Цель туториала – рассмотреть возможность создания executable джара 3 способами:

1. **Maven plugin**
2. **Intellij idea build artifact processing**
3. **Консоль**

В качестве примера/базового класса возьмем javaFX приложение, сам source программы будет выглядить следующим образом:

package textfieldsample;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Insets;

import javafx.scene.Group;

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.stage.Stage;

import org.apache.commons.lang.math.NumberUtils;

public class SimpleGUI extends Application {

@Override

public void start(Stage stage) {

Group root = new Group();

Scene scene = new Scene(root, 300, 150);

stage.setScene(scene);

stage.setTitle("Text Field Sample");

NumberUtils.isNumber("");

GridPane grid = new GridPane();

grid.setPadding(new Insets(10, 10, 10, 10));

grid.setVgap(5);

grid.setHgap(5);

scene.setRoot(grid);

final TextField name = new TextField();

name.setPromptText("Enter your first name.");

name.setPrefColumnCount(10);

name.getText();

GridPane.setConstraints(name, 0, 0);

grid.getChildren().add(name);

final TextField lastName = new TextField();

lastName.setPromptText("Enter your last name.");

GridPane.setConstraints(lastName, 0, 1);

grid.getChildren().add(lastName);

final TextField comment = new TextField();

comment.setPrefColumnCount(15);

comment.setPromptText("Enter your comment.");

GridPane.setConstraints(comment, 0, 2);

grid.getChildren().add(comment);

Button submit = new Button("Submit");

GridPane.setConstraints(submit, 1, 0);

grid.getChildren().add(submit);

Button clear = new Button("Clear");

GridPane.setConstraints(clear, 1, 1);

grid.getChildren().add(clear);

final Label label = new Label();

GridPane.setConstraints(label, 0, 3);

GridPane.setColumnSpan(label, 2);

grid.getChildren().add(label);

submit.setOnAction(new EventHandler<ActionEvent>() {

public void handle(ActionEvent e) {

if (

(comment.getText() != null && !comment.getText().isEmpty())

) {

label.setText(name.getText() + " " +

lastName.getText() + ", "

+ "thank you for your comment!");

} else {

label.setText("You have not left a comment.");

}

}

});

clear.setOnAction(new EventHandler<ActionEvent>() {

public void handle(ActionEvent e) {

name.clear();

lastName.clear();

comment.clear();

label.setText(null);

}

});

stage.show();

}

public static void main(String[] args) {

launch(args);

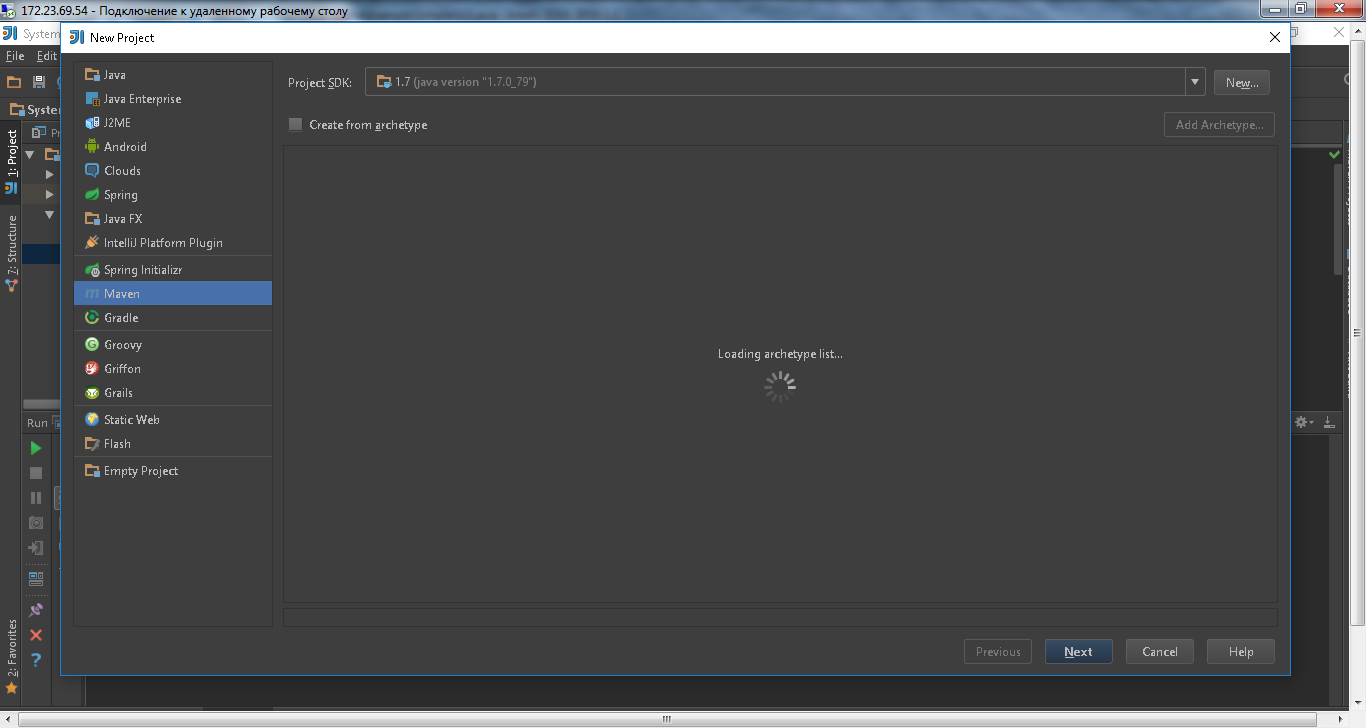
}

}

Основная идея программы – определение: является ли строка, введенная пользователем - числом или нет

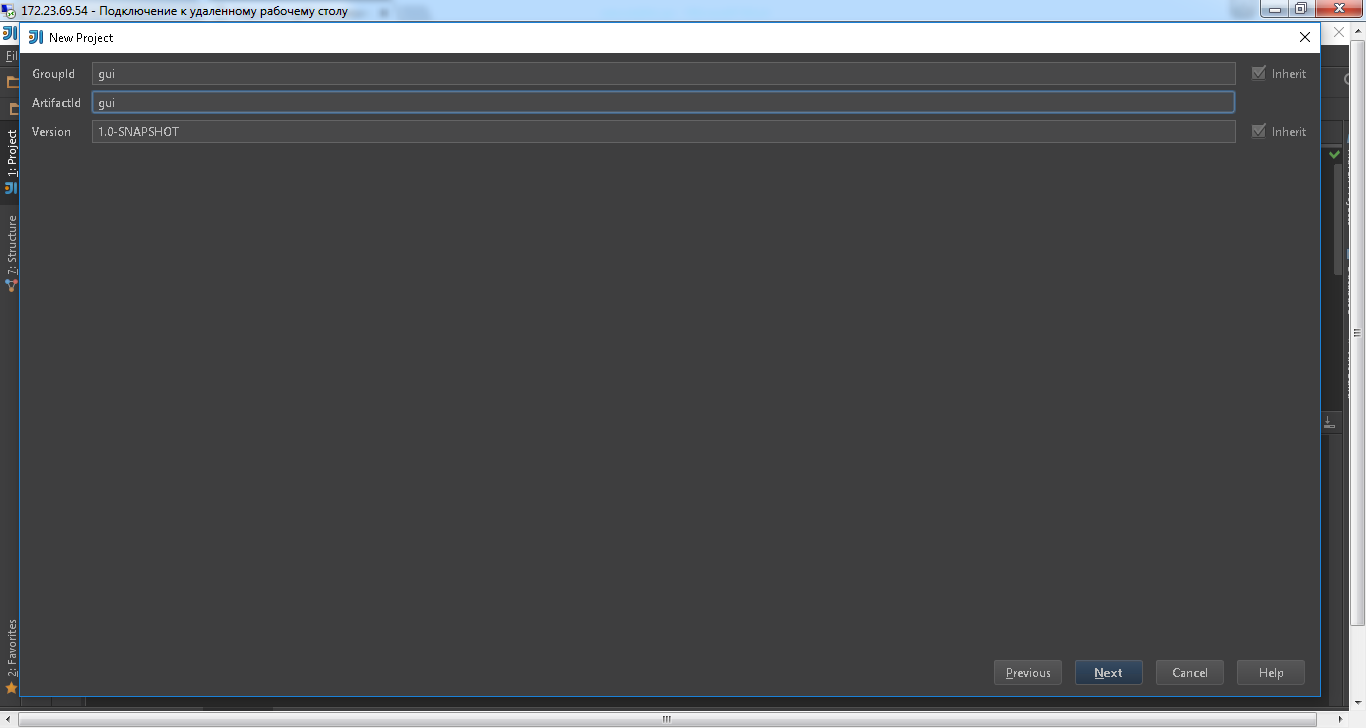
**Итак:**

* Запустим IDE -> Intellij IDEA
* File->New project

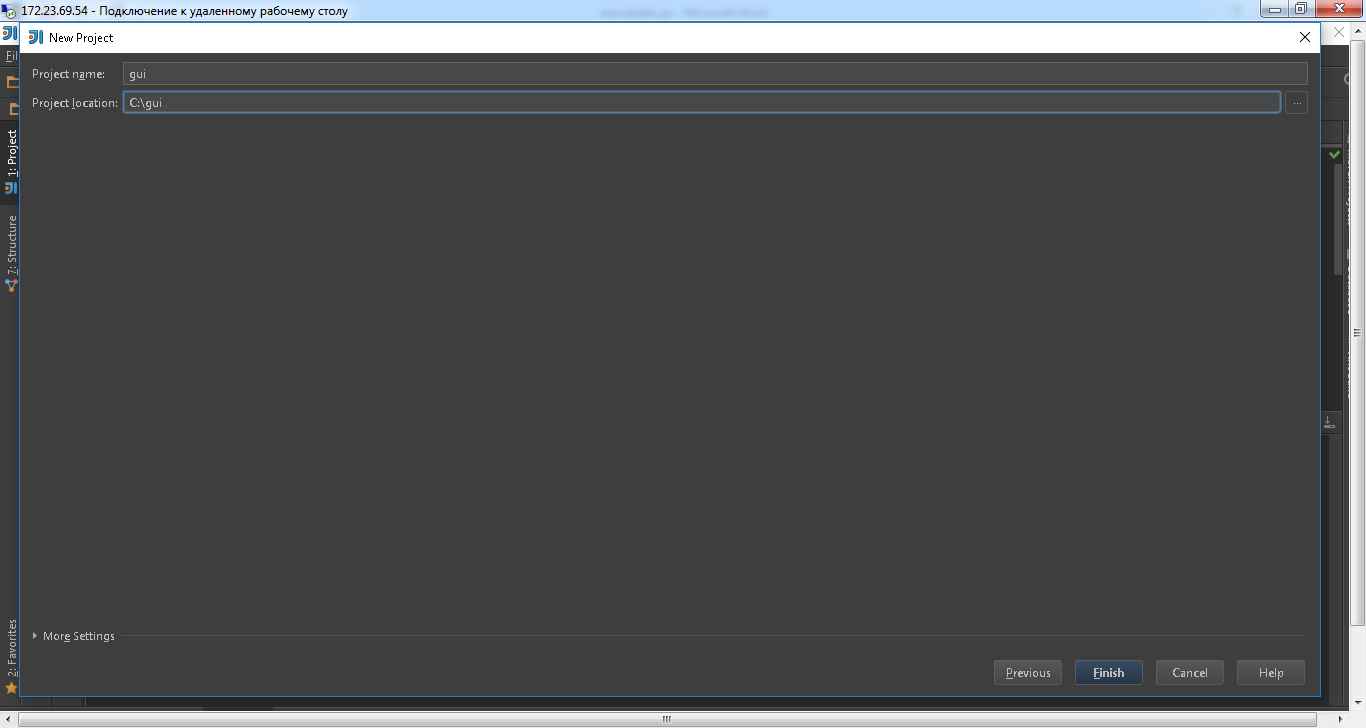


**GroupID -> gui**

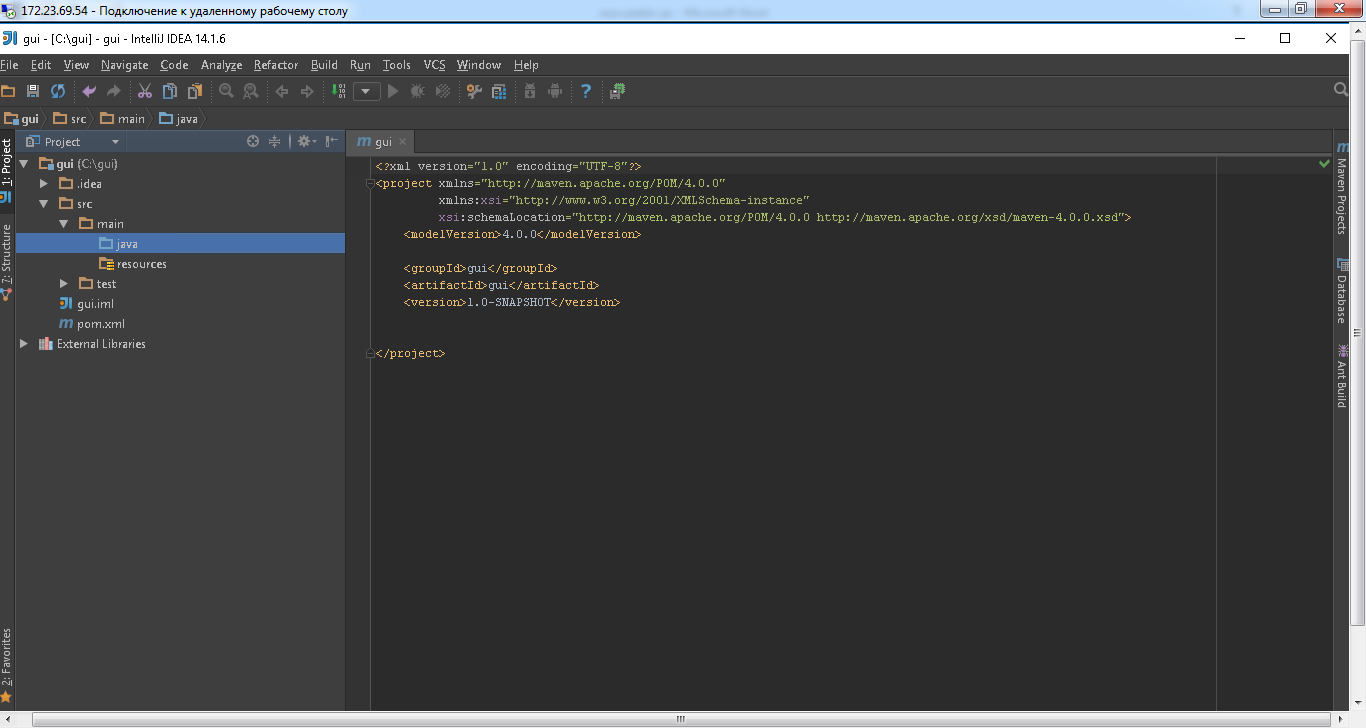
**ArtifactID -> gui**



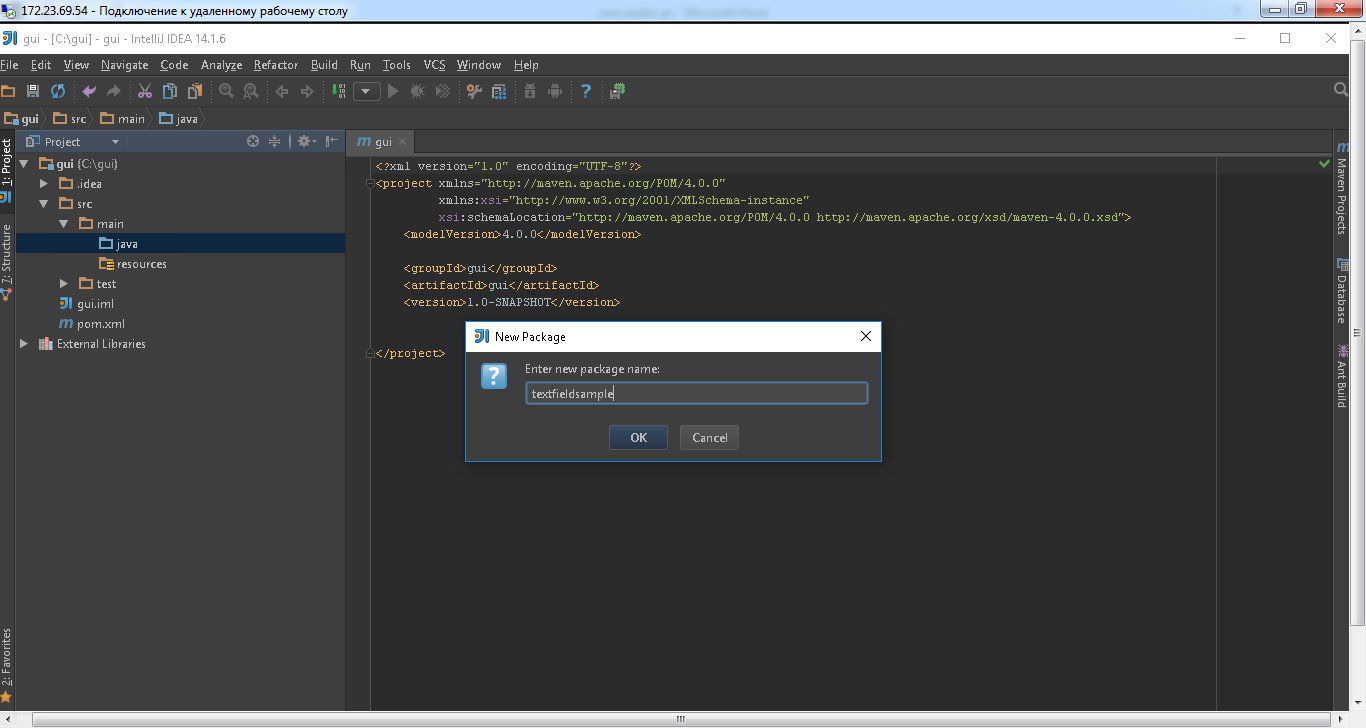
* **Project name -> gui**



Общая структура проекта

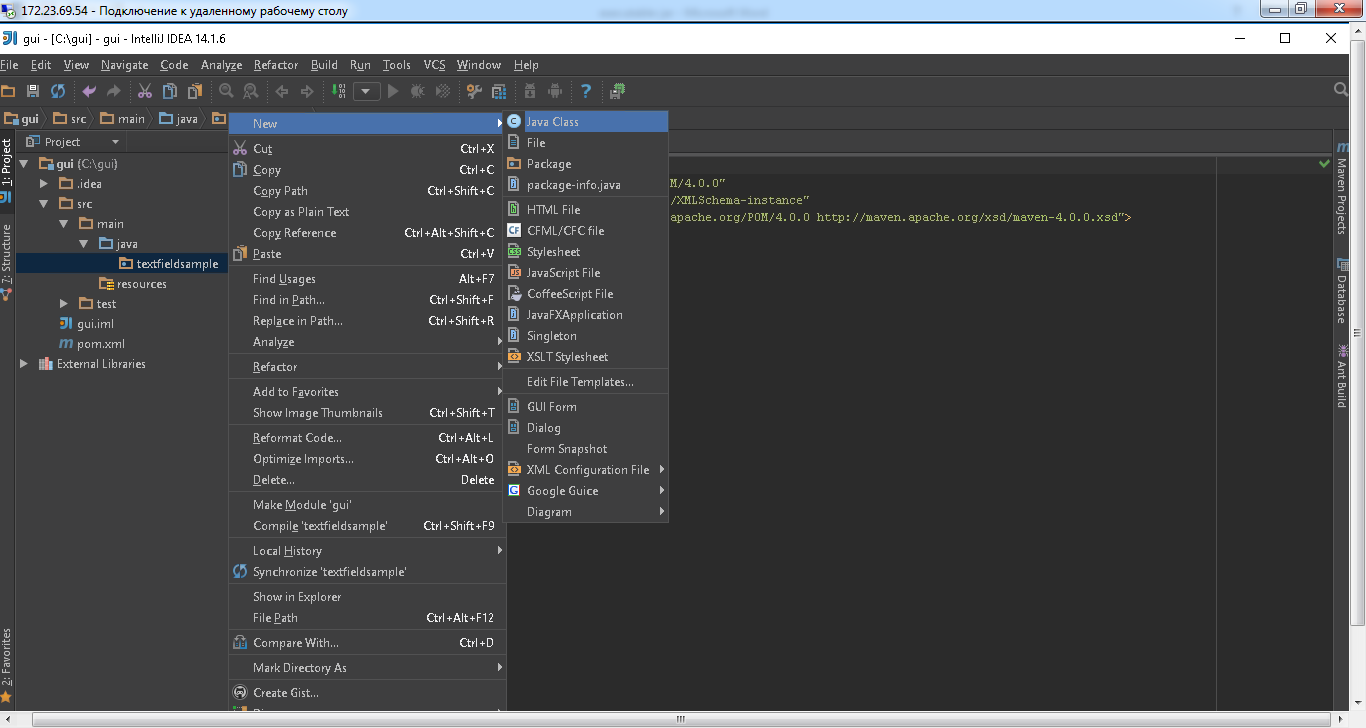


* Создадим пакет – в нашем случае, это будет **textfieldsample**



* Создадим класс SimpleGUI – в пределах нашего пакета

Правой кнопочкой щелкнем по нашему пакету (**textfieldsample**), затем New -> Java class ->



Прошу заметить, что для определения является ли введенная пользователе строка числом – мы юзаем NumberUtils – сторонний API, поэтому:

1. Проверьте **pom.xml** -> должна быть указана зависимость

На одну из версию джара <https://mvnrepository.com/artifact/commons-lang/commons-lang> ( в котором данный класс-утилита фигурирует)

в моем случае, это

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>gui</groupId>  
 <artifactId>gui</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <dependencies>  
 <dependency>  
 <groupId>commons-lang</groupId>  
 <artifactId>commons-lang</artifactId>  
 <version>2.3</version>  
 </dependency>  
 </dependencies>  
  
</project>

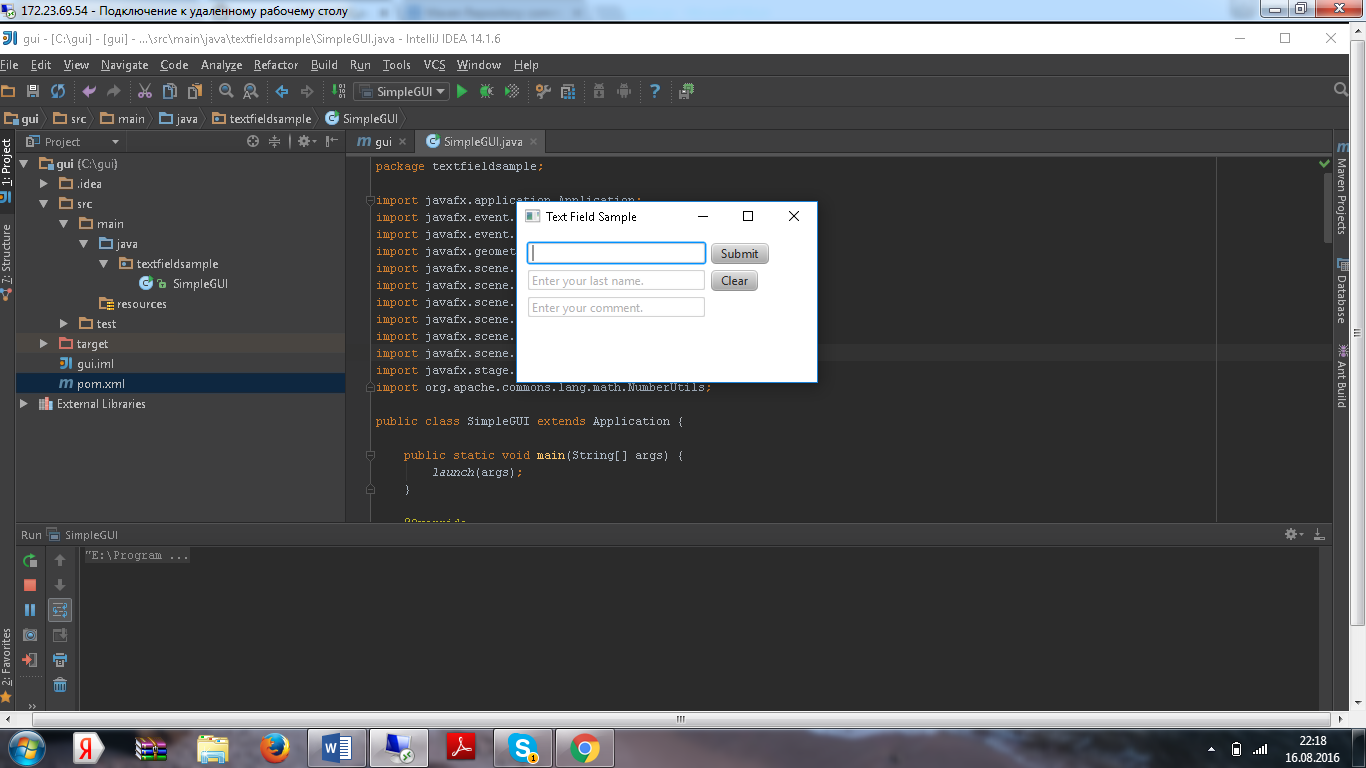
1. Проверьте также, что у Вас есть импорт утилиты в основном классе

import org.apache.commons.lang.math.NumberUtils;

Ок, с этим справились – молодцы ☺

Теперь -- попробуйте запустить программу из IDE ->

CTRL+SHIFT+F10



После проделанного, давайте попробуем вернуться к нашему плану

1. Создание exe джара с помощью мавен плагина

Для этого откроем **pom.xml**

И в пределах **build** тегов добавим следующее:

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<configuration>

<archive>

<manifest>

<mainClass>fully.qualified.MainClass</mainClass>

</manifest>

</archive>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

</configuration>

<executions>

<execution>

<id>make-assembly</id> <!-- this is used for inheritance merges -->

<phase>package</phase> <!-- bind to the packaging phase -->

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

</plugin>

В результате Вы должны получить(опять же к версии джара, указанного в депендеси мы не привязываемся: у каждого она своя) – следующ. pom.xml

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>gui</groupId>  
 <artifactId>gui</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <dependencies>  
 <dependency>  
 <groupId>commons-lang</groupId>  
 <artifactId>commons-lang</artifactId>  
 <version>2.3</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <artifactId>maven-assembly-plugin</artifactId>  
 <configuration>  
 <archive>  
 <manifest>  
 <mainClass> textfieldsample.SimpleGUI </mainClass>  
 </manifest>  
 </archive>  
 <descriptorRefs>  
 <descriptorRef>jar-with-dependencies</descriptorRef>  
 </descriptorRefs>  
 </configuration>  
 <executions>  
 <execution>  
 <id>make-assembly</id>  
 *<!-- this is used for inheritance merges -->* <phase>package</phase>  
 *<!-- bind to the packaging phase -->* <goals>  
 <goal>single</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

Что важно понимать – что после того, как Вы указали в качестве фазы сборки – package, Ваш джар будет создаваться после процессинга\выполнения команды

**mvn [clean] package**

**[clean] --** опционально

<executions>

<execution>

<id>make-assembly</id> <!-- this is used for inheritance merges -->

<phase>package</phase> <!-- bind to the packaging phase -->

<goals>

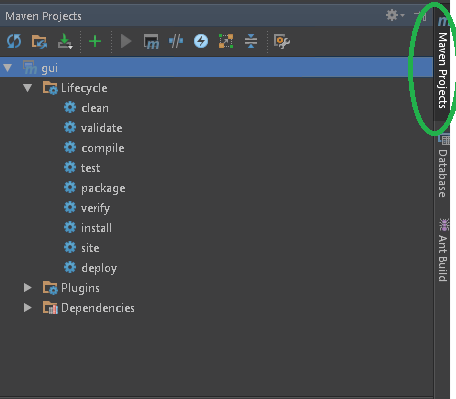
<goal>single</goal>

</goals>

</execution>

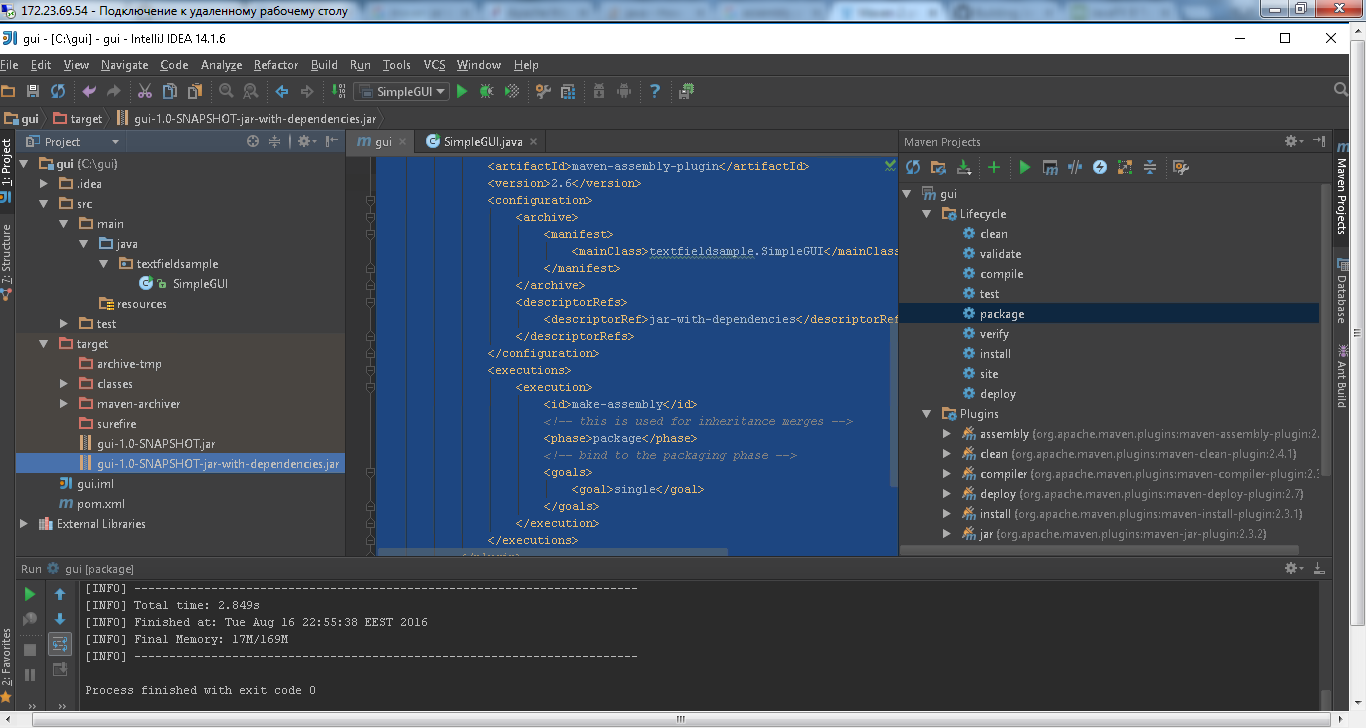
</executions>

Для запуска найдите слева



Затем кликните **clean** затем **package**

После – перейдите в **target** директорию



Попробуйте запустить джар с зависимостями

* java –jar ваш\_ехе\_джар
* javaw –jar ваш\_ехе\_джар
* двойным щелчком

важно понимать как вы запускаете ваш джар:

есть нюанс, если у вас в системе фигурирует 2-3 и больше пакетов JDK разных версий, ТО в случае:

- если качестве основной выставлена 7-ая, **НО** на уровне идеи выставлена допустим 8-ая (то-есть версия на уровне IDE и версия на уровне системы – разные) -> то запуск Вы должны осуществлять исходя из версии джавы, которую Вы юзали во время сборки джара**(IDE)**

в моем случае:

- Java 8 в IDE,

- JAVA\_HOME выставлена 7

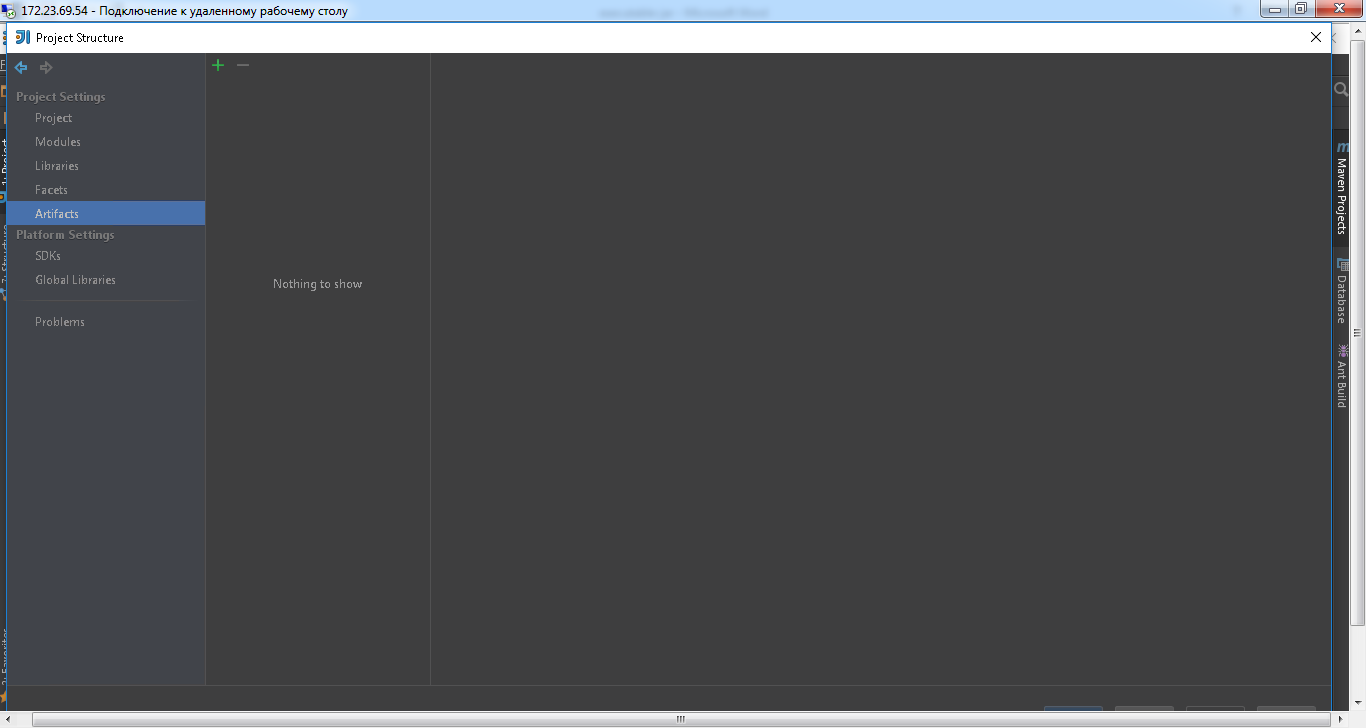
запуск из консольки – это использование ехе **8-ой Java**

E:\Program Files\Java**\jdk1.8.0\_45\**bin**>**

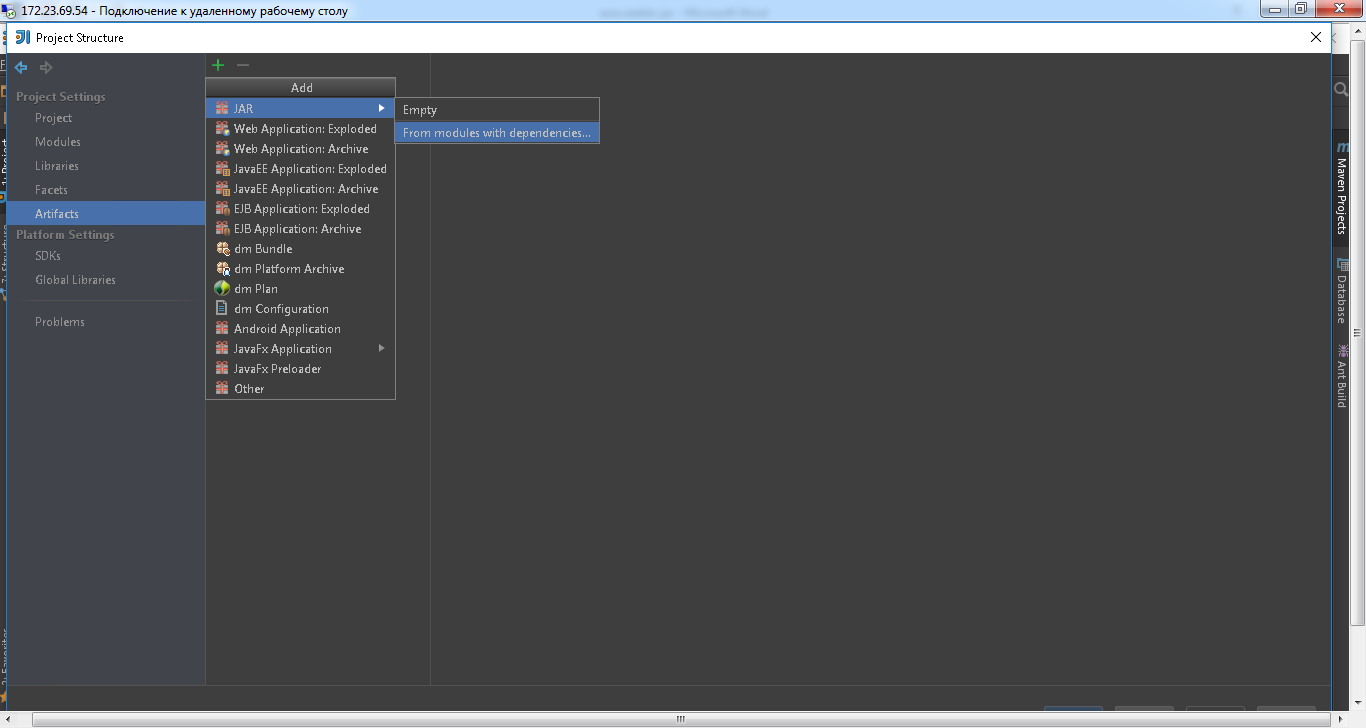
**java.exe -jar наш\_сгенерированный\_джар**

1. **Intellij idea build artifact processing**

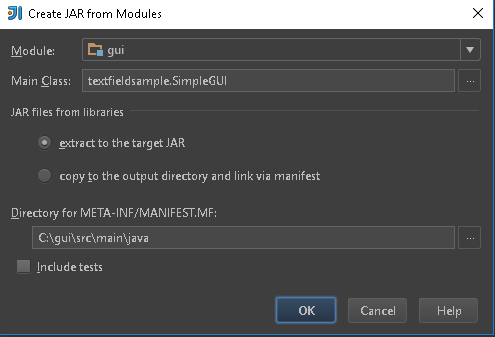
File -> Project structure -> Artifacts



Нажимаем на + -> **JAR -> module with dependencies**



Заполним параметры, исходя из нашей конфигурации/структуры проекта

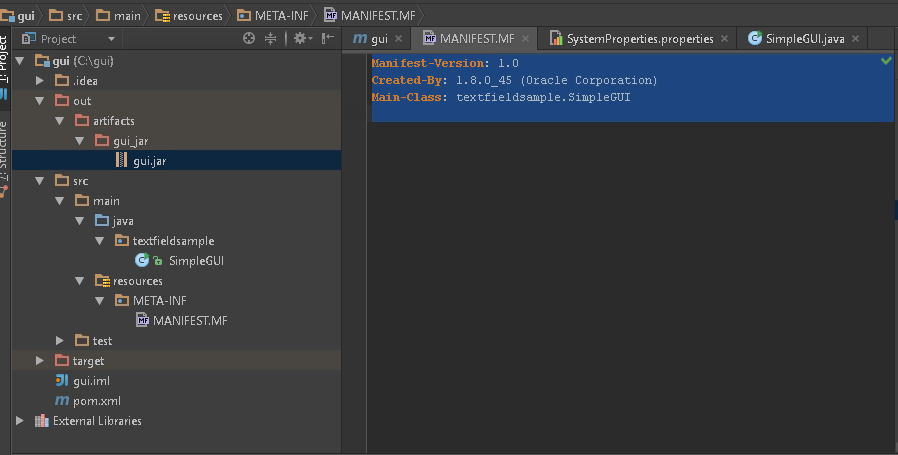


Затем нажмем «ok»

После перетяните Вашу папочку META-INF в ресурсы + если у Вас в манифесте не прописалась версия Java – сделайте это вручную

Мой манифест имеет следующий вид

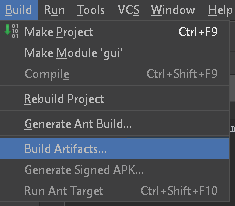
**Manifest-Version**: 1.0  
**Created-By**: 1.8.0\_45 (Oracle Corporation)  
**Main-Class**: textfieldsample.SimpleGUI



, давайте попробуем сгенерировать джар

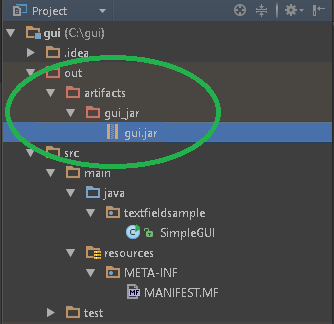
Выполним команду

**Build -> build artifacts**

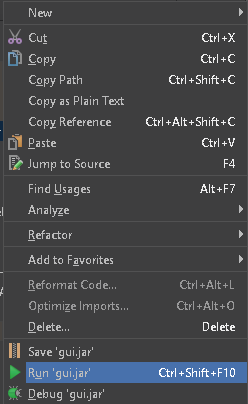
****

**Build or rebuild**

**В src -> out найдем нужный нам джар**

****

Кликнем правой кнопкой на джаре, затем



1. Консолька ☺

Давайте определимся – по сути у нас есть 2 варианта создания jar(а) из консольки

1. Использование entry point

Например для практики - попробуем заархивировать нашу программку **Hello.java**

**Компиляция:**

javac –cp . путь\_к\_Вашим\_сорсам\_с\_учетом\_расширения

**для нас это**

javac –cp . Hello.java

**Исполнение/выполнение/запуск:**

java –cp . путь\_к\_Вашим\_сорсам\_без\_учета\_расширения

**для нас это**

java –cp . Hello

**Архивация**

jar –cvfe название\_Вашего\_джара.jar Ваш\_мейн\_класс байт\_код

**для нас это**

jar –cfve my\_first\_hello.jar Hello.class

c – create

f – file

v – please speak with me/log in console

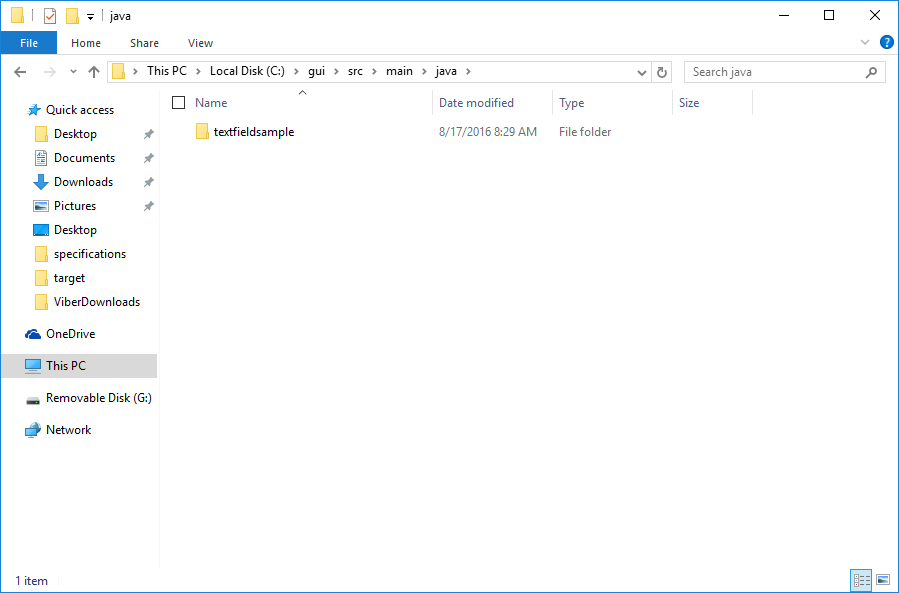
e – entry point (provide the main class name after jar name)

1. Другой способ архивации -- использование манифеста

Манифест – файл, с расширением **mf || txt,** содержащий информацию о джаре: когда был создан, версия, кем, мейн класс, основные зависимости

* Итак – переходим в «рут» директорию нашего проекта – глазками должны видеть пакет, в котором находится наш GUI класс, исходя из того, что прописано у нас в SimpleGUI классе это

package **textfieldsample;**



* Давайте подумаем, для нормальной компиляции нам необходим какой джар? Все правильно – один из <https://mvnrepository.com/artifact/commons-lang/commons-lang> ( джар, содержащий нужный нам **NumberUtils** –т.к именно его мы юзаем в нашей программке)

у нас есть 2 варианта получения этого джара

1. Выкачать его напрямую с интернета

как вариант можно юзать эту ссылку

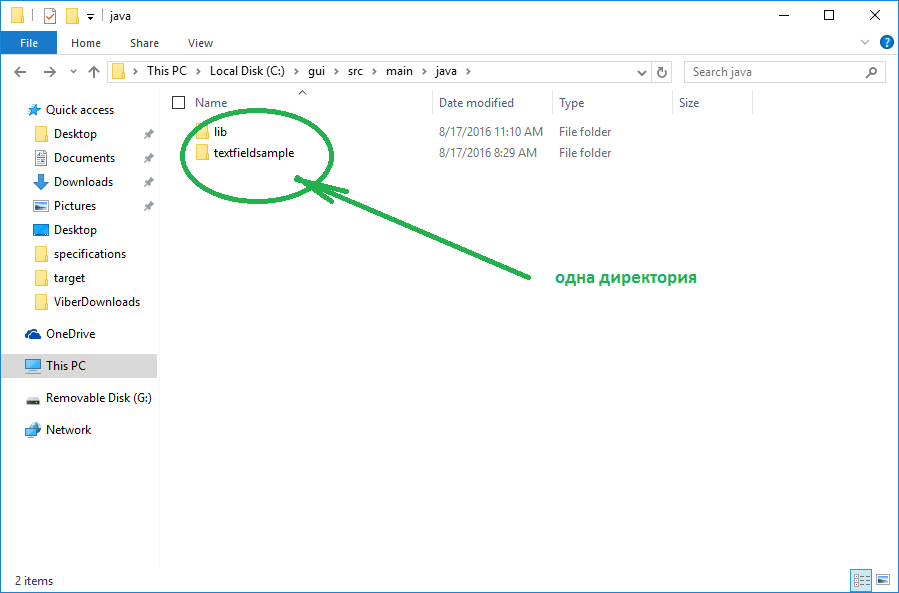
<http://www.java2s.com/Code/Jar/a/Downloadapachecommonslangjar.htm>

1. Есть возможность поискать нужный нам джар в пределах локального мавен репозитория, для этого

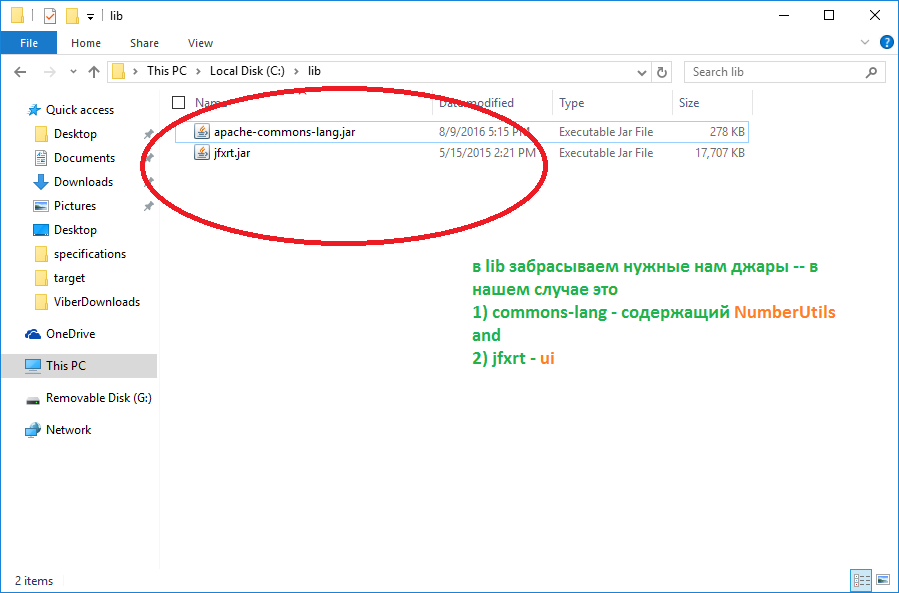
переходим в **%HOMEPATH%\.m2\repository\commons-lang\commons-lang,** определяемся с версией, и забираем commons-lang-xx.jar

для меня это будет **2.3\commons-lang-2.3.jar**

* создадим в пределах нашей директории с проектом папочку **lib**

****

* перенесем туда найденный джар – **commons-lang-xx.jar and jfxrt.jar (чтобы не было привязки к версии джава) **

****

Теперь все по схеме

**Компиляция:**

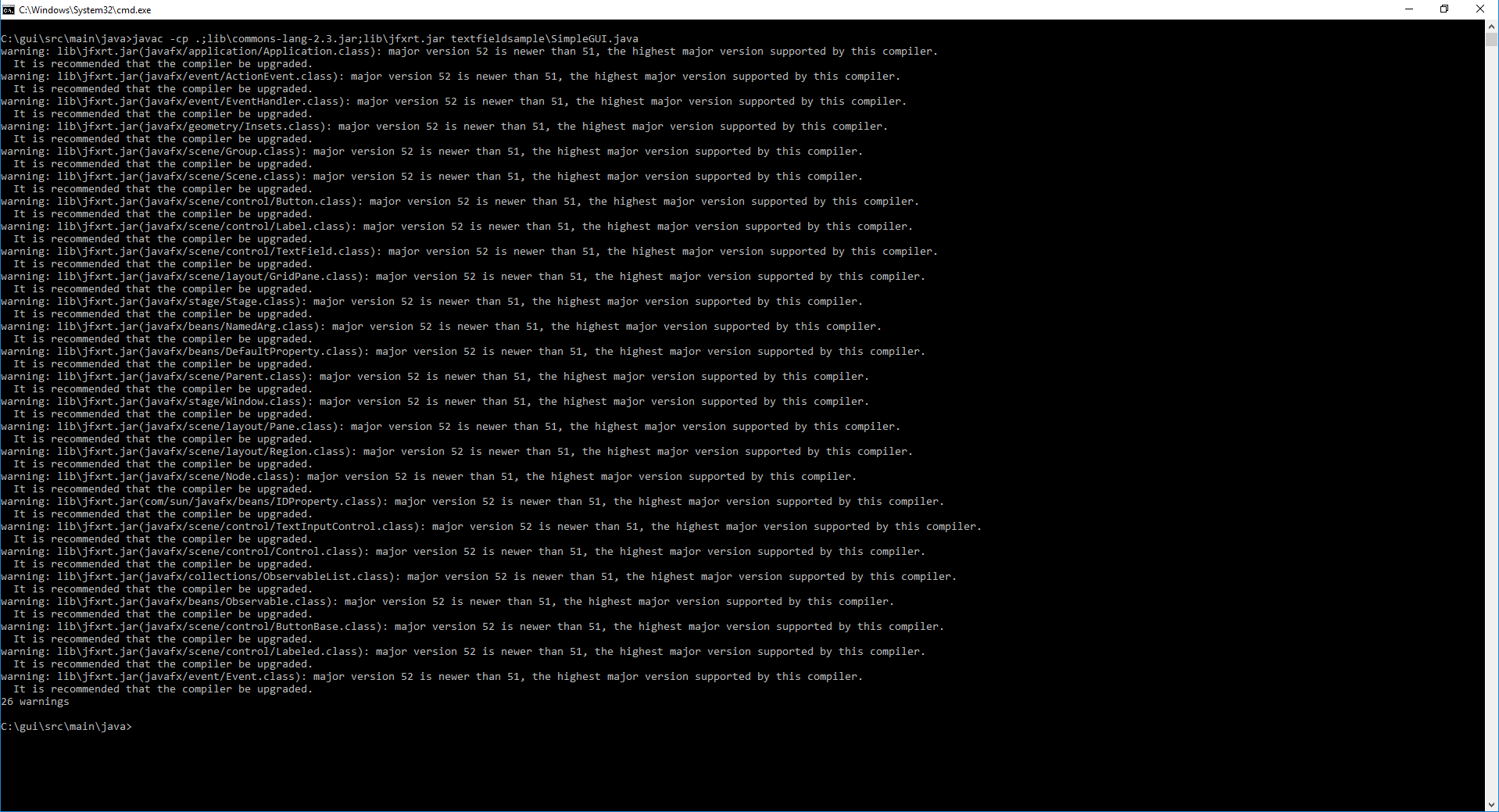
javac –cp . путь\_к\_Вашим\_сорсам\_с\_учетом\_расширения

**для нас это**

javac –cp .;lib\commons-lang-2.3.jar;lib\jfxrt.jar textfieldsample\SimpleGUI.java

НЕ ЗАБУДЬТЕ, ЧТО для **Linux** and **IOS** вместо **;**

необходимо использовать **:** для всех джаров, указанных в класспасе

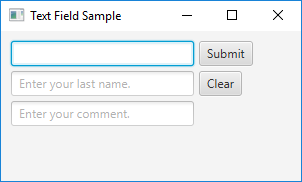


**Исполнение/выполнение/запуск:**

java –cp . путь\_к\_Вашим\_сорсам\_без\_учета\_расширения

**для нас это**

java –cp .;lib\commons-lang-2.3.jar;lib\jfxrt.jar textfieldsample.SimpleGUI



**Архивация**

**Создадим манифест файлик – откроем блокнот**

Пропишем туда

Manifest-Version: 1.0

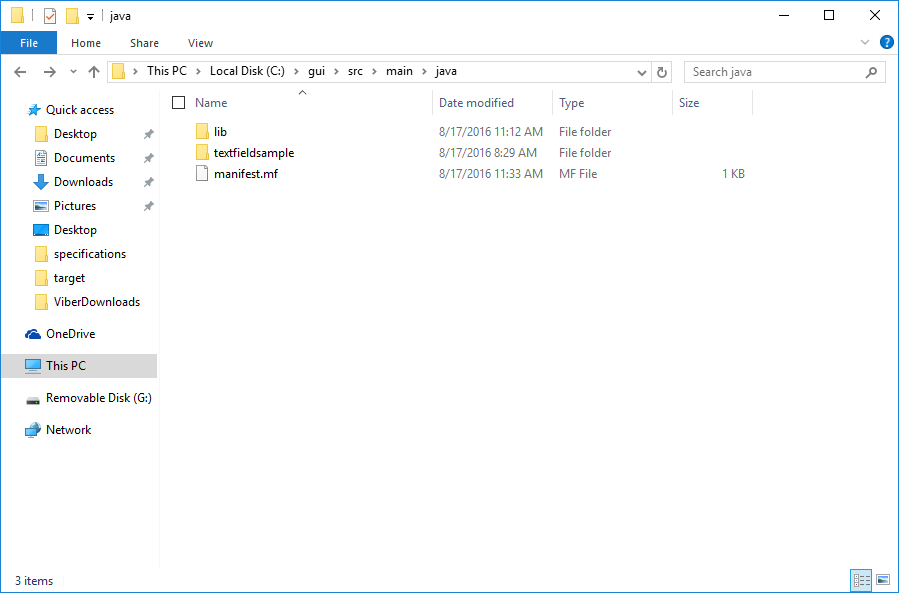
Created-By: 1.8.0\_45 (Oracle Corporation)

Class-path: lib\commons-lang-2.3.jar

Main-Class: textfieldsample.SimpleGUI

Опять же – вы должны ориентироваться на версию джава, которая есть у Вас – для javaFX это java8 + последние версии java7 – будьте внимательны

После сохранения



Пробуем создать джар

jar –cvfm название\_Вашего\_джара.jar манифест\_файл байт\_код

**для нас это**

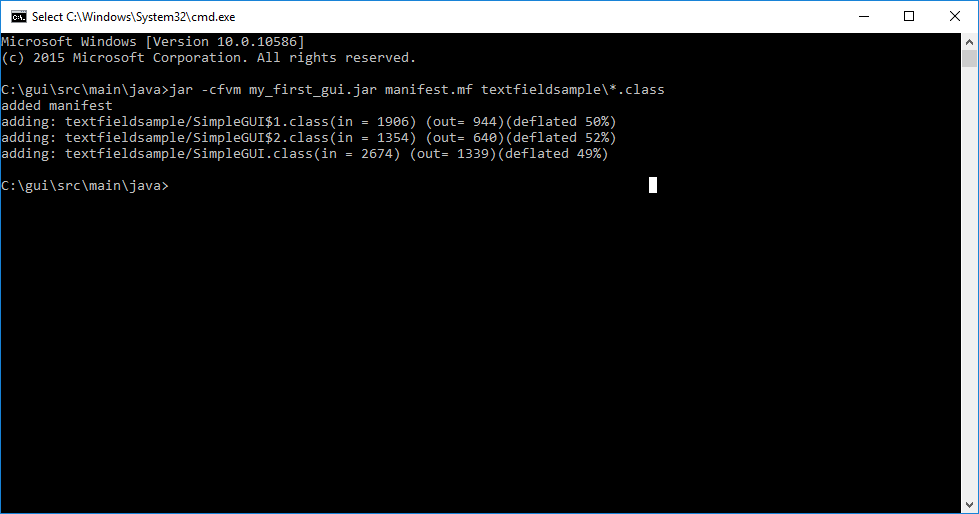
jar –cfvm my\_first\_gui.jar manifest.mf textfieldsample\\*.class

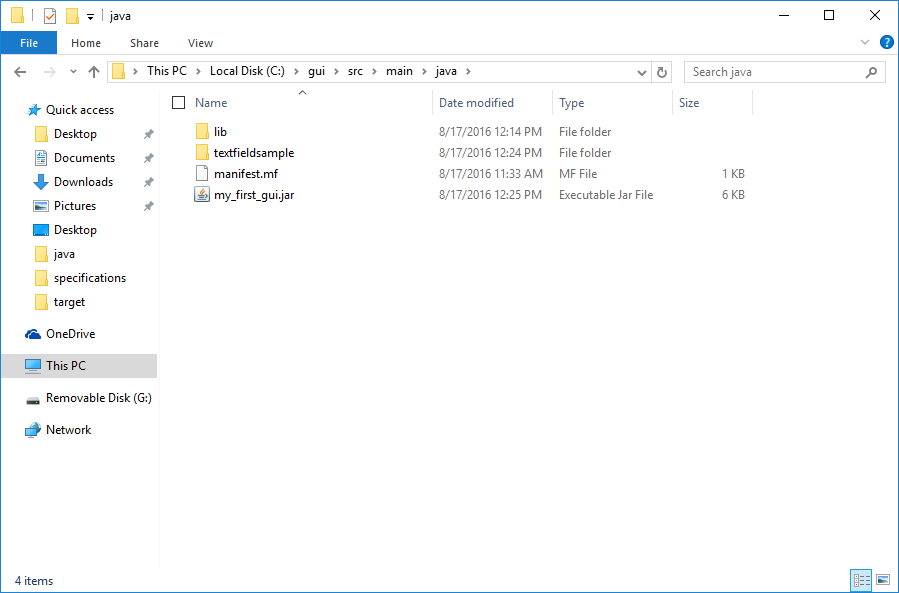
c – create

f – file

v – please speak with me/log in console

m – manifest





Запуск:

* java –jar my\_first\_gui.jar
* javaw –jar my\_first\_gui.jar
* двойным щелчком