Setting Up Image Editor Unit Tests

Downloads

First you will need to visit the Projects section of the current CS 240 website and download the "Test Files (Zip)" folder. Inside this zip you will find 4 folders: jars, key_images, out_images, source_images, and test_files.

Create a Project

Open IntelliJ and create a new Project.



This tutorial will be setting up as a normal Java project. If you wish to create a Maven or Gradle project you will be responsible for setting it up on your own and getting dependencies working correctly.

Select "Java" for your project type and in "Project SDK" select whatever version of Java you want (You should select whatever is the latest version of Java installed on your computer). You do not need to select any additional libraries or frameworks. Click "Next".

Continue to click "Next" until you reach this window:

| (2) | New Project | × |
|-------------------|------------------------------------|---|
| Project name: | ImageEditor | |
| Project location: | ~/ImageEditor | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| More Settings | | |
| | Previous Finish Cancel Help | |

Name your project whatever you want though we suggest using ImageEditor. Click "Finish"

Now your project will build and will open:



Open the zip file you downloaded earlier and move jars, key_images, out_images, and source_images into the main folder of your project.



Right click on your src folder and add two new packages. One needs to be called passoff, the other needs to be called editor.

| 🔲 Project 👻 | ⊕ <u>÷</u> ¢ - |
|--|---|
| ImageEditor ~/ImageEditor idea jars key_images out_images source_images | |
| ■ src New ■ ImageEdi | Image: Second system Image: Second system Ctrl+X Image: Second system Ctrl+Shift+C Image: Second system Ctrl+Alt+Shift+C Image: Second system Ctrl+V Image: Second system |
| Find <u>U</u> sages Find in <u>P</u> ath Repl <u>a</u> ce in Path Analvze | Alt+F7 Ctrl+Shift+F Ctrl+Shift+R Ctrl+Shift+R & HTML File & Stylesheet |

Move the files from test_files into passoff

| ImageEditor ~/ImageEditor | | <u>e</u> 5 |
|---|---|------------|
| 🕨 🖿 .idea | | |
| ► 🖿 jars | | |
| key images | | |
| out images | | |
| source images | | |
| V src | <pre>15 private static final String KEY IMAGE DIRCTORY = "key images/";</pre> | |
| a editor | 16 private static final String (<u>kkF</u> = "Iseg.ppm"; | |
| V Dassoff | 17 private static final string one_boss not = contessing cy.ppm; 18 private static final String PPAGIUTUS = "Penguins.ppm": | |
| ImageEditorTest | <pre>19 private static final String SUNSET = "sunset.ppm";</pre> | |
| Pixel | | |
| | | |
| Illi External Libraries | 22 private static final String GRAYSCALE = "grayscale"; interface static final String GRAYSCALE = "grayscale"; | |
| The Seratches and Consolar | 23 private static final string EmbOSS = "embOSS"; 24 private static final String MTTTM = "metionblue"; | |
| - o scraccies and consoles | 25 | |
| | | |
| | | |
| | | |
| | 29 System.out.print(n("invert ling"); Christ() armend (Counce There Proceeder, There are proceeder, Singer the There are a structure thereas | |
| | 30 String[command = [SUBRE_IMAGE_DIRECTORY + TIMT, OUT_IMAGE_DIRECTORY + TIMET. + TIMT, INVERT]; | |

You'll see a lot of red errors, but that is okay for now.

Dependencies

The last step we need to do is add dependencies to your project. Since these tests run on JUnit 5 you need to have the files for JUnit 5 so that IntelliJ can access them and use them. The folder that is labeled "jars" is where all this code is stored as .jar files. So we will add dependencies to your project for these files so that your project knows where to go in order to access the code.

Select File > Project Structure (or use Ctrl+Alt+Shift+S) and you'll see this screen

| 2 | Project Structure | × |
|---|---|-----|
| Project Settings Project Modules Libraries Facets Artifacts Platform Settings | Project Structure Project name: ImageEditor Project SDK: This SDK is default for all project modules. A module specific SDK can be configured for each of the modules as required. ImageEdite ImageEdite Project language level: | × |
| SDKs Global Libraries Problems | This language level is default for all project modules. A module specific language level can be configured for each of the modules as required. SDK default (11 - Local variable syntax for lambda parameters) Project compiler output: This path is used to store all project compilation results. A directory corresponding to each module is created under this path. This directory contains two subdirectories: Production and Test for production code and test sources, respectivel A module specific compiler output path can be configured for each of the modules as required. | ly. |
| | /users/guest/w/westenm/ImageEditor/out | |
| ? | OK Cancel Apply | |

Double-click Modules and click the tab labeled Dependencies

| ۹ | | Project Structure | |
|--------------------------|-------------|---|---|
| $\leftarrow \rightarrow$ | + - 6 | Name: ImageEditor | |
| Project Settings | lmageEditor | inage dicor | |
| Project | | Sources Paths Dependencies FindBugs-IDEA | |
| Modules | | | |
| Libraries | | Module SDK: Rev Edit | |
| Facets | | Export Scope · | ÷ |
| Platform Settings | | 🔚 11 (1) (java version "11.0.4") | |
| SDKs | | Module source> | |
| Global Libraries | | | - |
| | | | |
| Problems | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | Dependencies storage format: IntelliJ IDEA (.iml) 🛛 🔻 | |
| | | | |
| ? | | OK Cancel Apply | |



Now click the little "+" icon to the far right of the window and select "JARs or directories"

Navigate to the jars folder in your project and select all of the jar files then click "OK"

| | Attach Files or Directories | × |
|------------------------|--|-----------|
| Select files or direct | tories in which library classes, sources, documentation or native libraries are | elocated |
| + 🗆 🖬 🖬 🛤 | N C P | Hide path |
| /users/guest/w/we | estenm/Desktop/JUnit5StudentFiles/ImageEditorTesting/jars/junit-5.0.jar | Ŧ |
| | EvilHangmanTesting ImageEditorTesting ImageEditorTesting | |
| | III junit-5.0.jar III junit-jupiter-api-5.2.0.jar III junit-jupiter-engine-5.2.0.jar III junit-platform-commons-1.2.0.jar III junit-platform-engine-1.2.0.jar III junit-platform-launcher-1.2.0.jar | |
| | Images | |
| ? | Drag and drop a file into the space above to quickly locate it in the tree | Cancel |

These will be added to the dependencies list. Click "OK"



All of the red errors should disappear except for the ones related to ImageEditor.main(command);

This is because you have not written your code yet. In your other package create a new Java class and call it ImageEditor



Give it a standard java main function which will look like this.



Now make sure that in your ImageEditorTest class it is importing from the correct location



All of the errors should be gone. Now right click on ImageEditorTest and select "Run

ImageEditorTest" or click on the green arrow next to the class declaration in the file itself

| passoff ImageEditorJUnitTests.zip | | | blue = Integer.pa realBlue = keySca |
|---|--|--|--|
| lmageEditorTest | New | 537 | Pixell studentPixe |
| Pixel ImageEditorTesting.im slctemple.ppm External Libraries Scratches and Consoles | 器 Cu <u>t</u> 喧 ⊆opy | | Ctrl+X Ctrl+C |
| | Copy Path Copy Reference | :e | Ctrl+Shift+C Ctrl+Alt+Shift+C |
| | Find <u>U</u> sages | | Alt+F7 |
| | Analy <u>z</u> e FindBugs | | • • |
| | <u>R</u> efactor | | ٠. |
| | Add to F <u>a</u> vorit | tes | • |
| | Browse Type H <u>R</u> eformat Cod | lierarchy e | Ctrl+H Ctrl+Alt+L |
| | Optimi <u>z</u> e Impo <u>D</u> elete | orts | Ctrl+Alt+O Delete |
| | Build <u>M</u> odule R <u>e</u> compile 'Im | 'ImageEditorTestin ageEditorTest.java | g' ' Ctrl+Shift+F9 |
| | ▶ R <u>u</u> n 'ImageEdi | itorTest' | Ctrl+Shift+F10 |

```
package passoff;
    import editor.ImageEditor;
    import org.junit.jupiter.api.*;
    import java.io.File;
    import java.io.FileNotFoundException;
    import static org.junit.jupiter.api.Assertions.*;
   Run Test ass ImageEditorTest {
J.
        private static final String SOURCE IMAGE DIRECTORY = "source
        private static final String OUT IMAGE DIRECTORY = "out image:
        private static final String KEY IMAGE DIRCTORY = "key images,
        private static final String FEEP = "feep.ppm";
        private static final String ONE DOES NOT = "one-does-not-simp
        private static final String PENGUINS = "Penguins.ppm";
        private static final String SUNSET = "sunset.ppm";
        private static final String TINY = "tiny.ppm";
        private static final String INVERT = "invert";
        private static final String GRAYSCALE = "grayscale";
        private static final String EMBOSS = "emboss";
        private static final String MOTION = "motionblur";
        @Test
        @DisplayName("Test Invert Tiny")
        void invertTinyTest(){
            System.out.println("Invert Tiny");
            String[] command = {SOURCE IMAGE DIRECTORY + TINY, OUT IM
                ImageEditor.main(command);
            catch(Throwable t){
```

This will run the tests. The tests will obviously fail because you have not written your code yet, but now you have the tests all ready to go so you can test as you code.