GUI Testing

- GUI testing, differences and challenges.
- Styles of testing
- Testing Frameworks, AssertJ and Costello

Ciências ULisboa

slides: João Neto

GUI Testing

- Graphical User Interfaces can constitute a great part of the code being developed
- Tools like Java Swing make GUI programming easier
  - But they also make GUI testing harder
- GUI is based on an event-based architecture
  - User actions result in events
  - User can produce events in any order
  - State machines can be used to model the GUI behavior, but they get really ugly really fast!
  - Testing must also produce events to mimic user usage

Challenges

- A direct simulation is not feasible
  - Users can click on any pixel of the screen
- Most GUI components have many parameters and complex behavior
- The state of a GUI is the set of all states of its components
- The public interface of a GUI app is the GUI!

Challenges

- Test case generation – which sequences of user actions?
  - What is the expected behavior?
- Regression testing – can we use tests from previous versions?
  - If a GUI changes its template, will it break tests?
    - Brittle tests are much more probable in this context
- How should we model the GUI in the testing environment?
- Test code often outruns the respective GUI.

Styles of Testing

- Black box: testing the GUI itself
  - Launch app, simulate mouse & keyboard events
  - Comparing screen results provides brittle tests
  - Other method: Costello
- Grey box: testing internal components
  - Launch app; obtain references to components and send events to them; assert component states afterwards
  - Tests harder to break, business model can be tested via GUI behavior
  - Tools: java.awt.Robot; Abbot; AssertJ Swing

Grey Box – Robot

- The Java API provides class `java.awt.Robot`
- It can be used to generate native system input events
  - These events will indeed move the mouse, click, etc.
- User interaction can be simulated by the robot
  - Very dependent of current GUI template
  - If something changes its place, tests break

GUI testing frameworks

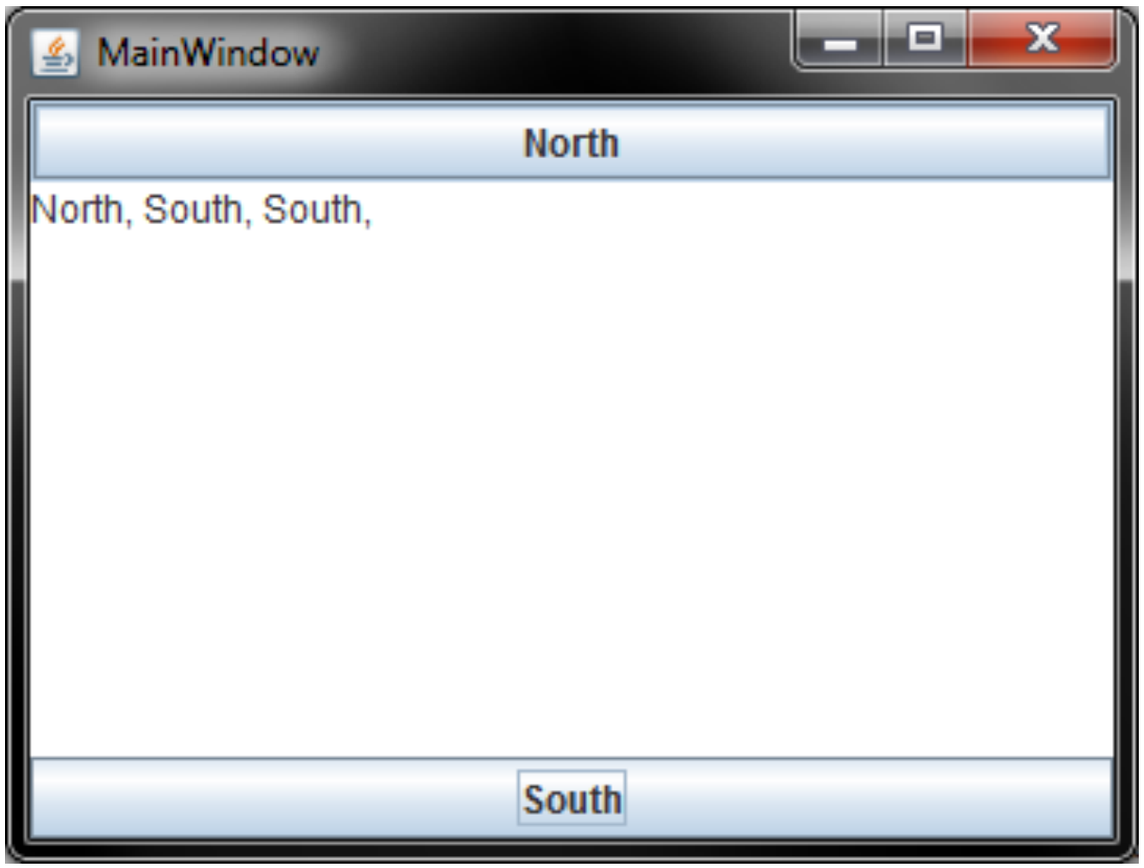
- Abbot and FEST Swing were two older GUI testing frameworks
  - They provide high-level interfaces for Swing UI testing
  - Integrated with JUnit
  - Still used but development stopped for both
- New tool: AssertJ Swing
  - joel-costigliola.github.io/assertj/assertj-swing.html

AssertJ Swing

- Reliable reproduction of user input
- High-level actions
- Reliable GUI component lookup
- Support for all Swing components
- JUnit compatible

Example

- Suppose a GUI with two buttons, where both write into a JTextArea



Example

```
public class MainWindow extends JFrame {
    ...

  private JPanel createContentPane() {
     JTextArea centerArea = new JTextArea();
     centerArea.setName("Center-Area");
     centerArea.setEditable(false);
     JButton northButton = this.createButton("North", centerArea);
     JButton southButton = this.createButton("South", centerArea);
     JPanel contentPane = new JPanel(new BorderLayout());
     contentPane.add(centerArea);
     contentPane.add(northButton, BorderLayout.NORTH);
     contentPane.add(southButton, BorderLayout.SOUTH);
     return contentPane;
  }
```

Example

- Some boilerplate code is needed to get a reference to the frame object

```
public class AbstractUiTest extends AssertJSwingTestCaseTemplate {
   protected FrameFixture frame;

   @BeforeEach
   public final void setUp() {
     this.setUpRobot();
     MainWindow mainWindow = GuiActionRunner.execute(
       new GuiQuery<MainWindow>() {
           protected MainWindow executeInEDT() throws Exception {
              return MainWindow.showWindow();
           }
       });
     this.frame = new FrameFixture(this.robot(), mainWindow);
     this.frame.show();
     onSetUp();
   }
```

Example

- Each test set class extends the previous class, and create fixtures for the components it needs for their tests

```
public class MainWindowTest extends AbstractUiTest {

  private JButtonFixture northButtonFixture;
  private JButtonFixture southButtonFixture;

  @Override
  protected void onSetUp() {
    this.northButtonFixture =
       this.frame.button(JButtonMatcher.withText("North"));
    this.southButtonFixture =
       this.frame.button(JButtonMatcher.withText("South"));
  }
```

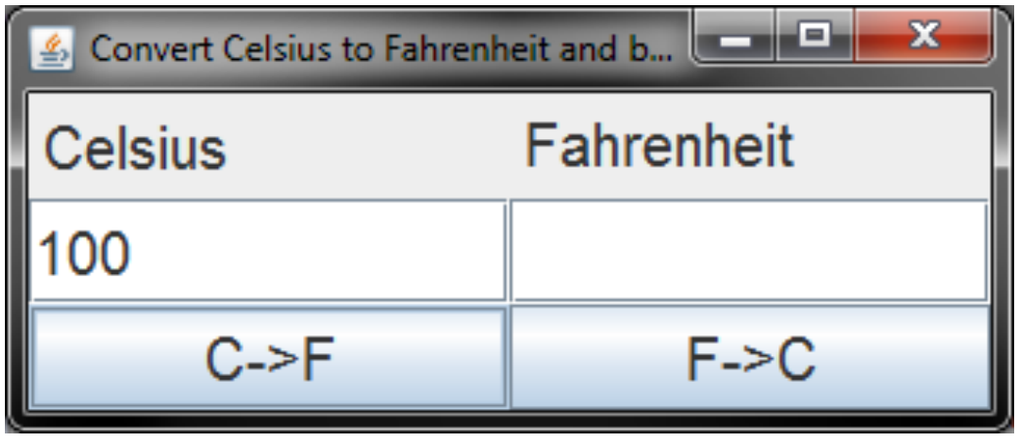
Example

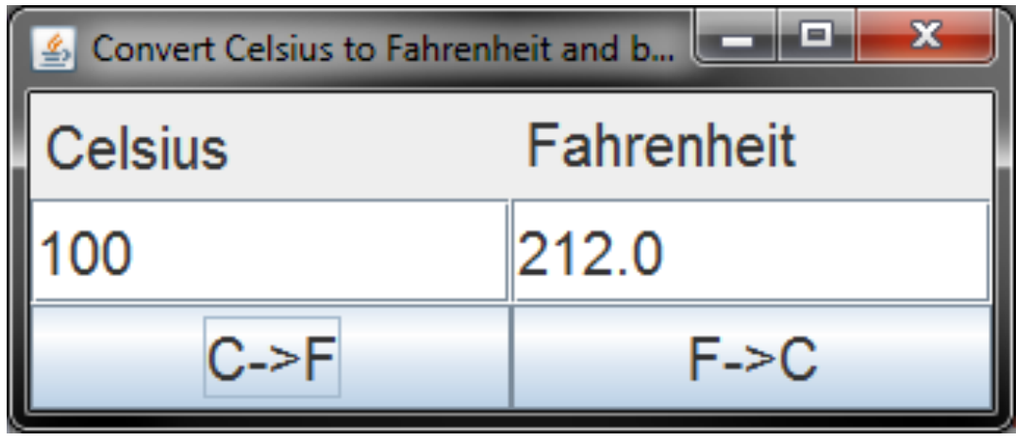
- A test case example:

```
@Test
public void testNorthClick() {
  // use JTextComponentMatcher.any() as there is only one text area
  this.frame.textBox(JTextComponentMatcher.any())
            .requireVisible()
            .requireEnabled()
            .requireNotEditable()
            .requireEmpty();
this.northButtonFixture.requireVisible()
                       .requireEnabled()
                       .click();
this.frame.textBox("Center-Area")
          .requireText("North, ");
```

Exercise 1

- Given the GUI for converting Celsius to Fahrenheit, check if the expected behavior for this test case occurs





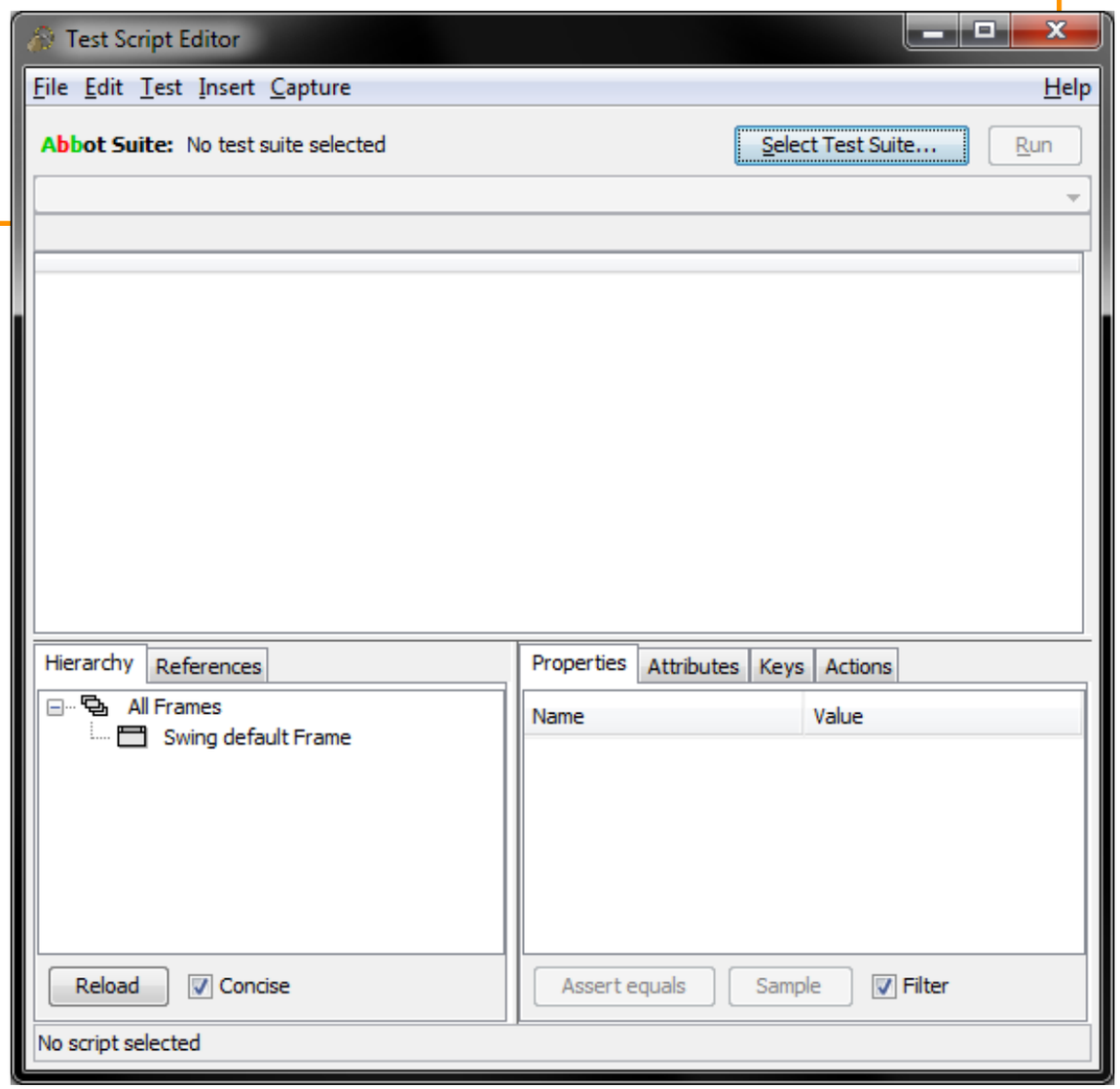
Black Box – Abbot and Costello

- Costello is a script recorder
- It allows the tester to run the GUI app, while saving all commands and translating into an XML script
- The script can be modified to add assertions

Running Costello inside Eclipse

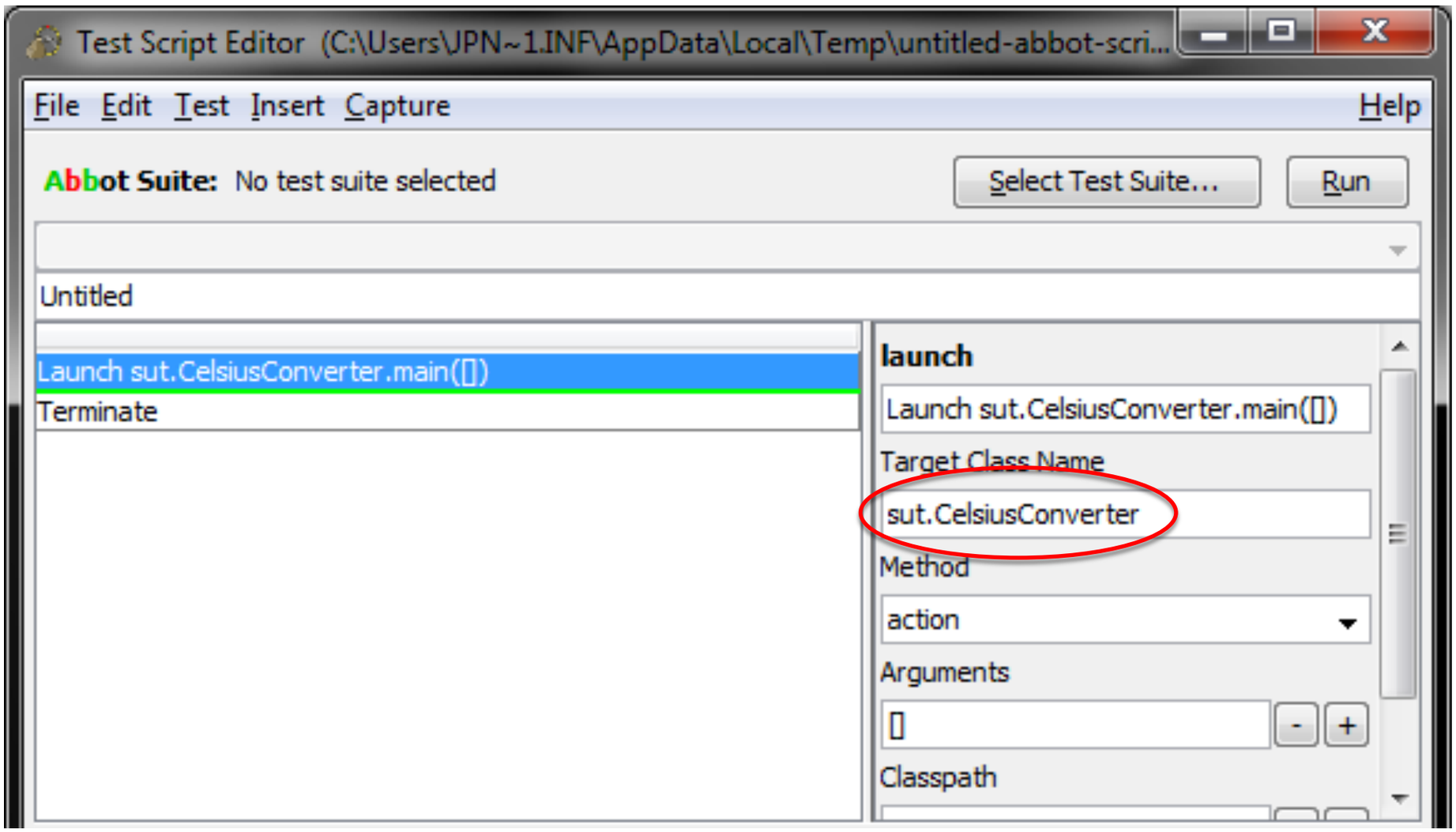
- Run this class in your test folder (import project `vvs_gui`)

```
public class CostelloRunnner {

  public static void main(String[] args) {
    abbot.editor.Costello.main(new String[] {});
  }

}
```



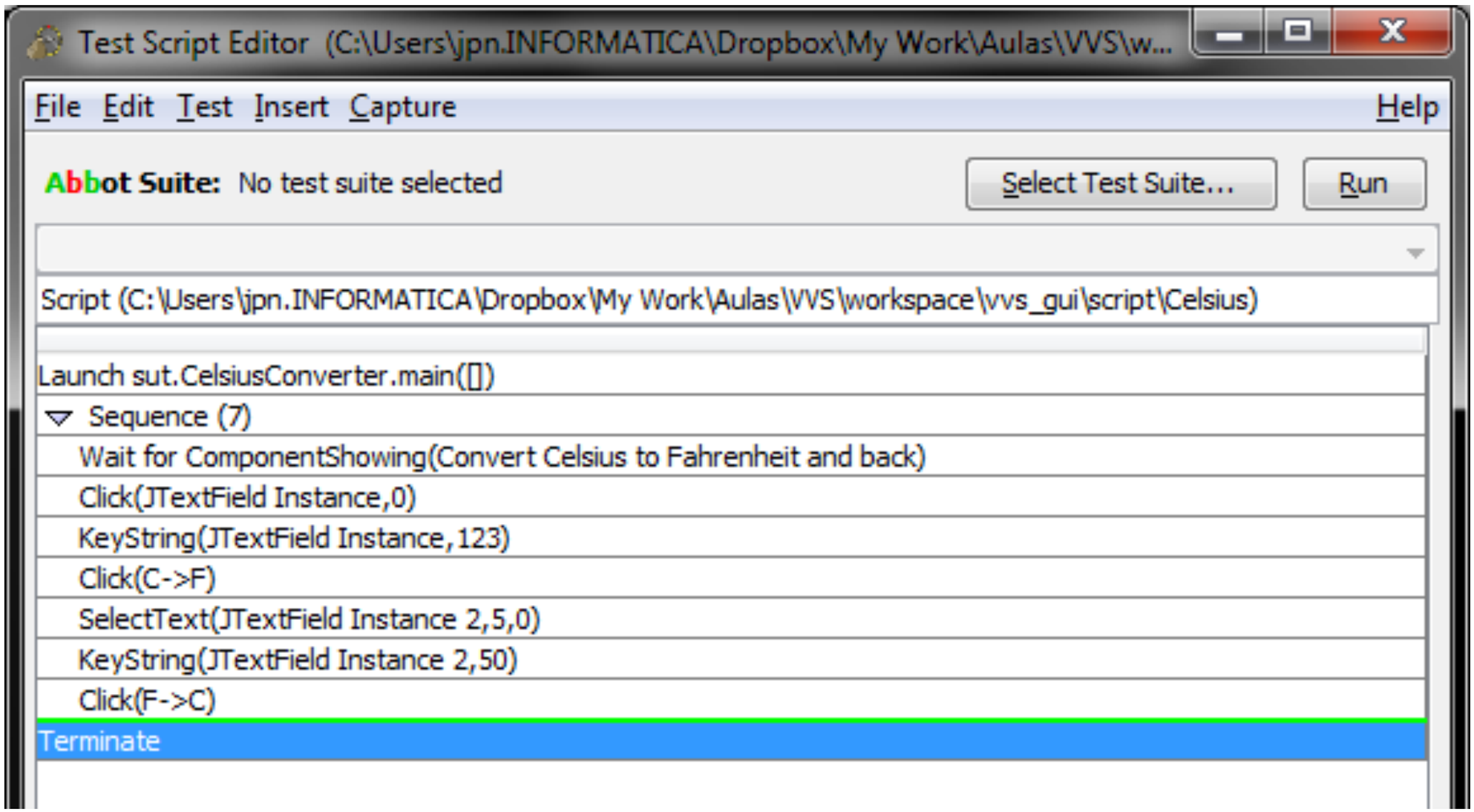
Running Costello inside Eclipse

- Create new script with File | New Script…
- Click "Insert your launch information here" and at the right a list of options appear
  - In the target class select the SUT



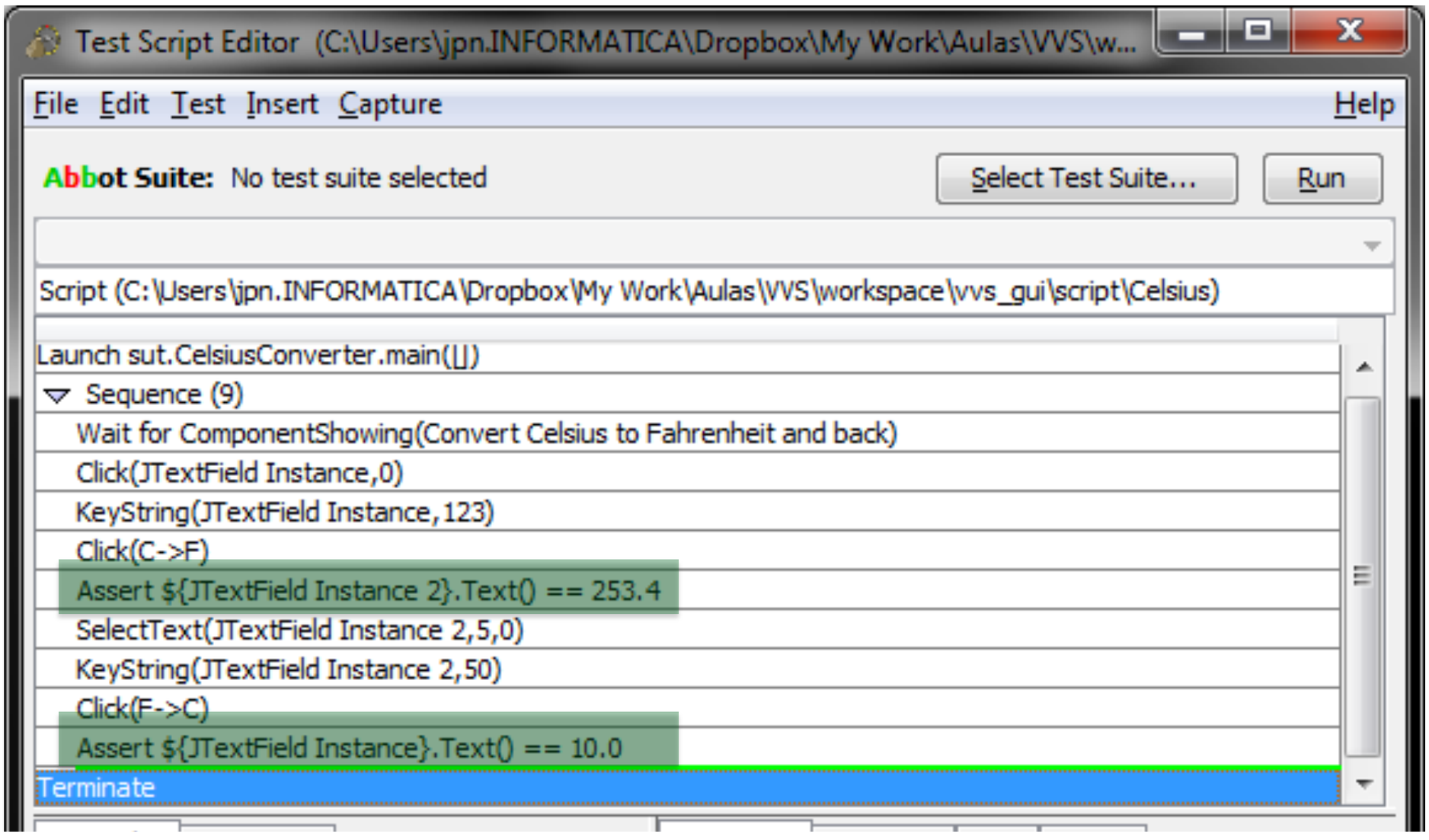
Running Costello inside Eclipse

- Select Capture | All Actions, then the GUI app starts and Costello will record your actions.



Running Costello inside Eclipse

- To include assertions, in the sequence click on an event
  - menu Insert | Assert | Text, and a new assertion will appear
  - select which component is to be checked, and provide the expected value



Running Costello scripts

- We need a test class to execute the script

```
public class CelsiusScriptTest extends ScriptFixture {

  public CelsiusScriptTest(String filename) {
    super(filename);
  }

  public static Test suite() {
    // all scripts in folder script/ will be executed
    return new ScriptTestSuite(CelsiusScriptTest.class, "script");
  }

  public static void main(String[] args) {
    TestHelper.runTests(args, CelsiusScriptTest.class);
  }
}
```