## Tests added

testGetCircleShapeType() checks the return value of a circle object when the getShapeType method is called. It will pass if the return value is a String with “Circle” as its value.

testGetRectangleShapeType() checks the return value of a rectangle when the getShapeType method is called. The test will pass if the return value is a String with “Rectangle” as its value.

testClear() tests the clear method from the MainActivity.java. It uses a vector from the mock instance of MainActivity and adds 2 objects to the vector, a circle object and a rectangle object. After that, it calls the clear() method, which should clear the vector of all its contents. The test will pass if the vector size is 0.

testUpdateRectShapeCount() tests the updateShapeCount() method from mainActivity.java. It adds 2 rectangles to the vector in the mock instance of mainActivity and then calls the updateShapeCount() method. If the vector count returns 2 for the number of rectangle counts, the test will pass.

testUpdateCircShapeCount() tests the updateShapeCount() method from the MainActivity.java file as well. It adds 2 circles to the vector in a mock instance of MainActivity then calls the updateShapeCount() method. If the vector count returns 2 for the number of circle counts, the test will pass.

## Approach

I used the built in create test function built into the IDE to get me started by right clicking on the class that I want to create a test with, and then creating the tests. After initially writing the tests, I realized that I could not create objects without context. I did some reading and found that Mockito was the framework to use to mock the app components. So I used that to mock the context, which got the simpler tests to work, such as the testRectangleShapeType and the testCircleShapeType.

To make sure that it was really working as intend, I wrote the tests to fail initially, which is did, and then write it to pass after. After that I tried applying the same logic of mocking the components for the rest of the tests but could not get it to work properly. Logically I think that the tests are sound, but something on the backend for the tests is not being mocked and so the tests were failing or pointing to a NullPointer. I tried to mock every class that was used that I could think of but it was still not enough.

When I found out I had to use mockito, I implemented it within the gradle build file after reading from the vogella site listed below.

## References

<http://www.vogella.com/tutorials/Mockito/article.html#mockito_installation_gradle>

for implementing mockito to use with my app

<https://www.youtube.com/watch?v=ZJE0MDKJOow>

learning to use the built in IDE test creation tools

<https://youtu.be/M_6z8L8qK8o>

<https://youtu.be/tkzJsP7NP54>

general learning about unit tests

## <http://stackoverflow.com/questions/38734073/using-mockito-to-test-function-that-uses-a-context>

<http://stackoverflow.com/questions/29616162/android-unit-test-how-to-mock-androids-context>

to learn how to mock the context

<http://stackoverflow.com/questions/31724445/mockito-mock-for-context-and-getapplicationcontext>

<http://stackoverflow.com/questions/2095695/android-unit-tests-requiring-context>

other attempts at mocking the context

<http://stackoverflow.com/questions/25368203/how-to-use-mockito-hamcrest-in-unit-tests-in-android-studio>

<https://youtu.be/79eXGJ2rKZs>

<https://youtu.be/Qq0uziWeMAA>

General learning about mockito

Also worked closely with classmate Marc Panlilio to figure out logic of how the code should work.