8 Designing Tests for State Change Abstracting with Function Objects

Goals

In the first part of this lab you will learn how to correctly design tests for the methods that change the state of an object.

In the second part of the lab you will learn to abstract over the functional behavior.

8.1 Designing Tests for State Change

For this part download the files in *Lab8Part1.zip*. The folder contains the files *ImageFile.java* and *Examples.java*.

Starting with partially defined classes and examples will give you the opportunity to focus on the new material and eliminate typing in what you already know. However, make sure you understand how the class is defined, what does the data represent, and how the examples were constructed.

Create a new **Project** *Lab8Part1* and import into it the files *ImageFile.java* and *Examples.java*. Also import the *TestHarness.jar* files and *jpt.jar* from the previous lab.

- Design the method *crop* that changes the dimensions of an *ImageFile* object to the given *width* and *height*. The *Examples* class contains comments on what needs to be done to design the tests. Follow the outline given by the comments to design the needed tests.
- Design the method *changeName* that allows us to change the name field of an *ImageFile* object. Design the tests.

8.2 Abstracting with Function Objects

For this part download the files in *Lab8Part2.zip*. The folder contains the files *ISelect.java*, *LoIF.java*, *MtIF.java*, *ConsLoIF.java*, and *Examples.java*.

Create a new *Project Lab8Part1* and import into it the file *ImageFile.java* from the *Part* 1 and the files *ISelect.java*, *LoIF.java*, *MtIF.java*, *ConsLoIF.java*, and *Examples.java* from the *Lab8Part2* folder. Again, import the test harness files and *jpt.jar* from the previous lab.

1. Design the method *smallerThan40000* in the class *ImageFile* that determines whether the file size is smaller that 40000 pixels. The size is computed as a product of the image file's width and height.

2. Design the method *allSmallerThan40000* that determines whether all items in a list are smaller that 40000 pixels. The size is computed as a product of the image file's width and height.

3. Design the method *nameShorterThan4* that determines whether the name in this *ImageFile* object is shorter than 4.

4. Design the method *allNamesShorterThan4* that determines whether all items in a list have a name that is shorter than 4 characters.

5. Design the class *SmallerThan40000* that implements the *ISelect* interface with a method that determines whether the size of the given *ImageFile* is less than 40000.

   Make sure in the class *Examples* you define an instance of this class and test the method.

6. Design the class *NameShorterThan4* that implements the *ISelect* interface with a method that determines whether the name in the given *ImageFile* object is shorter than 4.

   Make sure in the class *Examples* you define an instance of this class and test the method.

7. Design the method *allSuch* that that determines whether all items in a list satisfy the predicate defined by the *select* method of a given instance of the type *ISelect*. In the *Examples* class test this method by abstracting over the method *allSmallerThan40000* and the method *allNamesShorterThan4*.

8. If you have time left, follow the same steps as above to design the method *anySuch* that that determines whether there is an item a list that satisfies the predicate defined by the *select* method of a given instance of the type *ISelect*.