http://www.cvshome.org/

http://www.tortoisecvs.org/

Stremler’s remote CVS on Rohan page
http://www-rohan.sdsu.edu/~stremler/CS530/AS1/remote_cvs.html

UCB InfoSys 255 course information on CVS & Eclipse
http://www.sims.berkeley.edu/academics/courses/is255/f04/labs/lab090904/CVSHowTo/eclipseCvsHowTo.html

CVS Eclipse Plug-in FAQ
http://dev.eclipse.org/viewcvs/index.cgi/%7Echeckout%7E/platform-vcm-home/docs/online/cvs_features2.0/cvs-faq.html

Manual
http://www.cvshome.org/docs/manual/

There are a number of books on CVS
Pragmatic Version Control by Thomas & Hunt
http://www.pragmaticprogrammer.com/starter_kit/vc/
Testing & Subversion References

Object-Oriented Design Heuristics, Riel, 1996
JUnit Cookbook http://junit.sourceforge.net/doc/cookbook/cookbook.htm


JUnit Javadoc: http://www.junit.org/junit/javadoc/3.8/index.htm

Brian Marick’s Testing Web Site: http://www.testing.com/

Testing for Programmers, Brian Marick, Available at: http://www.testing.com/writings.html

Main Subversion Website, http://subversion.tigris.org/
Source Control
Why Use Source Control?
## Two Free & Common Source Control Systems

<table>
<thead>
<tr>
<th>CVS</th>
<th>Subversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrent Versions System</td>
<td>Claims to be a better CVS</td>
</tr>
<tr>
<td>Command line interface in Unix</td>
<td>Many commands are same as CVS</td>
</tr>
<tr>
<td>Various interfaces in Window</td>
<td>Runs on:</td>
</tr>
<tr>
<td></td>
<td>Mac OS X</td>
</tr>
<tr>
<td></td>
<td>Unix</td>
</tr>
<tr>
<td></td>
<td>Linux</td>
</tr>
<tr>
<td></td>
<td>Windows</td>
</tr>
<tr>
<td>For remote access need</td>
<td>Run locally or remote</td>
</tr>
<tr>
<td>Client</td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td></td>
</tr>
</tbody>
</table>
Basic Source Control Operations

Starting a new project
Adding code to a project
Modifying existing code
Retrieving past versions of code
Handling conflicts in code
Creating code branches
Merging code branches
CVS on Windows

TortoiseCVS

Nice CVS client and Server for Windows

Adds CVS menu options in Windows Explorer

Can use Rohan as remote server

http://www.tortoisecvs.org/

Eclipse has a GUI interface for accessing CVS repositories

Can access local & remote CVS repositories

See
CVS Eclipse Plugin- FAQ
UCB InfoSys 255 course information on CVS & Eclipse
CVS On Unix
CVS Root on Unix

One Time Only

To store your own files you need a cvs root (repository)

cvs -d cvsLocation init

Example

cvs -d /home/ma/whitney/cvsRoot init

On Rohan cvs commands are in

/usr/local/bin
Starting a New Project

rohan-> mkdir xmlrpcClient
rohan-> cd xmlrpcClient/
rohan-> cvs import -m "start assn1" cs580/xmlrpcClient yoyo start
cvs -d /home/ma/whitney/cvsRoot checkout C/D/Project2

cvs -d /home/ma/whitney/cvsRoot co C/D/Project2

cvs co C/D/Project2 #set CVSROOT

cvs co p2 #Create module
Setting CVSROOT

For tcsh or csh in .cshrc or .login file add a line

    setenv CVSROOT /home/ma/whitney/cvsRoot

For other shells you may need to use:

    set CVSROOT="/home/ma/whitney/cvsRoot"
Creating a CVS shortcut or Module

Get modules file
rohan-> cd ..
rohan-> cvs checkout CVSROOT/modules
rohan-> cd CVSROOT/
rohan-> ls
CVS/ modules

Edit modules file
Add the following line at the end the file
assn1 cs580/xmlrpcClient

Commit the changes
rohan-> cvs commit -m "Added assn1 module" modules

Release the module file
rohan-> cd ..
rohan-> cvs release -d CVSROOT/
Getting assn1 Project

rohan-> **cvs checkout assn1**
rohan-> cd assn1
rohan-> ls
CVS/
Adding Files

Create a file called client.java using an editor

Tell cvs about the file

rohan-> `cvs add -m "main client file" client.java`

Commit any changes

rohan-> `cvs commit -m "added main"

cvs commit: Examining .
RCS file: /home/ma/whitney/cvsRoot/cs580/xmlRpcClient/client.java,v
done
Checking in client.java;
/home/ma/whitney/cvsRoot/cs580/xmlRpcClient/client.java,v <--
client.java
initial revision: 1.1
done
rohan-> **cvs log**
cvs log: Logging.

RCS file: /home/ma/whitney/cvsRoot/cs580/xmlRpcClient/client.java,v
Working file: client.java
head: 1.2
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 2; selected revisions: 2
description:
main client file
-----------------------------
revision 1.2
date: 2002/09/12 22:45:17; author: whitney; state: Exp; lines: +2 -0
added foobar method
-----------------------------
revision 1.1
date: 2002/09/12 22:44:16; author: whitney; state: Exp;
added main

---

**Accessing the log data**

---
Comparing Files

rohan-> **cvs diff -r 1.1 -r 1.2 client.java**
Index: client.java

RCS file: /home/ma/whitney/cvsRoot/cs580/xmlRpcClient/client.java,v
retrieving revision 1.1
retrieving revision 1.2
diff -r1.1 -r1.2
1a2,3
> // main
> // more code
Files by Date

rohan-> **cvs checkout -D yesterday assn1**

rohan-> **cvs checkout -D "2002-9-11 20:00" assn1**

rohan-> **cvs checkout -D "2002-9-11" assn1**

rohan-> **cvs checkout -d "1 hour ago" assn1**
Files by Version

rohan-> cvs checkout -r 1.1 assn1
Creating a Branch

rohan->cvs co assn1
rohan->cd assn1
rohan->cvs tab -b branchA

Branch now created, but is in repository
Must check out branch to work on it

cvs co -r branchA assn1
Merging

Check out a branch

cvs co assn1

Merge another branch

cvs update -j branchA assn1

Now edit files to resolve conflicts
Remote Access to CVS via Unix

For complete instructions see the page by Stewart Stremler: http://www-rohan.sdsu.edu/~stremler/CS530/AS1/remote_cvs.html

Set CVS_RSH

setenv CVS_RSH /usr/bin/ssh

Set the CVSROOT environment variable

The general format is:

:ext:username@servermachine:/absolutePathToCVSRoot

For example I use:

setenv CVSROOT :ext:whitney@rohan.sdsu.edu:/home/ma/whitney/cvsRoot
Remote Access to Rohan CVS using TortoiseCVS

Right-click on Windows Explorer window
Select "CVS Checkout"

Protocol: Secure shell (:ext:)
Server: rohan.sdsu.edu
Port: (keep blank)
Repository folder: /Full/path/to/your/repo/itory
Username: your rohan login

Must select a Module

To avoid having to enter password each time see:
http://www.tortoisecvs.org/faq.html#sshkeys
Unit Testing
Testing

Johnson's Law

If it is not tested it does not work

The more time between coding and testing

More effort is needed to write tests
More effort is needed to find bugs
Fewer bugs are found
Time is wasted working with buggy code
Development time increases
Quality decreases
Unit Testing

Tests individual code segments

Automated tests
What wrong with:

Using print statements

Writing driver program in main

Writing small sample programs to run code

Running program and testing it be using it
We have a QA Team, so why should I write tests?
When to Write Tests

First write the tests

Then write the code to be tested

Writing tests first saves time

Makes you clear of the interface & functionality of the code

Removes temptation to skip tests
What to Test

Everything that could possibly break

Test values
  Inside valid range
  Outside valid range
  On the boundary between valid/invalid

GUIs are very hard to test
  Keep GUI layer very thin
  Unit test program behind the GUI, not the GUI
Common Things Programs Handle Incorrectly

Adapted with permission from “A Short Catalog of Test Ideas” by Brian Marick,
http://www.testing.com/writings.html

Strings
Empty String

Collections
Empty Collection
Collection with one element
Collection with duplicate elements
Collections with maximum possible size

Numbers
Zero
The smallest number
Just below the smallest number
The largest number
Just above the largest number
XUnit

Free frameworks for Unit testing

SUnit originally written by Kent Beck 1994

JUnit written by Kent Beck & Erich Gamma

Available at: http://www.junit.org/

Ports to many languages at:
  http://www.xprogramming.com/software.htm
JUnit Example

Goal: Implement a Stack containing integers.

Tests:
   Subclass junit.framework.TestCase
   Methods starting with 'test' are run by TestRunner

import junit.framework.*;
public class TestStack extends TestCase {

   public void testDefaultConstructor() {
      Stack test = new Stack();
      assertTrue("Default constructor", test.isEmpty() );
   }

   public void testSizeConstructor() {
      Stack test = new Stack(5);
      assertTrue( test.isEmpty() );
   }

}
Start of Stack Class

public class Stack {
    int[] elements;
    int topElement = -1;

    public Stack() {
        this(10);
    }

    public Stack(int size) {
        elements = new int[size];
    }

    public boolean isEmpty() {
        return topElement == -1;
    }
}
Running JUnit Using Eclipse

After creating your Stack Class

Select JUnit TestCase in Create Icons Menu
Running JUnit Using Eclipse
Fill in dialog window & create the test cases

Select Junit test case from the "Run as..." menu
Assert Methods

assertTrue()
assertFalse()
assertEquals()
assertNotEquals()
assertSame()
assertNotSame()
assertNull()
assertNotNull()
fail()

For a complete list see

Testing the Tests
If can be useful to modify the code to break the tests

package example;

public class Stack {
    int[] elements;
    int topElement = -1;

    etc.

    public boolean isEmpty() {
        return topElement == 1;
    }
}

Test Fixtures

Before each test setUp() is run
After each test tearDown() is run

package example;
import junit.framework.TestCase;

class StackTest extends TestCase {
    Stack test;

    public void setUp() {
        test = new Stack(5);
        for (int k = 1; k <= 5; k++)
            test.push(k);
    }

    public void testPushPop() {
        for (int k = 5; k >= 1; k--)
            assertEquals("Pop fail on element " + k, test.pop(), k);
    }
}

Testing Exceptions

```java
public void testIndexOutOfBoundsException() {
    ArrayList list = new ArrayList(10);
    try {
        Object o = list.get(11);
        fail("Should raise an IndexOutOfBoundsException");
    } catch (IndexOutOfBoundsException success) {}  
}
```

Example is from the JUnit FAQ