Piotr Slesarew

piotr.slesarew@gmail.com

on GitHub as Sliski

in

ROBUST UNIT TESTING IN ANDROID

TIPS & TRICKS

UNIT TESTING IS LIKE INSURANCE POLICIES...

A LOT OF EFFORT FOR SOMETHING YOU'LL NEVER USE.

UNIT TESTING IS LIKE INSURANCE POLICIES...

THEY COVER WHAT YOU INVEST IN ADVANCE.

WHAT POWERS DO WE NEED?

WHAT POWERS DO WE NEED?

```
public class AndroidDeveloperTest {

    Developer mAndroidDeveloper;

    @Before
    public void setUp() throws Exception {
        Skill skill = Skill.Builder
                .withTestingSkill(true)
                .build();
        mAndroidDeveloper = AndroidDeveloper.create(skill);
    }

    @Test
    public void androidDev_ShouldUnitTesting() throws Exception {
        assertTrue(
                "What? Android developer is not testing? Eghh...",
                mAndroidDeveloper.isUnitTesting()
        );
    }
}
```

WHAT POWERS DO WE NEED?

```
public class
    Developer
        Skill skill = mock(Skill.class);
        when(skill.isTestingSkill())
            .thenReturn(true);
            )
    }
```

WHAT POWERS DO WE NEED?

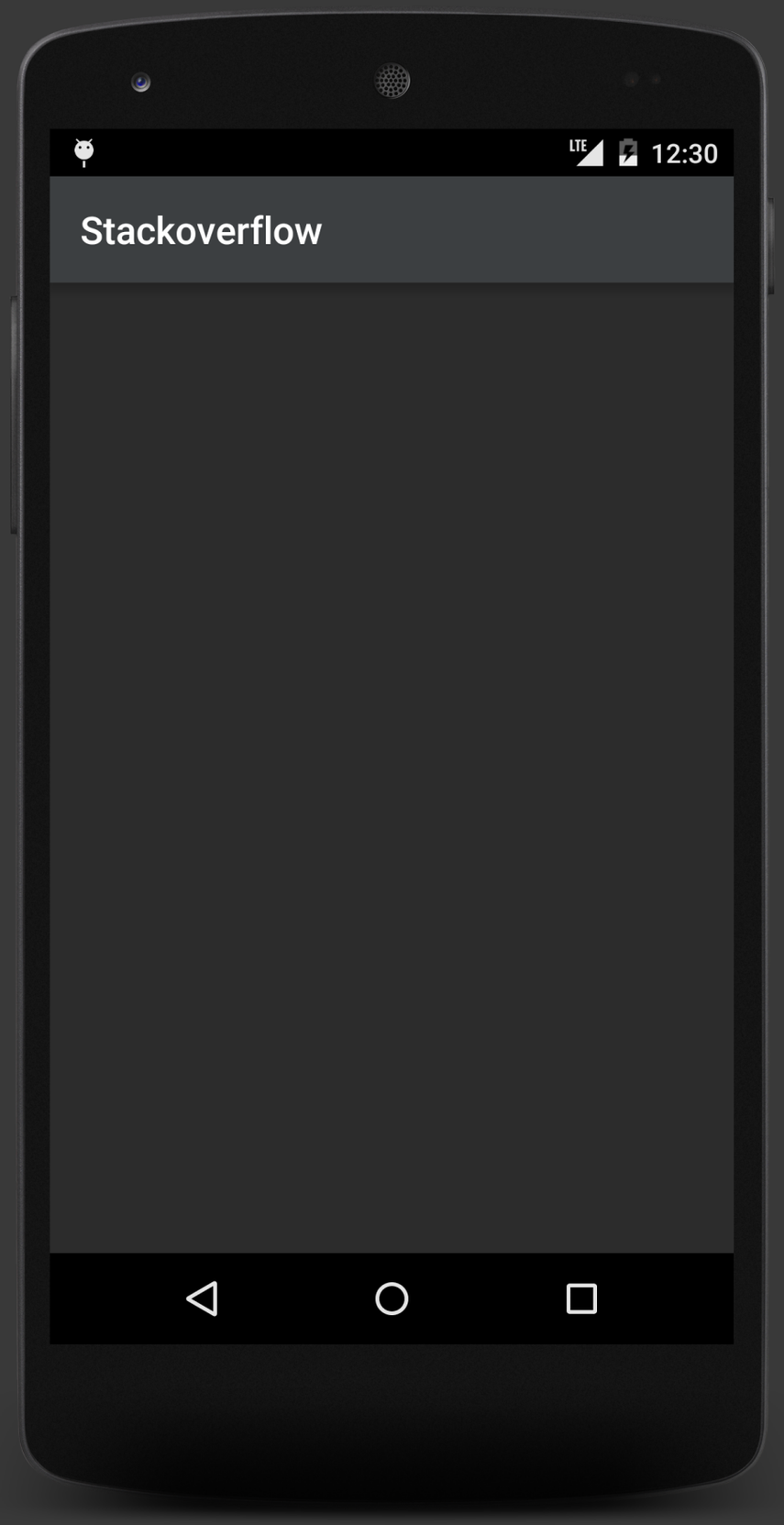
```
@RunWith(RobolectricTestRunner.class)
public class

    Developer

        Skill skill =

                .thenReturn(
        Application context = Robolectric.application;
        mAndroidDeveloper = AndroidDeveloper.create(context, skill);

        )

}
```

WHAT POWERS DO WE NEED?

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {
    void inject(Skill skill);
}

@Module
public class AppModule {
    @Provides
    public Skill provideSkill() {
        return Skill.Builder
                .withTestingSkill(true)
                .build();
    }
}
```

ANDROID GRADLE PLUGIN BRINGS SUPPORT FOR JUNIT!

REMEMBER! HEROES DO NOT TEST BOILERPLATE.


LTE
12:30
Stackoverflow

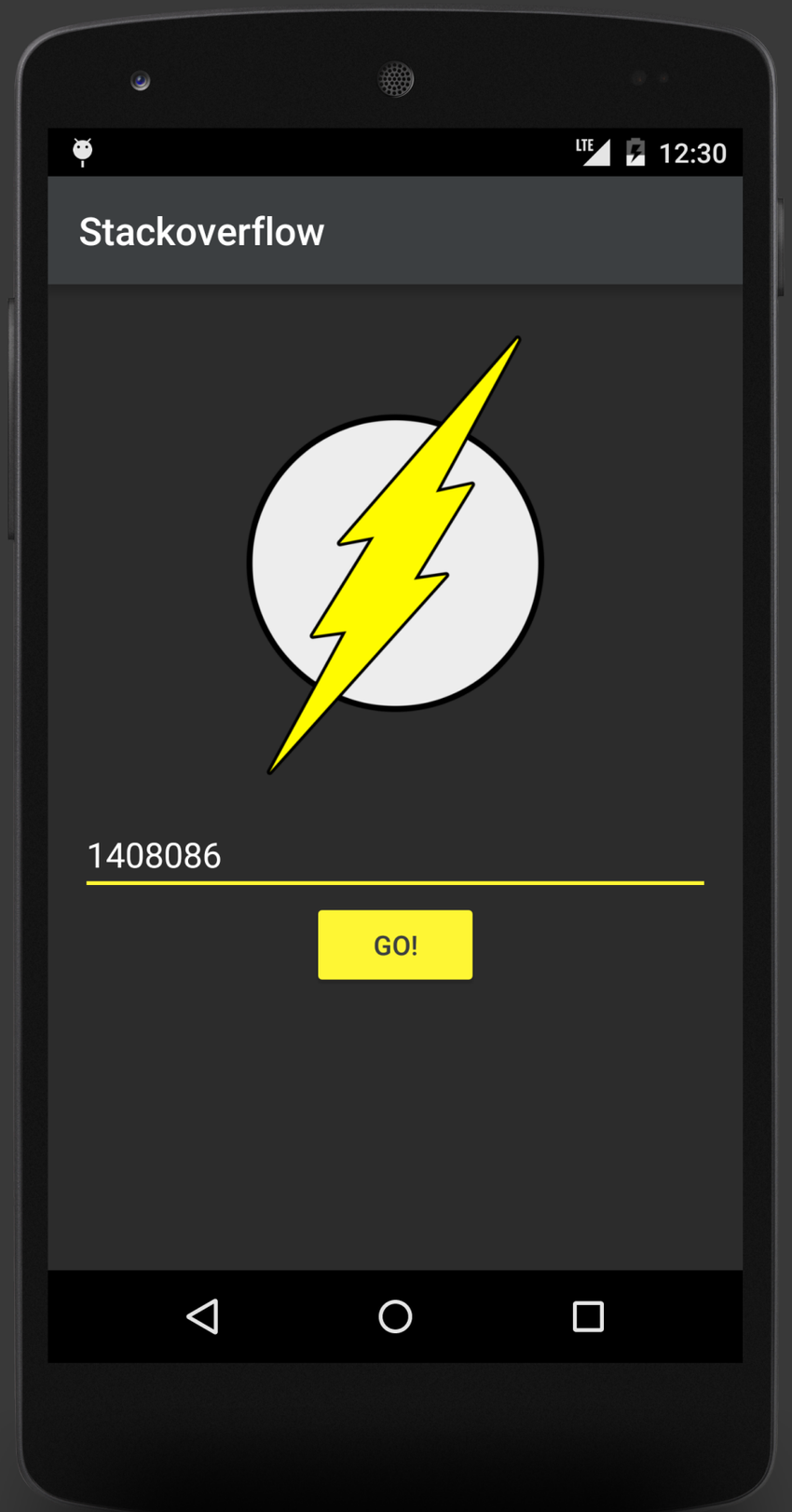
```
public class MainActivity : ActionBarActivity(), MainActivityView {

    public var presenter: MainActivityPresenter? = null
        [Inject] set

    override fun onCreate(savedInstanceState: Bundle?) {
        super<ActionBarActivity>.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        App.mGraph.inject(this)
        presenter?.setView(this)
        presenter?.addFragment(savedInstanceState)
    }

    override fun addLoginFragment() {
        addFragment(LoginFragment())
    }

    override fun addUserInfoFragment() {
        addFragment(UserInfoFragment())
    }

    private fun addFragment(fragment: Fragment) {
        addFragment(R.id.container, fragment)
    }
}
```

```
public class MainActivity : ActionBarActivity(), MainActivityView {

    public var presenter: MainActivityPresenter? = null
        [Inject] set

    override fun

        setContentView(R.layout.
        App.

        presenter?.addFragment(savedInstanceState)
    }

    override fun addLoginFragment() {
        addFragment(LoginFragment())
    }

    override fun addUserInfoFragment() {
        addFragment(UserInfoFragment())
    }

    private fun addFragment(fragment: Fragment) {
        addFragment(R.id.container, fragment)
    }
}
```

```
public trait MainActivityPresenter {
    public fun addFragment(savedInstanceState: Bundle?)
    public fun setView(view: MainActivityView)
    public fun getView() : MainActivityView?
}

public class MainActivityPresenterImpl : MainActivityPresenter {

    private var mainActivityView: MainActivityView? = null
    var preferencesManager: PreferencesManager? = null

    override fun addFragment(savedInstanceState: Bundle?) {
        if (savedInstanceState == null) {
            val userId = preferencesManager?.userId()?.getOr(0L)
            if (userId == 0L) {
                mainActivityView?.addLoginFragment()
            } else {
                mainActivityView?.addUserInfoFragment()
            }
        }
    }

    override fun setView(view: MainActivityView) {
        mainActivityView = view
    }

    override fun getView(): MainActivityView? {
        return mainActivityView
    }
}
```

```java
public class MainActivityTest {

    Fragment fragment;

    @Before
    public void setUp() throws Exception {
        MainActivity mainActivity = Robolectric.buildActivity(MainActivity.class)
                .create()
                .start()
                .visible()
                .get();
        fragment = mainActivity
                .getSupportFragmentManager()
                .findFragmentById(R.id.container);
    }

    @Test
    @Config(application = App.class)
    public void addFragment_ShouldAddUserIdFragment() throws Exception {
        assertTrue(fragment instanceof LoginFragment);
    }

    @Test
    @Config(application = TestApp.class)
    public void addFragment_ShouldAddUserInfoFragment() throws Exception {
        assertTrue(fragment instanceof UserInfoFragment);
    }
}
```

```java
public class MainActivityPresenterImplTest {

    private MainActivityPresenterImpl presenter;

    @Before
    public void setUp() throws Exception {
        presenter = new MainActivityPresenterImpl();
        presenter.setView(mock(MainActivityView.class));
        presenter.setPreferencesManager(getPreferencesManager());
    }

    @Test
    public void addFragment_ShouldNotAddFragment() throws Exception {
        presenter.addFragment(mock(Bundle.class));
        MainActivityView mainActivityView = presenter.getView();
        verify(mainActivityView, never()).addLoginFragment();
        verify(mainActivityView, never()).addUserInfoFragment();
    }

    @Test
    public void addFragment_ShouldAddLoginFragment() throws Exception {
        presenter.getPreferencesManager().userId().put(0L).commit();
        presenter.addFragment(null);
        verify(presenter.getView()).addLoginFragment();
    }

    @Test
    public void addFragment_ShouldAddUserInfoFragment() throws Exception {
        presenter.getPreferencesManager().userId().put(1408086L).commit();
        presenter.addFragment(null);
        verify(presenter.getView()).addUserInfoFragment();
    }
}
```


12:30
Stackoverflow
1408086
GO!

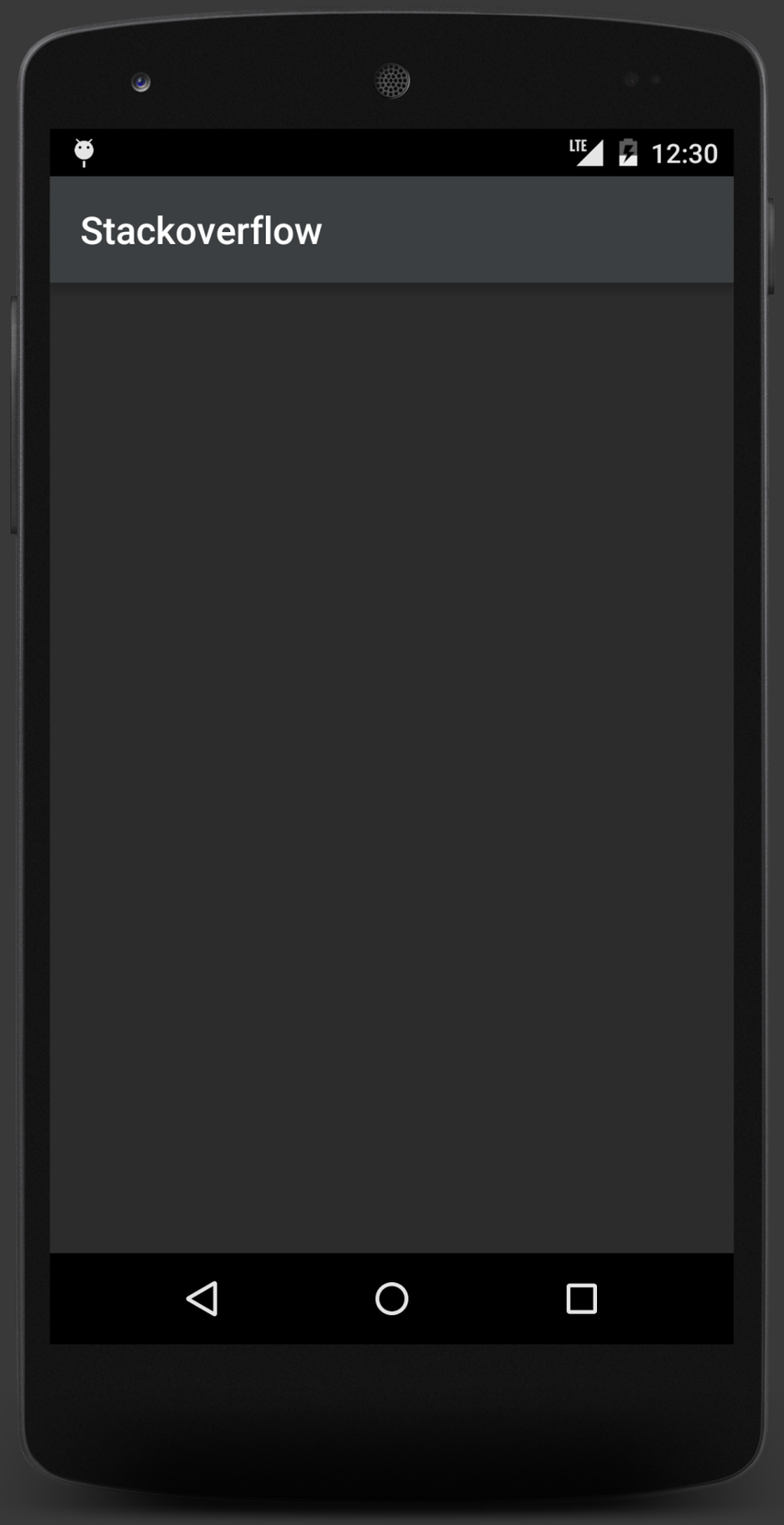
```
public class LoginFragment : Fragment(), LoginFragmentView, View.OnClickListener {

    public var presenter: LoginFragmentPresenter? = null
        [Inject] set

    override fun onCreate(savedInstanceState: Bundle?) {
        super<Fragment>.onCreate(savedInstanceState)
        App.mGraph.inject(this)
        presenter?.setView(this)
    }

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View {
        return inflater.inflate(R.layout.user_id_fragment, container, false)
    }

    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super<Fragment>.onActivityCreated(savedInstanceState)
        findView<View>(R.id.go_button).setOnClickListener(this)
    }

    override fun showBadFormatInfo() {
        showToastShort(R.string.bad_format_info)
    }

    override fun replaceWithUserInfoFragment() {
        replaceWith(UserInfoFragment())
    }

    override fun onClick(v: View) {
        val userId = findView<EditText>(R.id.user_id_editText).getText().toString()
        presenter?.login(userId)
    }
}
```

```
public class LoginFragment : Fragment(), LoginFragmentView, View.OnClickListener {

    public var presenter: LoginFragmentPresenter? = null
        [Inject] set

    override fun

        App.

    }

    override fun
savedInstanceState: Bundle?): View {

    }

    override fun

    }

    override fun showBadFormatInfo() {
        showToastShort(R.string.bad_format_info)
    }

    override fun replaceWithUserInfoFragment() {
        replaceWith(UserInfoFragment())
    }

    override fun onClick(v: View) {
        val userId = findView<EditText>(R.id.user_id_editText).getText().toString()
        presenter?.login(userId)
    }
}
```

```
public trait LoginFragmentPresenter {
    fun login(userId: String)
    public fun setView(view: LoginFragmentView)
    public fun getView() : LoginFragmentView?
}

public class LoginFragmentPresenterImpl : LoginFragmentPresenter {

    private var loginFragmentView: LoginFragmentView? = null
    var preferencesManager: PreferencesManager? = null

    override public fun login(userId: String) {
        try {
            val userIdToLong = userId.toLong()
            preferencesManager
                    ?.userId()
                    ?.put(userIdToLong)
                    ?.commit()
            loginFragmentView?.replaceWithUserInfoFragment()
        } catch(e : NumberFormatException) {
            loginFragmentView?.showBadFormatInfo()
        }
    }

    override fun setView(view: LoginFragmentView) {
        loginFragmentView = view
    }

    override fun getView(): LoginFragmentView? {
        return loginFragmentView;
    }
}
```

```java
public class LoginFragmentTest {

    private LoginFragmentView userIdFragment;

    @Before
    public void setUp() throws Exception {
        userIdFragment = new LoginFragment();
        FragmentTestUtil.startFragment(((LoginFragment) userIdFragment),
MainActivity.class);
    }

    @Test
    public void login_ShouldReplaceFragmentWithUserInfoFragment() throws Exception {
        userIdFragment.setId("2");
        userIdFragment.onClickGoButton();
        Fragment fragment = ((LoginFragment) userIdFragment)
                .getActivity()
                .getSupportFragmentManager()
                .findFragmentById(R.id.container);
        assertTrue(fragment instanceof UserInfoFragment);
    }
}
```

```java
public class LoginFragmentPresenterImplTest {

    private LoginFragmentPresenterImpl presenter;

    @Before
    public void setUp() throws Exception {
        presenter = new LoginFragmentPresenterImpl();
        presenter.setView(mock(LoginFragmentView.class));
        presenter.setPreferencesManager(getPreferencesManager());
    }

    @Test
    public void login_ShouldShowBadFormatInfo() throws Exception {
        presenter.login("");
        verify(presenter.getView()).showBadFormatInfo();
    }

    @Test
    public void login_ShouldReplaceFragmentAndSaveIdToPrefs() throws Exception {
        presenter.login("1408086");
        verify(presenter.getView()).replaceWithUserInfoFragment();

        assertEquals(
                Long.valueOf(1408086L),
                presenter.getPreferencesManager().userId().getOr(0l)
        );
    }
}
```


LTE
12:30
Stackoverflow

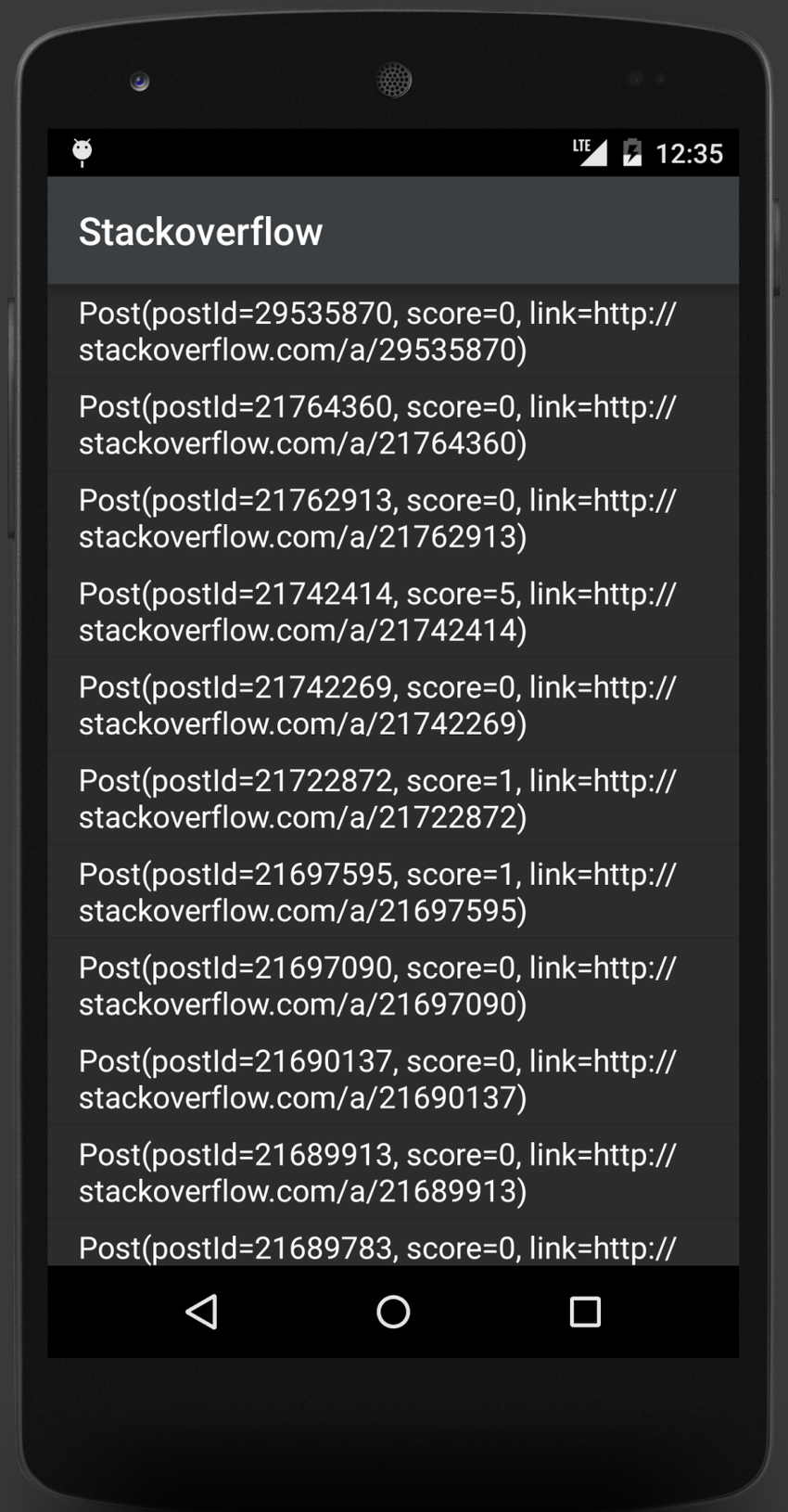
```
public class UserInfoFragment : Fragment(), UserInfoFragmentView {

    public var presenter: UserInfoFragmentPresenter? = null
        [Inject] set

    override fun onCreate(savedInstanceState: Bundle?) {
        super<Fragment>.onCreate(savedInstanceState)
        App.mGraph.inject(this)
        presenter?.setView(this)
    }

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View {
        return inflater.inflate(R.layout.user_info_fragment, container, false)
    }

    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super<Fragment>.onActivityCreated(savedInstanceState)
        presenter!!.addFragments(savedInstanceState, isTablet())
    }

    override fun addPostListFragment() {
        addChildFragment(R.id.list_container, PostListFragment())
    }

    override fun addPostDetailsFragment() {
        addChildFragment(R.id.preview_container, PostDetailsFragment())
    }
}
```

```
public class UserInfoFragment : Fragment(), UserInfoFragmentView {

    public var presenter: UserInfoFragmentPresenter? = null
        [Inject] set

    override fun

        App.

    }

    override fun
savedInstanceState: Bundle?): View {

    }

    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super<Fragment>.onActivityCreated(savedInstanceState)
        presenter!!.addFragments(savedInstanceState, isTablet())
    }

    override fun addPostListFragment() {
        addChildFragment(R.id.list_container, PostListFragment())
    }

    override fun addPostDetailsFragment() {
        addChildFragment(R.id.preview_container, PostDetailsFragment())
    }
}
```

```
public trait UserInfoFragmentPresenter {
    public fun addFragments(savedInstanceState: Bundle?, isTablet: Boolean)
    public fun setView(view: UserInfoFragmentView)
    public fun getView() : UserInfoFragmentView?
}

public class UserInfoFragmentPresenterImpl : UserInfoFragmentPresenter {

    private var userInfoFragmentView: UserInfoFragmentView? = null

    override fun addFragments(savedInstanceState: Bundle?, isTablet: Boolean) {
        if (savedInstanceState == null) {
            userInfoFragmentView?.addPostListFragment()
            if (isTablet) {
                userInfoFragmentView?.addPostDetailsFragment()
            }
        }
    }

    override fun setView(view: UserInfoFragmentView) {
        userInfoFragmentView = view
    }

    override fun getView(): UserInfoFragmentView? {
        return userInfoFragmentView
    }
}
```

```java
public class UserInfoFragmentTest {

    private int childFragmentCount;

    @Before
    public void setUp() throws Exception {
        UserInfoFragment userInfoFragment = new UserInfoFragment();
        FragmentTestUtil.startFragment(userInfoFragment);
        childFragmentCount = userInfoFragment
                .getChildFragmentManager()
                .getFragments()
                .size();
    }

    @Test
    public void fragment_ShouldContainsOneChild() throws Exception {
        assertEquals(1, childFragmentCount);
    }

    @Test
    @Config(qualifiers = "layout-large")
    public void fragment_ShouldContainsTwoChildren() throws Exception {
        assertEquals(2, childFragmentCount);
    }
}
```

```java
public class UserInfoFragmentPresenterImplTest {

    private UserInfoFragmentPresenterImpl presenter;

    @Before
    public void setUp() throws Exception {
        presenter = new UserInfoFragmentPresenterImpl();
        presenter.setView(mock(UserInfoFragmentView.class));
    }

    @Test
    public void addFragments_ShouldNotAddFragments() throws Exception {
        presenter.addFragments(mock(Bundle.class), true);
        verify(presenter.getView(), never()).addPostListFragment();
        verify(presenter.getView(), never()).addPostDetailsFragment();
    }

    @Test
    public void addFragments_ShouldAddPostDetailsFragment() throws Exception {
        presenter.addFragments(null, true);
        verify(presenter.getView()).addPostDetailsFragment();
    }

    @Test
    public void addFragments_ShouldNotAddPostDetailsFragment() throws Exception {
        presenter.addFragments(null, false);
        verify(presenter.getView(), never()).addPostDetailsFragment();
    }
}
```


12:35
Stackoverflow
Post(postId=29535870, score=0, link=http://stackoverflow.com/a/29535870)
Post(postId=21764360, score=0, link=http://stackoverflow.com/a/21764360)
Post(postId=21762913, score=0, link=http://stackoverflow.com/a/21762913)
Post(postId=21742414, score=5, link=http://stackoverflow.com/a/21742414)
Post(postId=21742269, score=0, link=http://stackoverflow.com/a/21742269)
Post(postId=21722872, score=1, link=http://stackoverflow.com/a/21722872)
Post(postId=21697595, score=1, link=http://stackoverflow.com/a/21697595)
Post(postId=21697090, score=0, link=http://stackoverflow.com/a/21697090)
Post(postId=21690137, score=0, link=http://stackoverflow.com/a/21690137)
Post(postId=21689913, score=0, link=http://stackoverflow.com/a/21689913)
Post(postId=21689783, score=0, link=http://

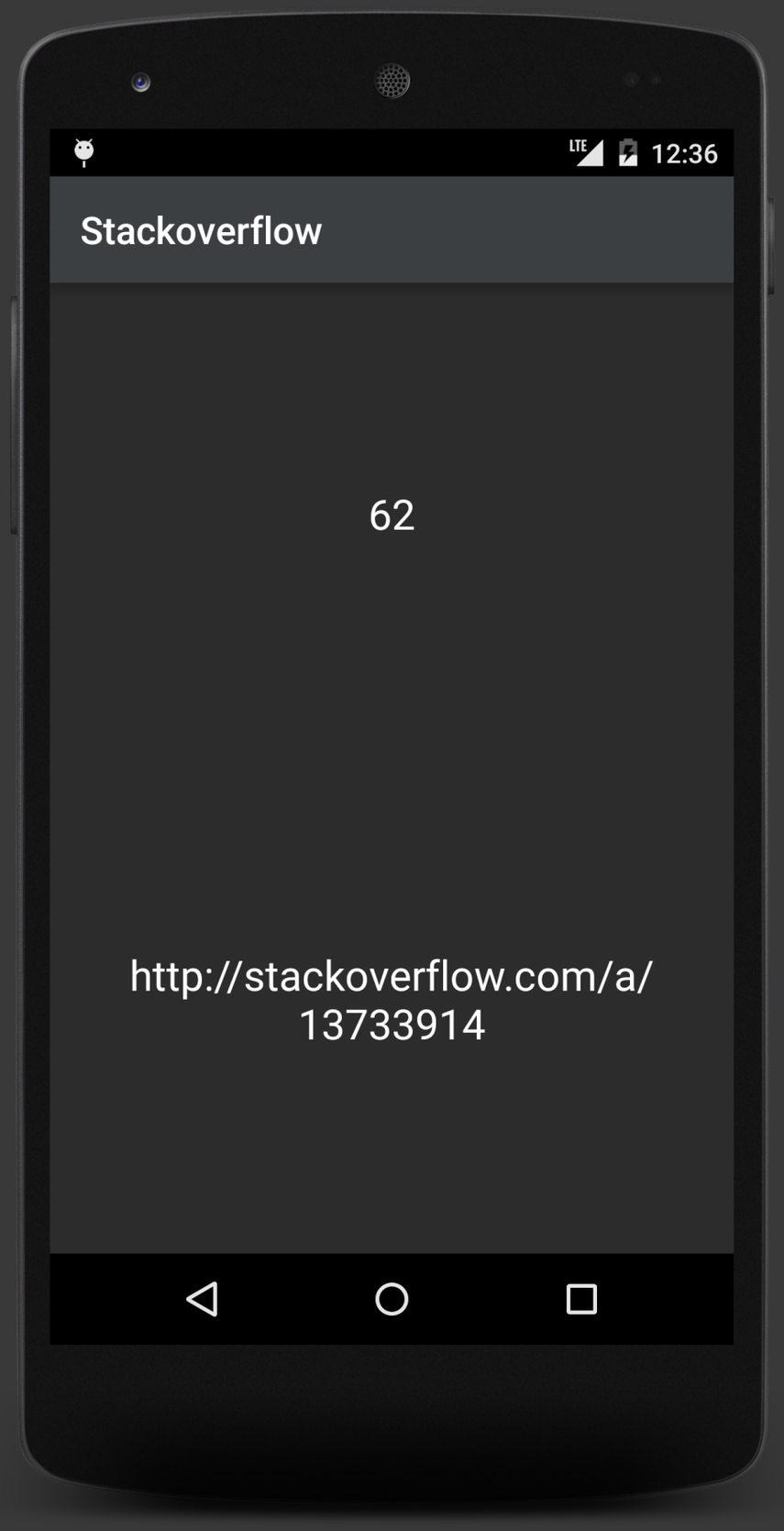
```
public class PostListFragment : ListFragment(), PostListFragmentView,
        AdapterView.OnItemClickListener {

    public var presenter: PostListFragmentPresenter? = null
        [Inject] set

    override public fun onCreate(savedInstanceState: Bundle?) {
        super<ListFragment>.onCreate(savedInstanceState)
        App.mGraph.inject(this)
        presenter?.setView(this)
    }

    override public fun onActivityCreated(savedInstanceState: Bundle?) {
        super<ListFragment>.onActivityCreated(savedInstanceState)
        getListView().setOnItemClickListener(this)
        presenter?.getPosts()
    }

    override fun onPause() {
        super<ListFragment>.onPause()
        presenter?.setView(null)
    }

    override fun onItemClick(pv: AdapterView<*>, v: View, p: Int, id: Long) {
        presenter?.onItemClick(p, isTablet())
    }

    override fun setAdapter(posts: ArrayList<Post>?) {
        val layoutId = android.R.layout.simple_list_item_1
        setListAdapter(ArrayAdapter(getActivity(), layoutId, posts))
    }

    override fun addToBackStack(post: Post) {
        addToBackStack(R.id.list_container, PostDetailsFragment.getInstance(post))
    }
}
```

```
public class PostListFragment : ListFragment(), PostListFragmentView,
        AdapterView.OnItemClickListener {

    public var presenter: PostListFragmentPresenter? = null
        [Inject] set

    override public fun

        App.

    }

    override public fun

        getListView().setOnItemClickListener(
        presenter?.getPosts()
    }

    override fun


    }

    override fun onItemClick(pv: AdapterView<*>, v: View, p: Int, id: Long) {
        presenter?.onItemClick(p, isTablet())
    }

    override fun setAdapter(posts: ArrayList<Post>?) {
        val layoutId = android.R.layout.simple_list_item_1
        setListAdapter(ArrayAdapter(getActivity(), layoutId, posts))
    }

    override fun addToBackStack(post: Post) {
        addToBackStack(R.id.list_container, PostDetailsFragment.getInstance(post))
    }
}
```

```
public trait PostListFragmentPresenter {
    public fun getPosts()
    public fun onItemClick(position: Int, isTablet: Boolean)
    public fun setView(view: PostListFragmentView?)
    public fun getView() : PostListFragmentView?
}

public class PostListFragmentPresenterImpl() : PostListFragmentPresenter {

    private var postListFragmentView : PostListFragmentView? = null
    var client: Client? = null
    var preferencesManager: PreferencesManager? = null
    var postList: ArrayList<Post>? = null
    var eventBus: EventBus? = null

    override fun getPosts() {
        if (postList != null) {
            postListFragmentView?.setAdapter(postList)
        } else {
            eventBus?.register(this)
            val userId = preferencesManager?.userId()?.getOr(0L);
            client?.getPosts(userId as Long);
        }
    }

    override fun onItemClick(position: Int, isTablet: Boolean) {
        val post = postList?.get(position) as Post
        if (isTablet) {
            eventBus?.post(OnPostClickedEvent(post))
        } else {
            postListFragmentView?.addToBackStack(post)
        }
    }

    public fun onEvent(event: GetPostsResponseEvent) {
        postList = event.posts
        postListFragmentView?.setAdapter(ArrayList(event.posts))
    }
}
```

```java
public class PostListFragmentPresenterImplTest {

    private PostListFragmentPresenterImpl presenter;

    @Before
    public void setUp() throws Exception {
        presenter = new PostListFragmentPresenterImpl();
        presenter.setPostListFragmentView(mock(PostListFragmentView.class));
        presenter.setClient(mock(Client.class));
        presenter.setEventBus(mock(EventBus.class));
    }

    @Test
    public void getPosts_ShouldLoadPostsFromCache() throws Exception {
        ArrayList<Post> arrayList = new ArrayList<>();
        presenter.setPostList(arrayList);
        presenter.getPosts();
        verify(presenter.getView()).setAdapter(arrayList);
    }

    @Test
    public void getPosts_ShouldLoadPostFromServer() throws Exception {
        PreferencesManager preferencesManager = getPreferencesManager(1408086L);
        presenter.setPreferencesManager(preferencesManager);
        presenter.setPostList(null);
        presenter.getPosts();
        verify(presenter.getClient()).getPosts(1408086L);
    }
}
```

```java
public class PostListFragmentPresenterImplTest {

    private PostListFragmentPresenterImpl presenter;

    @Before
    public void setUp() throws Exception {
        presenter = new PostListFragmentPresenterImpl();
        presenter.setView(mock(PostListFragmentView.class));
        presenter.setClient(mock(Client.class));
        presenter.setEventBus(mock(EventBus.class));
    }

    @Test
    public void onItemClick_ShouldPostOnPostClickedEvent() throws Exception {
        ArrayList<Post> posts = new ArrayList<>();
        posts.add(new Post(1, 0, ""));
        presenter.setPostList(posts);
        presenter.onItemClick(0, true);
        verify(presenter.getEventBus()).post(Mockito.any(OnPostClickedEvent.class));
    }

    @Test
    public void onItemClick_ShouldAddToBackStack() throws Exception {
        ArrayList<Post> posts = new ArrayList<>();
        Post post = new Post(1, 0, "");
        posts.add(post);
        presenter.setPostList(posts);
        presenter.onItemClick(0, false);
        verify(presenter.getView()).addToBackStack(post);
    }
}
```


12:36
Stackoverflow
62
http://stackoverflow.com/a/
13733914

```
public class PostDetailsFragmentTest {

    @Test
    public void fragment_ShouldDisplayInfoAboutPost() throws Exception {
        PostDetailsFragment postDetailsFragment =
PostDetailsFragment.Factory.getInstance(new Post(1, 100, "www.stack.com/aa"));
        FragmentTestUtil.startFragment(postDetailsFragment);
        TextView score = ((TextView)
postDetailsFragment.getView().findViewById(R.id.score));
        TextView link = ((TextView) postDetailsFragment.getView().findViewById(R.id.link));
        assertEquals(
                "100",
                score.getText().toString()
        );
        assertEquals(
                "www.stack.com/aa",
                link.getText().toString()
        );
    }

    @Test
    public void fragment_ShouldNotDisplayInfoAboutPost() throws Exception {
        PostDetailsFragment postDetailsFragment = new PostDetailsFragment();
        FragmentTestUtil.startFragment(postDetailsFragment);
        TextView score = ((TextView)
postDetailsFragment.getView().findViewById(R.id.score));
        TextView link = ((TextView) postDetailsFragment.getView().findViewById(R.id.link));
        assertEquals(
                "",
                score.getText().toString()
        );
        assertEquals(
                "",
                link.getText().toString()
        );
    }
}
```

Safari File Edit View History Bookmarks Window Help

https://github.com/sliskiCode/Robust-unit-testing-in-Android

This repository Search Explore Gist Blog Help

sliskiCode / **Robust-unit-testing-in-Android**

QUESTIONS