SE 333 Software Testing

Assignment 4 Testing with Mocks

Due Date: October 11, 2021, 11:59 pm
Late penalties: 1% per day for a maximum of 7 days, not accepted after 7 days

The Objectives

The objective of this assignment is to effectively use mock objects in a test scenario.

Problem

In this assignment, you are given two production classes: FileMonitor and FileService. The FileMonitor uses the FileService to interact with the file system. The FileMonitor has a function clean(String aDirectory) that needs to be tested. The job of clean() is to:

1. Ask the FileService for a listing of the given directory. The FileService has a function getDirectoryContents(String aDirectory) that is used for this purpose.

2. If the listing contains any files matching "*.tmp", ask the FileService to delete them. FileService has a function delete(String filename) for this purpose.

3. The clean function returns a List<String> of the files the monitor sent to FileService for removal.

   The constructor of FileMonitor takes a FileService as a parameter.

Assignment
Create test cases in FileMonitorTest that test:

1. When there are "*.tmp" files in the directory listing, FileMonitor returns the expected sub list from a call to clean().

2. When there are no "*.tmp" files in the listing, FileMonitor makes no delete() calls to FileService. Notice this is not a test of the return value of clean().

3. FileMonitor does not throw an exception when FileService returns an empty list as the directory listing.

These tests must not actually depend on a real FileService. We do not want to have to create real files so that directory listings contain something to delete. We can't count on files being there and we can't count on the files being deletable.

To perform these tests, do the following:

1. Download and import the maven project **week4-file-service.zip** from Contents/Week 4/homework.

2. Edit the build.gradle to personalize the project name, output name, etc.

3. Edit FileMonitorTest.java and add tests using the following techniques:
    a. Use the mock() function to make a mock FileService
    b. Use the when() function to give the mock it required behavior
    c. Create a FileMonitor and send messages to it
    d. Use Junit assert functions and Mockito verify(), to determine if the desired behavior took place.

Overview of the template project

Once the project is installed, you should see this:

1. There are 2 test classes (FileMonitorTest and FileServiceTest). The assignment only involves adding tests to FileMonitorTest. Notice that this class has no meaningful members yet. I provided it so that you shouldn't have to concern yourself with import statements. If you do see a need for more imports however, please feel free to add them.

2. You have the source code for the class you are testing (FileMonitor) and the class you need to mock (FileService).

3. There is also the FileApp class that demonstrates how these classes work together. This should help you see what steps the tests must take to arrange and then execute the required actions. This demo creates a file, cleans the directory the file was created in, then cleans again to show that the file was removed.
    IMPORTANT:
    a. your test should not create any temp files for testing. One of the reasons we are using mocks is to avoid this.
    b. You are not testing FileApp. Is exists solely to demonstrate the relationship between FileService and FileMonitor

Discussion

1. There is a lot to figure out here. Please use Slack to discuss this problem freely among yourselves (I will contribute as well), just don't post any actual solutions to the problem.
2. A lot of coding is not required. The assignment section above lists just 3 requirements to be tested. There is no benefit to going beyond these requirements.

Grading Criteria

1. All required tests are provided
2. Coding style guidelines were followed
3. Test design best practices were followed

Deliverables

1. The usual gradle-generated zip file