CMSC 131

Object-Oriented Programming I

Course Software

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane

Tools for Writing Programs

- Good old days ☺
  - Text Editor → create source code files
  - Compiler → generate executables from source
  - Debugger → trace programs to find errors
  - Cycle between editing, compiling, running, debugging
- Today, IDE (Integrated Development Environment)
  - Editor, compiler, debugger, version control rolled in one
- IDE Examples
  - Eclipse → Free!! What we will use
  - Visual Studio
  - NetBeans
  - Others

Eclipse Fundamentals

- Eclipse IDE allow us to create, edit, compile, run, debug, programs
- Eclipse can use for several languages; in this class Java
- Eclipse installation
  - http://www.cs.umd.edu/eclipse/EclipseTutorial/install.html
- Eclipse tutorial
  - http://www.cs.umd.edu/eclipse/EclipseTutorial/index.html

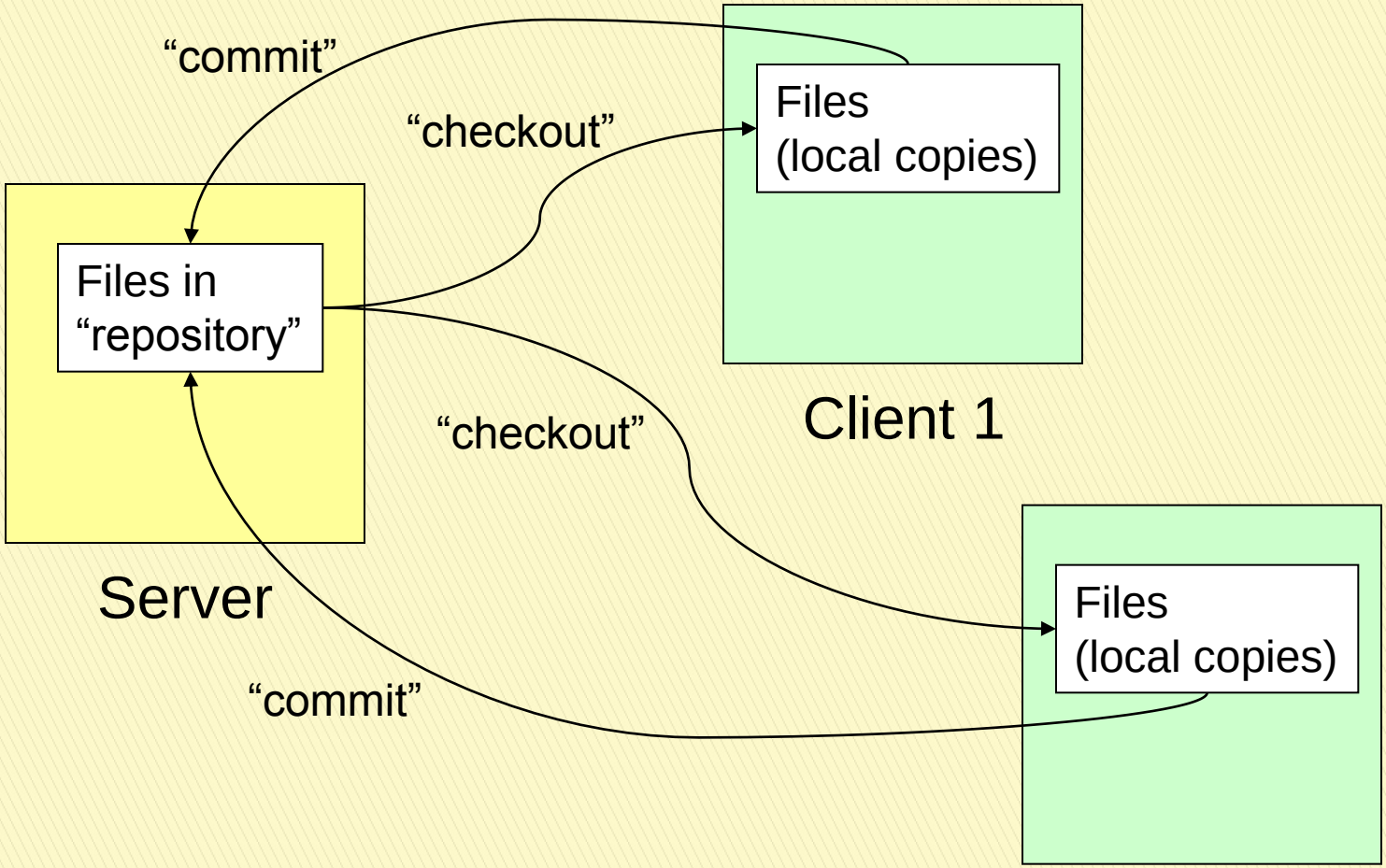
Eclipse Terminology

- Let's run Eclipse
- Workspace → folder/directory where your files are stored
  - You can switch workspaces
- Projects
  - Project → collection of related files
  - Creating a program in Eclipse requires creating a project
  - Let's create a project
- Perspective
  - Framework for viewing/manipulation programs
  - Three important perspectives
    - Java → creating, running programs
    - Debug → tracing, removing errors in programs
    - CVS repository → accessing/managing project files
  - Let's change perspectives
  - Resetting perspective
- Let's create a program

CVS (Concurrent Versions System)

- What is CVS?
  - Tool for project management
  - Maintains versions, etc.
  - Allows different sites to work on same project
- We will use CVS in this class to
  - Obtain (checkout) files we provide for a project
  - Save (commit) files you have worked on
  - Submit your homework
  - Work from different locations

CVS (Concurrent Versions System)



CVS (Concurrent Versions System)

- CVS server maintains current versions of files in “CVS repository”
- To access files from another machine (“client”) repository files must be “checked out”
- Changes to files on client may be “commited” to server, with changed files becoming new version
- We have created a “CVS repository” for each of you. The linuxlab account provides access to this repository
- You need to connect to the “CVS repository” from Eclipse
  - You only need to do this once a semester from every computer you plan to use.
  - How to connect? Let’s see the “CVS Repository Account” section in Resources (class web page)

CVS (Connecting/Checking Out)

- Let's connect to the "CVS Repository"
  - We need to be in the CVS perspective
  - Let's gather our linuxlab account information from the grades server
  - Let's complete the CVS form
- Let's check out the first class project
  - Let's go to the Java Perspective to see the project
  - Make sure you are in the Java Perspective to edit the project
- Each time you save a changes will be send to the CVS repository
- How to submit?
  - Right-click on the project and select "Submit Project"
  - This will send a copy of your project to the Submit Server

Submit Server

- Submit server → https://submit.cs.umd.edu/
- When you select "Submit Project" a copy a sent to the submit server
- Submit server allows you to see results of tests, among other things
- Different kind of tests
  - Public tests – Provided with your project distribution
  - Release tests –
    - You can retrieve limited feedback about tests
    - You have a limited number of opportunities to see the results (e.g., 3 tokens in a 24 hour period)
- Let's submit a project and look at the submit server
  - When project does not compile
  - When project does not generate expected results
  - When project generates expected results