7 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.

7.1 Designing Tests for State Change

For this part download the files in Lab7-Part1-f07.zip. The folder contains the files ImageFile.java, ISelect.java, SmallImageFile.java, AList.java, MTList.java, ConsList.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 Lab7-fl07 and import into it all of the given files. Also import tester.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.

7.2 Quiz

7.3 Abstracting with Function Objects

We will now practice the use of function objects. The only purpose for defining the class SmallImageFile is to implement one method that determines whether the given ImageFile object has the desired property. An instance of this class can then be used as an argument to a method that deals with ImageFiles.
1. In the Examples class design the tests for the class SmallImageFile.

2. Design the method `allSmallerThan40000` that determines whether all items in a list are smaller than 40000 pixels. The method should take an instance of the class SmallImageFile as an argument.

3. 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.

4. Design the method `allNamesShorterThan4` that determines whether all items in a list have a name that is shorter than 4 characters. The method should take an instance of the class NameShorterThan4 as an argument.

5. Design the method `allSuch` 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`.

6. Design the class GivenKind that implements the ISelect interface with a method that produces `true` for all ImageFiles that are of the given kind. The desired kind is given as a parameter to the constructor, and so is specified when a new instance of the class GivenKind is created.
   Hint: Add a field to represent the desired kind to the class GivenKind.

7. In the Examples class use the method `allSuch` and the class GivenKind to determine whether all files in a list are jpg files. Do it again, but now ask about the giff files.

8. If you have some time left, design the method `filter` that produces a list of all ImageFiles that satisfy the ISelect predicate. Test it with as many of your predicates as you can.

9. For the first portfolio program, at home, follow the same steps as above to design the method `anySuch` 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.