COMP 504: Graduate Object-Oriented Programming and Design

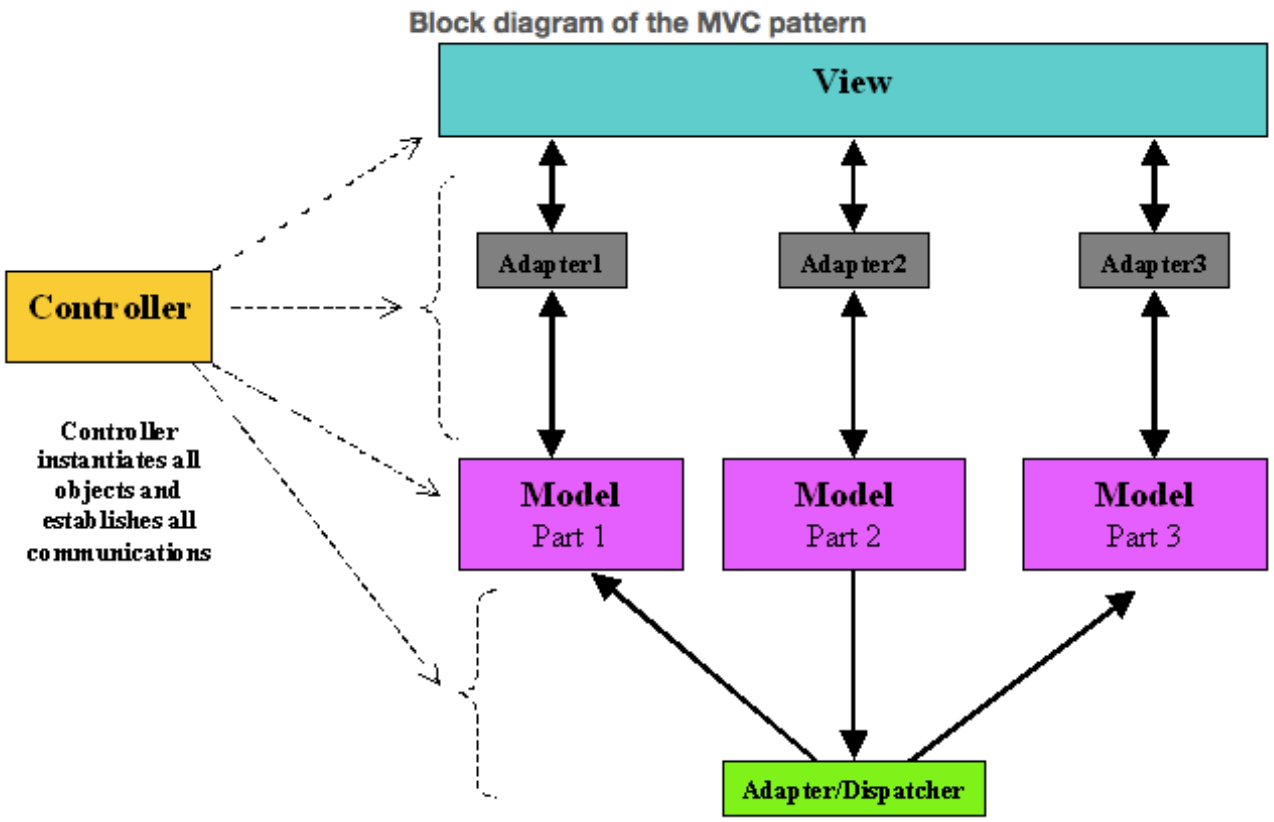
Lecture 5: MVC Design Pattern and Unit Testing

Mack Joyner (mjoyner@rice.edu)

https://www.clear.rice.edu/comp504

MVC Design Pattern

Controller may use an adapter to communicate with model and view



MVC Design Pattern

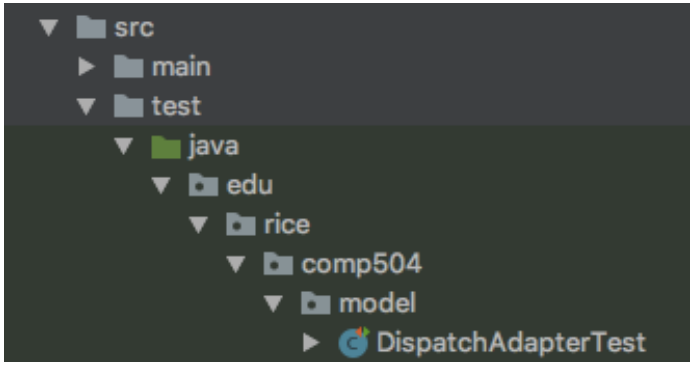
- View operates independent of model
  - Changing how user interacts with view shouldn't affect model
  - View should be tailored to meet user needs
- View interface may not match model interface
- Model should work with many views
- View should work with many models
- View is unaware of model and needs an adapter to connect them

Testing

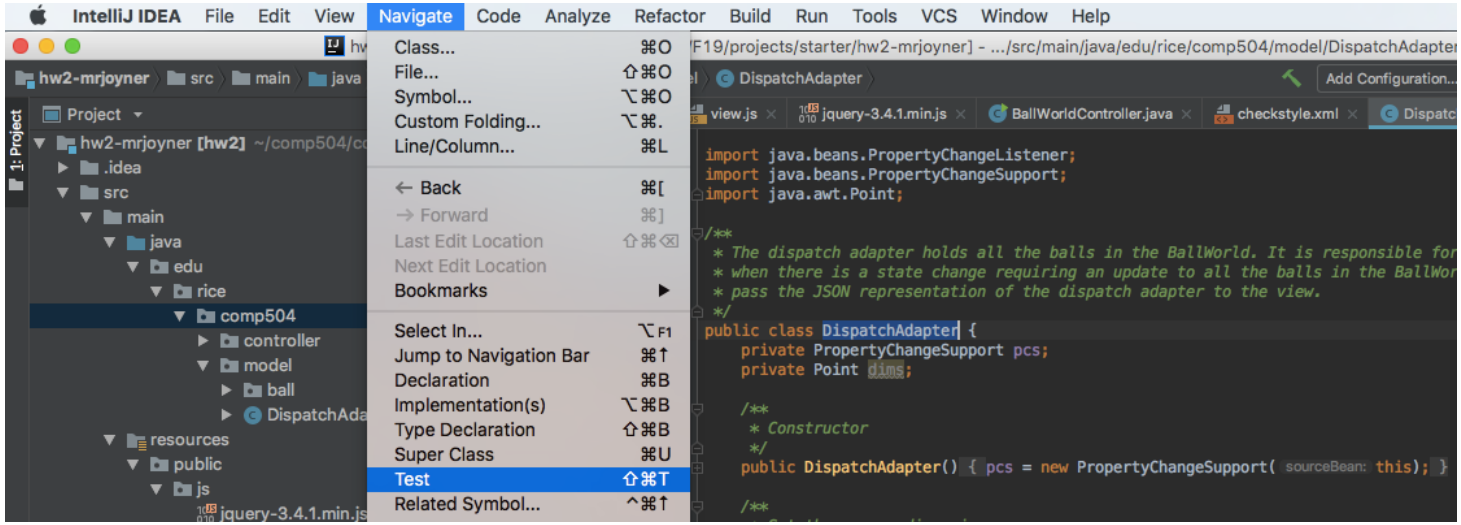
- Testing code is part of software development process
  - Illustrate expected code behavior
  - Not a good feeling to find bugs in never tested code!
- Units tests ensure code changes don't negatively impact existing code
  - Unknown assumptions exist with millions of lines of code in code base
- Better to find and fix bugs before releasing software
  - Cost of fixing bug after production can be 10x
  - Customers are sometimes restricted from using new releases

JUnit

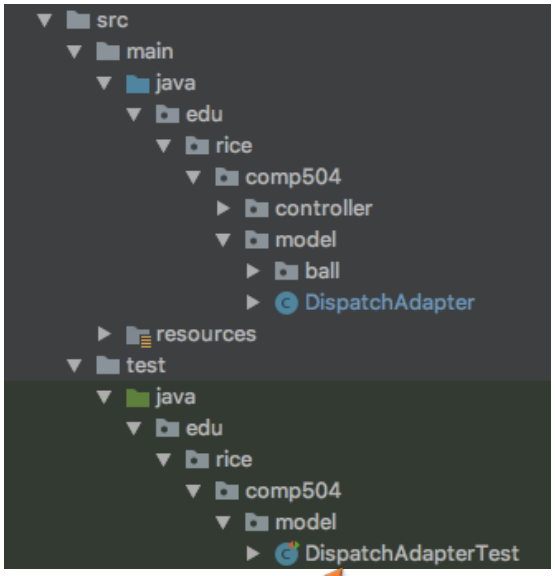
- Unit testing will be done with JUnit (version 4)
- Tests will go in *src/test/java* directory
  - set as test directory in *File -> Project Structure -> Modules*
- Test classes should extend *junit.framework.TestCase* class
- Unit test method names should begin with "test"
- Use asserts to test expected behavior
  - assertEquals
  - assertTrue
  - assertFalse



Creating a JUnit Test



New JUnit Test Created




```
6   public class DispatchAdapterTest extends TestCase {
7
8       public void testLoadBall() {
9           DispatchAdapter da = new DispatchAdapter();
10          ABall stBall = da.loadBall("straight");
11          assertEquals("load straight ball type test", "straight", stBall.getName());
12
13           ABall nBall = da.loadBall("unknown");
14          assertEquals("load unknown ball type test","null", nBall.getName());
15      }
```

preserves package structure

Asserts

```
6   public class DispatchAdapterTest extends TestCase {
7
8       public void testLoadBall() {
9           DispatchAdapter da = new DispatchAdapter();
10          ABall stBall = da.loadBall("straight");
11          assertEquals("load straight ball type test", "straight", stBall.getName());
12
13           ABall nBall = da.loadBall("unknown");
14          assertEquals("load unknown ball type test","null", nBall.getName());
15      }
```

clear test description

JUnit Test Run Configuration

Run/Debug Configurations

JUnit
DispatchAdapterTest
Templates

Name: DispatchAdapterTest
Share
Single instance only

Configuration | Code Coverage | Logs

Test kind: Class
Fork mode: none
Repeat: Once
1

Class: edu.rice.comp504.model.DispatchAdapterTest

VM options: –ea

Program arguments:

Working directory: $MODULE_WORKING_DIR$

Environment variables:

Use classpath of module: hw2

JRE: 11

Shorten command line: user-local default: none - java [options] classname [args]

Before launch: Build, Activate tool window

Build

Show this page
Activate tool window

Cancel | Apply | OK

JUnit Test Failure

Test assertEquals failure shows expected and actual result

Oops…made a mistake with expected value

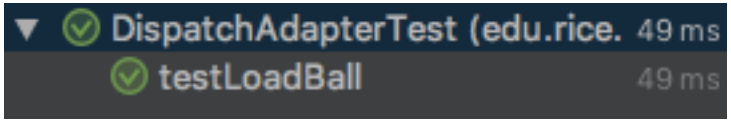
```
▼ DispatchAdapterTest (edu.rice. 49 ms
    testLoadBall 49 ms

junit.framework.ComparisonFailure: load unknown ball type test
Expected :nul
Actual   :null
 <Click to see difference>

<1 internal call>
    at junit.framework.TestCase.assertEquals(TestCase.java:261)
    at edu.rice.comp504.model.DispatchAdapterTest.testLoadBall(DispatchAdapterTest.java:14) <18 internal calls>

Process finished with exit code 255
```

JUnit Test Failures

- Failures can happen for multiple reasons
  - Test doesn't match expected behavior (bad test)
  - Bug in code (unexpected failure)
  - Functionality hasn't been implemented yet (expected failure)
- Every bug found in the code should have a unit test that exposes the bug
  - Should keep track of known issues (wiki, JIRA, etc…)
  - Provide bug status (in progress, fixed in next release, etc…)
  - Bug is fixed when test passes

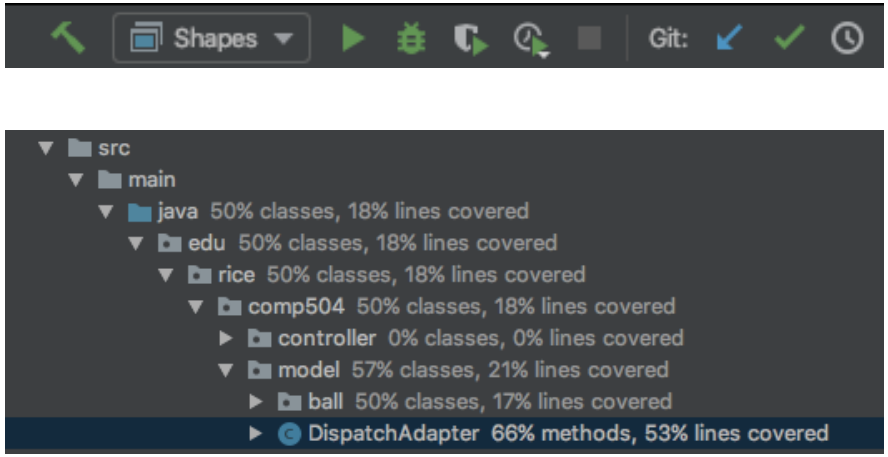
JUnit Test Success

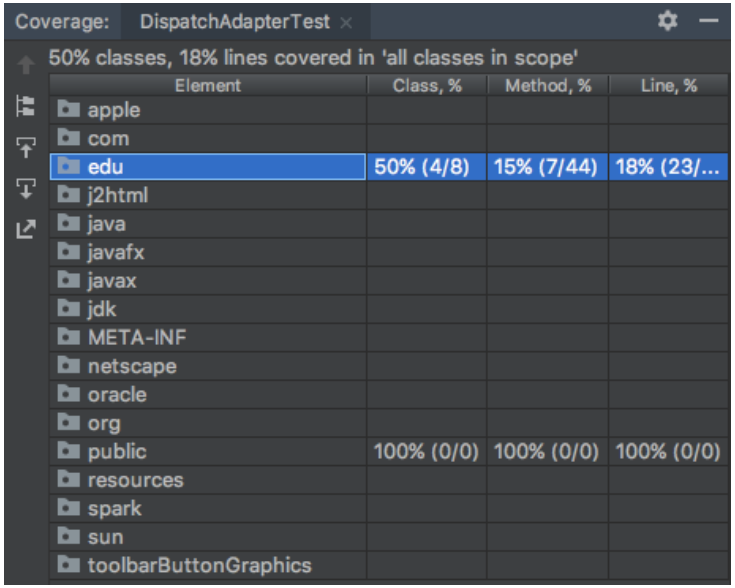
Green check box indicates test passed!



Coverage Testing

Coverage testing shows what percentage of source classes, methods, and lines are exercised by at least one test





Test Coverage

- Ideally, you want 100% test coverage
  - Often not practical (e.g. error handling code)

- Realistically, settle for 80% - 90% test coverage
  - keep adding tests until desired range is achieved

- Customers sometimes ask for code coverage results (COMP 539)

http://localhost:4567

- Currently, hw #1 app is hosted “locally”
  - accessible from the web
  - only from your computer

- Normally, we want it hosted somewhere else
  - others may need to access the app from the web

- A web hosting service provides support we need

PaaS Providers

heroku

Google App Engine

Elastic Beanstalk

Microsoft Azure

DigitalOcean

$5 /mo
$0.007 /hr
512MB Memory
1 Core Processor
20GB SSD Disk
1TB Transfer
SIGN UP

$10 /mo
$0.015 /hr
Most Popular Plan
1GB Memory
1 Core Processor
30GB SSD Disk
2TB Transfer
SIGN UP

HEROKU

Free
Ideal for experimenting with cloud applications in a limited sandbox.
SLEEPS AFTER 30 MINS OF INACTIVITY
MUST SLEEP 6 HOURS IN A 24 HOUR PERIOD
CUSTOM DOMAINS
512 MB RAM | 1 web/1 worker
Free

Hobby
Perfect for small scale personal projects and hobby apps.
ALL FREE FEATURES +
NEVER SLEEPS
MULTIPLE WORKERS FOR MORE POWERFUL APPS
512 MB RAM | 10 Process Types
$7 per dyno/month
prorated to the second

Deploying Spark Java on Heroku (JAR file)

- Heroku needs app jar file with dependencies to deploy app
- Configure Maven to produce jar file
  - create jar file in pom.xml (execution phase: *package*)
  - ensure assembly done during build (execution goal: *single*)
  - predefined descriptor (descriptorRef) to include dependencies
  - tell archiver which main class to use (edu.rice.comp504.controller.SimpleShapesController)

```xml
<plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <executions>
        <execution>
            <phase>package</phase>
            <goals>
                <goal>single</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <descriptorRefs>
            <!-- This tells Maven to include all dependencies -->
            <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
        <archive>
            <manifest>
                <mainClass>Main</mainClass>
            </manifest>
        </archive>
    </configuration>
</plugin>
```

For more information: http://sparkjava.com/tutorials/heroku

Deploying Spark Java on Heroku (Heroku App)

- Create a new Heroku application in your hw directory (unique name)

  >> heroku create [netid]-ex2-shape-world

- Configure Heroku plugin to launch app
  - will use JDK 11
  - replace appName spark-heroku-example with [netid]-ex2-shape-world

- Replace my-app-1.0…jar name with ex2-1.0-SNAPSHOT-jar-with-dependencies.jar

```
<plugin>
    <groupId>com.heroku.sdk</groupId>
    <artifactId>heroku-maven-plugin</artifactId>
    <version>0.4.4</version>
    <configuration>
        <jdkVersion>1.8</jdkVersion>
        <!-- Use your own application name -->
        <appName>spark-heroku-example</appName>
        <processTypes>
            <!-- Tell Heroku how to launch your application -->
            <!-- You might have to remove the ./ in front   -->
            <web>java -jar ./target/my-app-1.0-jar-with-dependencies.jar</web>
        </processTypes>
    </configuration>
</plugin>
```

-

Deploying Spark Java on Heroku (Port)

- Heroku assigns port number every time app is deployed.
- Listen for correct port number

```
import static spark.Spark.*;

public class Main {

    public static void main(String[] args) {
        port(getHerokuAssignedPort());
        get("/hello", (req, res) -> "Hello Heroku World");
    }

    static int getHerokuAssignedPort() {
        ProcessBuilder processBuilder = new ProcessBuilder();
        if (processBuilder.environment().get("PORT") != null) {
            return Integer.parseInt(processBuilder.environment().get("PORT"));
        }
        return 4567; //return default port if heroku-port isn't set (i.e. on l
    }

}
```

For more information: http://sparkjava.com/tutorials/heroku

Deploying Spark Java on Heroku

- Deploy app on heroku
  - cd to project directory (i.e. ex2)
  - set maven bin dir (e.g. /Users/mjoyner/apache-maven-3.5.2)

    >> PATH=$PATH:/Users/mjoyner/apache-maven-3.5.2/bin
  - Run maven heroku app deploy

    >> mvn heroku:deploy
- Open browser: *https://app-name.herokuapp.com* where app-name is the web app name

Heroku Hosted Web App

Announcements & Reminders

- HW #1 is due Friday, Sep 4th by 11:59pm
- (Update: Controller should use singleton to create shape)
- Use Piazza (public or private posts, as appropriate) for all communications re. COMP 504
    - Do not include code in a public post (could be considered an honor code violation)
- See course web site for syllabus, work assignments, due dates, office hours schedule.

Mini Exercise: Heroku

1. Create Heroku account (https://signup.heroku.com/dc)

2. Install Heroku (requires git to be installed first)

   https://devcenter.heroku.com/articles/heroku-cli

3. From the command line type:

>> heroku -v