7 Using Annotation for Testing Flexibility

So far we have mainly ignored the use of visibility modifiers and the subsequent information hiding commonly used in production programming. All methods we have tested have been defined as `public`, `protected`, or by default *package-friendly* ,and so the `Examples` class could invoke them and test their outcomes. However, a comprehensive testing library must provide some means for testing private methods as well. The *tester* library uses *Annotations* to extend the flexibility in how and where the test methods are defined.

A Brief Tutorial

Java allows the programmer to use *Annotations* to provide additional information about its classes, methods, and fields, that the program may use at the run time. The *tester* library uses *Annotations* to allow the programmer to specify that a class may contain test methods and to specify that a method with an arbitrary name is to be run as a test method. This methodology is much more powerful than the previous manner of invoking the *tester* library.

By adding the `@Example` above any class declaration, you can tell the *tester* to use that class as an *example* class (i.e. notifying the tester to look for the test methods defined in that class). This means that you can declare **any number** of classes to be *example* classes. The *tester* will run all of the tests contained within each class that is annotated as an *example* class.

The *tester* will still run every method whose name begins with 'test' in every such class; **however**, this naming convention can be bothersome. Or, what if you already have a function that you would like to use as a test method, but it is referenced in numerous places in your code? In any class marked by the `@Example` annotation, the *tester* library will also run as a test every method marked with a `@TestMethod` annotation.

Note that methods annotated by `@TestMethod` must still take a `Tester` object as an argument.

Here is an example:

```
import tester.*;
@Example //declares this class to be an example class
public class Song {
  String title;
  String artist;
  int duration;
  int rating;
  boolean released;

  public Song(){
```

```
    this.title = "Nothing";
    this.artist = "No One";
    this.duration = 0;
    this.rating = 0;
    this.released = false;
  }

  public Song(String title, String artist, int dur, int rate, boolean released){
    this.title = title;
    this.artist = artist;
    this.duration = dur;
    this.rating = rate;
    this.released = released;
  }

  //change the title of this Song
  public void changeName(String title){
    this.title = title;
  }

  //calculate the cost of this song
  public int cost(){
    if(this.released){
      return 2 * this.rating;
    }
    return 0;
  }

  //will be run as a test
  @TestMethod
  public void doSomeStuff(Tester t){
    t.checkExpect(true, "hahaha");
  }

  //will also be run as a test
  public void testCost(Tester t){
    if(this.released){
      t.checkExpect(this.cost(), 2 * this.rating, "TestCost");
    } else{
      t.checkExpect(this.cost(), 0, "TestCost");
    }
  }

  //also run as a test
  public void tests(Tester abc) {
    testCost(abc);
  }
}
```

Note that making one or more of your classes serve as an *example* class does not invalidate those classes in any way. Being an example class merely indicates that the class contains test data in addition to all of the material it is meant to contain as a part of your program.

Lab Tasks

Start a new project and import into it the file **AnnotationsSample.java**. Add the *tester* library to the class path.

1. Run a configuration with `tester.Main` as the main method. Notice that all three tests ran.

2. Change the visibility of the `cost` method to `private` and run the tests again. It should still work.

3. Add another `private` method to the class `Song`, add the tests in any version you choose, and run it again.

4. Import the file `AnnotationExample.java` to your project. Run its `main` method. Notice that even trough the name of the examples class does not start with `Examples`, the annotated test method is executed.

5. Now run again the configuration for the `AnnotationsSample`. Notice that it ran all tests - those in the file **AnnotationsSample.java** as well as those in the file **AnnotationExample.java**.

6. Import the file `AnnotationExample2.java` to your project. It has no `main` method and the name of the examples class does not start with `Examples`. Run again the configuration for the `AnnotationsSample`. Notice that it ran all tests - those in the file **AnnotationsSample.java** as well as those in the files **AnnotationExample.java** and **AnnotationExample2.java**.