Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №5**

**Дисциплина**: Проектирование мобильных приложений

**Тема**: UI Tests

Выполнил студент гр. 3530901/70203 И.Д. Иванов

(подпись)

Преподаватель И.С. Егорова

(подпись)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Санкт-Петербург

2019

**1. Цели работы**

Ознакомиться с принципами и получить практические навыки разработки UI тестов для Android приложений.

**2. Программа работы**

**2.1. Задача 1.** Простейший UI тест

Ознакомьтесь с Espresso Framework. Разработайте приложение, в котором есть одна кнопка (Button) и одно текстовое поле (EditText). При (первом) нажатии на кнопку текст на кнопке должен меняться.

Напишите Espresso тест, который проверяет, что при повороте экрана содержимое текстового поля (каким бы оно ни было) сохраняется, а надпись на кнопке сбрасывается в исходное состояние.

Внешний вид разработанного приложения (Рис.1.):



Рис.1. Внешний вид разработанного приложения

При нажатии на кнопку текст на ней меняется на “Something 2”. При повороте экрана на кнопке снова отображается текст “Something 1”, т.к. activity пересоздается.

Данные, введённые в поле editText, сохраняются при повороте экрана, т.к. editText использует методы onSaveInstanceState/onRestoreInstanceState для сохранения своего состояния.

Написаны два теста средствами Espresso Framework для проверки сохранения состояния кнопки и поля для ввода текста при переходе из портретного режима в альбомный и наоборот.

Основные компоненты Espresso:

Espresso – основной класс. Содержит в себе статические методы, позволяющие, например, нажимать на системные кнопки (Back, Home);

ViewMatchers – позволяет найти компонент на экране в текущей иерархии;

ViewActions – позволяет взаимодействовать с компонентом (click, longClick, doubleClick и т.д.);

ViewAssertions – позволяет проверить состояние компонента.

План тестирования: задать необходимую ориентацию, очистить поле editText, задать в него данные, нажать на кнопку, проверить, что текст на кнопке изменился, изменить ориентацию, проверить, что текст на кнопке вернулся в исходное состояние, а текст в поле editText сохранился.

Код успешно проходящих тестов представлен в приложении 1.

**2.2. Задача 2.** Тестирование навигации

Возьмите приложение из Лаб №3 о навигации (любое из решений). Напишите UI тесты, проверяющие навигацию между 4мя исходными Activity/Fragment (1-2-3-About).

В результате выполнения данного задания написаны четыре теста:

Первые три теста проверяют правильность переходов между Activity для, соответственно, первой, второй и третьей Activity, а также переходов в ActivityAbout. Кроме того, проверяется сохранение состояния при повороте экрана;

Четвёртый тест проверяет глубину backstack – производится несколько(более 3) переходов между Activity, завершающихся в Activity3, затем выполняется три нажатия на кнопку Back и ожидается завершение приложения.

Код успешно проходящих тестов представлен в приложении 2.

**3. Вывод**

В результаты выполнения данной лабораторной работы были изучены принципы разработки UI тестов для Android приложений, написаны тесты для проверки функциональности компонентов приложения, сохранения состояния, а также для проверки backstack.

**Приложение 1**

@Test  
fun rotateScreen1() {  
 mActivityTestRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT* onView(withId(R.id.*editText*)).perform(clearText())  
 onView(withId(R.id.*editText*)).perform(typeText("Some Text abcd"))  
 onView(withId(R.id.*button*)).perform(click())  
 onView(withId(R.id.*button*)).check(matches(withText("Something 2")))  
 onView(withId(R.id.*button*)).perform(click())  
 onView(withId(R.id.*button*)).check(matches(withText("Something 2")))  
 mActivityTestRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_LANDSCAPE* onView(withId(R.id.*button*)).check(matches(withText("Something 1")))  
 onView(withId(R.id.*editText*)).check(matches(withText("Some Text abcd")))  
  
}  
  
@Test  
fun rotateScreen2() {  
 mActivityTestRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_LANDSCAPE* onView(withId(R.id.*button*)).perform(click())  
 onView(withId(R.id.*button*)).check(matches(withText("Something 2")))  
 onView(withId(R.id.*button*)).perform(click())  
 onView(withId(R.id.*button*)).check(matches(withText("Something 2")))  
 onView(withId(R.id.*editText*)).perform(clearText())  
 onView(withId(R.id.*editText*)).perform(typeText("Hello"))  
 mActivityTestRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT* onView(withId(R.id.*button*)).check(matches(withText("Something 1")))  
 onView(withId(R.id.*editText*)).check(matches(withText("Hello")))  
}

**Приложение 2**

@Test  
fun act1Test() {  
 intentsRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT* Espresso.onView(withId(R.id.*btn\_from\_first\_to\_second*)).perform(ViewActions.click())  
 intended(hasComponent(Activity2::class.*java*.*name*))  
 intentsRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_LANDSCAPE* Espresso.onView(withId(R.id.*textView\_act2*)).check(matches(withText("Activity 2")))  
 Espresso.onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 Espresso.onView(withId(R.id.*textView\_act1*)).check(matches(withText("Activity 1")))  
 openActionBarOverflowOrOptionsMenu(getInstrumentation().*targetContext*)  
 Espresso.onView(Matchers.allOf(withText("About"))).perform(ViewActions.click())  
 intended(hasComponent(ActivityAbout::class.*java*.*name*))  
 Espresso.onView(withId(R.id.*textView*)).check(matches(withText("This is Activity 1")))  
 intentsRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT* Espresso.onView(withId(R.id.*textView*)).check(matches(withText("This is Activity 1")))  
 Espresso.onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 Espresso.onView(withId(R.id.*textView\_act1*)).check(matches(withText("Activity 1")))  
  
}  
  
@Test  
fun act2Test() {  
 intentsRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT* Espresso.onView(withId(R.id.*btn\_from\_first\_to\_second*)).perform(ViewActions.click())  
 intended(hasComponent(Activity2::class.*java*.*name*))  
 openActionBarOverflowOrOptionsMenu(getInstrumentation().*targetContext*)  
 Espresso.onView(Matchers.allOf(withText("About"))).perform(ViewActions.click())  
 intended(hasComponent(ActivityAbout::class.*java*.*name*))  
 Espresso.onView(withId(R.id.*textView*)).check(matches(withText("This is Activity 2")))  
 Espresso.onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 Espresso.onView(withId(R.id.*textView\_act2*)).check(matches(withText("Activity 2")))  
 Espresso.onView(withId(R.id.*btn\_from\_second\_to\_third*)).perform(ViewActions.click())  
 intended(hasComponent(Activity3::class.*java*.*name*))  
 intentsRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_LANDSCAPE* Espresso.onView(withId(R.id.*textView\_act3*)).check(matches(withText("Activity 3")))  
 Espresso.onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 Espresso.onView(withId(R.id.*textView\_act2*)).check(matches(withText("Activity 2")))  
 Espresso.onView(withId(R.id.*btn\_from\_second\_to\_first*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*textView\_act1*)).check(matches(withText("Activity 1")))  
 try {  
 Espresso.onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 Assert.fail()  
 } catch (e: NoActivityResumedException) {  
  
 }  
}  
  
@Test  
fun act3Test() {  
 intentsRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT* Espresso.onView(withId(R.id.*btn\_from\_first\_to\_second*)).perform(ViewActions.click())  
 intended(hasComponent(Activity2::class.*java*.*name*))  
 Espresso.onView(withId(R.id.*btn\_from\_second\_to\_third*)).perform(ViewActions.click())  
 intended(hasComponent(Activity3::class.*java*.*name*))  
 openActionBarOverflowOrOptionsMenu(getInstrumentation().*targetContext*)  
 Espresso.onView(Matchers.allOf(withText("About"))).perform(ViewActions.click())  
 intended(hasComponent(ActivityAbout::class.*java*.*name*))  
 Espresso.onView(withId(R.id.*textView*)).check(matches(withText("This is Activity 3")))  
 Espresso.onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 Espresso.onView(withId(R.id.*textView\_act3*)).check(matches(withText("Activity 3")))  
 Espresso.onView(withId(R.id.*btn\_from\_third\_to\_second*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*textView\_act2*)).check(matches(withText("Activity 2")))  
 Espresso.onView(withId(R.id.*btn\_from\_second\_to\_third*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*textView\_act3*)).check(matches(withText("Activity 3")))  
 Espresso.onView(withId(R.id.*btn\_from\_third\_to\_first*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*textView\_act1*)).check(matches(withText("Activity 1")))  
 intentsRule.*activity*.*requestedOrientation* = ActivityInfo.*SCREEN\_ORIENTATION\_LANDSCAPE* Espresso.onView(withId(R.id.*textView\_act1*)).check(matches(withText("Activity 1")))  
 try {  
 Espresso.onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 Assert.fail()  
 } catch (e: NoActivityResumedException) {  
  
 }  
}  
  
@Test  
fun backstackTest() {  
 Espresso.onView(withId(R.id.*btn\_from\_first\_to\_second*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*btn\_from\_second\_to\_third*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*btn\_from\_third\_to\_first*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*btn\_from\_first\_to\_second*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*btn\_from\_second\_to\_first*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*btn\_from\_first\_to\_second*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*btn\_from\_second\_to\_third*)).perform(ViewActions.click())  
 Espresso.onView(withId(R.id.*textView\_act3*)).check(matches(withText("Activity 3")))  
 Espresso.onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 Espresso.onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 try {  
 Espresso.onView(ViewMatchers.isRoot()).perform(ViewActions.pressBack())  
 Assert.fail()  
 } catch (e: NoActivityResumedException) {  
  
 }  
}