An example to tell how Abbot is been used to unit test.

Let's say we are trying to unit test a swing based JComponent, JRoundRectangle which has certain size and certain color, say Blue.

Test Code Snippet:

```
JRoundRectangle br = new JRoundRectangle(100, 250, 60, 60, 10, 10);
br.setColor(Color.BLUE);
```

Now when we ask Abbot to find the JComponent which is a JRoundRectangle and color blue. The unit tests initially called **findComponent** of the **BasicFinder** class and sent the matcher, for the type of object it is trying to find.

Prototype:

```
public Component find(Matcher m) throws ComponentNotFoundException, MultipleComponentsFoundException ;
```

It tells whether the component is present or else fails the test if it doesn't find one.

With the current Abbot implementation, let's say there are 2 or more JComponents which have same features like size, color and locations. When we try to find them, Abbot throws "**MultipleComponentsFoundException** ".

In order to use Abbot for finding multiple similar components, the current code has to be modified.
A similar new method called **findComponents** was adde in the **BasicFinde**r class, which will now find and return multiple components of the matcher type, if found.

Prototype:

```
public Component[] findComponents(Matcher m) throws ComponentNotFoundException;
```

To support finding the matches of multiple components, new methods were added.

```
public Component[] findComponents(Hierarchy h, Matcher m) throws ComponentNotFoundException ;
```

The above method further calls a protected newly added method to find the matching component.

```
protected void findMatchesComponents(Hierarchy h, Matcher m, Component c, Set found);
```

So the following test method will get the all the components that matches the match.

```
@Test
public void testWidthDimensionOfBlueRoundRectangle() throws Throwable {

//initial test code

JRoundRectangle rr2 = new JRoundRectangle(200, 250, 60, 60, 10, 10);
rr2.setColor(Color.BLUE);
panel.add(rr2);

JRoundRectangle rr3 = new JRoundRectangle(200, 250, 60, 60, 10, 10);
rr3.setColor(Color.BLUE);
panel.add(rr3);

// continued test code

Component[] components = getFinder().findComponents(new Matcher() {
        public boolean matches(Component c) {
                return c instanceof JRoundRectangle && ((JRoundRectangle) c).getColor() ==
        (Color.BLUE) && ((JRoundRectangle) c).getX() == 200 && ((JRoundRectangle) c).getY() == 250;
        }
});

System.out.println("comps.length=" + components.length); //Prints 2

//continued test code to test further things

}
```