Getting Started

Installation

All you need to do to use ApprovalTests is simply include the ApprovalTests.jar in your class path. Then use it with your favorite Testing Framework.

Parts of a Test

All tests (unit and otherwise) contain 2 parts:

Do
Verify

ApprovalTests is a way to handle the second part: Verification.
All call's will look about the same. Approvals.approve(objectToBeVerified)

Strings

Let's say you wanted to test if a string was being built correctly.

Do Verify	->	create a string with "Approval" append "Tests" to it Verify the resulting string	->	String s = "Approval"; s += "Tests"; Approvals.*approve*(s);

If you see "ApprovalTests" as the result, simply Approve The Result (see below).

Objects

Let's say you wanted to test a customized StringBuilder was creating text correctly.

Do
Verify
	->	create my String Builder
		append "Approval" to it
		append "Tests" to it
		Verify the object
			->	MyStringBuilder s = **new** MyStringBuilder();
				s.append("Approval");
				s.append("Tests");
				Approvals.*approve*(s.toString());

If you see "ApprovalTests" as the result, simply Approve The Result (see below).
It's important to notice that you will need to create a *useful* instance of the toString() Method for objects you want to use.

Arrays

Let's say you wanted to test an array of Strings

Do Verify	->	create a String Array set 1st index to “Approval” set 2nd index to “Tests” Verify the array	->	String[] s = **new** String[2]; s[0] = "Approval"; s[1] = "Tests"; Approvals.*approve*("Text", s);

Note the use of the Label “Text”. This is needed for
and the resulting approval file will contain them:

Text[0] = Approval
Text[1] = Tests

Again, simply Approve The Result (see below).

Swing / AWT
Let’s say you wanted to test you’ve created a JPanel correctly. (this works for anything that extends java.awt.Component : awt, swing, JFrame, Label, etc...)

Do Verify	->	create a TvGuide select show for 3pm Verify the TvGuide	->	TvGuide tv = **new** TvGuide(); tv.selectTime("3pm"); Approvals.*approve*(tv);

First, I want to note that even though there is a UI and a select box for times, I’m not “poking” it to select the time. Just because we are looking at the UI at the end, doesn’t mean I need to manipulate it directly. We are programmers, and are not limited by the constraints of the UI. I simple expose a selectTime(String value) function.

Second, this will produce a screen shot of the JPanel as a result. Simply Approve The Result when it’s ready(see below).

Approving The Result

When you run a test with an approval, it will generate a file named “YourTestClass.youTestMethod.received.txt” (or png, html, etc) and place it in the same directory as your test.

For the test to pass, this file must match:
YourTestClass.youTestMethod.approved.txt

There are many ways to do this.
1) Rename the .received file
2) run the “move” command that is displayed (also added to your clipboard) in the command line
3) Use “use whole file” on a diff reporter
4) Use the ‘approve’ command with the approval plugin

It's doesn't matter how you do it.
Note: if the files match, then the received file will be deleted.
Note: you must include the received files in your source control.

Reporters

If an approval fails, then a report will be called that will report the ".received" and ".approved" files. There are many reporters, and you can create your own.

The simplest way to have your reporter called is to use the Annotation
@UseReporter(Reporter.**class**)
you can annotate at either the method or class level.

Here are some common Reporters and uses

DiffReporter	Launches an instance of TortoiseSvnDiff
FileLauncherReporter	Opens the .received file
ImageDiffReporter	Launches an instance of TortoiseSvnImageDiff
ImageWebReporter	Opens the files in a web browser
JunitReporter	Text only, displays the contents of the files as a AssertEquals failure
NotePadLancher	Opens the .received file in notepad
QuietReporter	outputs the move command to the console. Great for build systems
TextWebReporter	Opens the files in a web browser