Spring 2019

Lecture 11: Mobile Application and Product Development

NYC Tech-in-Residence Corps
rdomanski@sbs.nyc.gov

Eric Spector
CUNY John Jay College

Bhargava Chinthirla
CUNY John Jay College

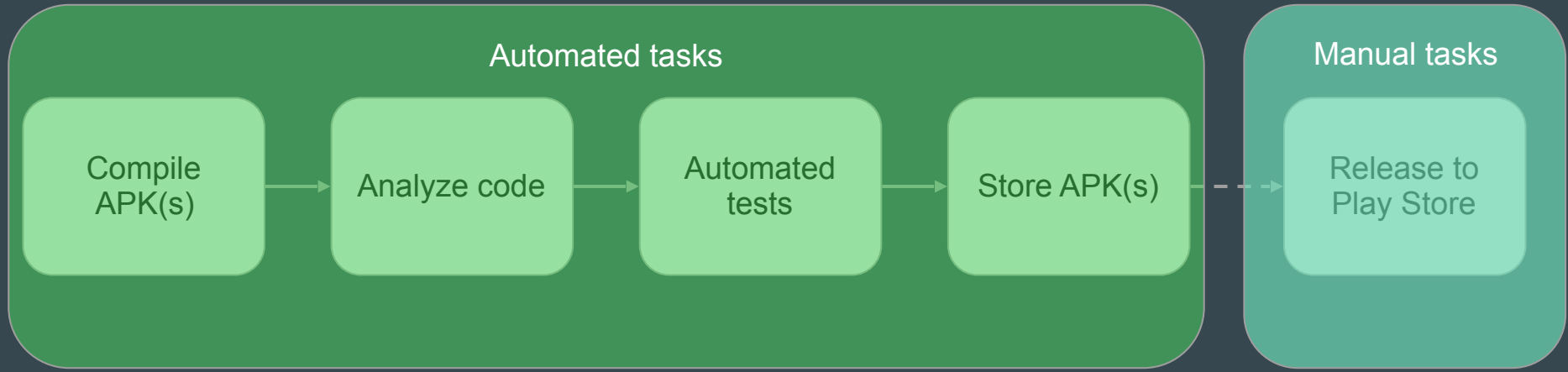
CSCI 380-04

Mobile Application and Product Development

04/10/2019

Recap

Continuous integration pipeline



Analyze code - lint

- Helps to find poorly structured code that can impact the reliability and maintainability of your app
- To run lint: `./gradlew lint`

Unused resources

../../src/main/res/values/strings.xml:3: The resource `R.string.spotify_api_key` appears to be unused

```
1 <resources>
2     <string name="app_name">Spotify Browser</string>
3     <string name="spotify_api_key">BQBC6N-epPUCTixWr7ecHNLVkfmtfxTD8W32GvQFjoR-4OwwUzSXBlEzsYXd2Q4CF
4     <string name="categoryicon">categoryIcon</string>
5     <string name="view_random_playlist">View Random Playlist</string>
6 </resources>
```

UnusedResources | Performance | Warning | Priority 3/10

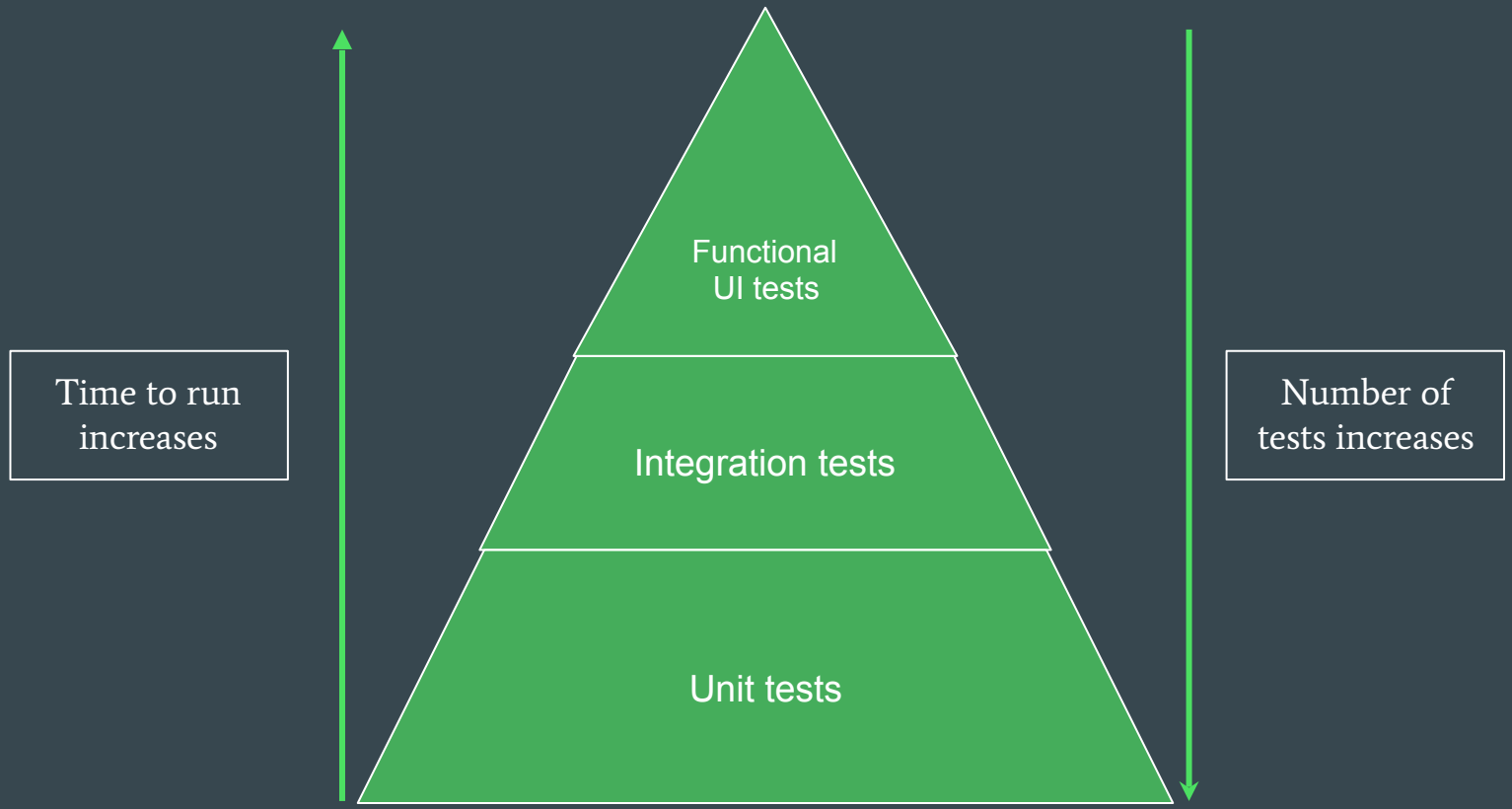
Store APK(s)

- If your pipeline was able to successfully compile APK(s), analyze code, and run automated tests, then it's safe to say that we can store our APK(s) as release candidates
- We don't want to store our APK(s) on some random hard drive, there are existing tools that are meant for this specific purpose:
  - Artifactory
  - Archiva
  - Nexus
  - etc.

Release to Play Store

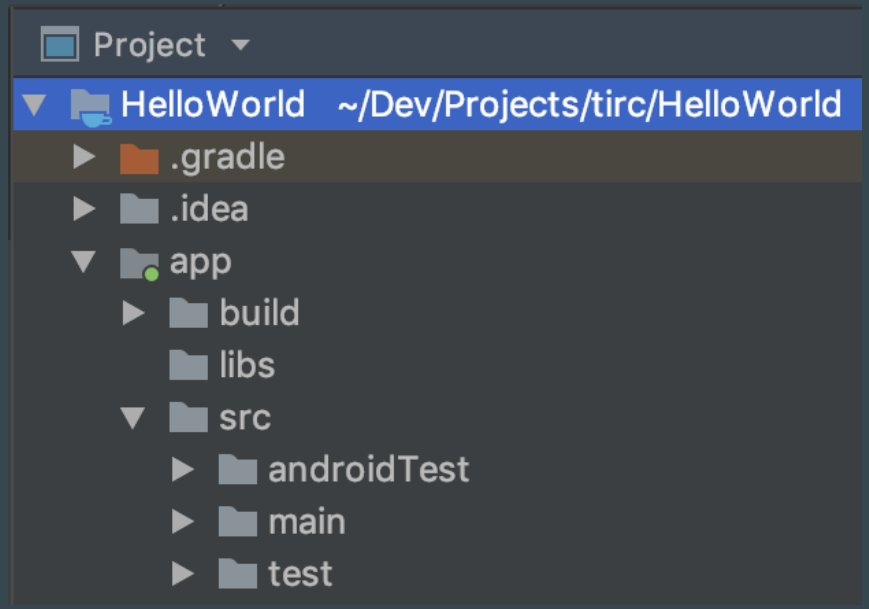
- The automated tasks are automatically triggered every time there is a change to the `master` branch on GitHub
- However, we don't want to release to the Play Store every time there is a change to the `master` branch
- Even though releasing to the Play Store is listed as a manual task, we want to make this task as automated as possible

Automated tests - testing pyramid



Test locations

- Each test suite has its own location in the project repo, outside of the main application code:



Unit testing

- JUnit - unit testing framework for Java
- Goal is to instantiate the class under test, and call any accessible (public, protected, package-private) method to test its behavior
- Methods usually have inputs/outputs, but not strictly necessary
- Dependencies can be mocked out, using `mockito`:

```
testImplementation 'junit:junit:4.12'
testImplementation 'org.mockito:mockito-all:1.10.19'
```

Unit testing - your turn

```
class Calculator {

    static int add(int a, int b) {

    }

    static float divide(float dividend, float divisor) {

    }
}
```

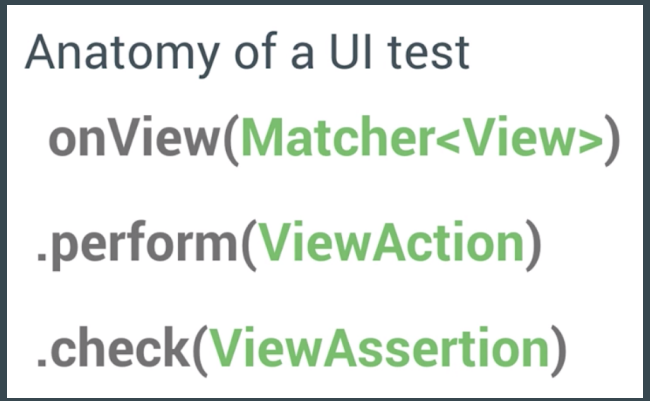
```
public class CalculatorTest {

    @Test
    public void testAdd() {
        assertEquals(Calculator.add(1, 2), 3);
        assertEquals(Calculator.add(4, 0), 4);
    }

    @Test
    public void testDivide() {
        assertEquals(Calculator.divide(15, 3), 5, 0);
        assertEquals(Calculator.divide(16, 4), 4, 0);
        assertEquals(Calculator.divide(15, 4), 3, 0.75);
        assertEquals(Calculator.divide(15, 0), 0, 0);
    }
}
```

Functional UI tests

- Espresso - UI testing framework for Android apps
- Goal is to automatically launch a specific activity in your app and interact with different views by either performing input or checking output
- Useful tutorial: https://www.youtube.com/watch?v=kL3MCQV2M2s



Midterm review

- All questions from the quiz are up for grabs
- Make sure you don't memorize the answers, but rather understand the concepts
- Let's review some data layer concepts:
  - What is the difference between SharedPreferences' commit() method and apply() method?
  - What four HTTP methods correspond to CREATE (POST), READ (GET), UPDATE (PUT), and DELETE (DELETE)?
  - Which status code range corresponds to informational status code types?
  - True or False: HTTP does not specify any limit to the number of headers that can be on a request/response.
  - True or False: All HTTP requests require a body.

Requirements for final project

- Must use some sort of API (similar to how you used Spotify's API for assignment 3)
  - Talk to me now or email me if you can't find a suitable API for your project's idea!
- Must use SharedPreferences in some sort of manner to persist user data
- Make sure to follow the Mobile Application Layers that we've been talking about, it'll help with unit testing the domain layer (we'll talk more about testing next week)
- Send me a link to your project's public GitHub repo
- Domain layer logic MUST be unit tested