CSE 332: Data Structures and Parallelism Autumn 2018

Setting Up Your CSE 332 Environment

This document guides you through setting up Eclipse for CSE 332. The first section covers using `gitlab` to access and update your project repos (required). The second section introduces Viz for Eclipse (optional), which allows you to visualize your programs and aids in debugging your projects. The third section provides setup instructions for Saros (optional), a tool which can be useful for online collaboration when peer programming. If you run into any problems or questions when setting up or using any of these plug-ins, feel free to ask on Piazza or during office hours!

Part 1: GitLab and Submitting Projects

We will be using `gitlab.cs.washington.edu` to submit homeworks and give feedback. `gitlab` is a web-based `git` hosting service which is similar to `github` but hosted locally by the CSE Department. You will need to learn basic `git` to be able to work on and submit your homework.

Downloading Eclipse

The first thing you should do is download **the newest version** of Eclipse. This is important, because we will be using plugins that were not included in old versions of Eclipse. To download Eclipse, go to:

`https://www.eclipse.org/downloads/packages/release/2018-09/r/eclipse-ide-java-developers`

Generating an SSH Key

The first time you clone from `gitlab`, you will need to create an *SSH key*. To do this, follow these steps:

(1) Open Eclipse Preferences and type "ssh" into the box, then select the "SSH2" pane.

(2) Click the "Key Management" tab and click "Generate RSA Key".

(3) Type your `UWNetID` into the "Comment" box.

(4) Click "Save Private Key" and choose somewhere safe.

(5) Copy the text starting with "ssh-rsa" in the box and go to

`https://gitlab.cs.washington.edu/profile/keys`

(6) Paste the text you copied into the box on `gitlab`, choose a title for the key, and click "Add key".

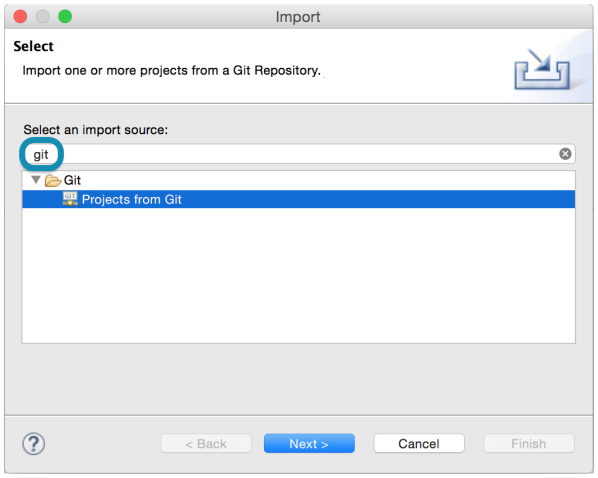
(7) You have now installed an SSH key, and you're ready to use `git`!

Creating The Eclipse Project

Eclipse uses the concept of *projects* to organize your files. Each project in the course will have its own project in Eclipse. We will create the project in Eclipse by cloning a `git` repository on `gitlab` which contains the starter code. To do this, follow these steps:

(1) Go to `File > Import` in the Eclipse menubar.

(2) Type "`git`" into the box; continue by choosing "Projects from Git" and "Clone URI":



Import

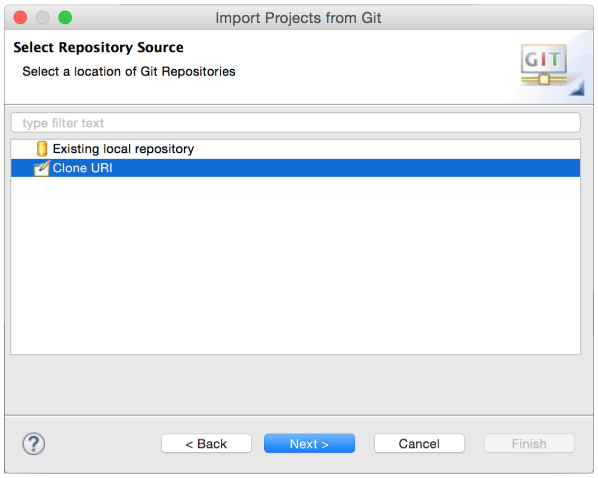
Select

Import one or more projects from a Git Repository.

Select an import source: git

Git

Projects from Git



Import Projects from Git

Select Repository Source

Select a location of Git Repositories

type filter text

Existing local repository

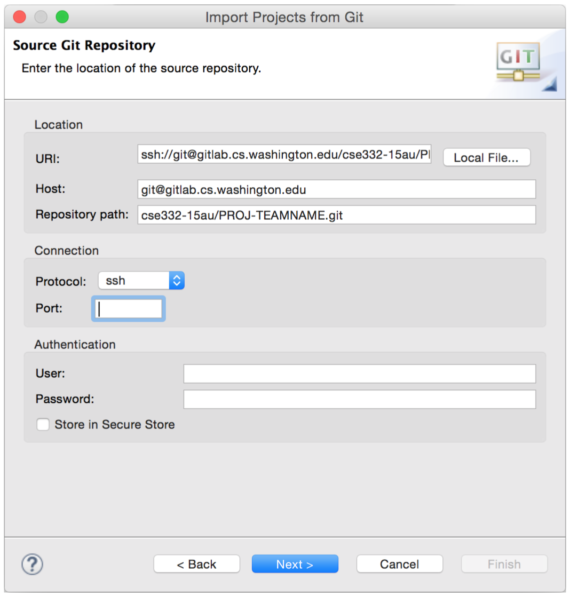
Clone URI

(3) The next dialog will ask you to enter the source of your `git` repository.
Ignore the URI field, and fill in the following fields:

- **Host:** `git@gitlab.cs.washington.edu`
- **Repository Path:** `cse332-`**18au**`/p1-`**bulbasaur**`.git`
- **Protocol:** `ssh`

A **bold** phrase in the repository path means you should substitute it with your equivalent.

Do *not forget* the `.git` at the end!



Import Projects from Git

Source Git Repository

Enter the location of the source repository.

Location

URI: ssh://git@gitlab.cs.washington.edu/cse332-15au/Pl

Host: git@gitlab.cs.washington.edu

Repository path: cse332-15au/PROJ-TEAMNAME.git

Connection

Protocol: ssh

Port:

Authentication

User:

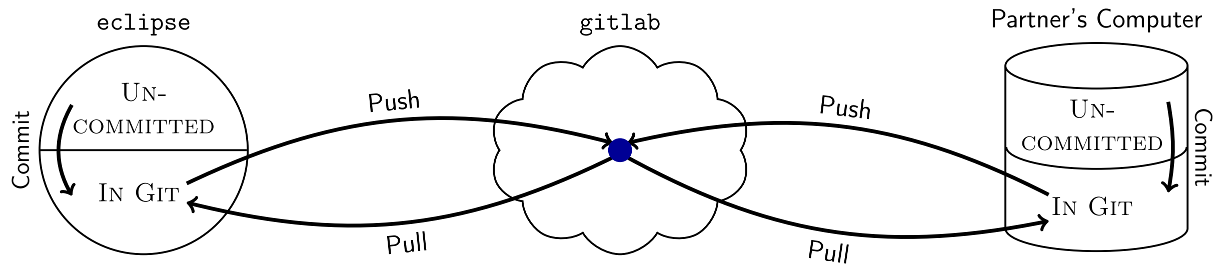
Password:

Store in Secure Store

(4) Click "Next" repeatedly, leaving all subsequent settings at their default values.
In particular, make sure to leave the "Wizard for project import" selection set to "Import existing Eclipse projects". Finally, click "Finish ", and you should have a new project in Eclipse!

Committing, Pushing, and Pulling

git is the *version control system* (VCS) underlying gitlab. Most real projects are kept in a VCS to avoid losing data, and for the ability to easily revert code back to older versions. Another major reason VCS's are important is that they allow you to effectively work together with other people. They allow you to combine ("merge") several different versions of your codebase together.



As shown in the diagram, there are several major actions you can do with respect to your git repository:

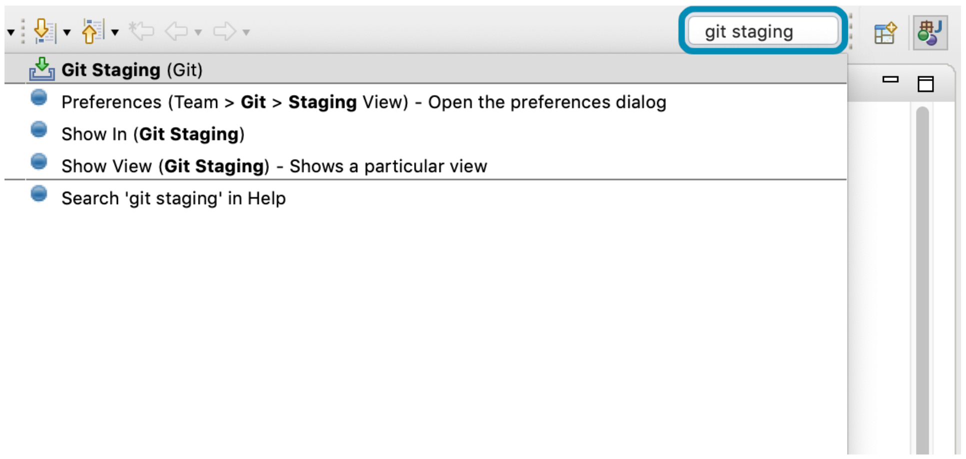
- **Commit:** A "commit" is a set of changes that go together. By "committing" a file, you are asking git to "mark" that it has changed. git requires that you give a message summarizing the effect of your changes. An example commit message might be "Add error handling for the empty queue case in ListFIFOQueue".
- **Push:** A "push" sends your commits to another version of the repository (in our case, this will almost always be gitlab). If you do not push your commits, nobody else can see them!
- **Pull:** A "pull" gets non-local commits and updates your version of the repository with them. If you and someone else both edited a file, git will ask you to explain how to merge the changes together (this process is called "resolving a merge conflict").

Using Git in Eclipse

Eclipse provides a GUI for all of the git operations. We now explain how to handle a git workflow in Eclipse.

Open Git Staging View

Your main tool to use git in Eclipse is called the "staging view". This view allows you to see changed files, make commits, and write commit messages. To open the staging view, type "git staging" into the box labeled "Quick Access" in the top right of the Eclipse window, then select "Git Staging (Git)" from the list of results.



git staging

Git Staging (Git)

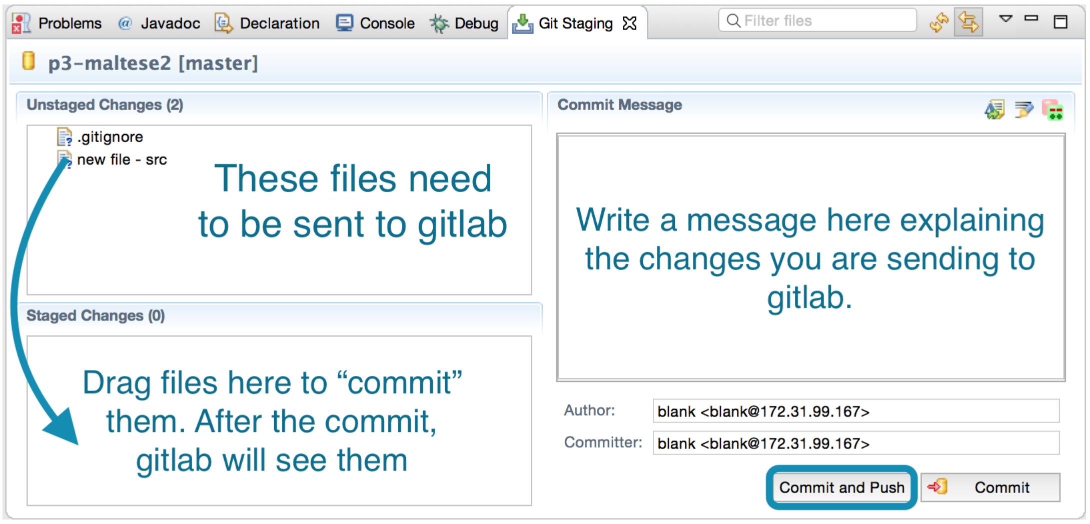
Preferences (Team > **Git** > **Staging** View) - Open the preferences dialog

Show In (**Git Staging**)

Show View (**Git Staging**) - Shows a particular view

Search 'git staging' in Help

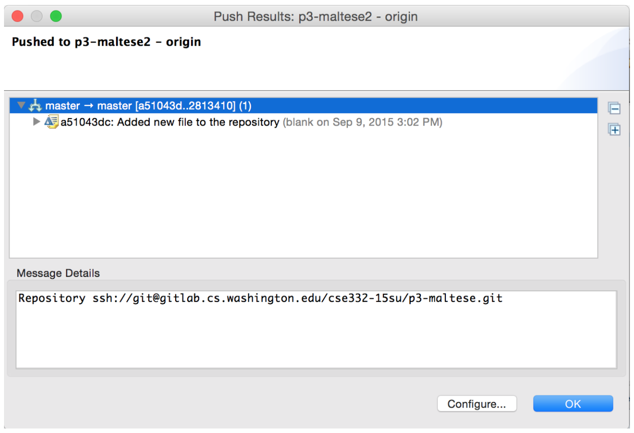
How Staging/Committing Works


Problems
Javadoc
Declaration
Console
Debug
Git Staging
Filter files
p3-maltese2 [master]
Unstaged Changes (2)
.gitignore
new file - src
These files need to be sent to gitlab
Staged Changes (0)
Drag files here to “commit” them. After the commit, gitlab will see them
Commit Message
Write a message here explaining the changes you are sending to gitlab.
Author:
blank <blank@172.31.99.167>
Committer:
blank <blank@172.31.99.167>
Commit and Push
Commit

After Pushing

After you “Commit and Push”, you will get one of several message dialogs:

Success!



Push Results: p3-maltese2 - origin

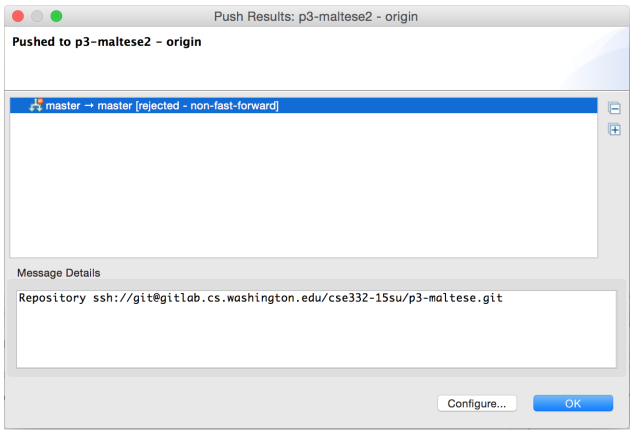
Pushed to p3-maltese2 – origin

master → master [a51043d..2813410] (1)
a51043dc: Added new file to the repository (blank on Sep 9, 2015 3:02 PM)

Message Details

```
Repository ssh://git@gitlab.cs.washington.edu/cse332-15su/p3-maltese.git
```

Rejected!



Push Results: p3-maltese2 - origin

Pushed to p3-maltese2 – origin

master → master [rejected - non-fast-forward]

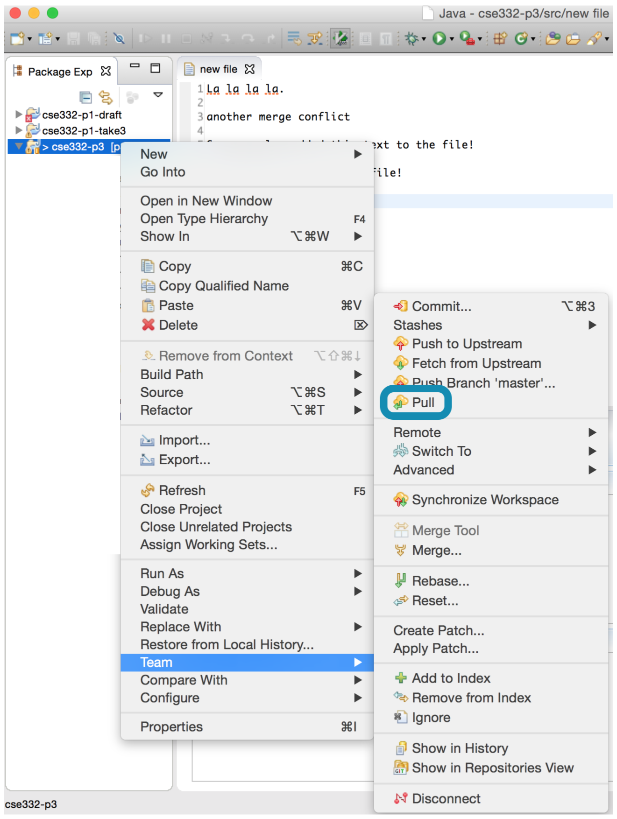
Message Details

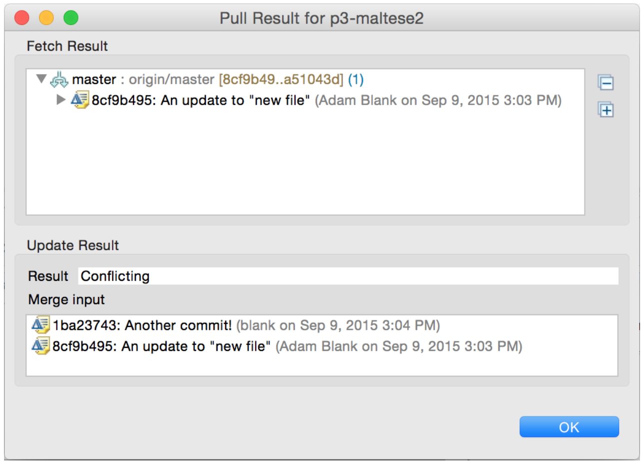
```
Repository ssh://git@gitlab.cs.washington.edu/cse332-15su/p3-maltese.git
```

If you get a “success” message, then your code has been pushed, and you’re good to go. If you get a “rejected” message, it means your partner pushed code; you will need to *pull* before pushing will work. Note that when you pull, you *might* have a merge conflict which you will have to fix before pushing.

Pulling

There are two reasons to pull: (1) you want to get the changes your partner made, or (2) you want to push your changes, but they were rejected because of a conflict.
To pull in Eclipse, click the following menu option (also available as "Pull" in Quick Access):





Pull Result for p3-maltese2

Fetch Result

master : origin/master [8cf9b49..a51043d] (1)
8cf9b495: An update to "new file" (Adam Blank on Sep 9, 2015 3:03 PM)

Update Result

Result Conflicting

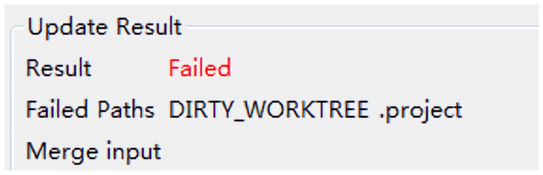
Merge input

1ba23743: Another commit! (blank on Sep 9, 2015 3:04 PM)
8cf9b495: An update to "new file" (Adam Blank on Sep 9, 2015 3:03 PM)

OK

The result will either indicate that you pulled cleanly or that there is a merge conflict.
The best way to tell if you currently have a conflict is to look for the red diamond icon in the list of files.

If you see a `DIRTY_WORKTREE` error when attempting to pull, then you have uncommitted local changes which conflict with changes on `gitlab`.



Update Result

Result Failed

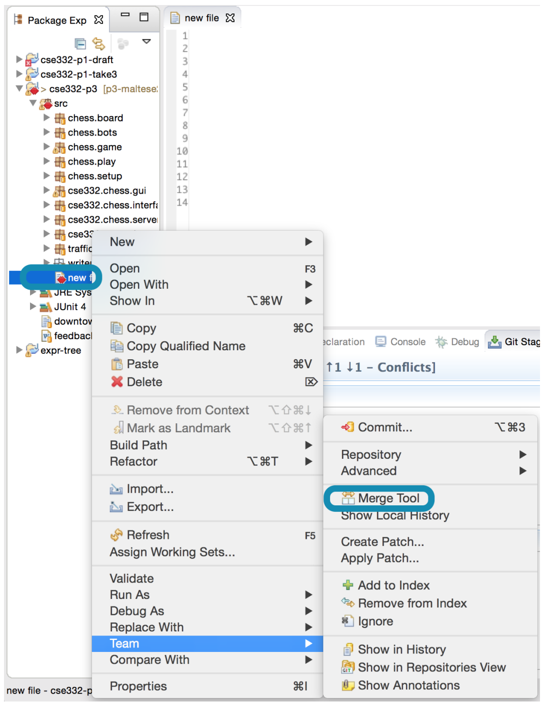
Failed Paths DIRTY_WORKTREE .project

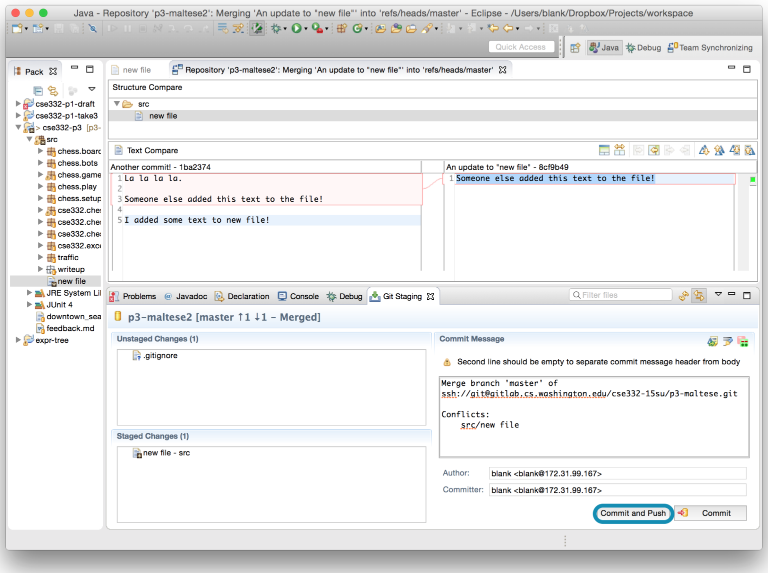
Merge input

To resolve this error, you should first **commit (but do not push)** all local changes, then **pull**, and *then* **merge any conflicts** using the procedure described on the following page.

Merging a Conflict

If you have any items that have a conflict (red diamond icon), you can fix them using the *merge tool*:




Java - Repository 'p3-maltese2': Merging 'An update to "new file"' into 'refs/heads/master' - Eclipse - /Users/blank/Dropbox/Projects/workspace
Repository 'p3-maltese2': Merging 'An update to "new file"' into 'refs/heads/master'
Structure Compare
Text Compare
Git Staging
p3-maltese2 [master ↑1 ↓1 – Merged]
Unstaged Changes (1)
.gitignore
Staged Changes (1)
new file - src
Commit Message
Commit and Push
Commit

Another commit! - 1ba2374

```
La la la la.

Someone else added this text to the file!

I added some text to new file!
```

An update to "new file" - 8cf9b49

```
Someone else added this text to the file!
```

Second line should be empty to separate commit message header from body

```
Merge branch 'master' of
ssh://git@gitlab.cs.washington.edu/cse332-15su/p3-maltese.git

Conflicts:
    src/new file
```

Author: blank <blank@172.31.99.167>

Committer: blank <blank@172.31.99.167>

The merge tool allows you to see your changes (on the left) and other peoples' changes (on the right). Your job is to make the file on the left be the result you actually want. Once you've done this, drag the merged file to the "staged changes" panel like usual. You will notice that there is an auto-filled commit message about a "merge". Go ahead and "Commit and Push". Now, you've pushed your commits and other people can see them!

Submitting Your Final Version

In some courses, you are asked to "tag" your final commit. We will not ask you to do this. The last commit you make within the deadline will be the one we grade.

Resetting SSH Issues

Sometimes Eclipse will get in a state where SSH cloning of projects does not work (even when you've created a new key from scratch). A superset of the steps required to reset *any* Eclipse SSH issue is:

(a) Remove all previous keys in `Preferences > General > Network Connections > SSH2 > Private Keys`.

(b) Move, rename, or delete the previous SSH2 home folder. For example, if my SSH2 home folder was in `~/.ssh`, I could move it to `~/.ssh2`.

(c) Generate a new SSH key and save it in a *new* location, then add it to `gitlab`.

(d) **Restart Eclipse** (This is important!)

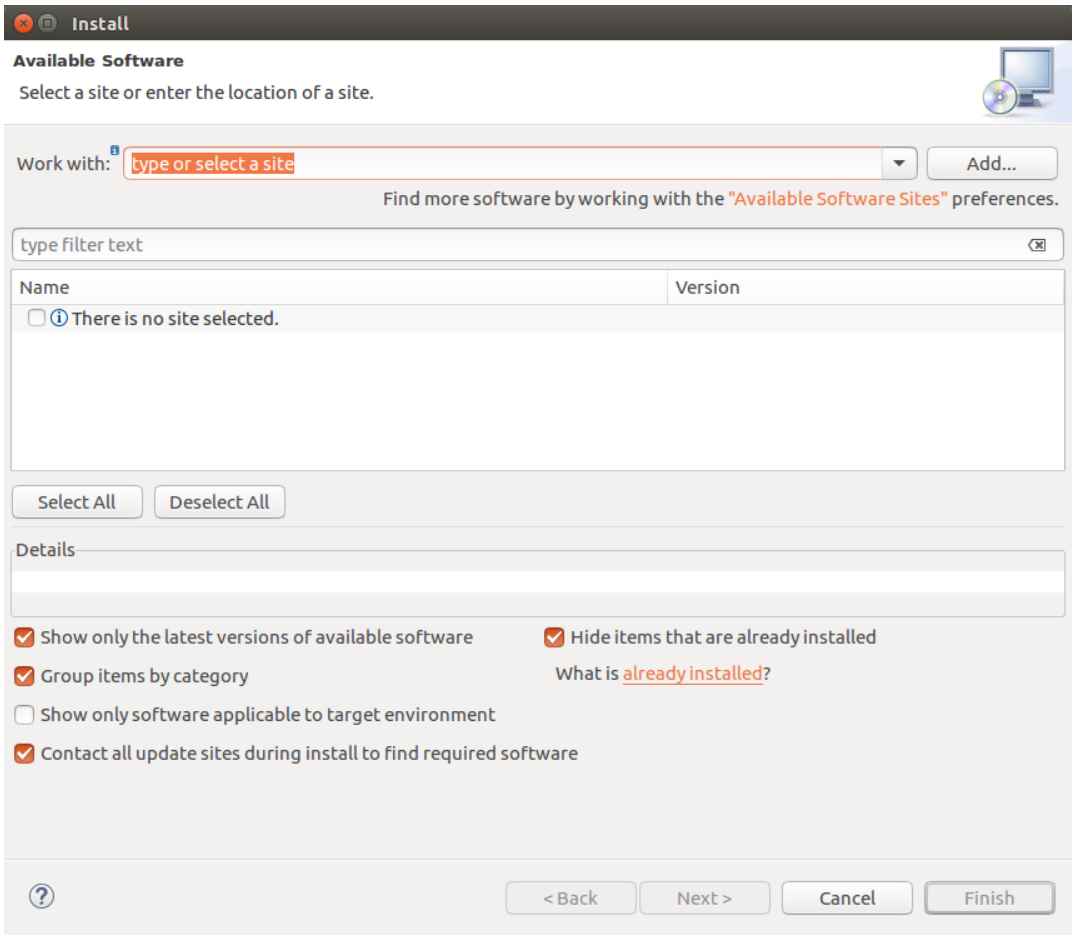
(e) Clone the project in Eclipse using the instructions provided at the top of this document.

Completing these steps should resolve Eclipse SSH issues (which typically surface as a password prompt during the project cloning process).

Part 2: Setting up Viz for Eclipse

This section will help you get started with Viz for Eclipse. Viz allows you to visualize your program's data structures, which can be useful for debugging.

(a) After installing Eclipse (see first section of this document), start Eclipse and go to `Help > Install New Software...`
The following window should appear:



Available Software

Select a site or enter the location of a site.

Work with: type or select a site

Find more software by working with the "Available Software Sites" preferences.

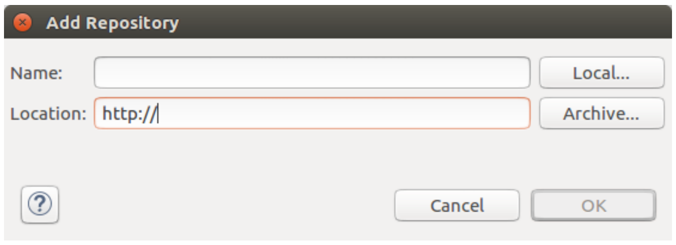
type filter text

There is no site selected.

- [✓] Show only the latest versions of available software
- [✓] Hide items that are already installed
- [✓] Group items by category
- [ ] Show only software applicable to target environment
- [✓] Contact all update sites during install to find required software

What is already installed?

Click on the `Add...` button and you should see the following window appear:



Name:

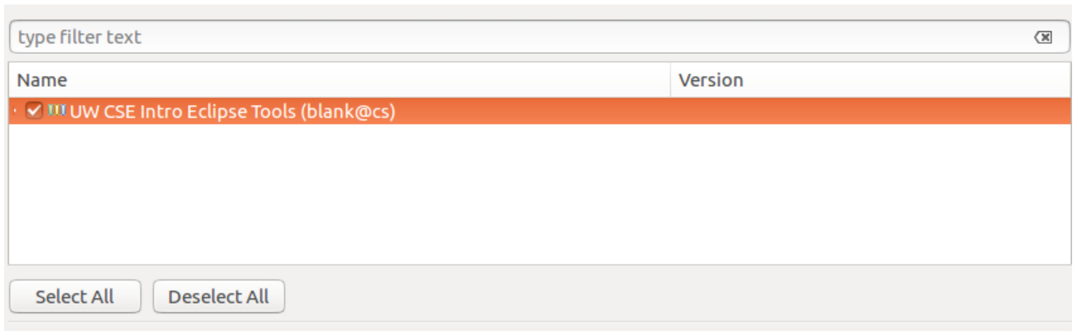
Location: http://

(b) In the input labeled "Location", enter

```
http://courses.cs.washington.edu/courses/cse332/17wi/eclipse/site.xml
```

and click `OK`.

You should then see the following:

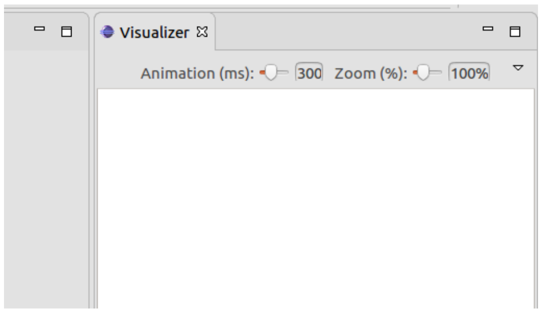


type filter text

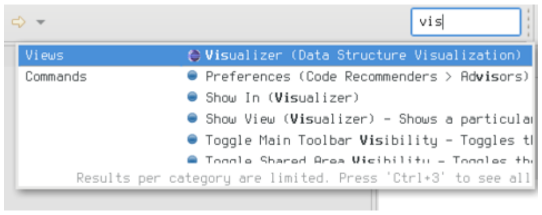
Name	Version
✓ UW CSE Intro Eclipse Tools (blank@cs)	

Make sure that the checkbox is checked, and then click Next twice. Eclipse should then prompt you to confirm license agreements and finish the installation process (do so), after which you should be prompted to restart Eclipse.

(c) After restarting Eclipse, you should see a new tab on the right of the screen for Viz.



This is the "Viz" visualization tool, which will allow you to see the state of your program and data structures during a debugging session.
If you don't see it, type "vis" into the Quick Access bar and wait for a suggestion to pop up. You should see an option to open the Visualization view, as shown below (if you don't, there's probably something else going on, and you should post your issue on Piazza or go to office hours for staff to help):



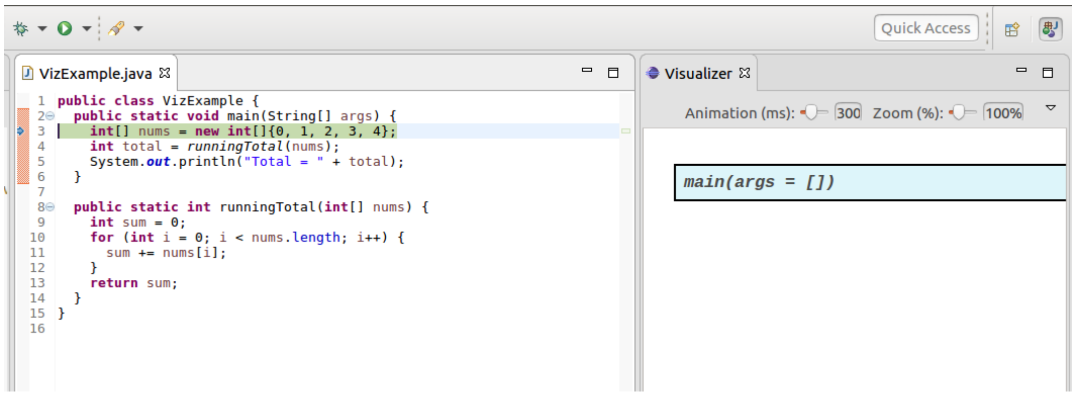
```
vis
Views      Visualizer (Data Structure Visualization)
Commands   Preferences (Code Recommenders > Advisors)
           Show In (Visualizer)
           Show View (Visualizer) - Shows a particula
           Toggle Main Toolbar Visibility - Toggles tl
           Toggle Shared Area Visibility - Toggles th
      Results per category are limited. Press 'Ctrl+3' to see all
```

(d) Viz is now ready to go! To test it out, try writing a small program involving a data structure (e.g., an array of integers) and starting a debugging session. We provide an example walkthrough of a Viz debugging session in the following section.

Example Viz Walkthrough

In this example, we will use Viz to see how a running total of an array of integers is calculated, demonstrating method calls, data structure visualization (for an array, in this case), and loop iteration.

(a) To start a debugging session, click the small bug icon at the top bar of your Eclipse environment (shown in the top-left corner of the screenshot below) and you should then see the following in the Viz pane:

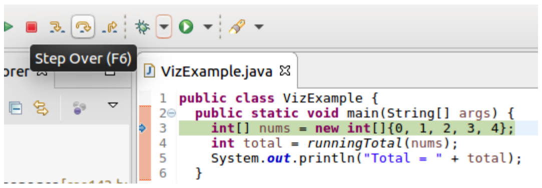


Animation (ms): 300 Zoom (%): 100%

```
1  public class VizExample {
2      public static void main(String[] args) {
3          int[] nums = new int[]{0, 1, 2, 3, 4};
4          int total = runningTotal(nums);
5          System.out.println("Total = " + total);
6      }
7
8      public static int runningTotal(int[] nums) {
9          int sum = 0;
10         for (int i = 0; i < nums.length; i++) {
11             sum += nums[i];
12         }
13         return sum;
14     }
15 }
16
```

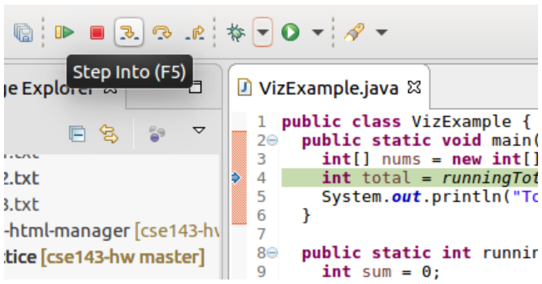
```
main(args = [])
```

(b) To move through a program, you can use the "Step Over" command to step to the next line in the current method, and the "Step Into" command to step into a method at the current line. Here, we want to step over until we get to line 4 where the `runningTotal` method is called.



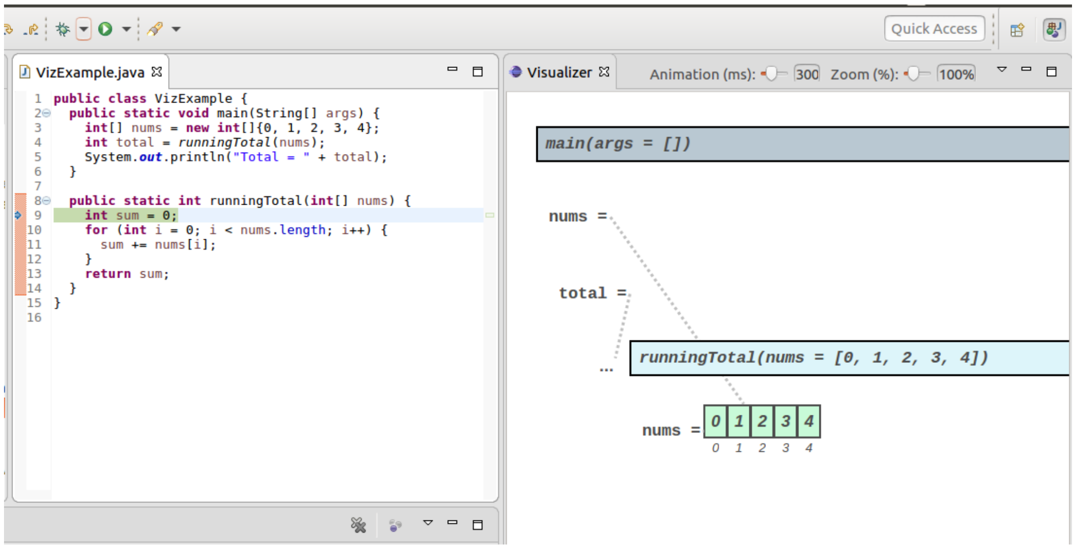
```
1  public class VizExample {
2      public static void main(String[] args) {
3          int[] nums = new int[]{0, 1, 2, 3, 4};
4          int total = runningTotal(nums);
5          System.out.println("Total = " + total);
6      }
```

Once you see the blue arrow (indicating your current line) at line 4, click on the "Step into" command:



```
1  public class VizExample {
2      public static void main(
3          int[] nums = new int[]
4          int total = runningTot
5          System.out.println("Tc
6      }
7
8      public static int runnir
9          int sum = 0;
```

You should then see the following in Viz, indicating that you are now in the `runningTotal` method:



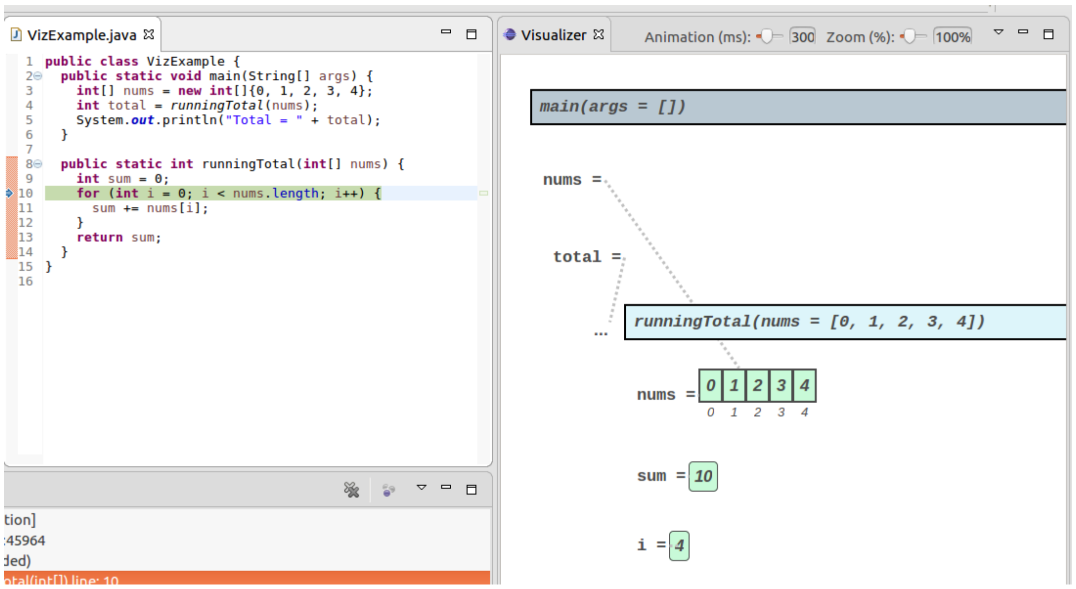
Animation (ms): 300 Zoom (%): 100%

```
public class VizExample {
    public static void main(String[] args) {
        int[] nums = new int[]{0, 1, 2, 3, 4};
        int total = runningTotal(nums);
        System.out.println("Total = " + total);
    }

    public static int runningTotal(int[] nums) {
        int sum = 0;
        for (int i = 0; i < nums.length; i++) {
            sum += nums[i];
        }
        return sum;
    }
}
```

```
main(args = [])
nums =
total =
...
runningTotal(nums = [0, 1, 2, 3, 4])
nums = 0 1 2 3 4
       0 1 2 3 4
```

(c) Now is where you'll really get to see Viz in action! Use the "Step Over" command to iterate through the loop and see the `sum` value update as it is incremented in each iteration. Once you reach the end of your method, the blue arrow will point to the last statement in the method:



Animation (ms): 300 Zoom (%): 100%

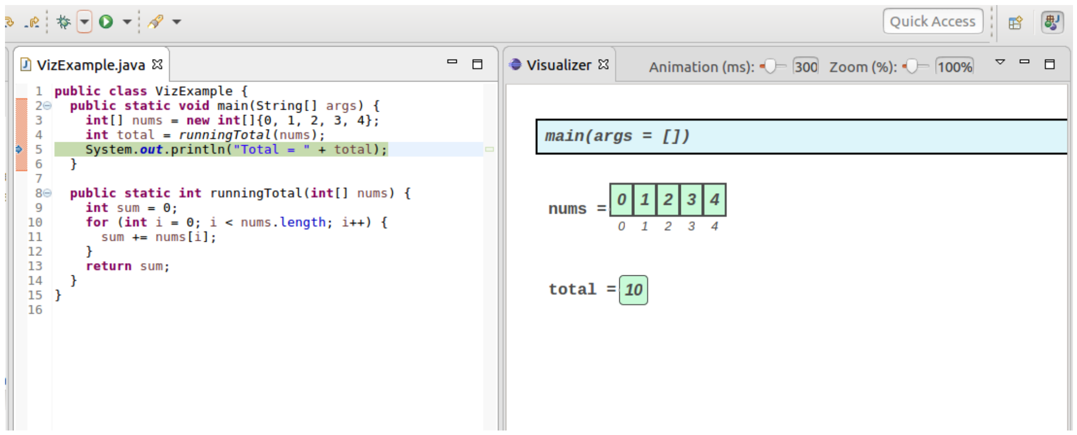
```
public class VizExample {
    public static void main(String[] args) {
        int[] nums = new int[]{0, 1, 2, 3, 4};
        int total = runningTotal(nums);
        System.out.println("Total = " + total);
    }

    public static int runningTotal(int[] nums) {
        int sum = 0;
        for (int i = 0; i < nums.length; i++) {
            sum += nums[i];
        }
        return sum;
    }
}
```

```
tion]
:45964
ded)
otal(int[]) line: 10
```

```
main(args = [])
nums =
total =
...
runningTotal(nums = [0, 1, 2, 3, 4])
nums = 0 1 2 3 4
       0 1 2 3 4
sum = 10
i = 4
```

(d) Once you click "Step over" from here, the method and its variables will be removed from the Viz pane, and Viz will return to the caller of the removed method. In our example, this looks like the following:



```
public class VizExample {
  public static void main(String[] args) {
    int[] nums = new int[]{0, 1, 2, 3, 4};
    int total = runningTotal(nums);
    System.out.println("Total = " + total);
  }

  public static int runningTotal(int[] nums) {
    int sum = 0;
    for (int i = 0; i < nums.length; i++) {
      sum += nums[i];
    }
    return sum;
  }
}
```

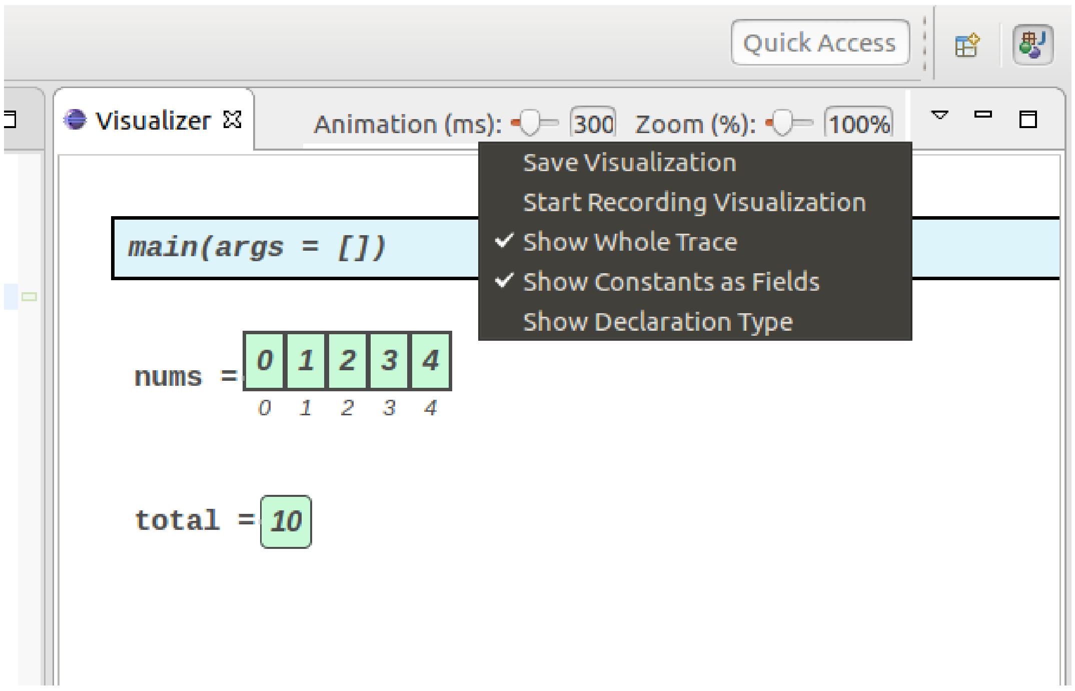
```
main(args = [])

nums = [0, 1, 2, 3, 4]

total = 10
```

From here, you can continue stepping over until you reach the end of `main`, where your debugging session will exit.

We've demonstrated the use of Viz for a simple program, but its value as a debugging tool is most useful when dealing with nested methods and complex data structures.
In addition to helping you visualize programs, Viz also lets you save an image or gif of your visualization at any time when you select the associated option ("Save Visualization" or "Star Recording Visualization"), as shown below:



Additionally, you can change the animation speeds (which affect how fast different data structures update/change references) and zoom in/out using the corresponding sliders on the Viz toolbar. Enjoy experimenting with Viz as you learn new data structures throughout the quarter, and if you run into any questions, you can use Piazza and office hours for additional help!

Part 3: Using Saros for Peer Programming

This guide will help you get started with Saros and using it to peer program with your CSE 332 project partner.

Installing Saros

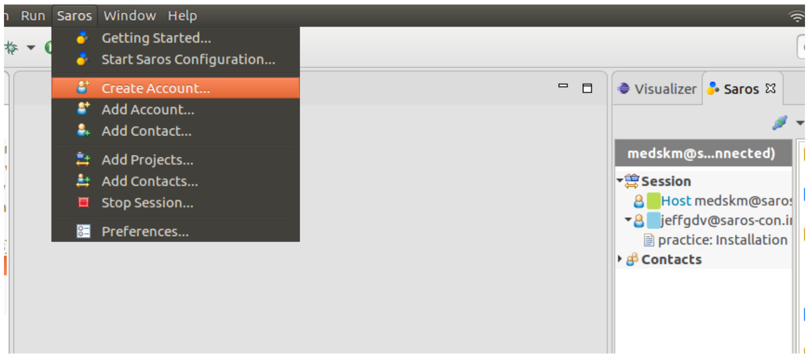
(a) Download Saros from Sourceforge.

(b) Uncompress the downloaded folder, and then use your file viewer to find your Eclipse app. If you see an Eclipse folder, enter it, otherwise if you only see the Eclipse application icon, right click on the Eclipse icon, and select "open package contents" or the equivalent. Inside of the resulting folder, find the "dropins" folder and drag your uncompressed folder into this folder.

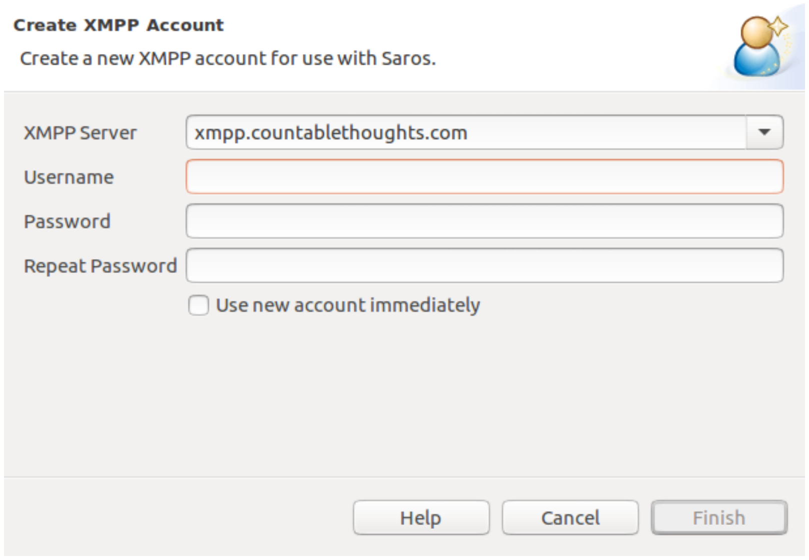
(c) Restart Eclipse. If you don't see a new tab for Saros appear when Eclipse starts up, type in "Saros" in the Quick Access bar. If no results appear, this likely means you need to download the Graphical Editing Framework. To do this, download the most recent version on the linked page, uncompress the downloaded folder, and drag this folder in the "dropins" folder as you did with the Saros download, and restart Eclipse.

Using Saros

(a) After restarting Eclipse, open Saros (either auto-populated in an Eclipse tab or by typing "Saros" in the Quick Access bar). Once you have the Saros tab open, you will need to create an account to use Saros. On the Saros option in the top menu bar, select "Create Account":



You should see the following window appear. Fill in the information to create your account, entering "saros-con.imp.fu-berlin.de" as the server name and providing your student username and a password of your choice. [Note: The screenshots that follow incorrectly refer to xmpp.countablethoughts.com, use saros-con.imp.fu-berlin.de instead.]Make sure to select the "Use new account immediately" option to prevent having to log in again, and click "Finish".


Create XMPP Account

Create a new XMPP account for use with Saros.

XMPP Server: xmpp.countablethoughts.com

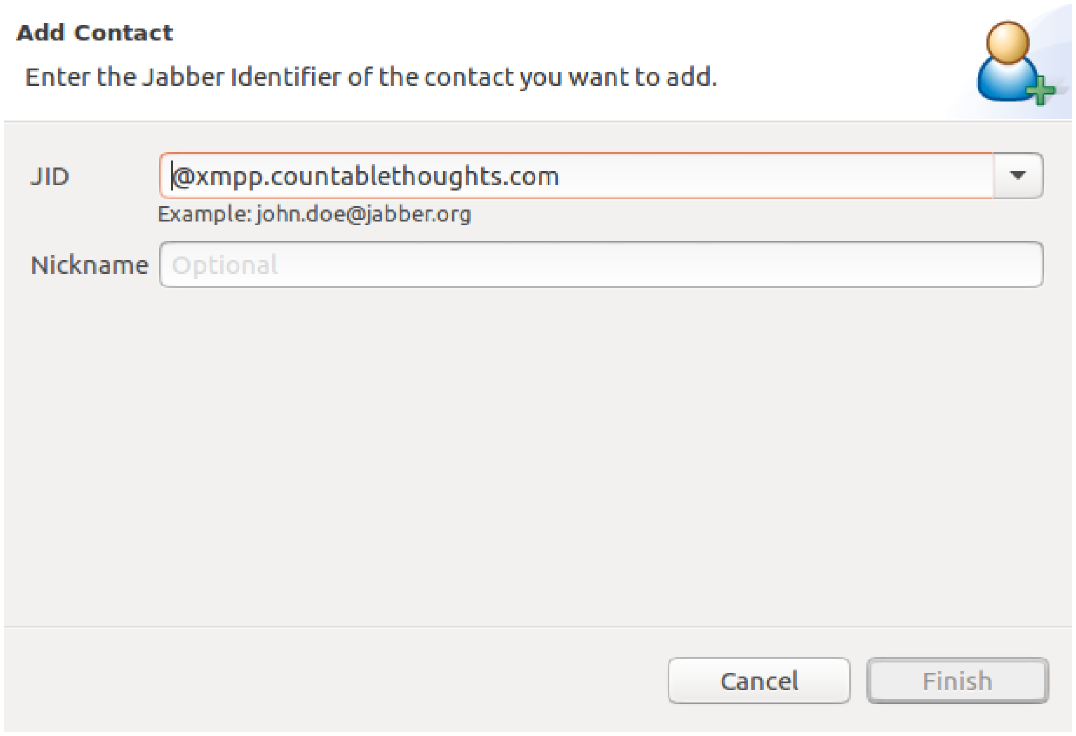
Username:

Password:

Repeat Password:

☐ Use new account immediately

(b) Now you will need to add your project partner as a Saros contact to enable real-time collaboration. On the same Saros menu bar from Step a, select "Add contact". In the "Add Contact" window that appears (see below), add your project partner's username to the left of the @-symbol in the "JID" text area.


Add Contact

Enter the Jabber Identifier of the contact you want to add.

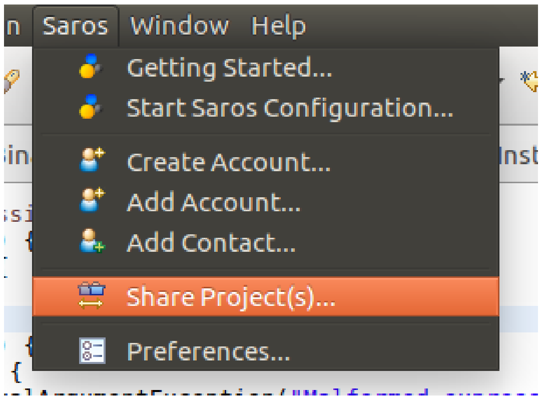
JID: @xmpp.countablethoughts.com

Example: john.doe@jabber.org

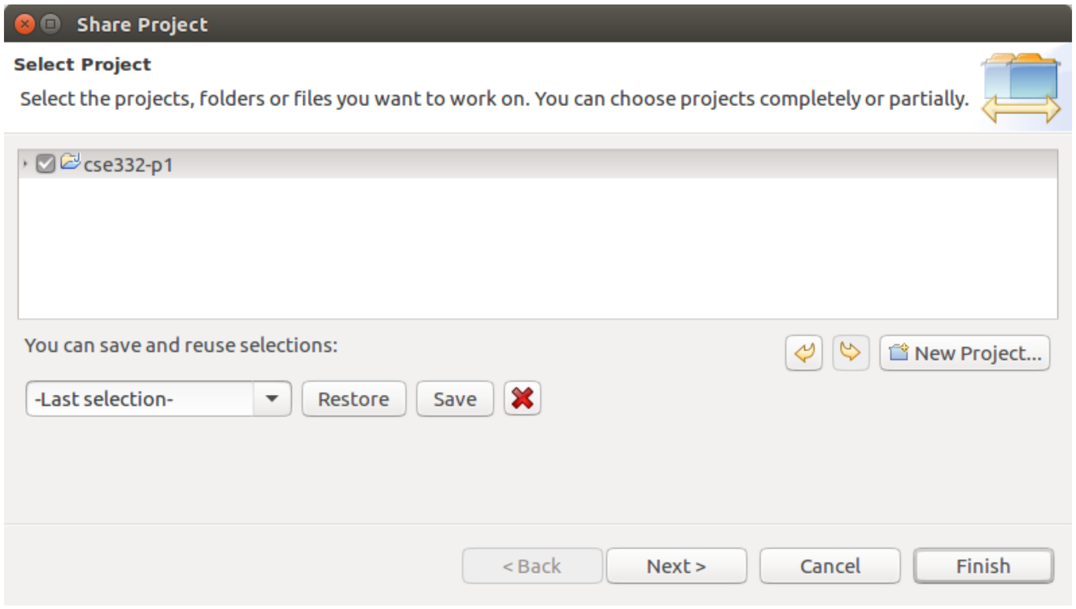
Nickname: Optional

Once you select "Finish", they should receive a contact request on their Eclipse app, which they should accept (only one of you needs to send a contact request for you both to be contacts).

(c) Once you have added your partner as a contact, you will want to add a project to Saros so that you can peer-code with your partner. To do this, first open up the Saros menu on the top menu bar and select "Share Project...".



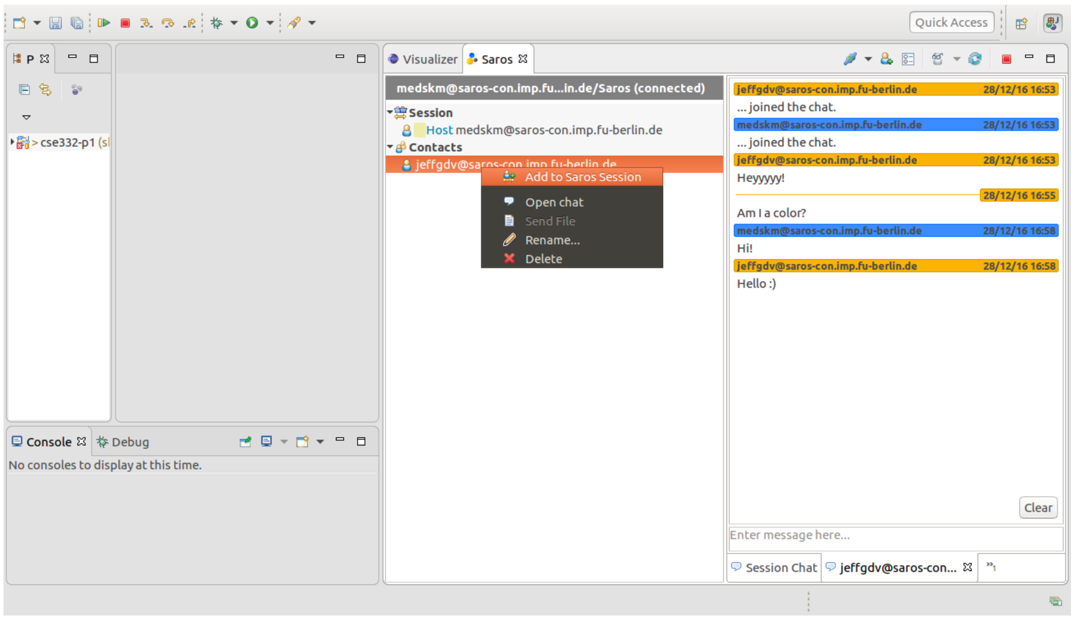
All projects in your current Eclipse workspace will appear, and you should select the one that corresponds to your project repo.



Select the projects, folders or files you want to work on. You can choose projects completely or partially.

You can save and reuse selections:

(d) Once you've shared your project, you will still need to add contacts to that project. To do so, right-click on your partner's name in the Saros window and click "Add to Saros Session" (see below). Your partner will have to accept your invitation on their eclipse screen. If your partner is already in a Saros Session, you will receive an error message.



No consoles to display at this time.

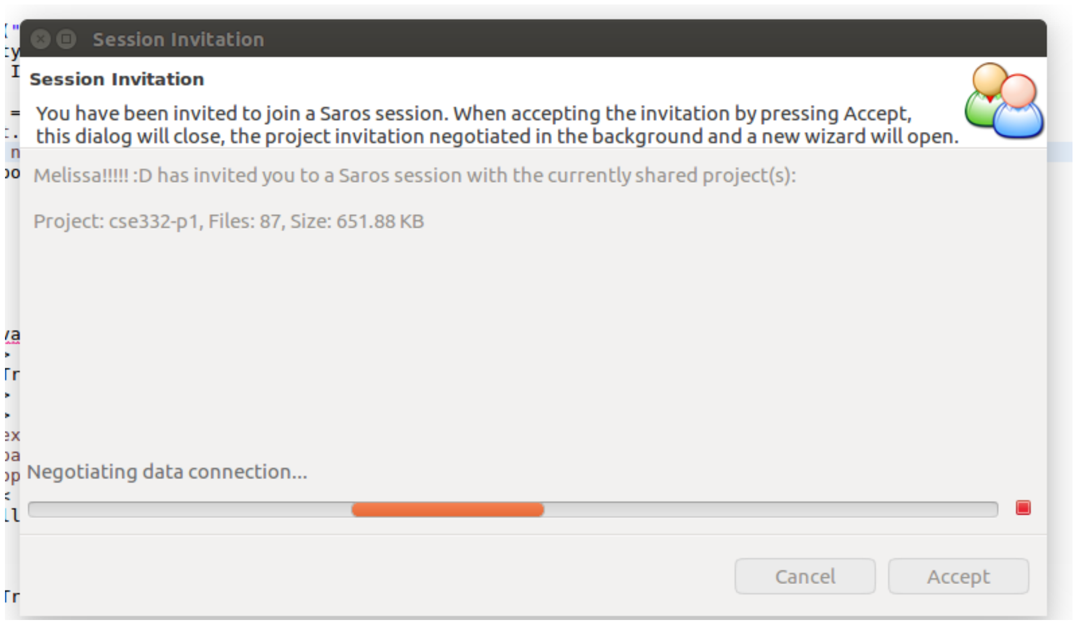
medskm@saros-con.imp.fu...in.de/Saros (connected)

Host medskm@saros-con.imp.fu-berlin.de

jeffgdv@saros-con.imp.fu-berlin.de

jeffgdv@saros-con.imp.fu-berlin.de 28/12/16 16:53
... joined the chat.
medskm@saros-con.imp.fu-berlin.de 28/12/16 16:53
... joined the chat.
jeffgdv@saros-con.imp.fu-berlin.de 28/12/16 16:53
Heyyyyy!
28/12/16 16:55
Am I a color?
medskm@saros-con.imp.fu-berlin.de 28/12/16 16:58
Hi!
jeffgdv@saros-con.imp.fu-berlin.de 28/12/16 16:58
Hello :)

Enter message here...


Session Invitation

Session Invitation

You have been invited to join a Saros session. When accepting the invitation by pressing Accept, this dialog will close, the project invitation negotiated in the background and a new wizard will open.

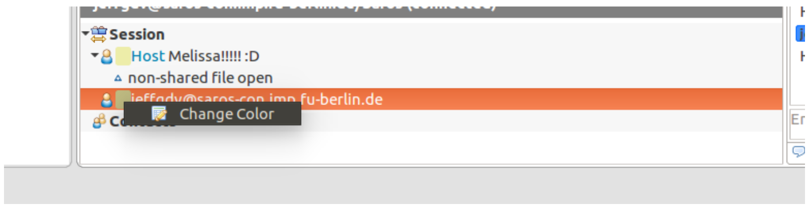
Melissa!!!!! :D has invited you to a Saros session with the currently shared project(s):

Project: cse332-p1, Files: 87, Size: 651.88 KB

Negotiating data connection...

(e) The Add Project menu will pop up once both partners accept their invitations. On that screen, simply give a name to the project on which you are collaborating. Once both partners select "Finish" in that menu and shared files are transferred, collaboration can begin.

(f) Your partner and their cursor on the project will be identified as the color next to their name. If you wish to change the color used to indicate which partner is writing what, simply right-click on the color icon under your name in the Saros menu and click on "change color" (see below):



(g) Start peer programming! When you are finished with your programming session, you can exit the session with the "Stop session" icon (the red square on the Saros tab). The student who shared their project space on the session should be the one committing and pushing the project changes to your group's repo, and the other partner should then pull the updated version to their local repository.

If you would like to learn more about the different features of Saros, you can find helpful "Getting Started" information here!