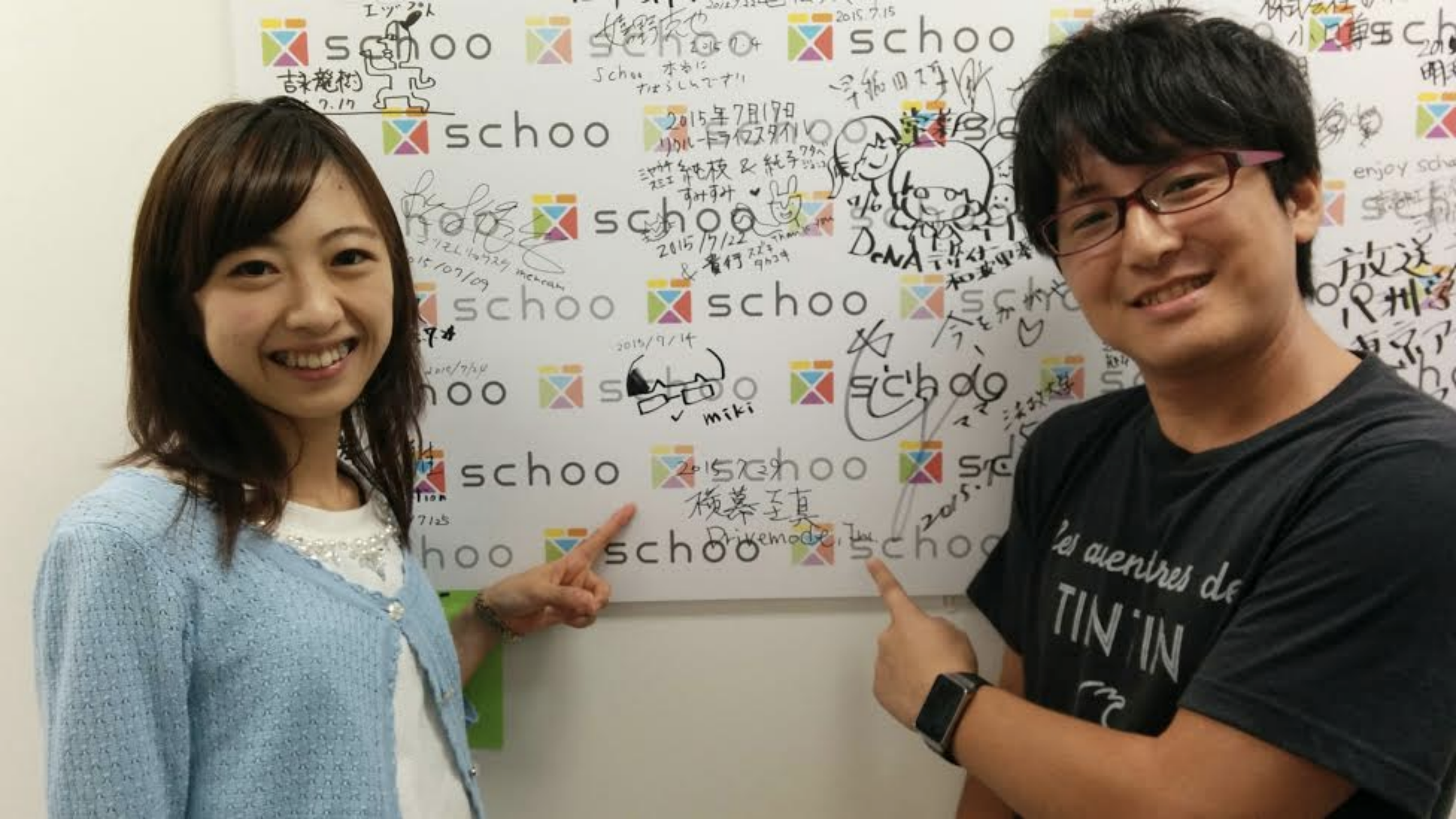
Make it compatible

Keishin Yokomaku (KeithYokoma) @ Drivemode, Inc.

Shibuya Apk #2

Profile

- Keishin Yokomaku at Drivemode, Inc.
- Social: @KeithYokoma
- Publication: Android Training
- Product:
- Like: Bicycle, Photography, Tumblr
- Nickname: Qiita Meister


schoo
miki
Les aventures de
TIN TIN

Compatibility

Compatibility

- between **Android versions**
- between **phone models**

Make it compatible with Android versions

Android versions

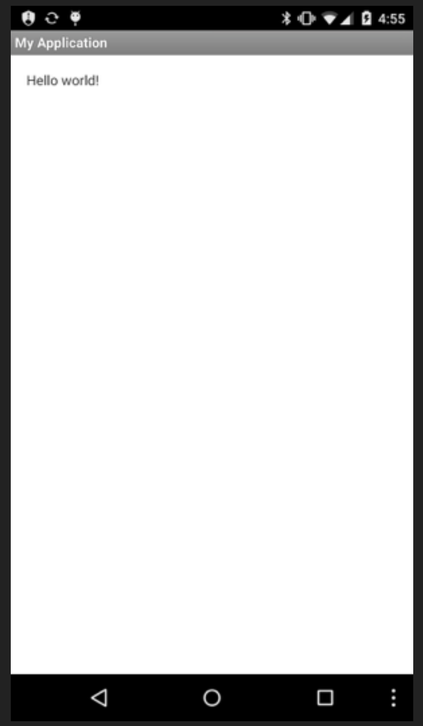
1. SDK versions
2. Compatibility mode
3. Annotations and branching by if-statement
4. Build compatibility classes
5. Branching object with Dagger

**SdkVersion

- **minSdkVersion**
  - Application will support this version at least.
- **targetSdkVersion**
  - Application is fine on this version.
- **maxSdkVersion**
  - Not recommended to specify
  - If specified and system is updated to above the version, the application will automatically uninstalled.

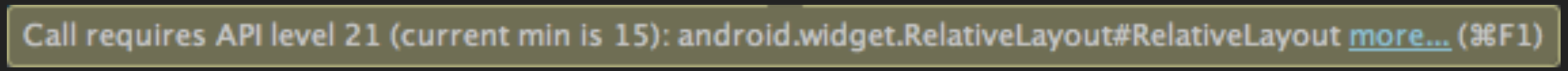
targetSdkVersion

- **Compatibility mode**
  - If targetSdkVersion is set ‘10’
    - Application behaves as API Level 10 even if it is running on 22.



Call requires...

- You likely to see this message...

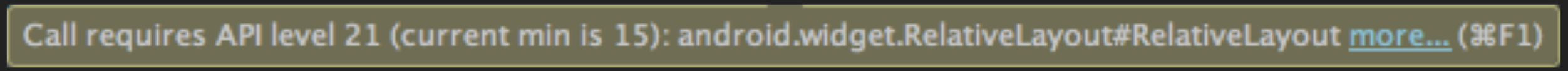


‘if’

```
public void doSomething() {
  if (Build.VERSION.SDK_INT == Build.VERSION_CODES.LOLLIPOP) {
    // for Lollipop
  } else {
    // for Kitkat and below
  }
}
```

Call requires...

- Still you see this warning…



Which is preferred?

- **2 Annotations**
  - @SuppressLint(“NewApi”)
  - @TargetApi(Build.VERSION_CODES.LOLLIPOP)

Which is preferred?

- **@SuppressLint(“NewApi”)**
  - Suppress all warnings of “NewApi”
  - Even if someone call newer API later on, no lint warnings appear

- **@TargetApi(Build.VERSION_CODES.LOLLIPOP)**
  - **Recommended**
  - Suppress warning up to specified API Level
  - If someone call newer API later on, new lint warning appears

Branching, branching, branching...

```
public class Something {
  public void foo() {
    if (Build.VERSION.SDK_INT == Build.VERSION_CODES.LOLLIPOP) {
      // foo on Lollipop
    } else {
      // foo on pre-Lollipop
    }
  }

  public void bar() {
    if (Build.VERSION.SDK_INT == Build.VERSION_CODES.LOLLIPOP) {
      // bar on Lollipop
    } else {
      // bar on pre-Lollipop
    }
  }

  // …
}
```


Looks Not Good To Me

android

open source project

Learn from Android

AOSP style delegate pattern

- **Declare each classes** corresponding to support version
  - Enclose those classes in the service class
  - Delegate all methods to enclosed object
  - Basic architecture on Support Library

AOSP style delegate pattern

```
public class SomethingCompat {
  private final SomethingImpl mImpl; // initialize in ctor

  public void doSomething() {
    mImpl.doSomething();
  }

  private interface SomethingImpl {
    void doSomething();
  }

  private static class SomethingICS implements SomethingImpl {
    @Override
    public void doSomething() {}
  }
  private static class SomethingJB implements SomethingImpl {
    @Override
    public void doSomething() {}
  }
}
```

AOSP style delegate pattern

```
public class SomethingCompat {
  private final SomethingImpl mImpl; // initialize in ctor

  public SomethingCompat() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN) {
      mImpl = new SomethingJB();
    } else if (Build.Version.SDK_INT >=
        Build.VERSION_CODES.ICE_CREAM_SANDWICH) {
      mImpl = new SomethingICS();
    } else {
      mImpl = new SomethingDefault();
    }
  }
}
```

AOSP style delegate pattern

- **Pros**
  - Clean code
  - More testable, more flexible
- **Cons**
  - Hard to test for each enclosed classes
  - Sometimes they don’t put any codes for compatibility

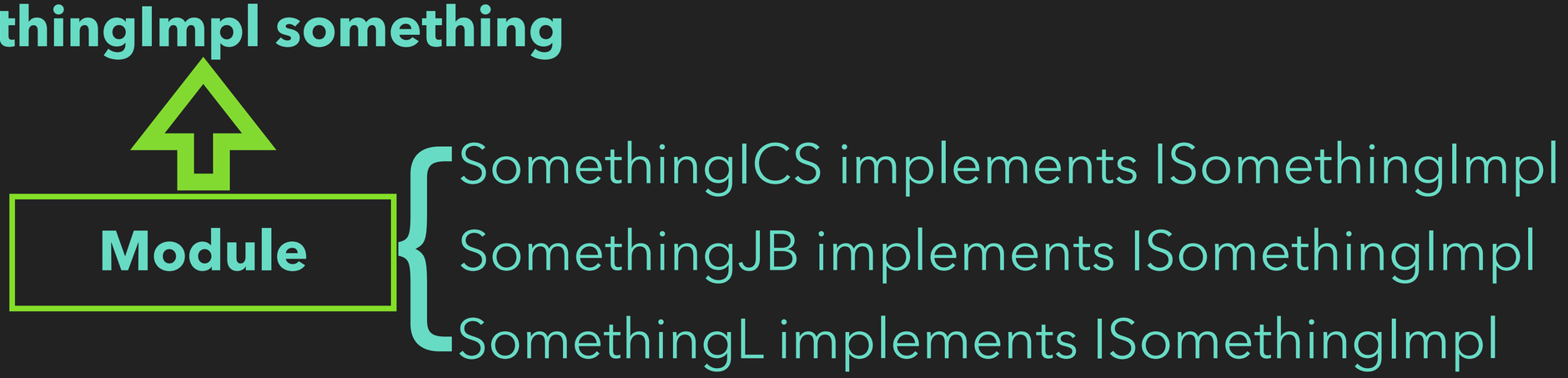
```
public void doSomething() {
  // TODO compatibility implementation
}

public Object returnSomething() {
  return null;
}
```

Service object depends on delegate object

- What if using dependency injection?
  - Inject object in constructor
    - ➡ Configure injected object outside of service object
    - ➡ Easily configure test condition

Android versions

- **Inject different objects** by version with Dagger
  - Declare common interface
  - Implement it enclosing API level specific processes
  - Configure modules which implementation to be injected



Package

- Package structure
  - com.example.model
    - SomethingService (or SomethingProvider, SomethingCompat)
    - impl
      - SomethingImpl
      - SomethingICS
      - SomethingJB
      - SomethingL

Make it compatible with Android models


and新生活
andはじめよう
andじぶんを
andおもいきり
android
みんなちがうから、
世界はたのしい。

アッハイ

Model and manufacturers

- S*msung
  - Galaxy
- L*
  - G
  - Optimus
- S*ny Ericsson
  - Xperia
- F*jitsu
  - Arrows

“Doesn‘t work on my device! It keeps crashing!!“


No way!

How to deal with it

1. No clue to predict problems specific to some device
2. Detect problems as early as possible
3. Use workaround utility and enclose dirty works in it

Test! test! test!

- **Remote Testing Services**
  - PerfectoMobile
  - Remote TestKit
  - Samsung Test Lab
  - Cloud Test Lab
  - Smartphone Test Farm

Make it quicker to notice errors

- **Crash reports**
  - Crashlytics
  - BugSense
  - ACRA(Application Crash Reports for Android)

- Integrate reports to
  - Email
  - Chat(Slack, HipChat, ChatWork…)

Encapsulate compatibility code

- Good to go with DI
  - like version compatibility architecture

- AndroidDeviceCompatibility
  - https://github.com/mixi-inc/Android-Device-Compatibility
  - History from API 7

Still it doesn’t work?

Problems

- Framework has bugs in
  - **Java layer**
    - Various type of bugs and result will be…
      - ➡ Some of ‘RuntimeException’ or some of ‘Error’ causes crash
  - **native code layer**
    - Bad lib** implementation
    - Bad memory management
    - Result
      - ➡ Process will be killed and no crash dialog

Problems

- Some of ‘RuntimeException’
  - Unexpectedly access `null` (NullPointerException)
  - Something is wrong (RuntimeException, IllegalStateException…)

- Some of ‘Error’
  - Class path conflict (VerifyError)
  - Unknown method definition on interface (AbstractMethodError)

“Doesn‘t work for me. No crash though…”

(Photo by SAKURAKO - Do not get mad !/ MJ/TR (´・ω・))

Big unknowns a lot, a lot, a lot…

- (ノ °□°) ノ ‿┻━┻

- Collect logs from area that seems to have some bug
  - Collect everything you need to debug…
  - View properties, object state, etc…
- stackoverflow.com

Be quick, be hacker

- No solid solution. Just you need to be quick to get info.
- If you get the device causing issue, let’s **hack it**!

Make it compatible

Keishin Yokomaku (KeithYokoma) @ Drivemode, Inc.

Shibuya Apk #2