70094 14

Mock Objects

Submitters

sf23	Shihan Fu
zj123	Ze Jin

```
Final Tests: Summary for sf23 of v5
-----------------------------------

  Public Tests:
    Compiles:                    1 / 1
    Tests Pass:                  1 / 1
    Coverage and Style Checks:   1 / 1

Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_spring/SE_Design_Ex4_sf23.git
Commit ID: 69779
```

✓ Good.

Great work. See some comments below. 17/20

```java
package ic.doc.camera;

import org.jmock.Expectations;
import org.jmock.integration.junit4.JUnitRuleMockery;
import org.junit.Rule;
import org.junit.Test;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

public class CameraTest {

  @Rule public JUnitRuleMockery context = new JUnitRuleMockery();

  Sensor sensor = context.mock(Sensor.class);
  MemoryCard memoryCard = context.mock(MemoryCard.class);

  @Test
  public void switchingTheCameraOnPowersUpTheSensor() {
    Camera camera = new Camera(sensor, memoryCard);
    // write your test here
    context.checking(
        new Expectations() {
          {
            exactly(1).of(sensor).powerUp();
          }
        });
    camera.powerOn();
    assertTrue(camera.get_power());
  }

  @Test
  public void switchingTheCameraOffPowersDownTheSensor() {
    Camera camera = new Camera(sensor, memoryCard);
    // write your test here
    context.checking(
        new Expectations() {
          {
            exactly(1).of(sensor).powerUp();
            exactly(1).of(sensor).powerDown();
          }
        });
    camera.powerOn();
    camera.powerOff();
    assertFalse(camera.get_power());
```

Tell, don't ask.

```java
  }

  @Test
  public void pressShutterWhenCameraPowerOff() {
    Camera camera = new Camera(sensor, memoryCard);

    context.checking(
        new Expectations() {
          {
            never(sensor);
          }
        });
    // assumption: after creating a camera, its mode is power off
    camera.pressShutter();
  }

  @Test
  public void pressingShutterWithPowerOnCopiesData() {
    Camera camera = new Camera(sensor, memoryCard);
    context.checking(
        new Expectations() {
          {
```

```java
            exactly(1).of(sensor).powerUp();
            exactly(1).of(sensor).readData();
            will(returnValue(new byte[0]));
            exactly(1).of(memoryCard).write(with(any(byte[].class)));
          }
        });
```

memory card should be written with exactly what was read

```java

    camera.powerOn();
    camera.pressShutter();
    assertTrue(camera.get_isWritingData());
  }

  @Test
  public void switchingTheCameraOffWithoutPowersDownTheSensor() {
    Camera camera = new Camera(sensor, memoryCard);
    context.checking(
        new Expectations() {
          {
            exactly(1).of(sensor).powerUp();
            exactly(1).of(sensor).readData();
            will(returnValue(new byte[0]));
            oneOf(memoryCard).write(with(any(byte[].class)));
            never(sensor).powerDown();
          }
        });

    camera.powerOn();
    camera.pressShutter();
    camera.powerOff();
  }

  @Test
  public void completeWritingPowerDown() {
    Camera camera = new Camera(sensor, memoryCard);
    context.checking(
        new Expectations() {
          {
            exactly(1).of(sensor).powerUp();
            exactly(1).of(sensor).readData();
            will(returnValue(new byte[0]));
            oneOf(memoryCard).write(with(any(byte[].class)));
            exactly(1).of(sensor).powerDown();
          }
        });
    camera.powerOn();
    camera.pressShutter();
    camera.powerOff();
    camera.writeComplete();
    assertFalse(camera.get_isWritingData());
  }
}
```

```java
package ic.doc.camera;

public class Camera implements WriteListener {

  private final Sensor sensor;
  private final MemoryCard memoryCard;
  private boolean power = false;
  private boolean isWritingData = false;

  /**
   * return the status of isWritingData.
   *
   * @return boolean
   */
  public boolean get_isWritingData() {
    return this.isWritingData;
  }

  /**
   * constructor.
   *
   * @param sensor Sensor
   * @param memoryCard MemoryCard
   */
  public Camera(Sensor sensor, MemoryCard memoryCard) {
    this.memoryCard = memoryCard;
    this.sensor = sensor;
  }

  /** press shutter. */
  public void pressShutter() {
    if (power) {
      isWritingData = true;
      memoryCard.write(sensor.readData());
    }
  }

  /** camera power on. */
  public void powerOn() {
    sensor.powerUp();
    this.power = true;
  }

  /** camera power off. */
  public void powerOff() {

    if (!isWritingData) {
      sensor.powerDown();
    }
    this.power = false;
  }

  /** write complete. */
  @Override
  public void writeComplete() {
    isWritingData = false;
    if (!power) {
      sensor.powerDown();
    }
  }

  /**
   * return the status of power.
   *
   * @return boolean
   */
```

```java
  public boolean get_power() {
    return this.power;
  }
}
```

```
-------- Test Output --------
Running LabTS build... (Tue  6 Feb 16:32:01 UTC 2024)

Submission summary...
You made 2 commits
  - 30a909d implement the basic function and test file [2 files changed, 146 insertions, 7 deletions]
  - 6977908 implement the testing logic [1 file changed, 19 insertions, 6 deletions]

Preparing...

BUILD SUCCESSFUL in 961ms

Compiling...
BUILD SUCCESSFUL in 11s

Running tests...

ic.doc.camera.CameraTest > switchingTheCameraOffWithoutPowersDownTheSensor PASSED

ic.doc.camera.CameraTest > completeWritingPowerDown PASSED

ic.doc.camera.CameraTest > pressShutterWhenCameraPowerOff PASSED

ic.doc.camera.CameraTest > switchingTheCameraOffPowersDownTheSensor PASSED

ic.doc.camera.CameraTest > pressingShutterWithPowerOnCopiesData PASSED

ic.doc.camera.CameraTest > switchingTheCameraOnPowersUpTheSensor PASSED

BUILD SUCCESSFUL in 2s

Checking test coverage and code style...
BUILD SUCCESSFUL in 7s
Finished auto test. (Tue  6 Feb 16:32:36 UTC 2024)

-------- Test Errors --------

```