

Stubs, Mocks, and Architecture

OBJECTIVES

- Writing a stub (temporary code)
- Writing a mock (faking code)
- Architecting to not require stubs or mocks
- Continue practice of Git/GitHub

USING SEG3103_PLAYGROUND

- Create **/lab05** directory
 - Extract **grades.zip** and **twitter.zip**
 - Make sure you can
 - compile code
 - Run tests

Grades

Homework #1

89

Homework #2

92

Homework #3

100

Homework #4

48

Midterm

73

Final

83

Labs #1

100

Labs #2

100

Labs #3

25

Labs #4

100

Labs #5

100

Labs #6

25

Final Grade

Letter Grade

--

Numeric Grade

--

Percent

--

CALCULATE

mix phx.server
http://localhost:4000

Please read the errors!!!

```
Unchecked dependencies for environment dev:  
* telemetry_metrics (Hex package)  
  the dependency is not available, run "mix deps.get"  
* phoenix_live_view (Hex package)  
  the dependency is not available, run "mix deps.get"  
* telemetry_poller (Hex package)  
  the dependency is not available, run "mix deps.get"  
* phoenix_live_reload (Hex package)  
  the dependency is not available, run "mix deps.get"  
* jason (Hex package)  
  the dependency is not available, run "mix deps.get"  
* phoenix_html (Hex package)  
  the dependency is not available, run "mix deps.get"  
* phoenix (Hex package)  
  the dependency is not available, run "mix deps.get"  
* plug_cowboy (Hex package)  
  the dependency is not available, run "mix deps.get"  
** (Mix) Can't continue due to errors on dependencies
```

`mix deps.get`

```
[info] Running GradesWebEndpoint with cowboy 2.5.10 at 0.0.0.0:4000 (http)
[error] Could not start Node.js watcher because script "/Users/aforward/sin/courses/_/seg3
x03_internal/lab05_solution/grades/assets/node_modules/webpack/bin/webpack.js" does not ex
ist. Your Phoenix application is still running, however assets won't be compiled. You may
fix this by running "npm install" inside the "assets" directory.
```

```
cd assets && npm install
```

Grades

Homework #1

Homework #2

Homework #3

Homework #4

Midterm

Final

Labs #1

Labs #2

Labs #3

Labs #4

Labs #5

Labs #6

Final Grade

Letter Grade

--

Numeric Grade

--

Percent

--

CALCULATE

Button doesn't work


```
[error] GenServer #PID<0.2239.0> terminating
** (UndefinedFunctionError) function Grades.Calculator.letter_grade/1 is undefined
(module Grades.Calculator is not available)
    Grades.Calculator.letter_grade(%{final: "", homework: [ "", "", "", "" ], labs:
[ "", "", "", "", "", "" ], midterm: ""})
    (grades 0.1.0) lib/grades_web/live/page_live.ex:23:
GradesWeb.PageLive.handle_event/3
```

Stub Grades.Calculator

- percentage_grade
- letter_grade
- numeric_grade

Grades

Homework #1

89

Homework #2

92

Homework #3

100

Homework #4

48

Midterm

73

Final

83

Labs #1

100

Labs #2

100

Labs #3

25

Labs #4

100

Labs #5

100

Labs #6

25

Final Grade

Letter Grade

--

Numeric Grade

--

Percent

--

CALCULATE

Replace stubbed
module to make it
work for real

Grades

Homework #1

Homework #2

Homework #3

Homework #4

Midterm

Final

Labs #1

Labs #2

Labs #3

Labs #4

Labs #5

Labs #6

Final Grade

Letter Grade

--

Numeric Grade

--

Percent

--

CALCULATE

Note your observations.

Tasty mocking framework for unit tests in Java



build **passing** coverage **85%** license **MIT**

maven-central **v3.11.1**

Project status

Please see the [release notes page](#).

Updates are announced via [Twitter](#)  [Follow @mockitojava](#) and [mailing list](#) .

<https://site.mockito.org>

Unable to get Mockito to work as expected using JUnit5.

As an aside you can play with Mockito yourself and push back sample code if you get it working.

EASYMOCK

Easy mocking. Better testing.

Getting started

Download (v4.3)

<http://easymock.org/>

<https://easymock.org/user-guide.html>

```
twitter (main)$ ./bin/run
```

```
Twitter Text Feed
```

```
Hello to @you
```

```
twitter (main)$ ./bin/run
```

```
Twitter Text Feed
```

```
twitter (main)$ ./bin/run
```

```
Twitter Text Feed
```

```
Hello to @you
```

```
twitter (main)$ ./bin/run
```

```
Twitter Text Feed
```

```
I am tweet that likes to talk about @me
```

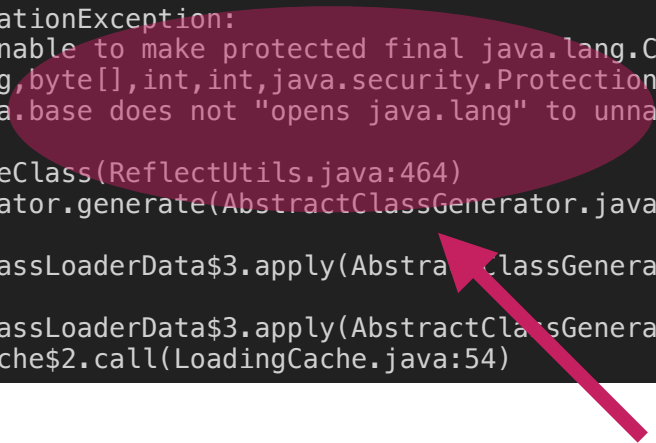
```
├─ JUnit Jupiter ✓  
  └─ TwitterTest ✓  
    ├── mock_full_object() ✓  
    ├── mock_partial_object() ✓  
    └─ actual_call() ✓  
└─ JUnit Vintage ✓
```

./bin/test


```

JUnit Jupiter:TwitterTest:mock_full_object()
  MethodSource [className = 'TwitterTest', methodName = 'mock_full_object', methodParameterTypes = '']
=> java.lang.ExceptionInInitializerError
  org.easymock.internal.ClassProxyFactory.createEnhancer(ClassProxyFactory.java:233)
  org.easymock.internal.ClassProxyFactory.createProxy(ClassProxyFactory.java:165)
  org.easymock.internal.MocksControl.createMock(MockControl.java:107)
  org.easymock.internal.MocksControl.createMock(MockControl.java:85)
  org.easymock.IMocksControl.mock(IMocksControl.java:67)
  [...]
  Caused by: org.easymock.cglib.core.CodeGenerationException:
java.lang.reflect.InaccessibleObjectException-->Unable to make protected final java.lang.Class
java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int,java.security.ProtectionDomain) throws
java.lang.ClassFormatError accessible: module java.base does not "opens java.lang" to unnamed module
@480d3575
  org.easymock.cglib.core.ReflectUtils.defineClass(ReflectUtils.java:464)
  org.easymock.cglib.core.AbstractClassGenerator.generate(AbstractClassGenerator.java:339)
org.easymock.cglib.core.AbstractClassGenerator$ClassLoaderData$3.apply(AbstractClassGenerator.java:96)
org.easymock.cglib.core.AbstractClassGenerator$ClassLoaderData$3.apply(AbstractClassGenerator.java:94)
  org.easymock.cglib.core.internal.LoadingCache$2.call(LoadingCache.java:54)

```



RTFE

--add-opens java.base/java.lang=ALL-UNNAMED

<https://github.com/easymock/easymock/issues/235>

```
| JUnit Jupiter ✓  
|   ↳ TwitterTest ✓  
|       ↳ mock_full_object() ✓  
|       ↳ mock_partial_object() ✓  
|       ↳ actual_call() ✗ Cannot invoke "String.cont  
<local2>" is null  
|   ↳ JUnit Vintage ✓
```

./bin/test



```
@Test
void mock_full_object() {

    Twitter twitter = createMock("twitter", Twitter.class);

    expect(twitter.loadTweet()).andReturn("hello @me");
    expect(twitter.loadTweet()).andReturn("hello @you");
    replay(twitter);

    String actual;

    actual = twitter.loadTweet();
    assertEquals("hello @me", actual);

    actual = twitter.loadTweet();
    assertEquals("hello @you", actual);
}
```

```
@Test
void mock_partial_object() {

    Twitter twitter = partialMockBuilder(Twitter.class)
        .addMockedMethod("loadTweet")
        .createMock();

    expect(twitter.loadTweet()).andReturn("hello @me").times(2);
    replay(twitter);

    boolean actual;

    actual = twitter.isMentionned("me");
    assertEquals(true, actual);

    actual = twitter.isMentionned("you");
    assertEquals(false, actual);
}
```

```
@Test
void isMentionned_lookForAtSymbol() {
    // Assuming a tweet like "hello @me"
    // isMentionned("me") should be true
    // isMentionned("you") should be false
}
```

```
@Test
void isMentionned_dontReturnSubstringMatches() {
    // Assuming a tweet like "hello @meat"
    // isMentionned("me") should be false
    // isMentionned("meat") should be true
}
```

```
@Test
void isMentionned_superStringNotFound() {
    // Assuming a tweet like "hello @me"
    // isMentionned("me") should be true
    // isMentionned("meat") should be false
}
```

```
@Test
void isMentionned_handleNull() {
    // Assuming no tweet is available (i.e. null)
    // isMentionned("me") should be false
    // isMentionned("meat") should be false
}
```

Test isMentionned()

SUBMISSION

- All work should be written under
 - **seg3103_playground/lab04**
- Git commit your work at each step
 - When the application starts (BEFORE YOU MAKE ANY CHANGES)
 - With the stubbed value
 - With the value from assignment #2
- Create **README.md** to summarize your work
- Share your repository with the teacher and TA (s)
 - Submissions to BrightSpace should clearly reference your GitHub repository
- Grades project
 - Your stubbed code
 - Results from putting in your *working* code from assignment #2
 - Observations from the stub
- Twitter
 - Implement the 4 missing test cases using mock objects
 - Show the results of those tests
 - Analyze the results by looking at the code of `isMentionned`
 - If necessary fix the code based on your testing