1. Начните создание проекта. Выберите пункт Create New Project или пункт меню File->New->Project.
2. В появившемся окне слева выберите тип проекта Gradle. Справа должна быть отмечена Java. Нажмите кнопку Next.
3. В появившемся окне заполните информацию о проекте: имя проекта, например ExampleFX. Затем раскройте информацию об артефакте, Artifact coodrinates и укажите дополнительные параметры: GroupId. Например example. Нажмите на кнопку Finish.
4. В окне с построенным проектом слева в структуре проекта раскройте папки src, main, java. На папке java вызовите контекстное меню, нажав правую кнопку мыши и выберите пункт меню New->Java Class. В появившемся окне введите имя класса example.Main и нажмите Enter.
5. Напишите код приложения в классе Main (после объявления пакета package ...;):
6. import javafx.application.Application;
7. import javafx.geometry.Pos;
8. import javafx.scene.Scene;
9. import javafx.scene.control.Label;
10. import javafx.scene.layout.HBox;
11. import javafx.stage.Stage;
12. public class Main extends Application {
13. @Override
14. public void start(Stage primaryStage) throws Exception {
15. Label label = new Label("First example JavaFX");
16. HBox box = new HBox();
17. box.setAlignment(Pos.*CENTER*);
18. box.getChildren().add(label);
19. Scene scene = new Scene(box, 400, 400);
20. primaryStage.setScene(scene);
21. primaryStage.show();
22. }
23. public static void main (String[] args) {
24. *launch*();
25. }

}

1. Загрузите изменения из build.gradle-файла в IDEA. Откройте панель Gradle справа. Нажмите на значок из вращающихся стрелочек
2. Запустите проект. В панели Gradle выберите пункт Tasks->Application->Run. Нажмите на него 2 раза.

При первых запусках IntelliJ IDEA, Gradle или приложения время работы может быть больше, чем при последующих.

Особенности настройки Gradle можно дополнительно посмотреть в документации на [openjfx.io](https://openjfx.io/openjfx-docs/#gradle). А также [пример первого приложения JavaFX от разработчиков](https://docs.oracle.com/javase/8/javafx/get-started-tutorial/hello_world.htm).

Не забудьте добавить к проекту новый класс example.Main (слева в структуре проекта в разделе java)

**Пример однооконного графического приложения с вводом данных**

Приведём код приложения. В методе start(Stage primaryStage) создаются корневой элемент root сцены, сцена с размерами scene, а также задаются настройки главного окна приложения primaryStage. Создание и расположение компонентов сцены описано в отдельном методе initRootLayout(), результатом работы которого является готовый объект - корневой элемент сцены - панель GridPane. В целом код приложения выглядит так.

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.scene.text.Text;

import javafx.stage.Stage;

import java.util.Calendar;

import java.util.Locale;

import java.util.TimeZone;

import java.util.Date;

public class Main extends Application {

public static GridPane initRootLayout(){

GridPane root = new GridPane();

root.setAlignment(Pos.*CENTER*);

root.setHgap(10);

root.setVgap(10);

root.setPadding(new Insets(25));

Text scenetitle = new Text("Calculation of age");

Font fontBig = Font.*font*("Tahoma", FontWeight.*NORMAL*, 26);

scenetitle.setFont(fontBig);

root.add(scenetitle, 0, 0, 2, 1);

Label userName = new Label("Name:");

userName.setFont(fontBig);

root.add(userName, 0, 1);

TextField nameTextField = new TextField();

nameTextField.setFont(fontBig);

root.add(nameTextField, 1, 1);

Label userYear = new Label("Year:");

userYear.setFont(fontBig);

root.add(userYear, 0, 2);

TextField valueYear = new TextField();

valueYear.setFont(fontBig);

root.add(valueYear, 1, 2);

Text resoult = new Text("Resoult");

resoult.setFont(fontBig);

root.add(resoult, 0, 6);

Button btn = new Button("OK");

btn.setFont(fontBig);

btn.setOnAction(event -> {

Calendar calendar = Calendar.*getInstance*(TimeZone.*getDefault*(), Locale.*getDefault*());

calendar.setTime(new Date());

String year = valueYear.getText();

if(year.matches("[0-9]+")) {

int age = calendar.get(Calendar.*YEAR*) - Integer.*parseInt*(year);

resoult.setText(nameTextField.getText()+ " is "+ age + " years old");

}

else {

resoult.setFill(Color.*FIREBRICK*);

resoult.setText("Wrong year");

}

});

root.add(btn, 1, 5);

return root;

}

@Override

public void start(Stage primaryStage) {

GridPane root = *initRootLayout*();

Scene scene = new Scene(root, 600, 500);

primaryStage.setTitle("Age");

primaryStage.setScene(scene);

primaryStage.show();

}

public static void main(String[] args) {

*launch*();

}

}

## Лабораторная работа 1

Создайте однооконное приложение для расчёта калорийности куска вареной колбасы по его весу. Известно, что 100 грамм вареной колбасы содержат примерно 301 Ккал. В окне приложения расположите поле для ввода веса продукта, метку с соответствующим пояснением для пользователя, метку или текст для отображения результата и кнопку «Расчет калорийности». После нажатия на кнопку выведите сообщение о количестве ккал. Если ввод данных неверный, то сообщите пользователю об ошибке.

В поле ответа на задание введите название класса, который используется для создания кнопки.

Дополнительно: добавьте возможность выбора разных продуктов. Справочно: Индейка 198 ккал; Омлет 209 ккал; Вафли 543 ккал; Вишня 52 ккал; Огурцы 13 ккал.