Unit Testing with JUnit and Mockito

Gaby Reilly
January 2020

Workshop Plan

Set up environment

Review starter Java program

Prepare for unit testing

- End to end testing, Unit tests, and "Testability"
- JUnit
- Dependencies

Write unit tests with Mockito

Comments on JUnit excellence!

https://gabyreilly.github.io/mockito

Step	Emoji when you've got it!
java -version	
IntelliJ installed	
git --version	
Sample code loaded in Intellij	

Sample Java program

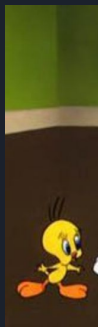Imagine we have written a program for use by a vet office.

The Reminder class takes a pet ID and sends an email to the pet's owners, reminding the owners to make an appointment.

Input: Pet ID

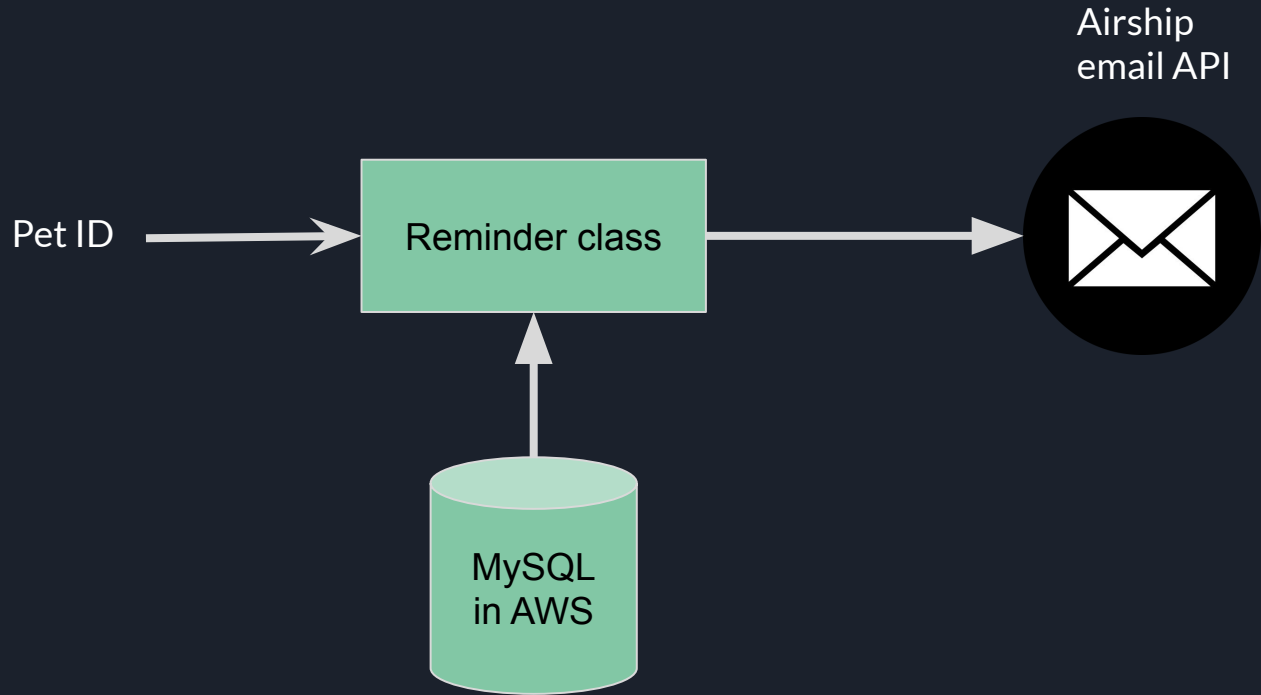Output: Emails

Pet 1:
Spike
Owned by Alice

Pet 2:
Tweety
Owned by Bob and Carlos

Pet 3:
Sylvester
No owner in database

System Diagram

Pet ID →

Reminder class → Airship email API ✉

MySQL in AWS ↑

Finding Dependencies

- Databases
- Files
- HTTP APIs
- Anything over the network

The direction (data in vs data out) doesn't really matter for this process

Our two dependencies are

- The database for storage
- The email service

Friends of this process

- Interfaces
- Variables passed into constructors

Enemies of this process

- Static methods
- Private methods
- Final classes

Refactoring the Reminder class

I'll be modifying the "first" module, refactoring the dependencies of the Reminder class.

You can use the "second" module for reference for where we are going to end up.

Step	Emoji when you've got it!
Database dependency	
Email dependency	

Finally! JUnit and Mockito!

What's left in the Reminder class? This is what we can write tests for.

I'll write tests for the 2-owner scenario.

Then, you can write tests for 1-owner and 0-owner.

Step	Emoji when you've got it!
1 owner	
0 owner	

Unit testing excellence

This is my style (and Airship's style):

- One test class per program class
- Declaring mocks and captors at the class level
- Asserting no more interactions

Regressions

- Let's break it.

Thank you!!