SAMSUNG

IT Академия Kotlin

Kotlin Базовый Курс

2.10. Отладка и тестирование приложений

Введение в тестирование и отладку

SAMSUNG IT Академия Kotlin

«Программы без ошибок можно написать двумя способами, но работает только третий»
Алан Перлис, американский ученый в области информатики

«Отладка кода вдвое сложнее, чем его написание. Так что если вы пишете код настолько умно, насколько можете, то вы по определению недостаточно сообразительны, чтобы его отлаживать»
Брайан Керниган, создатель языка Си

Отладка — это процесс определения и устранения причин ошибок.

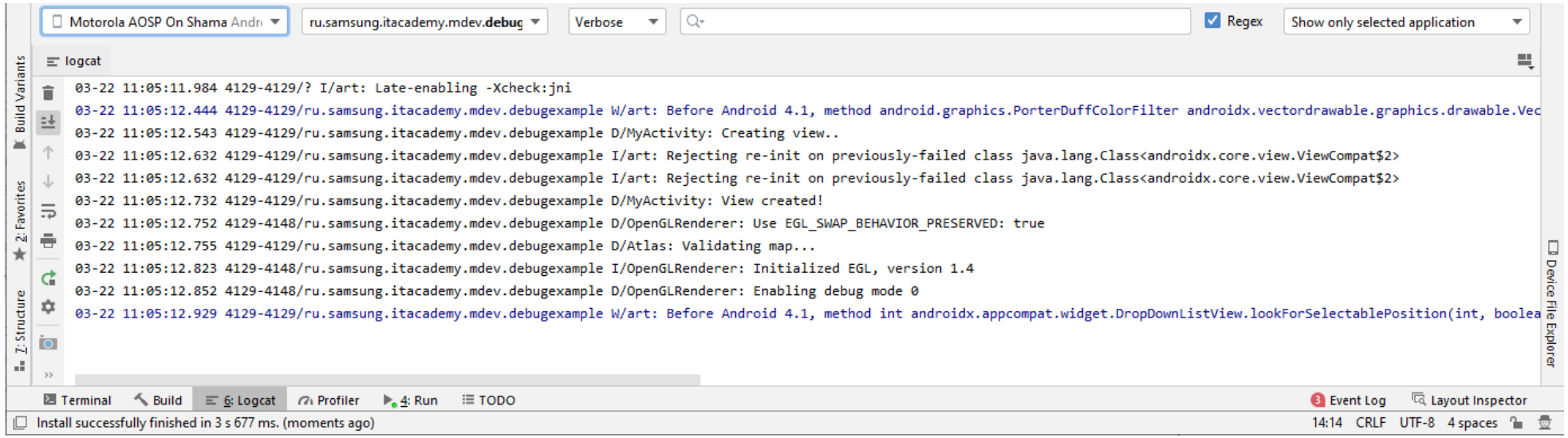
Тестирование — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Отладочный вывод и логирование

SAMSUNG

IT Академия Kotlin

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    Log.d(LOG_TAG, "Creating view..")
    setContentView(R.layout.activity_main)
    Log.d(LOG_TAG, "View created!")
}
```



Отладочный вывод и логирование

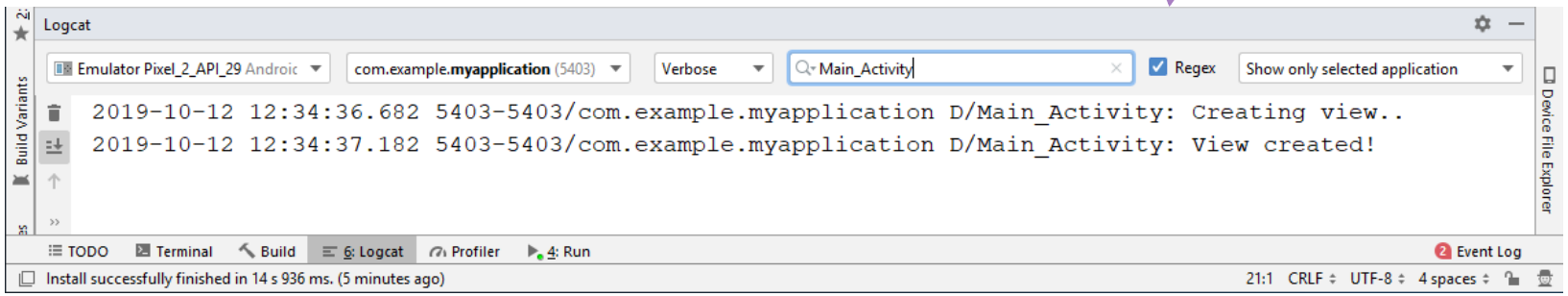
SAMSUNG

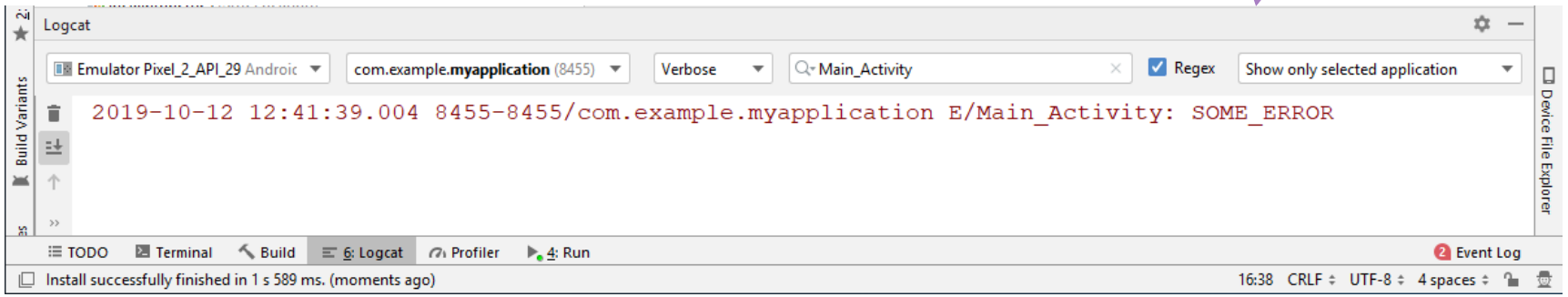
IT Академия Kotlin

- *Log.e() – ошибки (error);*
- *Log.w() – предупреждения (warning);*
- *Log.i() – информация (info);*
- *Log.d() – отладка (debug);*
- *Log.v() – подробности (verbose);*
- *Log.wtf() – очень серьезная ошибка.*

```
Log.d("Main_Activity", "Creating view…")
```

```
Log.e("Main_Activity", "SOME_ERROR")
```



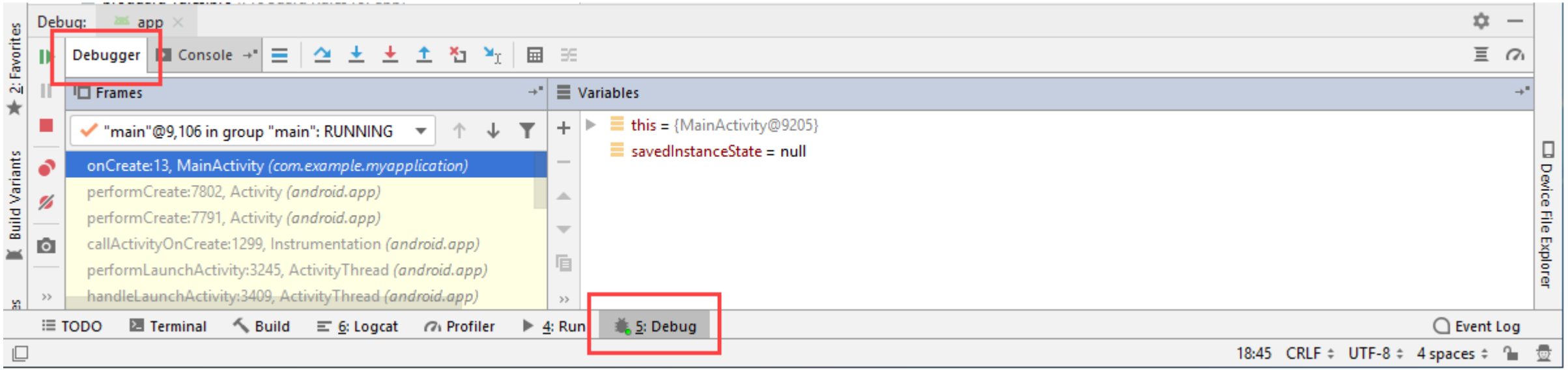


Использование отладчика

SAMSUNG

IT Академия Kotlin

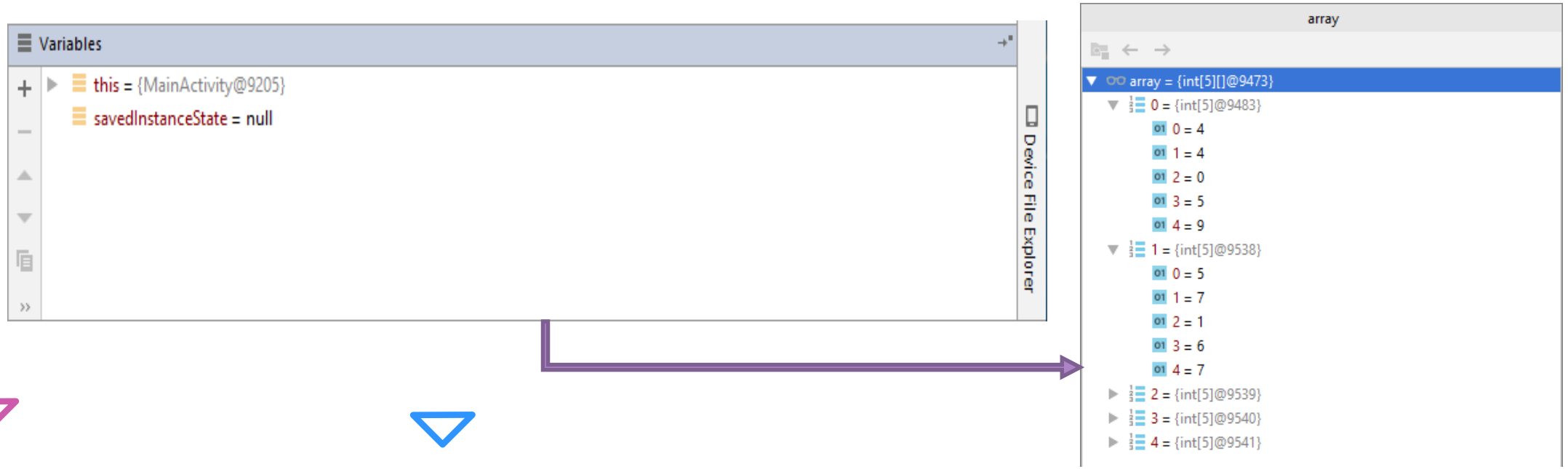
```
10    override fun onCreate(savedInstanceState: Bundle?) {  savedInstanceState: null
11        super.onCreate(savedInstanceState)  savedInstanceState: null
12        Log.d(LOG_TAG,  msg: "Creating view..")
13        setContentView(R.layout.activity_main)
14        Log.d(LOG_TAG,  msg: "View created!")
15        Log.v(LOG_TAG,  msg: "Matrix: \n" + createMatrix( n: 5,  m: 5)?.let { matrixToString(it) });
16    }
```



Использование отладчика

Для работы в режиме отладки удобно пользоваться панелью управления отладкой. На ней расположены кнопки:

Android	Eclipse	Описание
Resume Program (F9)	Resume (F8)	Продолжить выполнение программы
Pause Program	Suspend	Приостановка
Stop (Ctrl + F2)	Terminate (Ctrl + F2)	Прекращение режима отладки
Step Into (F7)	Step Into (F5)	Зайти внутрь метода
Step Over (F8)	Step Over (F6)	Переход к следующей строке
Step Out (Shift + F8)	Step Return (F7)	Выход из текущего метода



Модульное тестирование

SAMSUNG IT Академия Kotlin

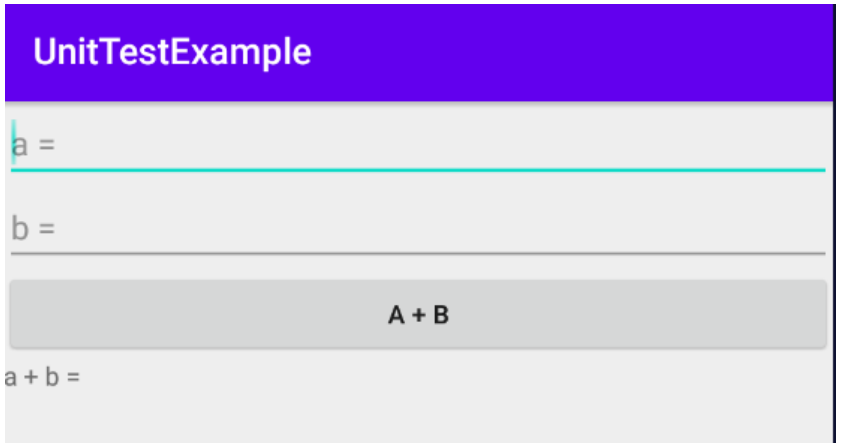
Локальные модульные тесты (unit-тесты) проверяют работу метода, класса, компонента. По сути вы тестируете код, который можно проверить без применения устройства или эмулятора. Подобные тесты находятся в папке Test проекта. Для создания для юнит-тестов можно использовать следующие инструменты:

- JUnit
- Mockito
- PowerMock

JUnit – это фреймворк, предназначенный для тестирования программ, разработанных с использованием технологии Java. Ключевая идея фреймворка – “сначала тесты, потом код”.

Тестовый случай (Test Case) в юнит тестировании – это код, который проверяет работу другого кода (класса, метода и т.п.) в соответствии с заданными требованиями.

Пример




```
class Calculator {
    fun add(a: Int, b: Int): Int {
        return a + b
    }
}
```

```
class CalculatorTest {
    private var calculator: Calculator? = null

    @Before fun setUp() {
        calculator = Calculator()
    }

    @Test fun addition() {
        Assert.assertEquals(3, calculator!!.add(1,
2).toLong())
    }

    @After fun tearDown() {
        calculator = null
    }
}
```

Другие виды тестирования

- Функциональное тестирование (functional testing)
- Системное тестирование (system testing)
- Тестирование проиводительности (performance testing)
- Регрессионное тестирование (regression testing)
- Модульное тестирование (unit testing)
- Тестирование безопасности (security testing)

SAMSUNG IT Академия Kotlin

Спасибо