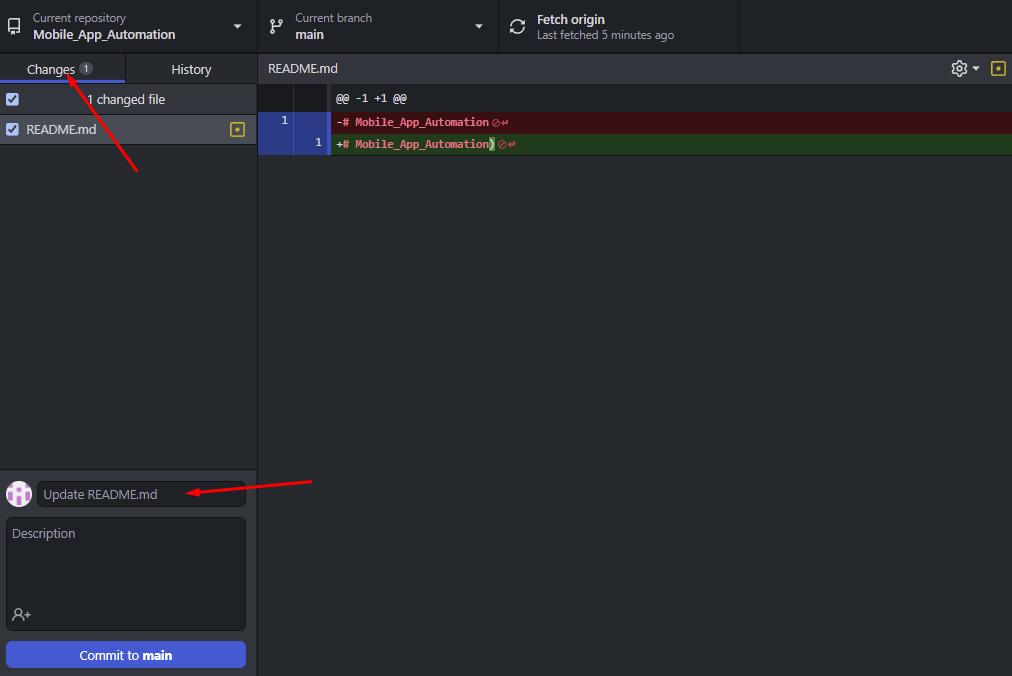
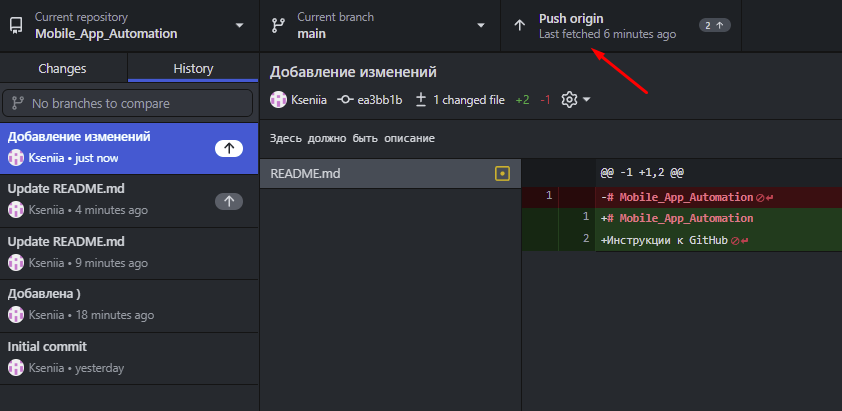
# Урок 0

## Инструкции к GitHub

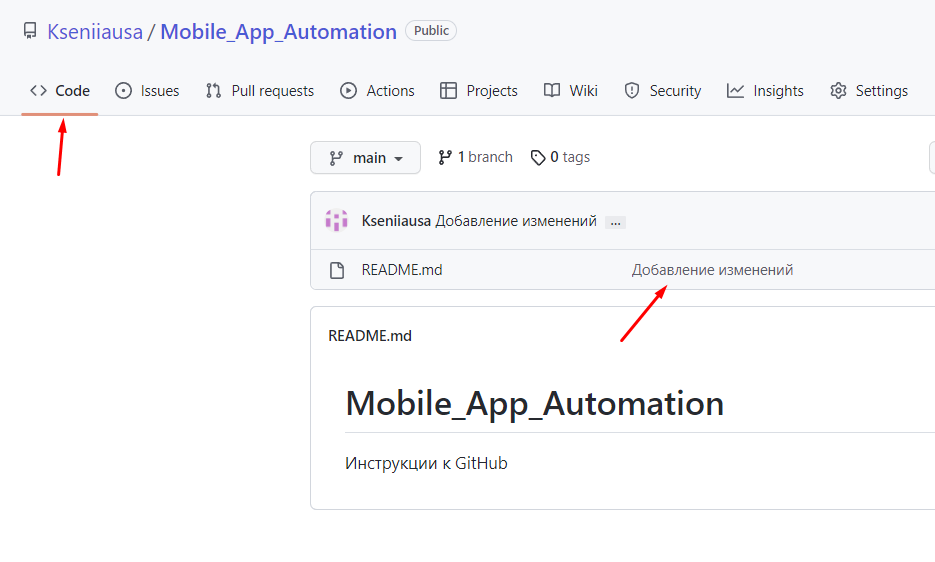
1. При внесении изменений можно посмотреть изменения и добавить комментарий:



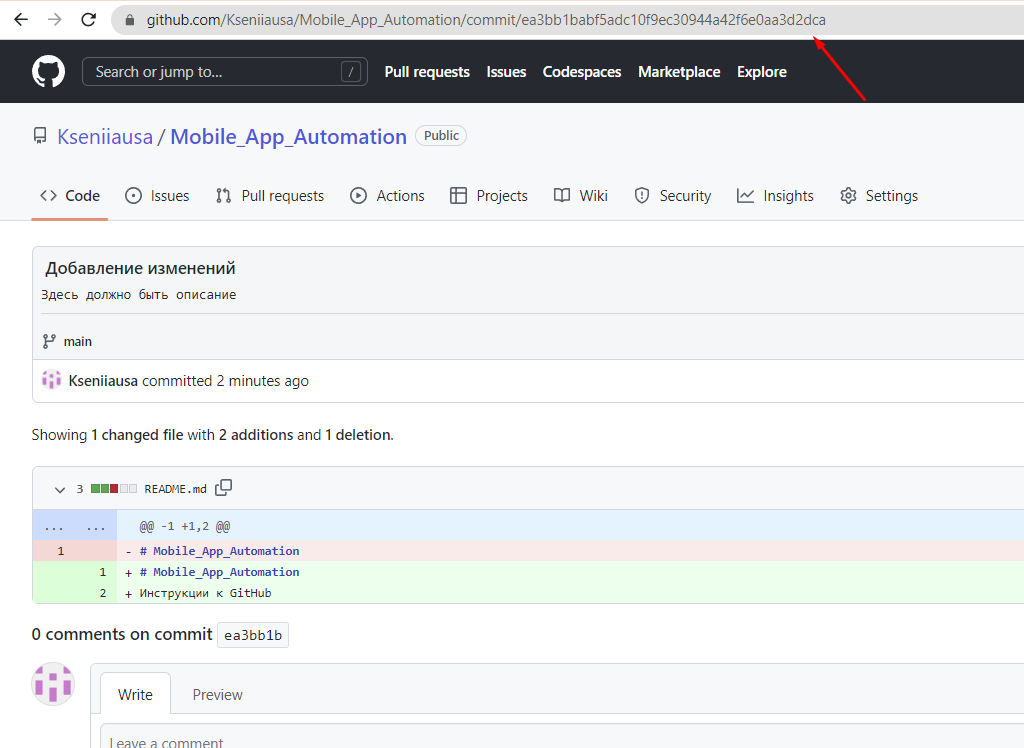
1. Добавляем комментарий и жмем кнопку коммит
2. Изменения сохранены локально, просмотр вкладка History
3. Для отправки на удаленную машину нажать PUSH – двухсторонняя синхронизация.



1. Изменения отображаются на удаленном доступе

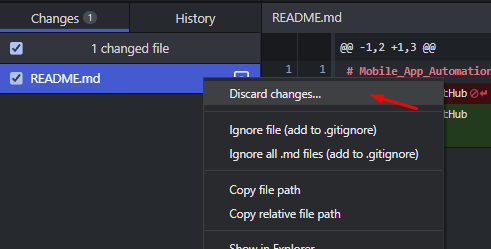


1. При переходе по коммиту открываются изменения, ссылку на которых необходимо отправлять на проверку



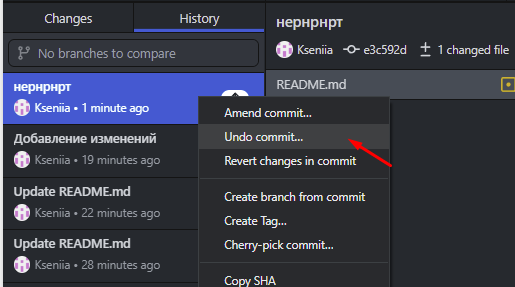
### Вернуть изменения

1. Внесли изменения в файл и сохранить
2. Перейти в ГитД и выбрать:



### Вернуть изменения с коммитом

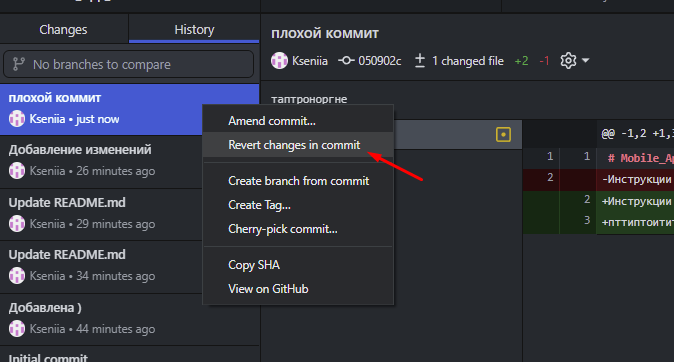
1. Внесли изменения в файл и сохранить
2. Отправить коммит
3. Перейти в ГитД и выбрать:



1. Для отмены изменений выполнить шаги [выше](#_Вернуть_изменения)

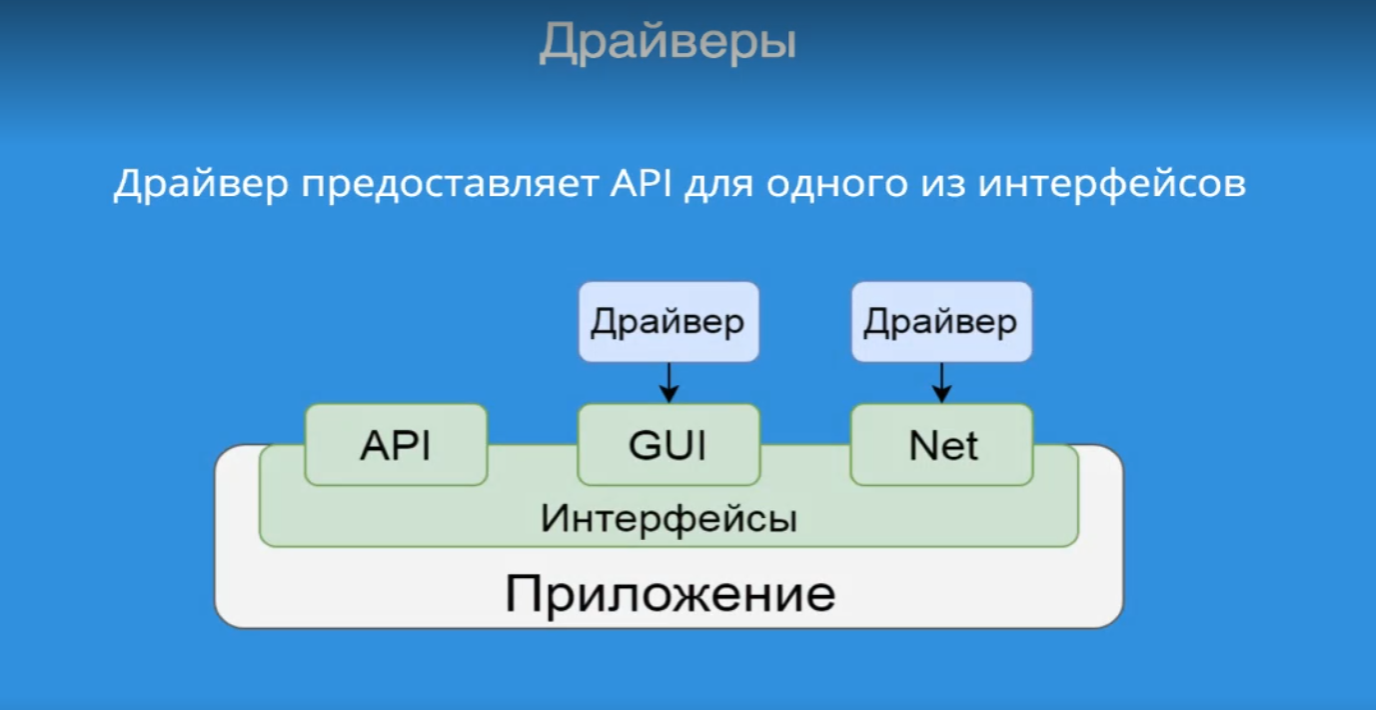
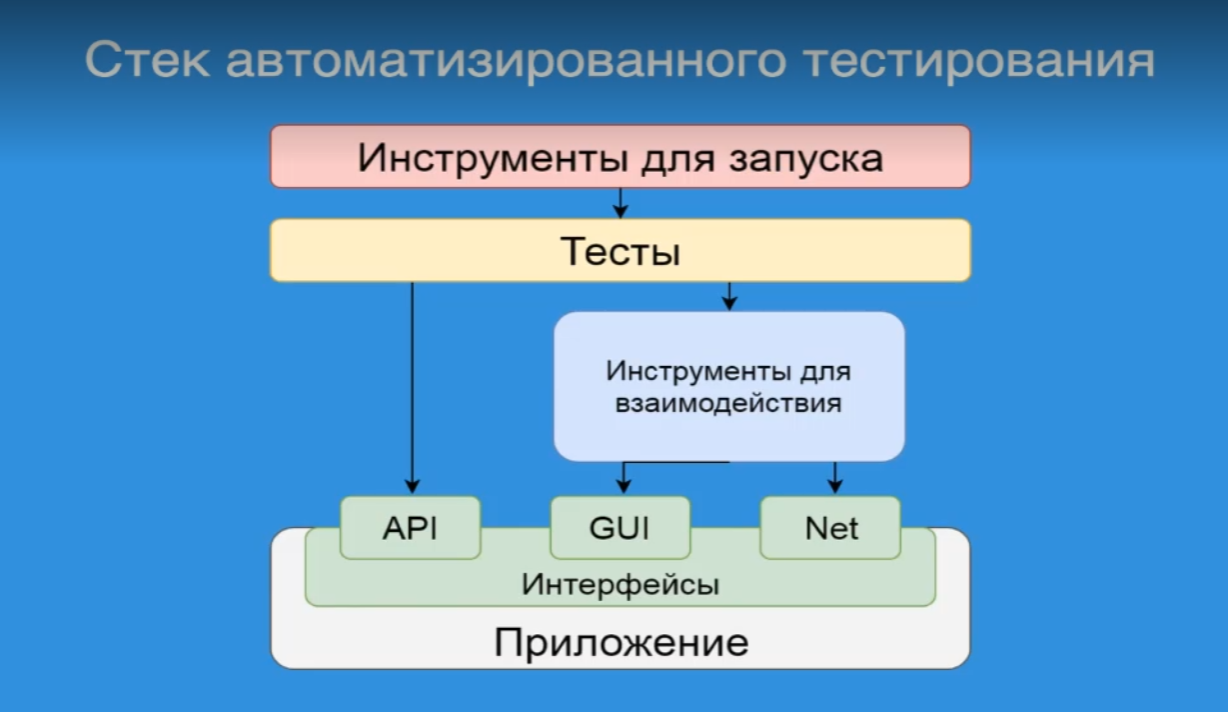
### Вернуть изменения с коммитом и уд. синхронизацией

1. Внесли изменения в файл и сохранить
2. Отправить коммит
3. Выполнить синхронизацию
4. Перейти в ГитД и выбрать Revert – локально изменения отменены
5. Для отправки на удаленную машину нажать PUSH

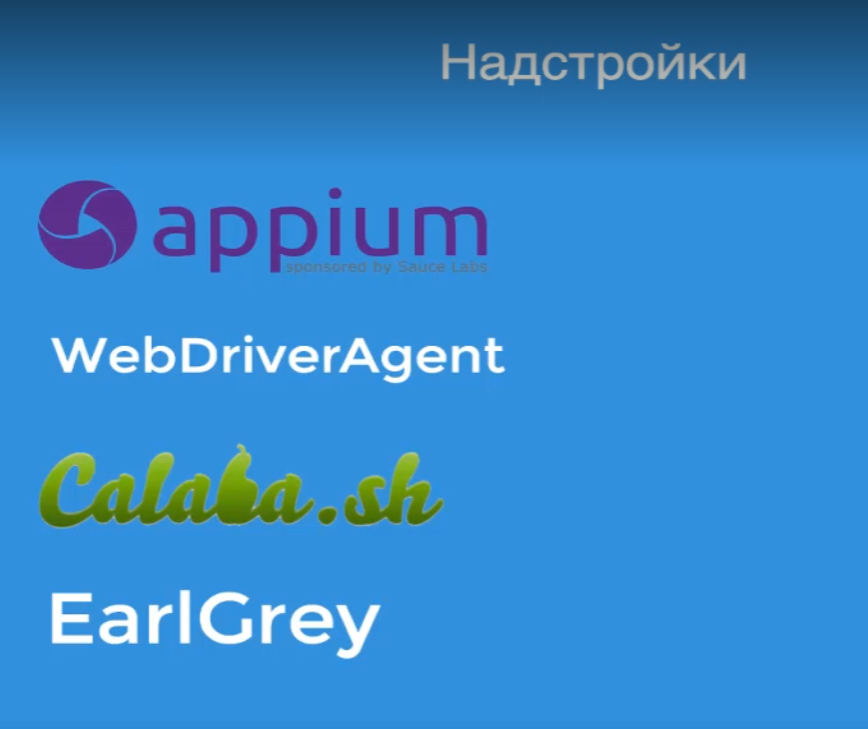
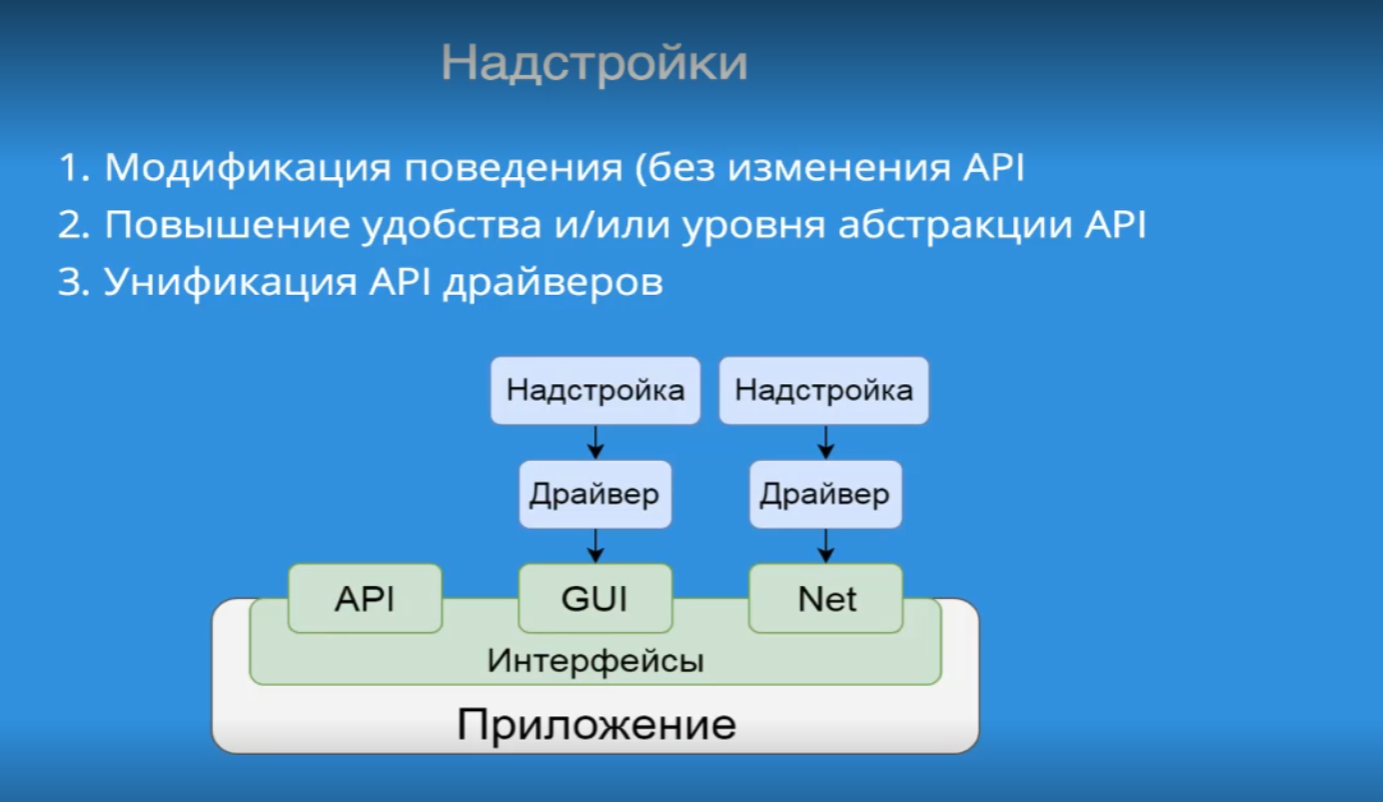


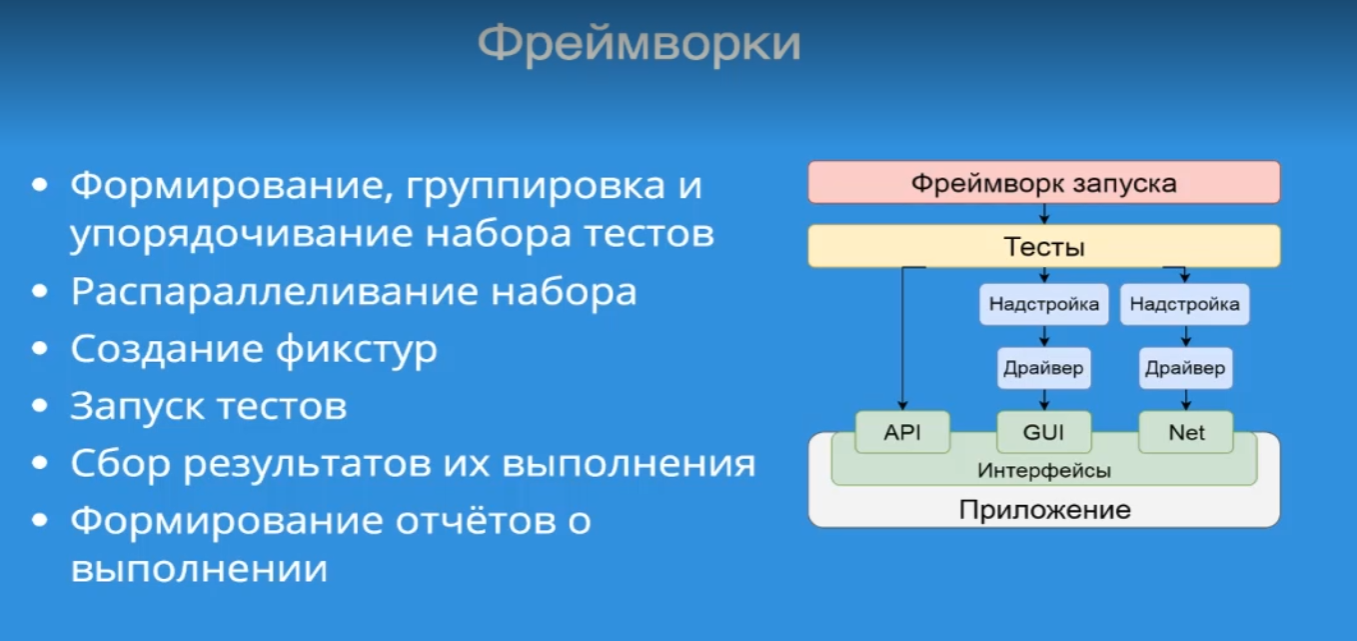
# Урок 1

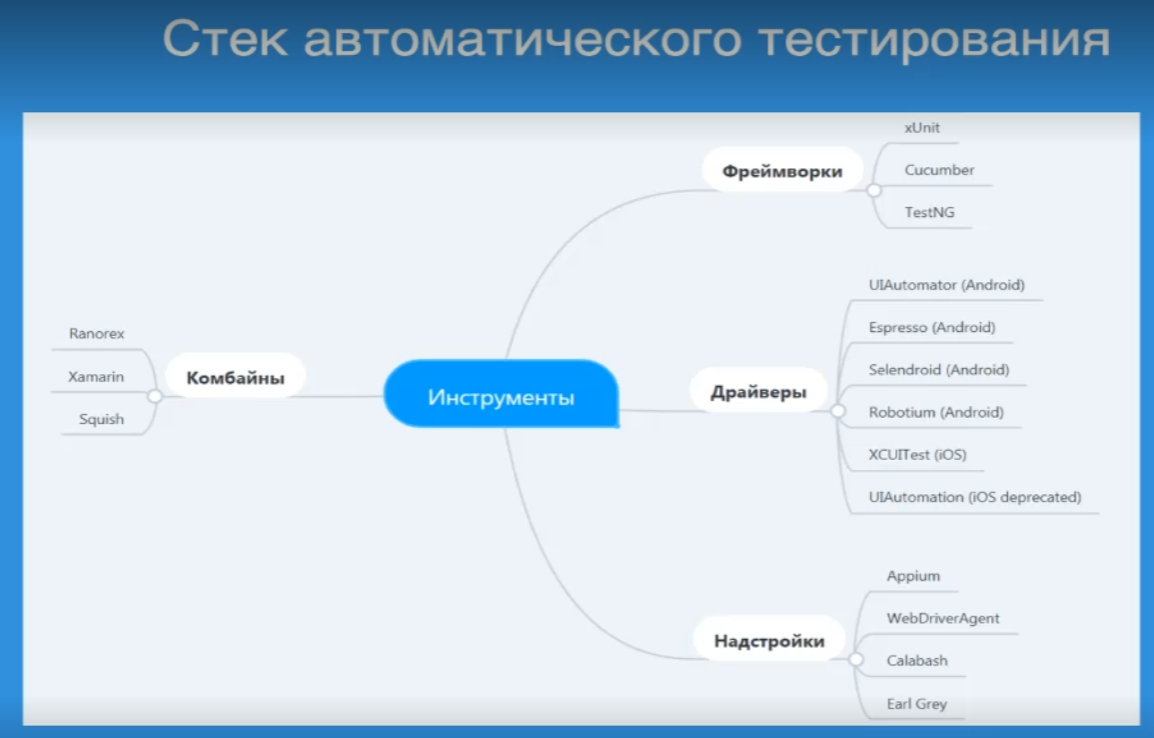
## Инструменты



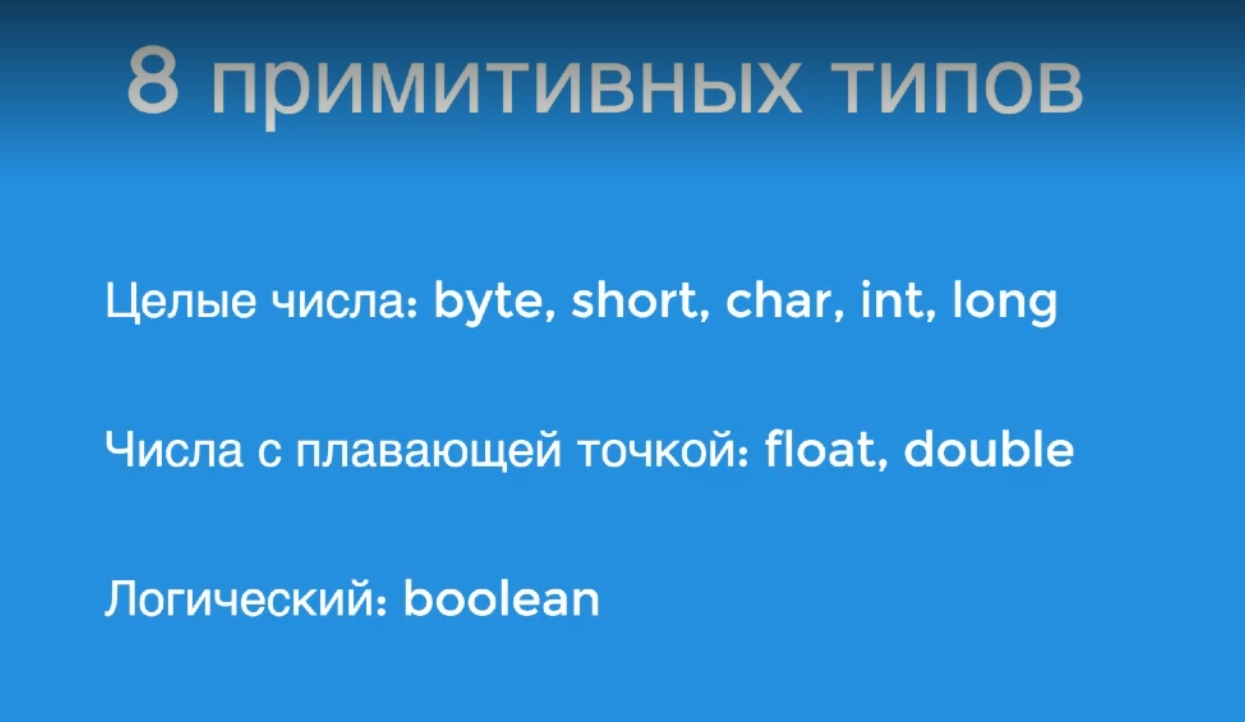




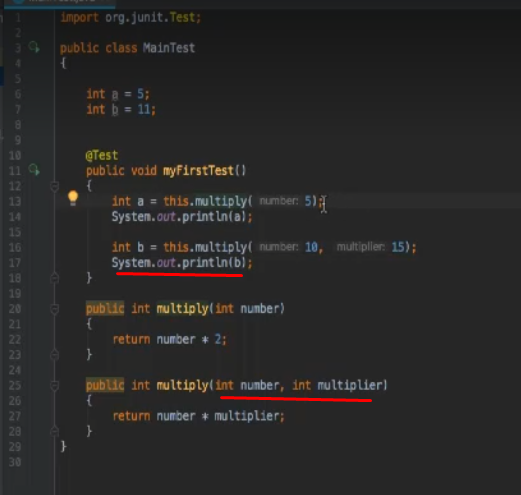
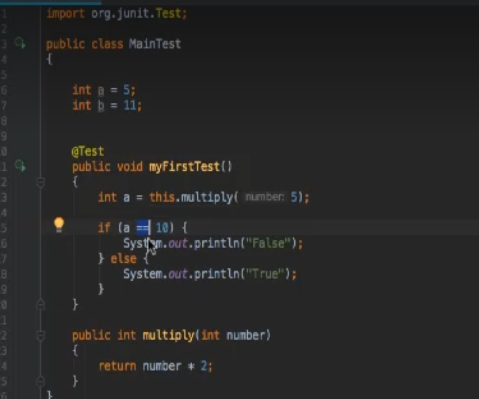




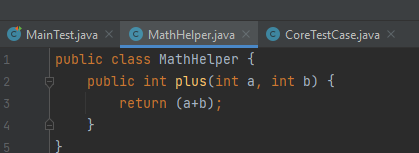
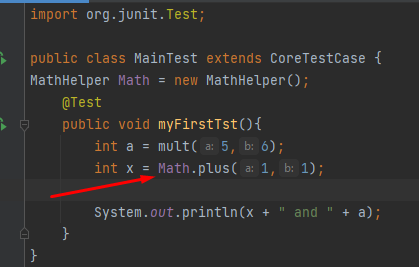
## Java



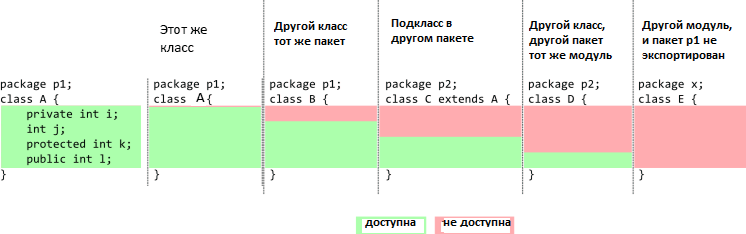
### Функции и this



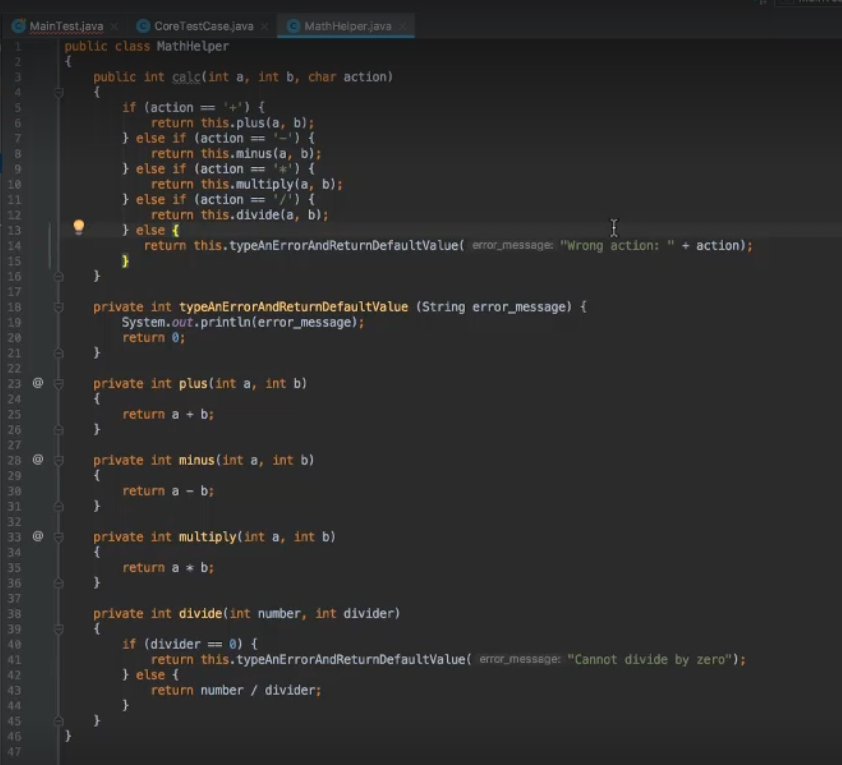
### Родительские классы



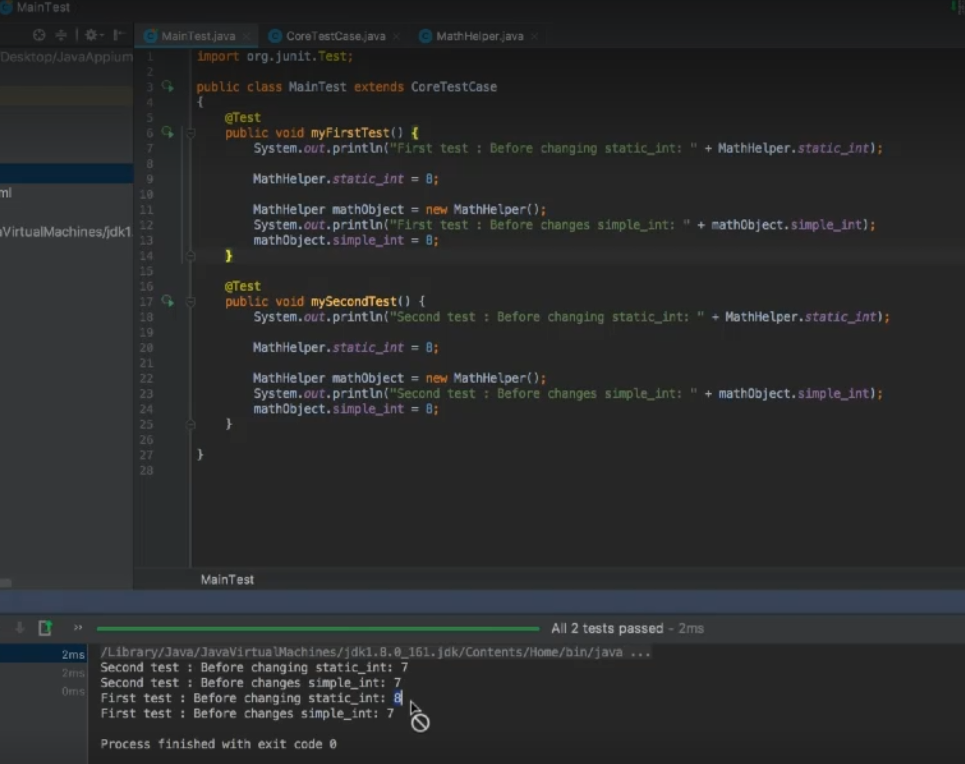
### Модификатор доступа



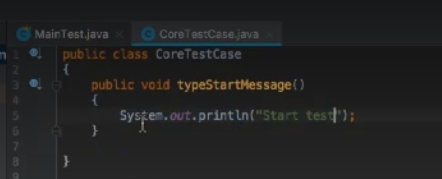
Пример:



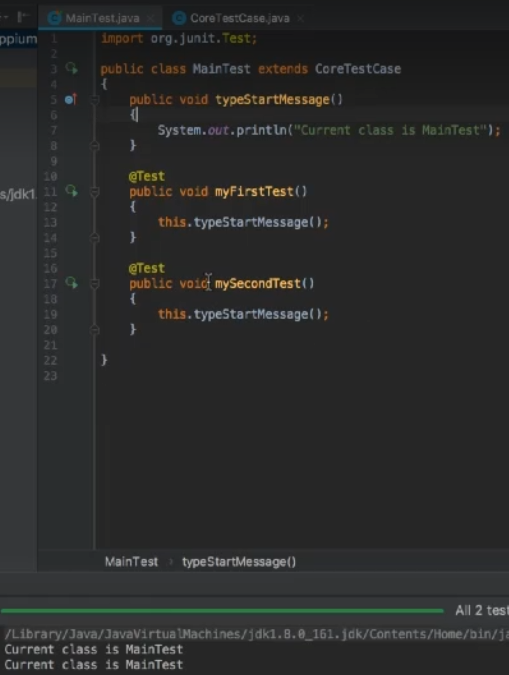
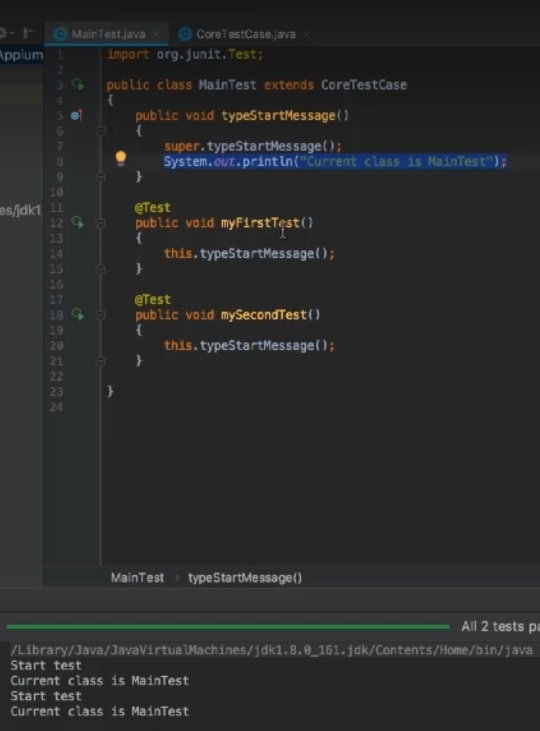
### Статические методы и поля?



### Super



При Super сначала родитель потом текущий. Для переопределения – убрать.



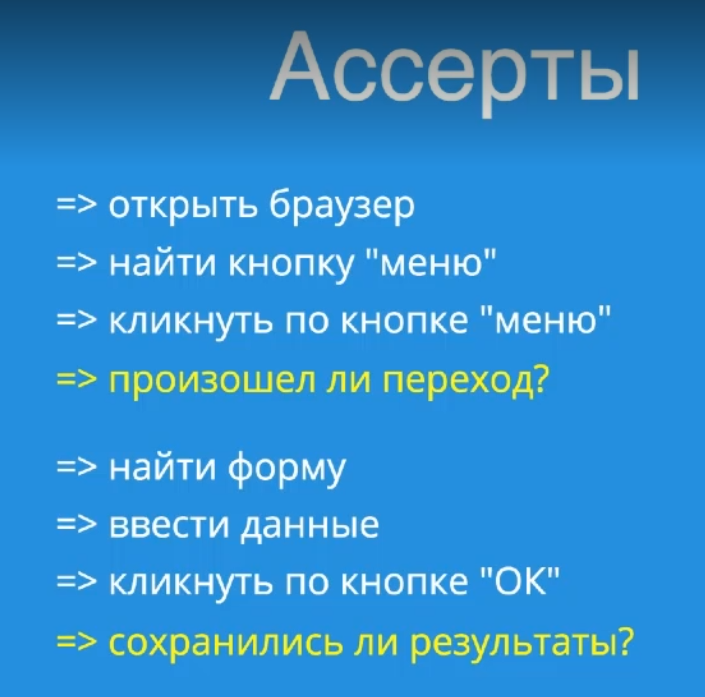
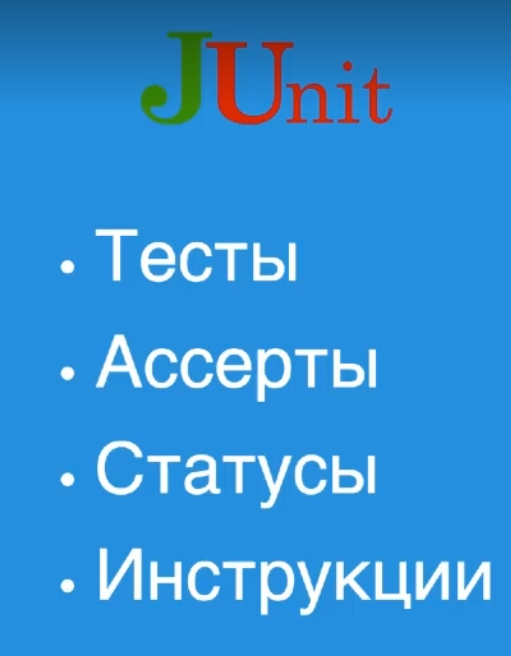
### ООП



### Junit

Ассерт - соответствие значения параметру предустановленного значения

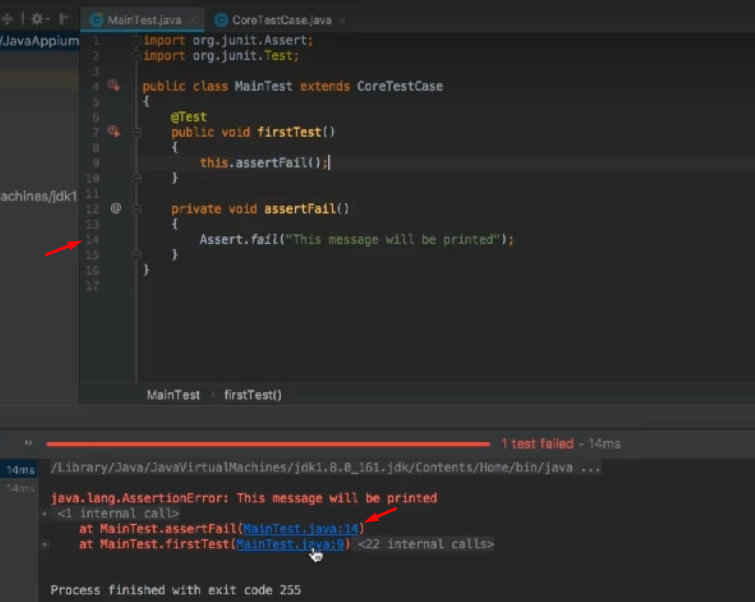
Инструкции – before, after



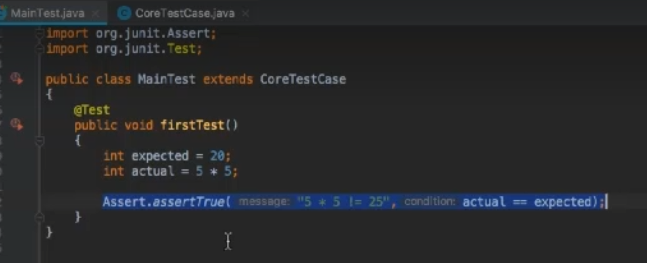
### Test

Alt(option) + enter – выбор

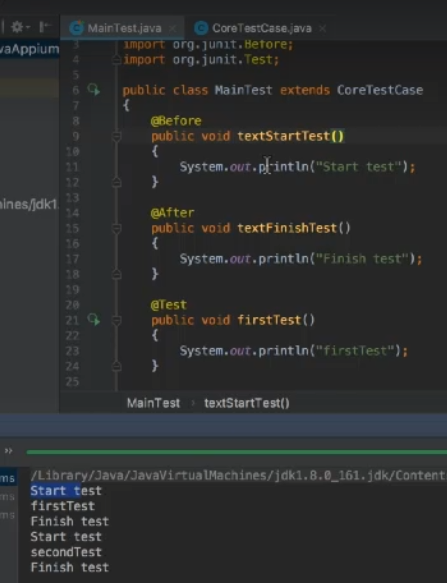
Trace показывает где ошибка



Пример успешного теста (справа как нужно писать):



Before – перед каждым тестом, After - после каждого теста:



# Урок 2

## Запуск эмулятор из Cmd

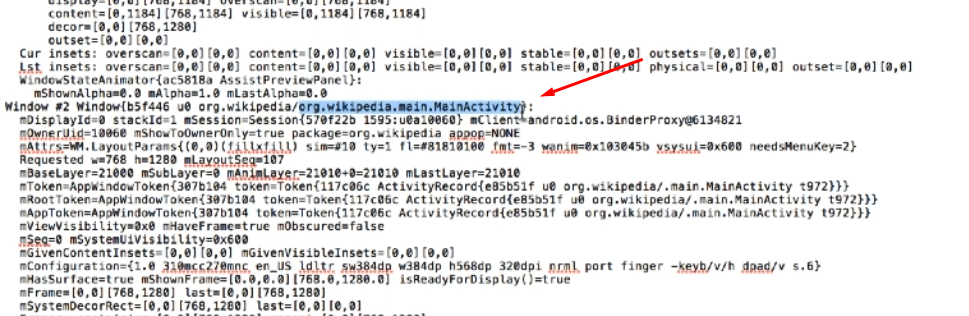
emulator @and80

## Packages and Capabilities

1. Запустить приложение
2. Проверить подключение : adb devices
3. Список всех пакетов всех приложений на устройстве: adb shell pm list packages
4. Найти пакет: adb shell pm list packages | grep "wiki"
5. Выдает активность на открытом экране приложения и записывает в файл:

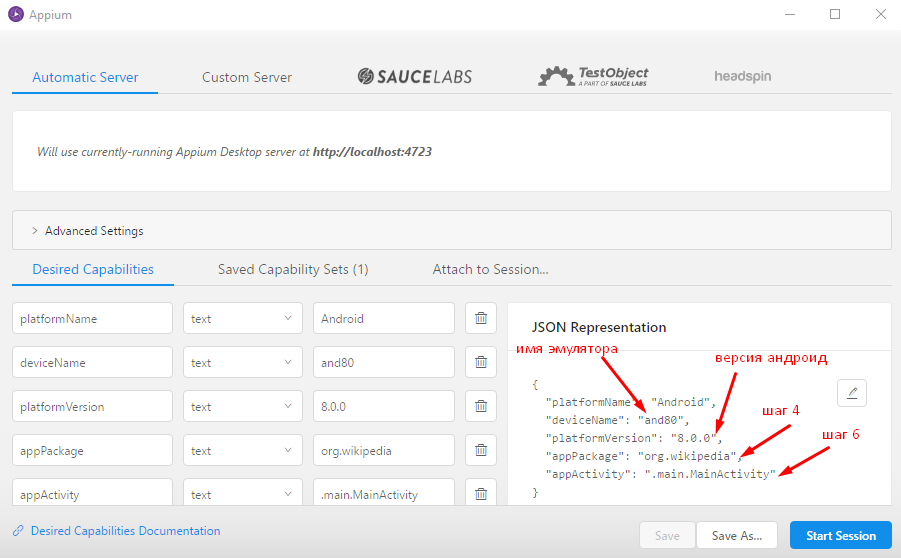
adb shell dumpsys window windows >> Desktop\activity.txt

1. Найти в файле по строке – шаг 4.



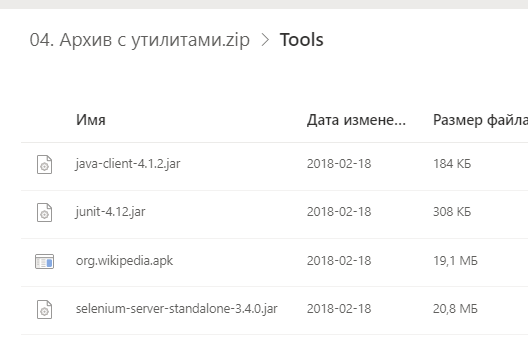
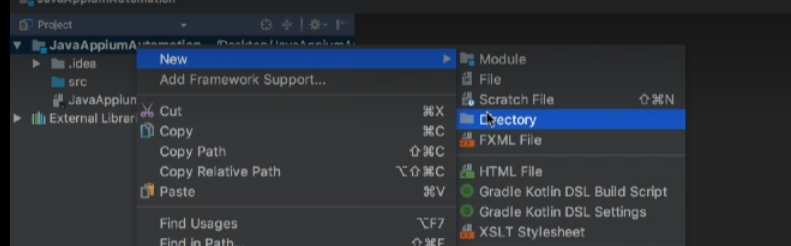
1. Перенести параметры в Appium

## Appium

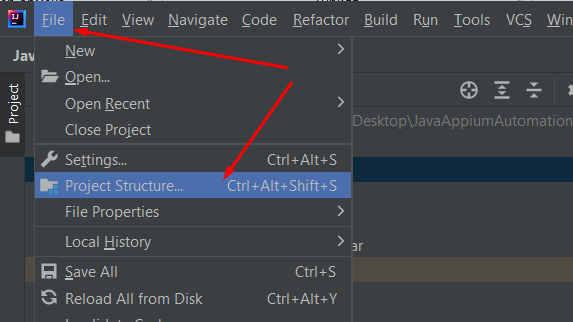
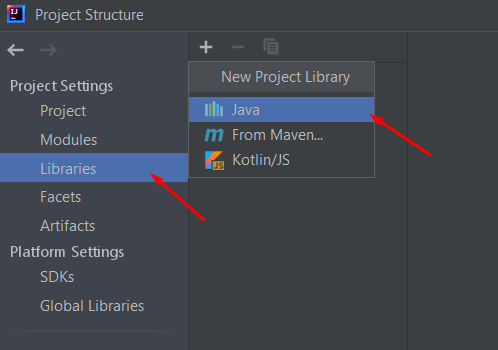


## Создание нового проекта

1. Скачать и импорт в проект в новую папку libs (фреймворки ниже)

1. Перейти в структуру и добавить их

1. Создать еще папку apks – перенести приложение

## Первый тест

В классе FirstTest

import io.appium.java\_client.AppiumDriver;  
import io.appium.java\_client.android.AndroidDriver;  
import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import org.openqa.selenium.remote.DesiredCapabilities;  
  
import java.net.URL;  
  
public class FirstTest {  
 private AppiumDriver driver;  
 @Before  
 public void setUp() throws Exception {  
 DesiredCapabilities capabilities = new DesiredCapabilities();  
 capabilities.setCapability("platformName","Android");  
 capabilities.setCapability("deviceName","and80");  
 capabilities.setCapability("platformVersion","8.0.0");  
 capabilities.setCapability("automationName","Appium");  
 capabilities.setCapability("appPackage","org.wikipedia");  
 capabilities.setCapability("appActivity",".main.MainActivity");  
 capabilities.setCapability("app","C:\\Users\\User\\Desktop\\JavaAppiumAutomation\\apks\\org.wikipedia.apk");  
  
 driver = new AndroidDriver(new URL("http://127.0.0.1:4723/wd/hub"),capabilities);  
 }  
  
 @After  
 public void tearDown ()  
 {  
 driver.quit();  
 }  
 @Test  
 public void firstTest(){  
 System.*out*.println("First run");  
 }  
}

# Урок 3

## Локаторы, клики, ввод в поле поиска

Узнать больше об XPath можно из этой довольно красочной документации: <http://pragmatictestlabs.com/2020/01/28/mastering-xpath-for-selenium-test-automation-engineers/>

(0)

//Поиск Строки поиска и клик:  
 WebElement element = driver.findElementByXPath( "//\*[contains(@text, 'Search Wikipedia')]");  
 element.click();

// Поиск строки ввода и ввод  
 WebElement element\_to\_enter\_search\_line = driver.findElementByXPath( "//\*[contains(@text, 'Search…')]");  
 element\_to\_enter\_search\_line.sendKeys( "Appium");

// - любая вложенность элементов \*- любой элемент и частичное совпадение

## Создадим свой Фреймворк (\*)

// Поиск элемента и ожидание его появления  
private WebElement waitForElementPresentByXpath(String xpath, String error\_message, long timeoutInSeconds)  
{  
WebDriverWait wait = new WebDriverWait(driver, timeoutInSeconds); // время ожидания  
wait.withMessage(error\_message + "\n"); // текст ошибки  
By by = By.*xpath*(xpath); // локатор  
return wait.until (ExpectedConditions.*presenceOfElementLocated*(by) ); // ожидаем элемент  
}

Можно переписать Поиск ввода (1)

// Поиск строки ввода и ввод с Фреймворком  
WebElement element\_to\_enter\_search\_line = waitForElementPresentByXpath(  
 "//\*[contains(@text, 'Search…')]",  
 "Cannot find search input",  
 5);  
  
 //было: driver.findElementByXPath("//\*[contains(@text, 'Search…')]");  
  
element\_to\_enter\_search\_line.sendKeys("Appium");

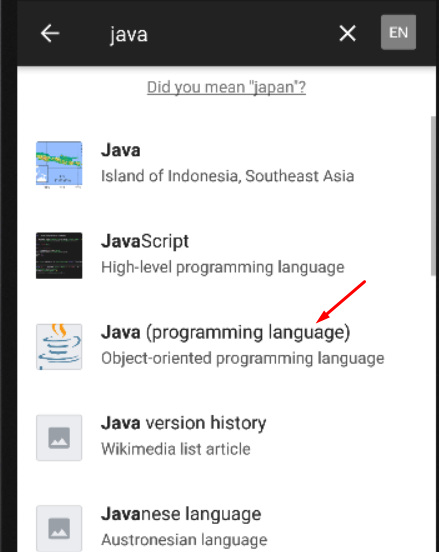
Переопределим Фреймворк (\*\*)

//Переопределим Фреймворк, чтобы не вводить время каждый раз  
private WebElement waitForElementPresentByXpath(String xpath, String error\_message){  
 return waitForElementPresentByXpath ( xpath, error\_message, 5);  
}

Можно переписать Поиск ввода (1\*)

// Поиск строки ввода и ввод с Фреймворком  
 WebElement element\_to\_enter\_search\_line = waitForElementPresentByXpath(  
 "//\*[contains(@text, 'Search…')]",  
 "Cannot find search input");  
 element\_to\_enter\_search\_line.sendKeys("Appium");  
}

## #5 Поиск строки в поиске по id и затем по тексту



1 или 1\* (поиск по java) + то, что ниже…

// Проверка наличия строки в поиске  
// от id искать текст ниже  
waitForElementPresentByXpath("//\*[@resource-id = 'org.wikipedia:id/page\_list\_item\_container']//\*[@text = 'Object-oriented programming language']",  
 "Cannot find 'Object-oriented programming language' topic searching by Java ",  
 15);

Промежуточный код:

@Test  
 public void firstTest() {  
 //Урок3  
 //Поиск Строки поиска и клик:  
 WebElement element = driver.findElementByXPath("//\*[contains(@text, 'Search Wikipedia')]");  
 element.click();  
  
 // Поиск строки ввода и ввод с Фреймворком  
 WebElement element\_to\_enter\_search\_line = waitForElementPresentByXpath(  
 "//\*[contains(@text, 'Search…')]", //было: driver.findElementByXPath("//\*[contains(@text, 'Search…')]");  
 "Cannot find search input");  
 element\_to\_enter\_search\_line.sendKeys("Java");  
  
 // Проверка наличия строки в поиске  
 // от id искать текст ниже  
 waitForElementPresentByXpath("//\*[@resource-id = 'org.wikipedia:id/page\_list\_item\_container']//\*[@text = 'Object-oriented programming language']",  
 "Cannot find 'Object-oriented programming language' topic searching by Java ",  
 15);  
 }  
  
 // Фреймворк Поиск элемента и ожадание его появления  
 private WebElement waitForElementPresentByXpath(String xpath, String error\_message, long timeoutInSeconds)  
 {  
 WebDriverWait wait = new WebDriverWait(driver, timeoutInSeconds); // время ожидания  
 wait.withMessage(error\_message + "\n"); // текст ошибки  
 By by = By.*xpath*(xpath); // локатор  
 return wait.until (ExpectedConditions.*presenceOfElementLocated*(by) ); // ожидаем элемент  
 }  
  
 //Переопределим Фреймворк, чтобы не вводить время каждый раз  
 private WebElement waitForElementPresentByXpath(String xpath, String error\_message){  
 return waitForElementPresentByXpath ( xpath, error\_message, 5);  
 }  
}

## #6 Методы Ожидание элемента клик и ввод

//Ожидание элемента и клик  
private WebElement waitForElementAndClick(String xpath, String error\_message, long timeoutInSeconds) {  
 WebElement element = waitForElementPresentByXpath (xpath, error\_message, timeoutInSeconds);  
 element.click();  
 return element;  
}  
  
//Ожидание элемента и ввод  
private WebElement waitForElementAndSendKeys(String xpath, String value, String error\_message, long timeoutInSeconds) {  
 WebElement element = waitForElementPresentByXpath (xpath, error\_message, timeoutInSeconds);  
 element.sendKeys(value);  
 return element;  
}

Стало:

//Поиск и клик  
waitForElementByXpathAndClick("//\*[contains(@text, 'Search Wikipedia')]",  
 "Cannot find Search Wiki input",  
 5);  
  
//Поиск и ввод  
waitForElementByXpathAndSendKeys( "//\*[contains(@text, 'Search…')]",  
 "Java",  
 "Cannot find search input",  
 5);

## #7 – 8 Поиск строки. Отмена поиска. Рефакторинг

@Test  
public void firstTest() {  
 //Поиск и клик  
 waitForElementAndClick(  
 By.*xpath*("//\*[contains(@text, 'Search Wikipedia')]"),  
 "Cannot find Search Wiki input",  
 5);  
  
 //Поиск и ввод  
 waitForElementAndSendKeys(  
 By.*xpath*("//\*[contains(@text, 'Search…')]"),  
 "Java",  
 "Cannot find search input",  
 5);  
  
 // Проверка наличия строки в поиске  
 // от id искать текст ниже  
 waitForElementPresent(  
 By.*xpath*("//\*[@resource-id = 'org.wikipedia:id/page\_list\_item\_container']//\*[@text = 'Object-oriented programming language']"),  
 "Cannot find 'Object-oriented programming language' topic searching by Java ",  
 15);  
}

@Test  
public void testCancelSearch() { //Тест на отмену поиска. По id  
  
 //Дожидаемся эл-та и кликаем  
 waitForElementAndClick(  
 By.*id*("org.wikipedia:id/search\_container"),  
 "Cannot find Search Wiki input",  
 5  
 );  
  
 //Поиск и клик  
 waitForElementAndClick(  
 By.*id*("org.wikipedia:id/search\_close\_btn"),  
 "Cannot find X to cancel search",  
 5  
 );  
  
 //Проверка отсутсвия эл-та на странице  
 waitForElementNotPresent(  
 By.*id*("org.wikipedia:id/search\_close\_btn"),  
 "X is still on the page",  
 5  
 );  
}

// Фреймворк. Поиск элемента и ожидание его появления  
 private WebElement waitForElementPresent(By by, String error\_message, long timeoutInSeconds)  
 {  
 WebDriverWait wait = new WebDriverWait(driver, timeoutInSeconds); // время ожидания  
 wait.withMessage(error\_message + "\n"); // текст ошибки  
 return wait.until (ExpectedConditions.*presenceOfElementLocated*(by) ); // ожидаем элемент  
 }  
  
 //Переопределим Фреймворк, чтобы не вводить время каждый раз  
 private WebElement waitForElementPresent(By by, String error\_message){  
 return waitForElementPresent(by, error\_message, 5);  
 }  
  
 //Ожидание элемента и клик  
 private WebElement waitForElementAndClick(By by, String error\_message, long timeoutInSeconds) {  
 WebElement element = waitForElementPresent(by, error\_message, timeoutInSeconds);  
 element.click();  
 return element;  
 }  
  
 //Ожидание элемента и ввод  
 private WebElement waitForElementAndSendKeys(By by, String value, String error\_message, long timeoutInSeconds) {  
 WebElement element = waitForElementPresent(by, error\_message, timeoutInSeconds);  
 element.sendKeys(value);  
 return element;  
 }  
  
 //Проверка отсутсвия эл-та на странице по ID  
 private boolean waitForElementNotPresent(By by, String error\_message, long timeoutInSeconds){  
 WebDriverWait wait = new WebDriverWait(driver, timeoutInSeconds);  
 wait.withMessage(error\_message + "\n");  
 return wait.until(ExpectedConditions.*invisibilityOfElementLocated*(by));  
 }

## Очистка элемента

//Очистка элемента  
private WebElement waitАoкElementAndClear(By by, String error\_message, long timeoutInSeconds){  
WebElement element = waitForElementPresent(by, error\_message, timeoutInSeconds);  
element.clear();  
return element;  
}

## Сравнение заголовков

@Test  
//Падение тестов - сравнение заголовков  
public void testCompareArticleTitle() {  
  
 waitForElementAndClick( //Поиск и клик  
 By.*xpath*("//\*[contains(@text, 'Search Wikipedia')]"),  
 "Cannot find Search Wiki input",  
 5);  
  
 waitForElementAndSendKeys( //Поиск и ввод  
 By.*xpath*("//\*[contains(@text, 'Search…')]"),  
 "Java",  
 "Cannot find search input",  
 5);  
  
 waitForElementAndClick( //Поиск и клик  
 By.*xpath*("//\*[@resource-id = 'org.wikipedia:id/page\_list\_item\_container']//\*[@text = 'Object-oriented programming language']"),  
 "Cannot find Search Wiki input",  
 25);  
  
 // ожидаем элемент и сохраняем  
 WebElement title\_element = waitForElementPresent(  
 By.*id*("org.wikipedia:id/view\_page\_title\_text"),  
 "Cannot find article title",  
 15  
 );  
  
 // запишем текст из заголовка по атрибуту text  
 String article\_title = title\_element.getAttribute("text");  
  
 //Сравнение заголовков  
 Assert.*assertEquals*(  
 "We see unexpected title!",  
 "Java (programming language)",  
 article\_title  
 );  
}

## Просмотр элементов uit как апиум

<https://www.guru99.com/uiautomatorviewer-tutorial.html>

# Урок 4

## Свайп

@Test  
  
//Свайп  
public void testSwipeArticle() {  
  
 waitForElementAndClick( //Поиск и клик  
 By.*xpath*("//\*[contains(@text, 'Search Wikipedia')]"),  
 "Cannot find Search Wiki input",  
 5);  
  
 waitForElementAndSendKeys( //Поиск и ввод  
 By.*xpath*("//\*[contains(@text, 'Search…')]"),  
 "Java",  
 "Cannot find search input",  
 5);  
  
 waitForElementAndClick( //Поиск и клик  
 By.*xpath*("//\*[@resource-id = 'org.wikipedia:id/page\_list\_item\_container']//\*[@text = 'Object-oriented programming language']"),  
 "Cannot find Search Wiki input",  
 25);  
  
 waitForElementPresent( // ожидаем элемент  
 By.*id*("org.wikipedia:id/view\_page\_title\_text"),  
 "Cannot find article title",  
 15  
 );  
  
 //вызов свайпа  
 swipeUp(2000);  
  
}

Метод:

// Свайп вверх  
protected void swipeUp(int timeOfSwipe) { // передаем время между нажатиями  
 TouchAction action = new TouchAction(driver); //импорт из appium  
 //для создания отн-ых координат  
 Dimension size = driver.manage().window().getSize(); // получаем параметры девайса  
  
 int x = size.width / 2;//нач. переменная по Х = конеч. переменная по Х  
 int start\_y = (int) (size.height \* 0.8); //нач. переменная по У - на 80% экрана т.е. снизу  
 int end\_y = (int) (size.height \* 0.2); //конеч. переменная по У  
  
 action.press(x, start\_y).waitAction(timeOfSwipe).moveTo(x, end\_y).release().perform();  
}

## Свайп до футера

(конца страницы)

1. Проверяем есть ли элемент
2. Ели нет то свайпаем и снова проверяем
3. Находим эл-т и по нему понимаем, что конец

@Test  
//Свайп до конца страницы  
public void testSwipeArticle() {  
  
 waitForElementAndClick( //Поиск и клик  
 By.*xpath*("//\*[contains(@text, 'Search Wikipedia')]"),  
 "Cannot find Search Wiki input",  
 5);  
  
 waitForElementAndSendKeys( //Поиск и ввод для Appium  
 By.*xpath*("//\*[contains(@text, 'Search…')]"),  
 "Appium",  
 "Cannot find search input",  
 5);  
  
 waitForElementAndClick( //Поиск и клик поиск среди заголовков по тексту  
 By.*xpath*("//\*[@resource-id = 'org.wikipedia:id/page\_list\_item\_title'][@text = 'Appium']"),  
 "Cannot find Search Appium input",  
 25);  
  
 waitForElementPresent( // ожидаем элемент  
 By.*id*("org.wikipedia:id/view\_page\_title\_text"),  
 "Cannot find article title",  
 35  
 );  
  
 //Свайп до футера (конца страницы)с ограничением до 6 свайпов  
 swipeUpToFindElement(  
 By.*xpath*("//\*[@text = 'View page in browser']"),  
 "Cannot find article title",  
 6  
 );  
}

Метод

//Свайп пока не найдем эл-т с ограничением по свайпам  
protected void swipeUpToFindElement(By by, String error\_message, int max\_swipes){  
 // driver.findElements(by); - находит все эл-ты что есть  
 // driver.findElements(by).size(); - число найденных эл-ов  
 int already\_swipes = 0;  
 while (driver.findElements(by).size() == 0){  
  
 // проверяем что эл-та все еще нет  
 if (already\_swipes > max\_swipes){  
 waitForElementPresent(by, "Cannot find element by swiping up. \n" + error\_message, 0);  
 return;  
 }  
 swipeUpQuick();  
 ++already\_swipes;  
 }  
}