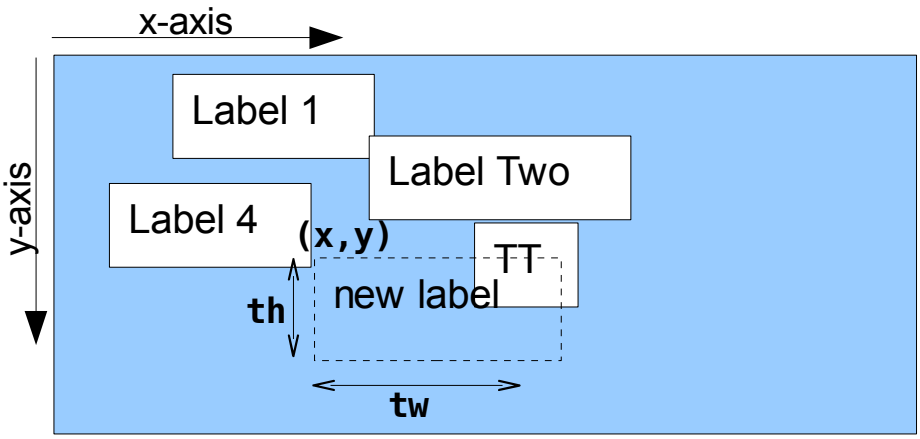
219343 Homework 4

You are developing a GUI application. In part of the information display, you have to show the labels of various objects in the window. See below for an example.



You want to place a new label on the screen. You have class `TextLocator` that computes the right position for the new label, given the "best" top-left corner of the label. This class also maintains the positions and sizes of all labels. (This is a bad design, but we do this to reduce dependencies.) A "template" of `TextLocator` is the following.

```
public class TextLocator {
    public void add(int x, int y, int tw, int th) { ... }

    public boolean isFreeFor(int x, int y, int tw, int th) { return true; }
}
```

Method `add` tells the position and the size of a label box which is already on the screen to an object of class `TextLocator`. The method in `TextLocator` that we want to test is `isFreeFor`; it takes the coordinate of the top-left corner and the size of the proposed box, and returns if that proposed area is free. The parameters to both methods describe the position and the size of a label box, as shown in the figure above.

In the example above, if the position and the size of the dashed box is given to `isFreeFor`, the return value should be `false`.

Your Homework

1. Write a JUnit test for this method. It can be either in JUnit 3.8 or JUnit 4. Your class should contain a few interesting test methods (at least 3). You should think about the guidelines (i.e., Right-BICEP). Also, try to use Fixtures (`@Before` and `@After`, or `setUp()` and `tearDown()`).

2. Complete parts of `TextLocator` so that it passes the test in problem 1.

An example

A rather trivial example of a test method is:

```
    @Test public void testSimpleOverlap() {
        TextLocator tl = new TextLocator();
        tl.add(10,10,200,20);
        assertEquals(true,tl.isFreeFor(100,100,100,100));
        assertEquals(false,tl.isFreeFor(100,15,100,20));
    }
```

(Note that you should not use this test method in your JUnit test.)