefWidth="150.0" />

<ChoiceBox fx:id="foundationTypeChoiceBox" layoutX="210.0" layoutY="188.0" prefWidth="150.0" />

</children>

</AnchorPane>

</content>

</Tab>

</tabs>

</TabPane>

</children>

</AnchorPane>

# **Список использованной литературы**

1. Свиридов А.В., Карцев Д.В. "Тестирование программного обеспечения с использованием JUnit". - М.: ДМК Пресс, 2015.
2. Чижов В.И., Рыжков В.В. "Автоматизированное тестирование программного обеспечения с использованием JUnit". - СПб.: БХВ-Петербург, 2004.
3. Бердоносов, В. Д. Объектно-ориентированное программирование: учеб. пособие / В. Д. Бердоносов, А. А. Животова. – Комсомольск-на-Амуре: ФГБОУ ВПО «КнАГТУ», 2015. – 135 с.
4. Тарасенко Д.А. "Программирование на языке Java: Лабораторный практикум. В 4 частях. Часть 2: Управление тестированием исходного кода средствами JUnit". - М.: Федоров, 2005.
5. Климов В.В., Лимаренко В.И. "Java. Тестирование программного обеспечения". - М.: Вильямс, 2018.
6. Машнин, Т. С. Eclipse: разработка RCP-, Web-, Ajax- и Android-приложений на Java / Т. С. Машнин. — Санкт-Петербург: БХВ- Петербург, 2013.— 384 с.
7. Литвинов В.Ю. "JUnit: Профессиональное программирование на Java". - СПб.: Питер, 2006.
8. Кулаков, К. А. Основы тестирования программного обеспечения [Электронный ресурс]: учебное электронное пособие для обучающихся Института математики и информационных технологий / К. А. Кулаков, В. М. Димитров; М-во образования и науки Рос. Федерации, Федер. гос. бюджет. образоват. учреждение высш. образования Петрозав. гос. ун-т. - Электрон. текст дан. – Петрозаводск: Изд-во ПетрГУ, 2018.