日本Androidの会 浜松支部 第４回 ミーティング　Aug 13, 2011

Androidテスト入門

有山圭二（日本Androidの会 関西支部）

http://goo.gl/b8vNs

はじめに

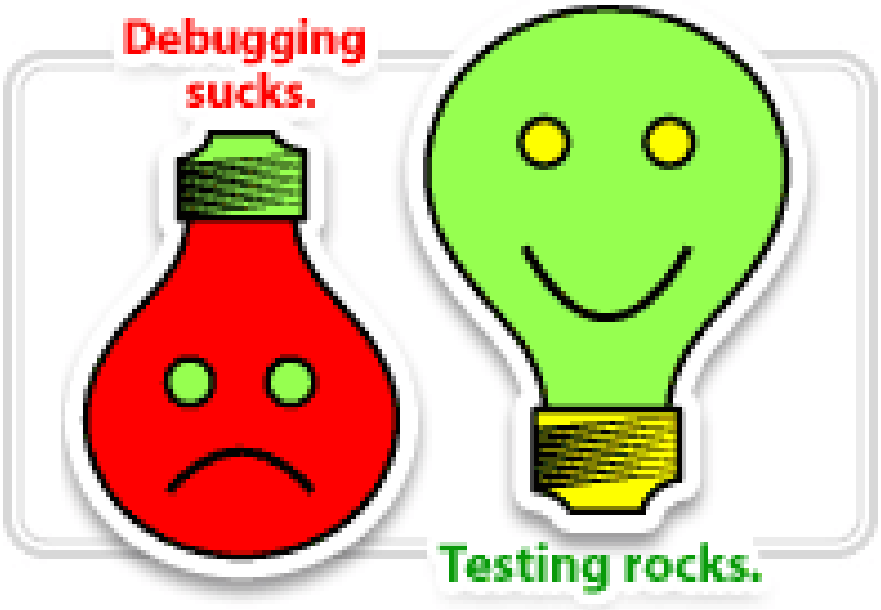
- Androidは、比較的モジュール化のしやすいプラットフォームです。
  - 4つのシステムコンポーネント
    (Activity, Service, ContentProvider, BroadcastReceiver)
  - Intentによる緩やかな連携（疎結合）

デバッグ、どうしてますか？

デバッグ、デバッグ、デバッグ…

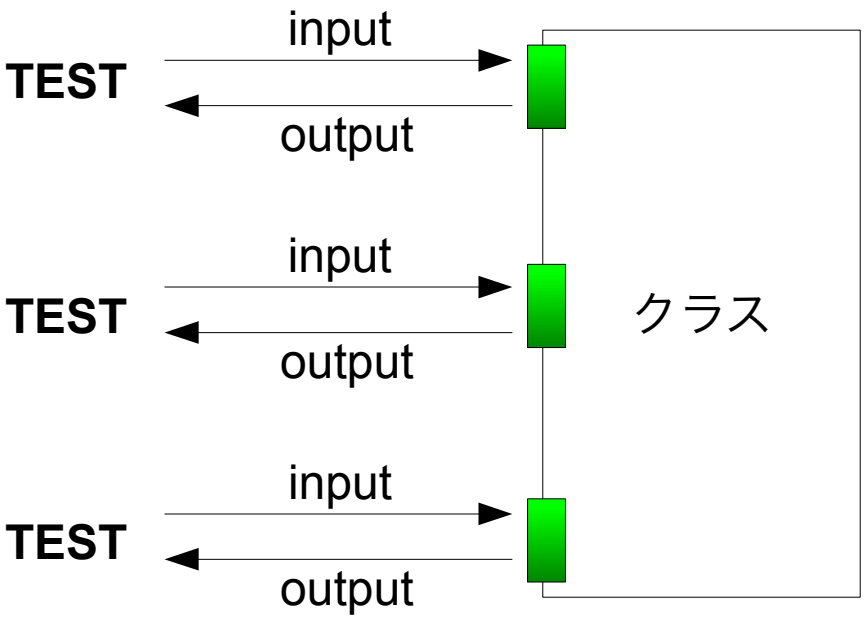
テストのススメ

- 不具合が起きてからデバッグをするのではなく、
  コードを変更するたびにテストをする
- コード変更で引き起こされる不具合を早期に見つけて、
  対処を可能にする



ユニットテスト

- クラス・メソッドなどの小さな単位で、テストをする（単体テスト）
- 入力(input)に対して正しい出力(output)がされているかをテストする

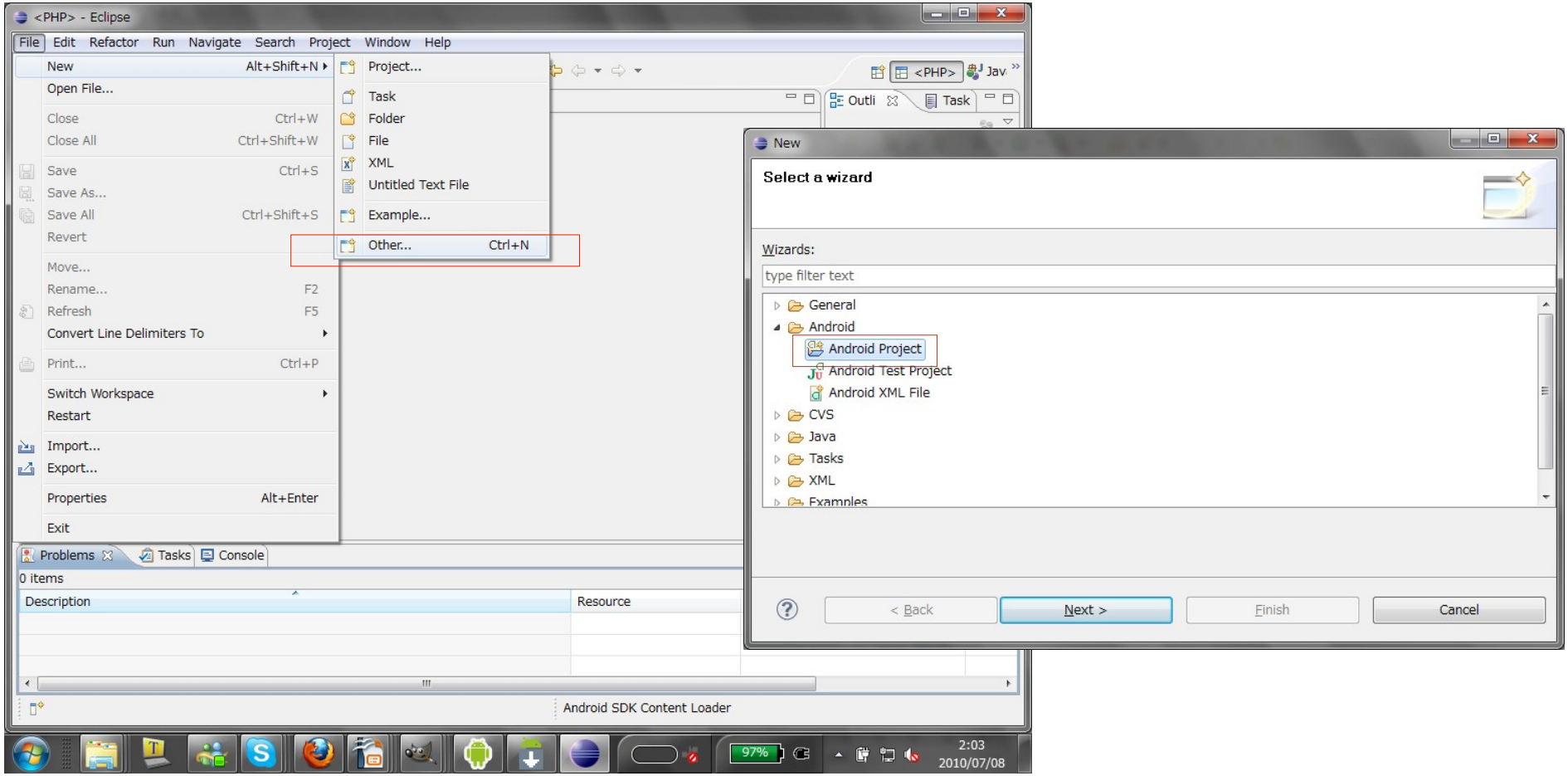


JUnit

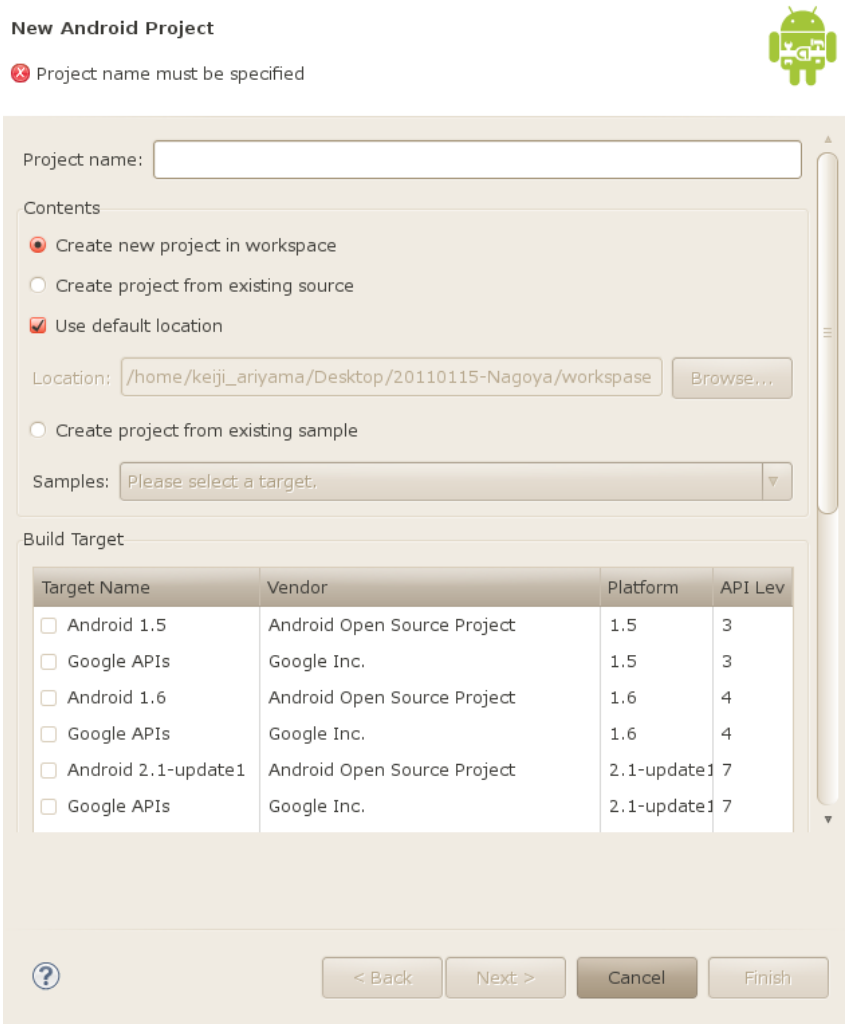
- Java用のユニットテストを自動化するためのフレームワーク
- AndroidのSDKは、JUnit3による自動テストをサポート

JUnitの基本

プロジェクトの作成



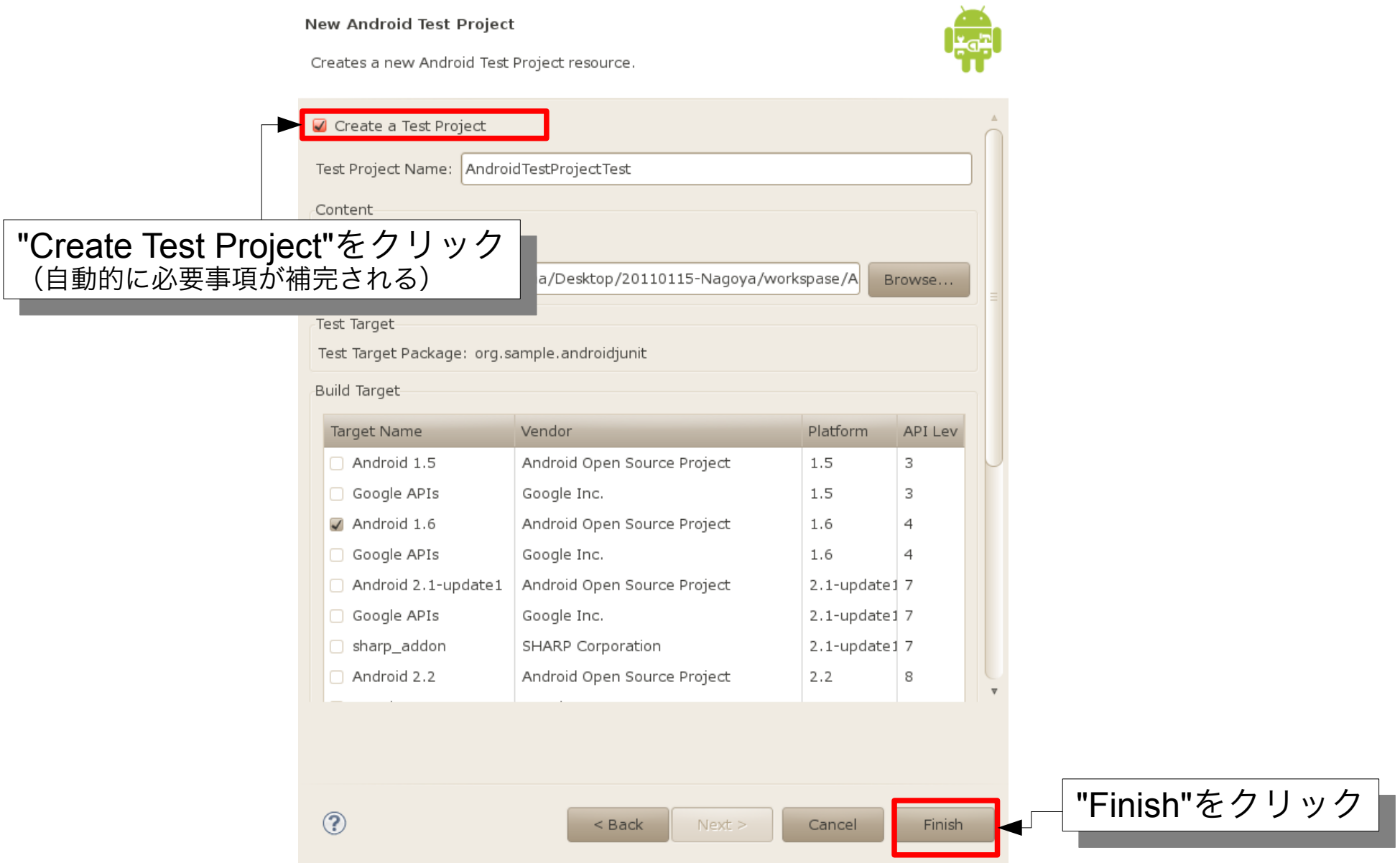
プロジェクトの作成



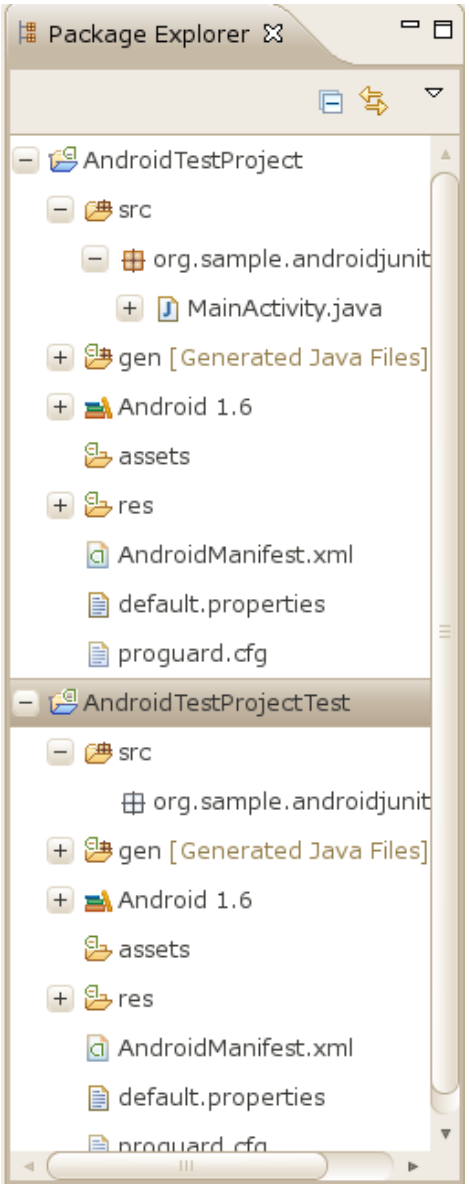
- Build Target : Android 1.6　　API Level 4
- Properties
  - Application name : AndroidJUnit
  - Package name : org.sample.androidtest
  - Create Activity : MainActivity
  - Min SDK Version : 4

入力後、"Next >" ボタンを押す

テストプロジェクトの作成



テストプロジェクトの作成



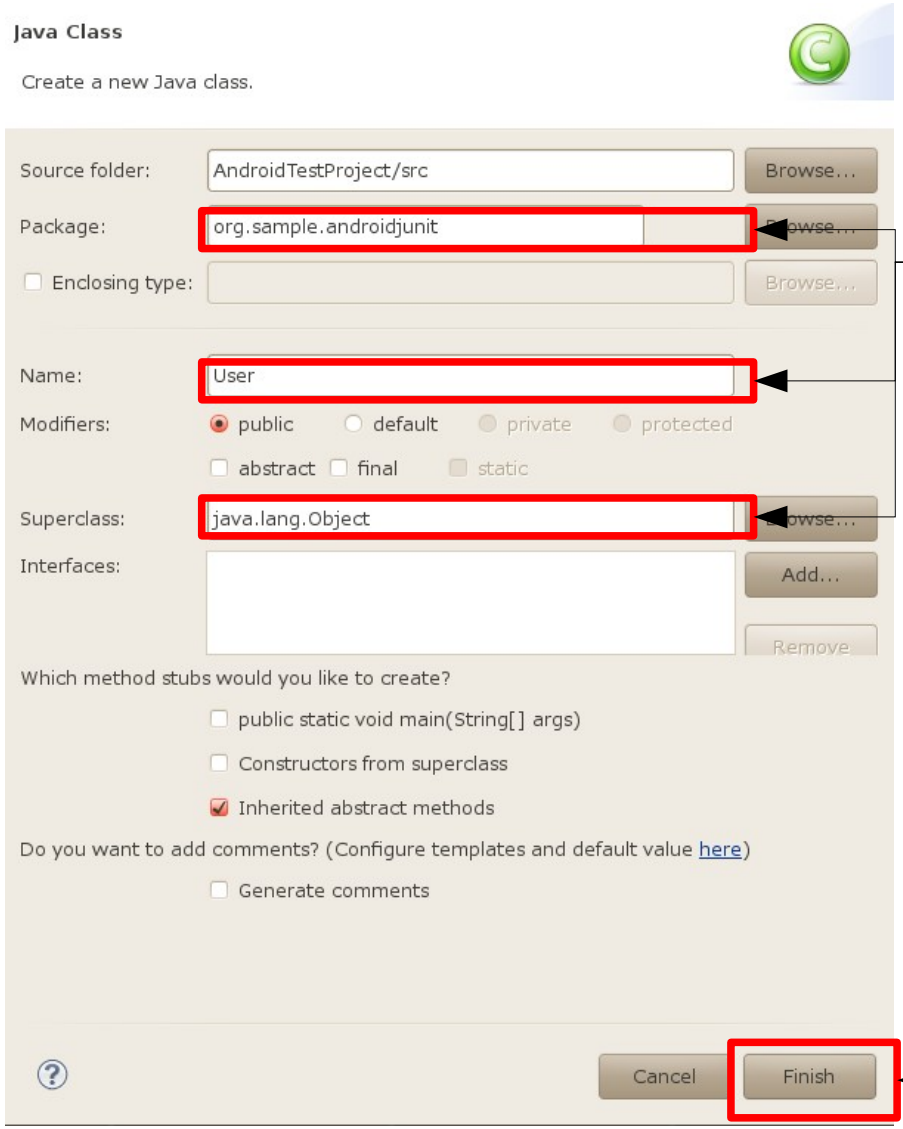
- テストプロジェクトの名前は、[プロジェクト名]Test
- テスト対象プロジェクトの各パッケージに対して、末尾に.testを付けて、テストケースを配置する（慣例）

例：

org.sample.androidjunit

org.sample.androidjunit.test

テスト対象クラスの作成



AndroidTestProject

Package: org.sample.androidjunit

Name: User

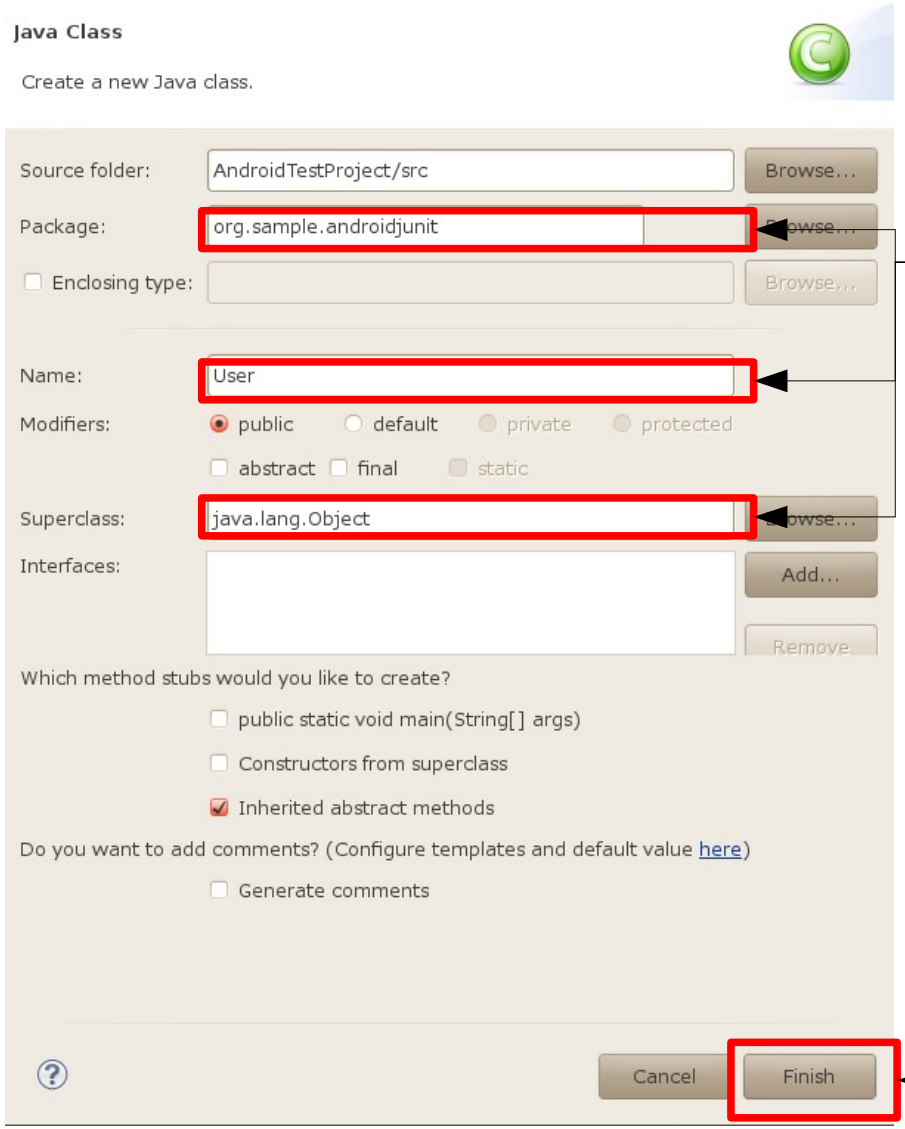
Superclass: java.lang.Object

"Finish"をクリック

テスト対象クラスの作成

```
package org.sample.androidjunit;

public class User {

	private int id = -1;
	private String name = null;

	public void setId(int id) {
		this.id = id;
	}
	public int getId() {
		return id;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getName() {
		return name;
	}
}
```

テストケースの作成



AndroidTestProjectTest

Package: org.sample.androidjunit.test

Name: UserTest

Superclass: junit.framework.TestCase

"Finish"をクリック

テストケースの作成

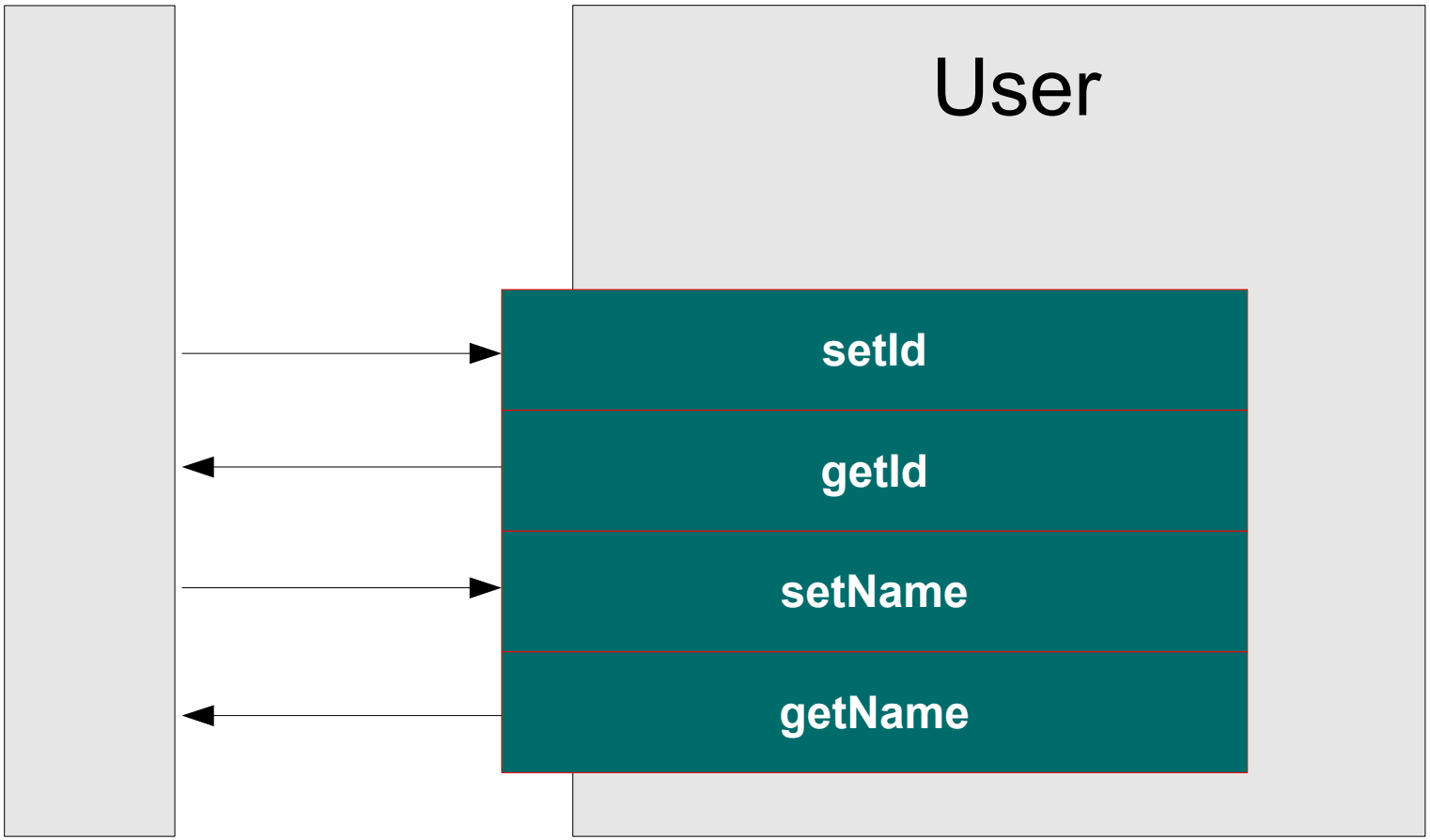
```
package org.sample.androidjunit.test

import junit.framework.TestCase;

public class UserTest extends TestCase {

}
```

- 空のテストケースが出来上がる

テストを書く

```
package org.sample.androidjunit;

public class User {

	private int id = -1;
	private String name = null;

	public void setId(int id) {
		this.id = id;
	}
	public int getId() {
		return id;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getName() {
		return name;
	}
}
```

テストを書く

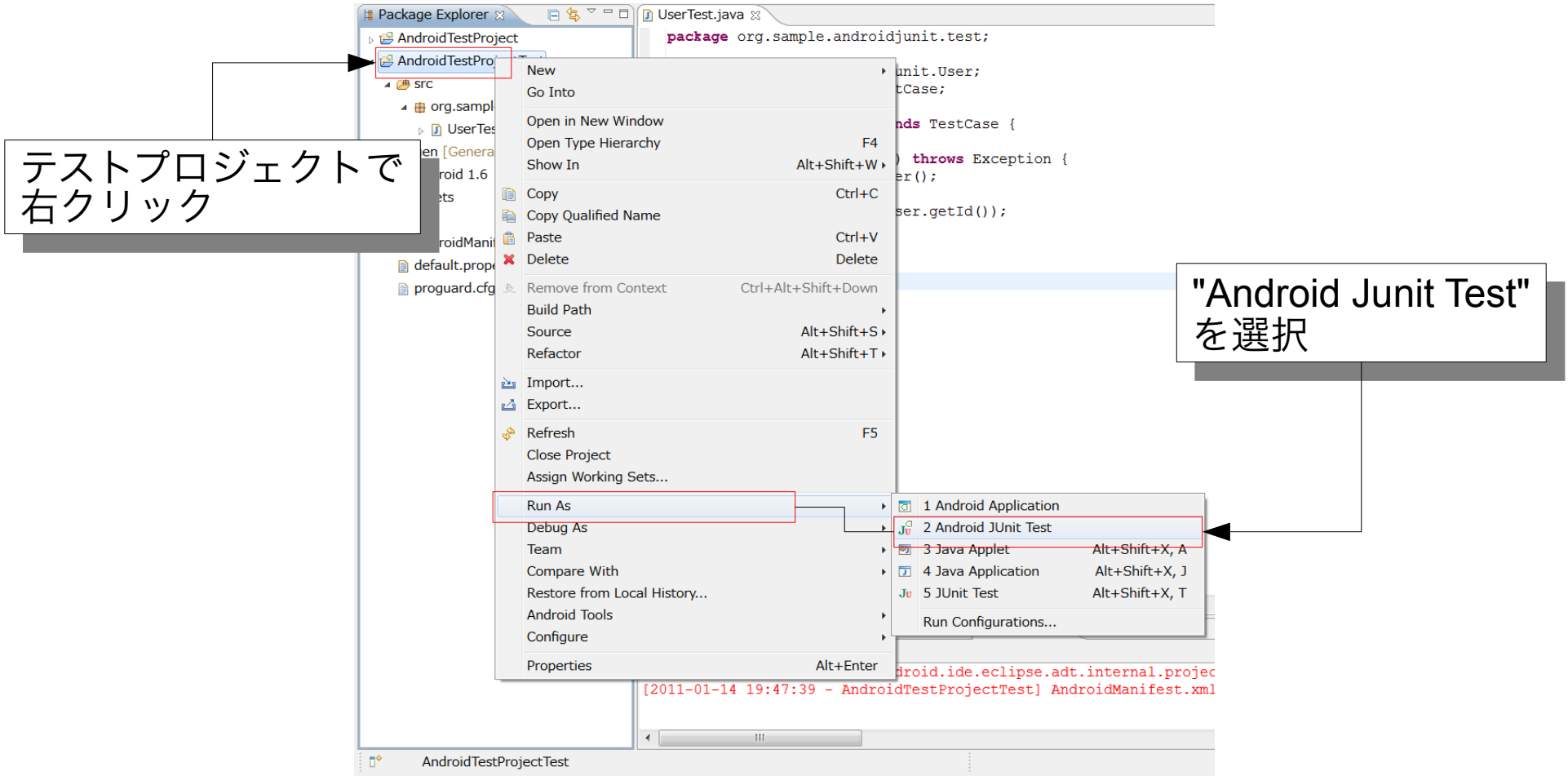


テストを書く

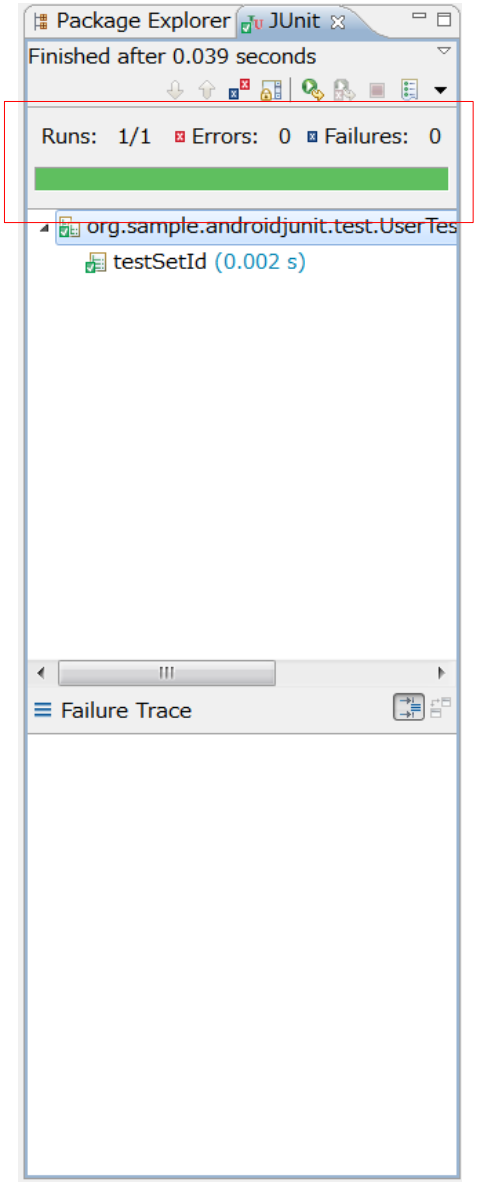
```
package org.sample.androidjunit.test

import org.sample.androidjunit.User;

import junit.framework.TestCase;

public class UserTest extends TestCase {

	public void testSetId() throws Exception {
		User user = new User();
		user.setId(20);
		assertEquals(20, user.getId());
	}

}
```

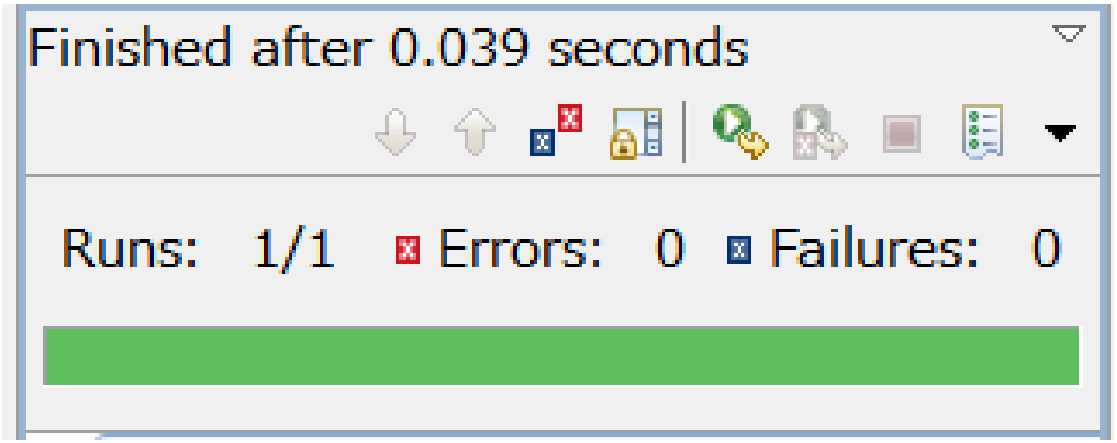
- テストメソッドは、メソッド名を "test" から始める
  - throws Exception を付ける
- assertEquals(期待する値, 値) で評価

テストの実行



テストの実行

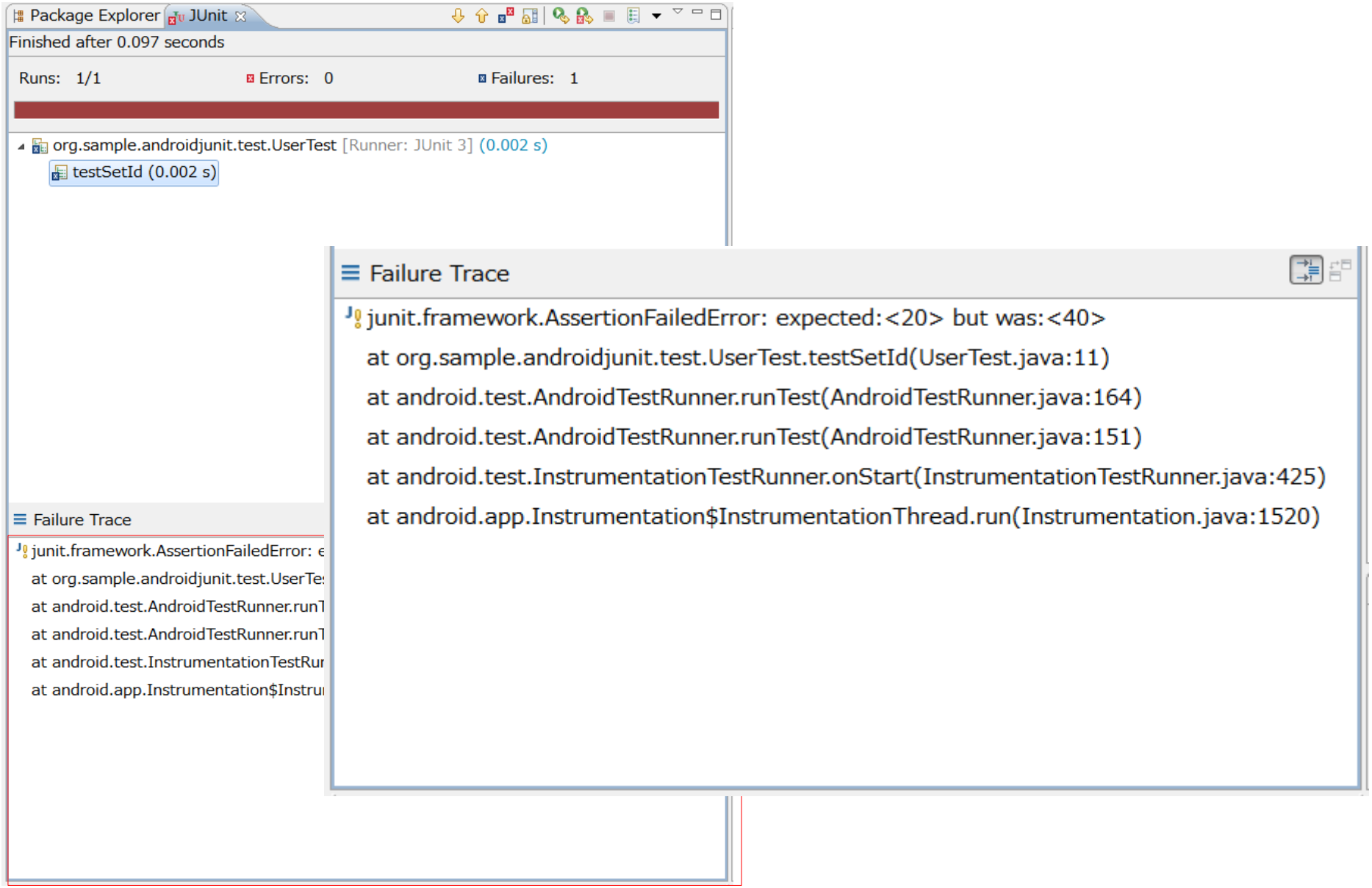




テストの失敗

```
package org.sample.androidjunit;

public class User {

	private int id = -1;
	private String name = null;

	public void setId(int id) {
		this.id = id*2;
	}
	public int getId() {
		return id;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getName() {
		return name;
	}
}
```

テストの失敗



テストの追加

```
package org.sample.androidjunit.test

import org.sample.androidjunit.User;

import junit.framework.TestCase;

public class UserTest extends TestCase {

	public void testSetId() throws Exception {
		User user = new User();
		user.setId(20);
		assertEquals(20, user.getId());
	}

	public void testSetName() throws Exception {
		User user = new User();
		user.setName("Keiji Ariyama");
		assertEquals("Keiji Ariyama", user.getName());
	}

}
```

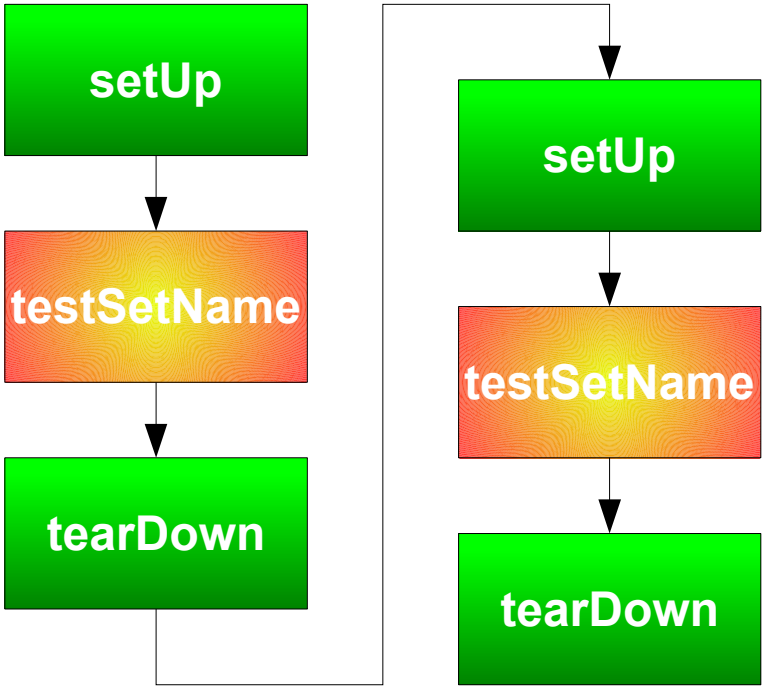
setUpとtearDown

```
package org.sample.androidjunit.test

import org.sample.androidjunit.User;

import junit.framework.TestCase;

public class UserTest extends TestCase {

        private User mUser1 = null;
        public void setUp() {
                mUser1 = new User();
                mUser1.setId(10);
                mUser1.setName("Keiji Ariyama");
        }

        public void tearDown() {
                mUser1 = null;
        }

        public void testSetId() throws Exception {
                mUser1.setId(20);
                assertEquals(20, mUser1.getId());
        }
```

```
        public void testSetName() throws Exception {
                mUser1.setName("Keiji Ariyama");
                assertEquals("Keiji Ariyama1",
                                mUser1.getName() );
        }

}
```

- setUpはテスト前、tearDownはテスト後に実行
- データベース接続の取得と開放、一時ファイルの作成と削除などに使う

よくある誤解



- setUpとtearDownは、それぞれのテストメソッドが実行される前と後に実行される

期待される例外のテスト

- setName(String name)の引数に、nullを与えた場合、IllegalArgumentException を throw する

```
package org.sample.androidjunit;

public class User {

	// 省略

	public void setName(String name) {
		if(name == null) throw new IllegalArgumentException();
		this.name = name;
	}

	// 省略
}
```

期待される例外のテスト

```
public void testSetNameNull()
        throws Exception {
    try {
        mUser1.setName(null);
        fail(); // 到達したら失敗
    } catch (IllegalArgumentException e) {
    }
    assertTrue(true);       // 成功
}
```

- fail()は、強制的にテストを失敗と判定する
- 例外が発生して、catchを通ると成功

Android固有のテスト

Android固有のテスト

- システムコンポーネントのテスト
  - Activity
  - Service
  - Content Providers
  - BroadcastReceiver

システムコンポーネントのテスト

- コンポーネントに対応した「テスト用クラス」が用意されている（BroadcastReceiver以外）

Activity	ActivityUnitTestCase ActivityInstrumentationTestCase2
Service	ServiceTestCase
ContentProviders	ProviderTestCase2
BroadcastReceiver	N/A

Activityのテスト

テスト対象のActivityを作る

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView android:id="@+id/tv_message"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<Button android:id="@+id/btn_execute"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="button"
    />
</LinearLayout>
```

res/layout/main.xml

テスト対象のActivityを作る

```
package org.sample.androidjunit;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;

public class MainActivity extends Activity implements OnClickListener {

    private TextView mTvMessage = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        findViewById(R.id.btn_execute).setOnClickListener(this);
        mTvMessage = (TextView) findViewById(R.id.tv_message);
    }

    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_execute:
                mTvMessage.setText("ボタンが押された");
                break;
        }
    }
}
```

src/org.sample.androidjunit/MainActivity.java

Activityのテストケースを作る

```
package org.sample.androidjunit.test;

import org.sample.androidjunit.MainActivity;

import android.app.Activity;
import android.app.Instrumentation;
import android.test.ActivityInstrumentationTestCase2;
import android.view.MotionEvent;
import android.widget.Button;
import android.widget.TextView;

import org.sample.androidjunit.R;

public class MainActivityTest extends ActivityInstrumentationTestCase2<MainActivity> {

    public MainActivityTest() {
        super("org.sample.androidjunit", MainActivity.class);
    }
```

src/org.sample.androidjunit.test/MainActivityTest.java

Activityのテストケースを作る

```
private Activity mActivity = null;
private Instrumentation mInstrumentation = null;

public void setUp() {
    mActivity = getActivity();
    mInstrumentation = getInstrumentation();
}

public void tearDown() {
    mActivity = null;
    mInstrumentation = null;
}
```

src/org.sample.androidjunit.test/MainActivityTest.java

Activityのテストケースを作る

```
    public void testPushButton() throws Exception {

        final Button btnExecute = (Button) mActivity.findViewById(R.id.btn_execute);
        final TextView tvMessage = (TextView) mActivity.findViewById(R.id.tv_message);

        mActivity.runOnUiThread(new Runnable() {
            public void run() {
                btnExecute.dispatchTouchEvent(MotionEvent.obtain(5, 0, MotionEvent.ACTION_DOWN, 0,
                        0, 0));
                btnExecute.dispatchTouchEvent(MotionEvent.obtain(5, 0, MotionEvent.ACTION_UP, 0, 0,
                        0));
            }
        });

        mInstrumentation.waitForIdleSync();

        assertEquals("ボタンが押された", tvMessage.getText());
    }

}     // public class MainActivityTest extends ActivityInstrumentationTestCase2<MainActivity>
```

src/org.sample.androidjunit.test/MainActivityTest.java

Activityのテスト - Instrumentation

- Activityのライフサイクル関係
  - callActivityOnCreate
  - callActivityOnResume
  - etc...
- ユーザー操作関係
  - sendKeySync
  - sendPointerSync
  - sendTrackballEventSync
  - etc...

Serviceのテスト

テスト対象のServiceを作る

```
package org.sample.androidjunit;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

public class BackgroundService extends Service {

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

}
```

- BackgroundService を作る

AndroidManifest.xmlに追加

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
	package="org.sample.androidjunit" android:versionCode="1"
	android:versionName="1.0">
	<application android:icon="@drawable/icon" android:label="@string/app_name">
		<activity android:name=".MainActivity" android:label="@string/app_name">
			<intent-filter>
				<action android:name="android.intent.action.MAIN" />
				<category android:name="android.intent.category.LAUNCHER" />
			</intent-filter>
		</activity>
		<service android:name="BackgroundService"></service>
	</application>
</manifest>
```

AIDLの作成

```
package org.sample.androidjunit;

interface IBackgroundService {

	int getState();
	boolean stop();

}
```

src/org.sample.androidjunit/IBackgroundService.aidl

- 作成して保存後、
  gen/org.sample.androidjunit/IBackgroundService.java
  が作成されているのを確認する
- 手動でビルドが必要な場合がある

テスト対象のServiceを作る

```
package org.sample.androidjunit;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.os.RemoteException;

public class BackgroundService extends Service {

    private int mState = 7;
    private IBackgroundService.Stub mBinder = new IBackgroundService.Stub() {
        public boolean stop() throws RemoteException {
            mState = 0;
            return true;
        }
        public int getState() throws RemoteException {
            return mState;
        }
    };

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }

}
```

Serviceのテストケース

```
package org.sample.androidjunit.test;

import org.sample.androidjunit.BackgroundService;
import org.sample.androidjunit.IBackgroundService;

import android.content.Intent;
import android.test.ServiceTestCase;

public class BackgroundServiceTest extends ServiceTestCase<BackgroundService> {

    public BackgroundServiceTest() {
        super(BackgroundService.class);
    }

    private IBackgroundService mBinder = null;

    protected void setUp() throws Exception {
        super.setUp();

        Intent intent = new Intent(getContext(), BackgroundService.class);
        mBinder = IBackgroundService.Stub.asInterface(bindService(intent));
    }
```

src/org.sample.androidjunit.test/BackgroundServiceTest.java

Serviceのテストケース

```
    public void testGetState() throws Exception {
        int state =  mBinder.getState();
        assertEquals(7, state);
    }

    public void testServiceStop() throws Exception {
        mBinder.stop();
        int state =  mBinder.getState();
        assertEquals(0, state);
    }

}       // public class BackgroundServiceTest extends ServiceTestCase<BackgroundService> {
```

src/org.sample.androidjunit.test/BackgroundServiceTest.java

ContentProvidersのテスト

ContentProviderの作成

```
package org.sample.androidjunit;

import android.content.ContentProvider;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;

public class InfoProvider extends ContentProvider {

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public String getType(Uri uri) {
        // TODO Auto-generated method stub
        return null;
    }
```

ContentProviderの作成

```
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public boolean onCreate() {
        // TODO Auto-generated method stub
        return false;
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
            String sortOrder) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
        // TODO Auto-generated method stub
        return 0;
    }

}       // public class InfoProvider extends ContentProvider {
```

AndroidManifest.xmlに追加

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
	package="org.sample.androidjunit" android:versionCode="1"
	android:versionName="1.0">
	<application android:icon="@drawable/icon" android:label="@string/app_name">
		<activity android:name=".MainActivity" android:label="@string/app_name">
			<intent-filter>
				<action android:name="android.intent.action.MAIN" />
				<category android:name="android.intent.category.LAUNCHER" />
			</intent-filter>
		</activity>
		<service android:name="BackgroundService"></service>
		<provider android:name="InfoProvider"
				android:authorities="org.sample.jandroidunit.provider">
		</provider>
	</application>
</manifest>
```

AndroidManifest.xmlに追加

```
package org.sample.androidjunit;

import android.content.ContentProvider;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;

public class InfoProvider extends ContentProvider {

    // 省略

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        String name = values.getAsString("name");
        return Uri.parse(uri.toString() +"/"+ name);
    }

    // 省略
}
```

ContentProviderのテストコード

```
package org.sample.androidjunit.test;

import org.sample.androidjunit.InfoProvider;

import android.content.ContentValues;
import android.net.Uri;
import android.test.ProviderTestCase2;

public class InfoProviderTest extends ProviderTestCase2<InfoProvider> {

    public InfoProviderTest() {
        super(InfoProvider.class, AUTHORITY);
    }

    private static final String AUTHORITY = "org.sample.jandroidunit.provider";
    private InfoProvider mProvider = null;
    private Uri mAuthority = null;

    protected void setUp() throws Exception {
        super.setUp();

        mProvider = getProvider();
        mAuthority = Uri.parse(AUTHORITY);
    }
```

ContentProviderのテストコード

```
    public void testInsert() throws Exception {
        ContentValues values = new ContentValues();
        values.put("name", "KeijiAriyama");

        Uri result = mProvider.insert(mAuthority, values);

        String expect = AUTHORITY + "/" + values.getAsString("name");
        assertEquals(expect, result.toString());
    }

}       // public class InfoProviderTest extends ProviderTestCase2<InfoProvider>
```

JUnit の まとめ

- ユニットテストを自動化して、定期的に実行することで、
  コードの変更に伴う意図しないエラーやバグの作り込みを防ぐ。
- 複雑なメソッドやクラスでも、ユニットテストの内容を見ることで仕様を
  把握できる。
- **ユーザーの意図しない操作による「新しいエラー」を
  見つけ出すのは不得意。**

monkey の 出番です！

monkeyとは

- AndroidのUIテストツール
- お猿のようにランダムで触って（正確にはUIイベントを起こして）、エラーが起こらないか見る

monkeyの使い方

- エミュレータを起動
- cmdを起動
  - SDK/platform-toolsへ移動(cd)
  - adb shell monkey -v -p com.android.browser 100

実行してみましょう

UI/Application Exerciser Monkey

http://developer.android.com/guide/developing/tools/monkey.html

Monkeyのログ

```
$ adb shell monkey -v -p com.android.browser 100
:Monkey: seed=0 count=100
:AllowPackage: com.android.browser
:IncludeCategory: android.intent.category.LAUNCHER
:IncludeCategory: android.intent.category.MONKEY
// Event percentages:
//   0: 15.0%
//   1: 10.0%
//   2: 15.0%
//   3: 25.0%
//   4: 15.0%
//   5: 2.0%
//   6: 2.0%
//   7: 1.0%
//   8: 15.0%
:Switch:
#Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10
000000;component=com.android.browser/.BrowserActivity;end
    // Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER]
cmp=com.android.browser/.BrowserActivity } in package com.android.browser
```

Monkeyのログ

```
:Sending Pointer ACTION_MOVE x=-4.0 y=2.0
:Sending Pointer ACTION_UP x=0.0 y=0.0
:Sending Pointer ACTION_DOWN x=47.0 y=122.0
    // Allowing start of Intent { act=android.intent.action.VIEW cat=[android.intent.category.BROWSABLE]
dat=http://www.google.com/m?client=ms-android-google cmp=com.android.browser/.BrowserActivity } in
package com.android.browser
    // Allowing start of Intent { act=android.intent.action.VIEW cat=[android.intent.category.BROWSABLE]
dat=http://www.google.com/m?client=ms-android-google cmp=com.android.browser/.BrowserActivity } in
package com.android.browser
:Sending Pointer ACTION_UP x=29.0 y=129.0
:Sending Pointer ACTION_DOWN x=255.0 y=99.0
:Sending Pointer ACTION_UP x=255.0 y=99.0
:Sending Pointer ACTION_DOWN x=295.0 y=63.0
:Sending Pointer ACTION_UP x=290.0 y=53.0
:Sending Pointer ACTION_MOVE x=-5.0 y=3.0
:Sending Pointer ACTION_MOVE x=0.0 y=-5.0
    // Rejecting start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.HOME]
cmp=com.android.launcher/.Launcher } in package com.android.launcher
:Sending Pointer ACTION_DOWN x=74.0 y=41.0
:Sending Pointer ACTION_UP x=74.0 y=41.0
:Sending Pointer ACTION_MOVE x=3.0 y=-2.0
:Sending Pointer ACTION_UP x=0.0 y=0.0
:Sending Pointer ACTION_MOVE x=-4.0 y=2.0
:Dropped: keys=0 pointers=0 trackballs=0 flips=0
## Network stats: elapsed time=10531ms (10531ms mobile, 0ms wifi, 0ms not connected)
// Monkey finished
```

質疑応答

ご清聴ありがとうございました。