Homework #2

• Read and critique
  – AbstractMapBag, HeapBag and TreeBag
  – from org.apache.commons.collections.bags
  – part of the apache jakarta commons collections project
  – http://commons.apache.org/collections/

• Bring in source code printouts with your notes on them

• Due next Tuesday, Sept 25th, in class
  – be prepared to discuss them
  – be prepared for a quiz on them
Source code control

• Way to store all versions of the files associated with a project or effort
• Lots of different version control systems
  – CVS, Subversion, Bitkeeper, GIT, Perforce, Mercurial, Microsoft SourceSafe
• CVS is one of the older ones
  – showing its age
• Subversion is the most widely used and available of the newer systems
  – need to add plugin to Eclipse to support Subversion
Straight line development
Multiple checkouts
Commit conflicts
Merging
CVS

- CVS is a kludge built on top of an even older source code control system, RCS
- When you commit changes to 10 files
  - it is performed as ten different commits
  - Nothing records the fact that these 10 changes were all committed at the same time
- Each file has its own version number
  - e.g. WebServer is version 1.9, MockSocket is 1.10, MySocket is 1.4
Atomic commits

• In subversion (and most other more modern systems), there is one version number for an entire repository
  – For FindBugs, there is one repository that holds FindBugs, test cases, the eclipse plugin and the bluej plugin... All separate buildable projects

• Each commit increments the version number
  – The commit has a unique identifier
Atomic commits

• CVS locks the repository while being accessed
  – so you are still guaranteed to see an atomic checkout; you can’t interleave a commit and an update, getting old versions of some files and new versions of others

• But much easier to name versions
  – compare two different versions
  – check out and build each atomic commit
Branches

• OK, you are working on the newest cool features
• But all of a sudden, you get a critical error report from a customer
  – Customer has version 1.0
    • you did a release, correct?
  – You are currently working on version 1.1
  – You are in the middle of a bunch of changes, it will be months before 1.1 is ready to ship
  – You need to release a patched version of 1.0
    • need to send it through Q/A, might take a few iterations to get right
    • might not be the last critical patch
Live outside the source code control system?

• Export a copy of the version 1.0 of the system
• Make the changes
• send a tar ball around with the patched code
  – or just send around a patch
• If iterations are needed, just keep updating the patched code
  – external to the source code control system

• No fun; bad, bad idea
Create a branch

• Create the 1.0 branch of your repository
  – Now, any one can either check out of HEAD or from the 1.0 branch.
Trunk, Branches and Tags

• Subversion encourages the following
  – trunk is your primary development version
  – You can create tags, which simply assign names to other versions
    • generally you don’t further evolve tags
  – Branches are snapshots of code that you will continue to evolve
    • will commit changes to

• All just naming conventions
Subversion copying and renaming

- In subversion, you can copy or rename a file or a directory
  - subversion will remember history across that change
  - CVS can’t handle copies or renames

- If you copy a file, it occurs symbolically
  - doesn’t copy all the files
  - but the original and the copy now evolve independently
Branching is just copying

• To make a copy of trunk as branch/v1.2.0

• just copy trunk to branch/v1.2.0

• that is it. Nothing special about branching, just another form of copying
Using subversion

- You can use subversion through the Eclipse plugin
- But you should also get the command line version installed
- Some features are only available through the command line interface
  - the features you need all the time are available through Eclipse
You can use subversion with a file system

- Subversion allows you to use a local file system as your repository
  - great for getting started
- Subversion also comes with a server, and you can install a apache plugin into a web server
Merging

• If you’ve made changes to a file
• and someone else has updated the repository
• You need to update/merge before you can commit
• If you’ve changed disjoint parts of the file, the merge is straightforward
  – but might generate something with a compile error
• If you’ve changed overlapping lines, you’ll need to manually merge those lines
• If someone reformatted the source code, god help you
Subversion recognizes binary files
You can’t merge changes to binary files
You may wish to lock binary files if you expect to change them
You can lock text files too, but generally it isn’t needed
  – Assuming commits are reasonably small and prompt, merges aren’t too bad
Moving changes from one branch to another

• Say you’ve fixed critical issues in version 1.0
• But the same problems also exist the current head
• You can use svn merge to move changes from a branch back to head
  – or to another branch

• You have to manually specify a lot
  – take the changes between versions 8127 and 8301 in the REL-1.0 branch and apply them to my current checked out version of the code
What I want
Tree based commits

- Supported by BitKeeper, GIT, some others
- but not Subversion
- coming in Subversion 1.5?