

## Setting Up Your CSE 332 Environment

In this document, we will provide information for 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 aid 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 Submission

We will be using [gitlab.cs.washington.edu](https://gitlab.cs.washington.edu) to submit homeworks and give feedback. gitlab is a version of github that is local to the CSE Department. As such, 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/eclipse-ide-java-developers/neon2>

#### Generating and Installing a Private Key

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

- (1) Open Eclipse Preferences and type “ssh” into the box.
- (2) Click the “Key Management” tab and click “Generate RSA Key”.
- (3) Type your UWNetID into the “Comment” box.
- (4) Then, click “Save Private Key” and choose somewhere safe.
- (5) Copy the text starting with “ssh-rsa” in the box and go to

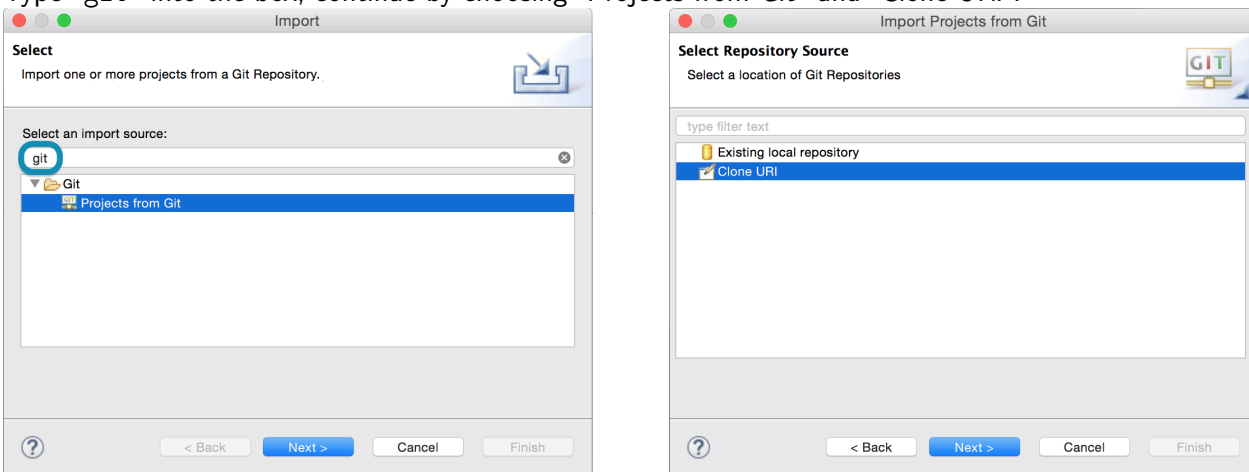
<https://gitlab.cs.washington.edu/profile/keys>.

- (6) Click “Add SSH Key”, choose a name for the key, and paste the text you copied into the box.
- (7) You have now installed an RSA public key, and you’re ready to use git!

## Creating The Eclipse Project

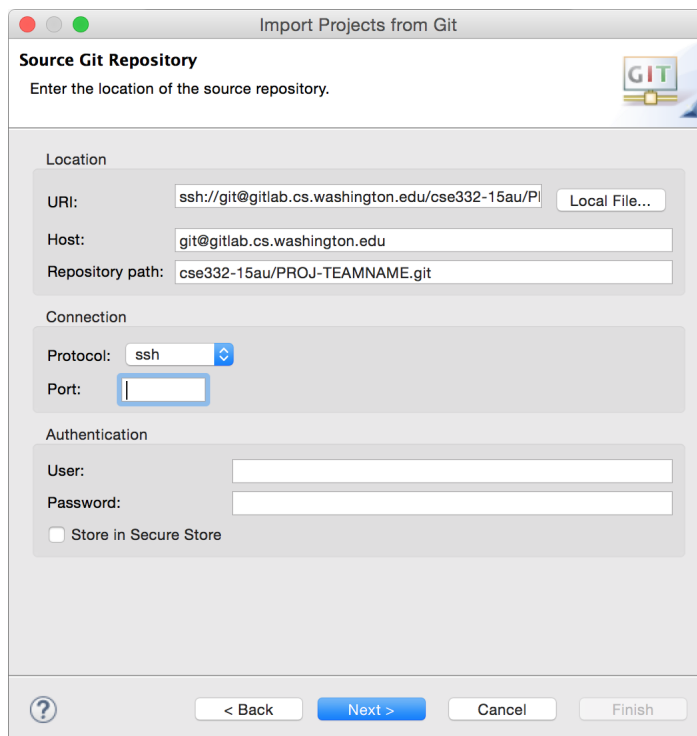
Eclipse uses a concept of *projects* to organize your files. Each project in the course will have its own project. We will create the project by using the template gitlab repository. To do this, follow these steps:

- (1) Go to File > Import
- (2) Type “git” into the box; continue by choosing “Projects from Git” and “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:** cse332-15au/p1-bulbasaur.git
  - **Protocol:** ssh

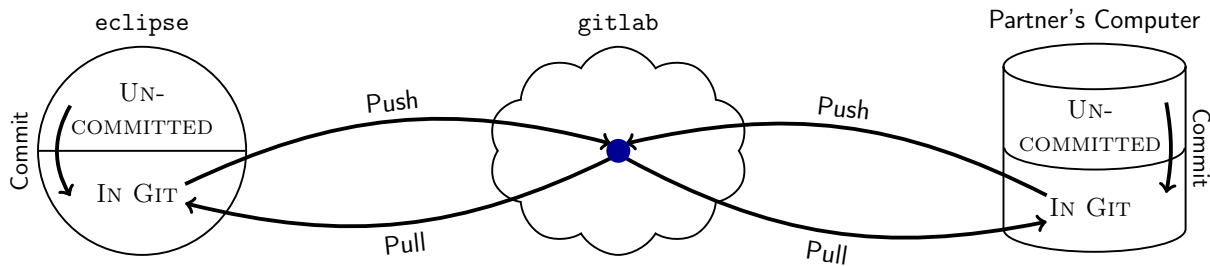
A **bold** phrase means you should substitute it with your equivalent.  
Do *not* forget the .git at the end!



- (4) Finish out the wizard, 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 the ability to go back to older versions of code. Another major reason VCS's are important is that they allow you to effectively work together with other people. They allow you to combine (called "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 you give a message indicating the high-level idea behind the changes. An example might be "Adds error handling for the empty queue case in dequeue".
- **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, the histories of the file diverged, and git asks you to explain how to merge the changes together. (This is called resolving a "merge conflict".)

## Using Git in Eclipse

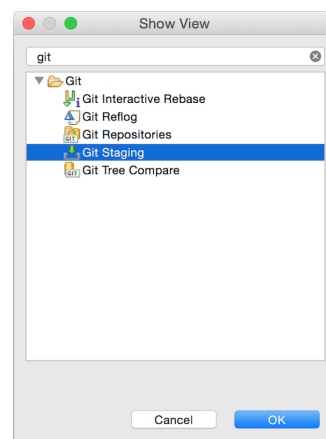
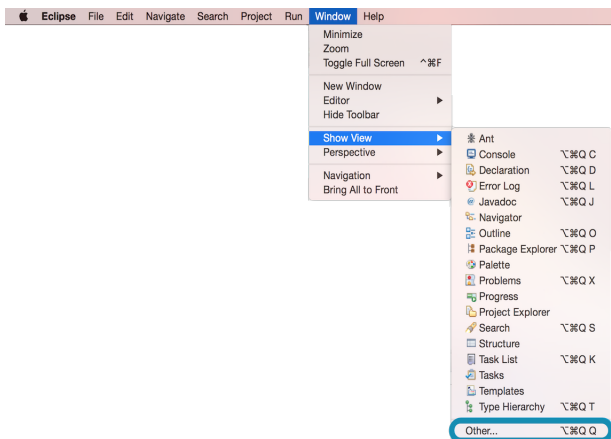
Eclipse provides GUIs for all of the git operations. We now explain how to handle a git workflow.

### Open Git Staging View

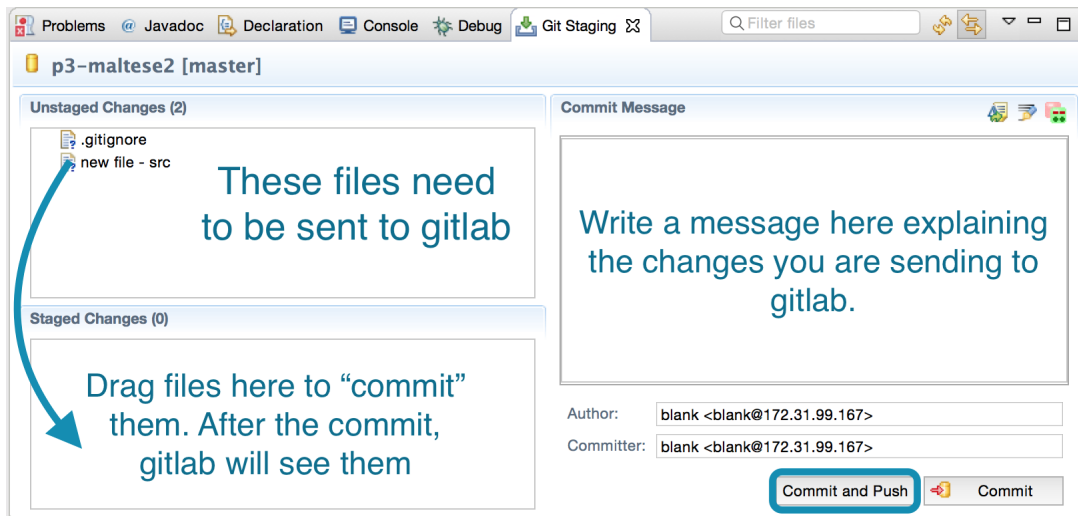
Your main tool to use git in eclipse is called the "staging view". This view allows you to see changed files, make commits, and write commit messages. To open the Staging View, follow these steps:

(1) Go to Window > Show View > Other

(2) Type "git"; choose "Git Staging"



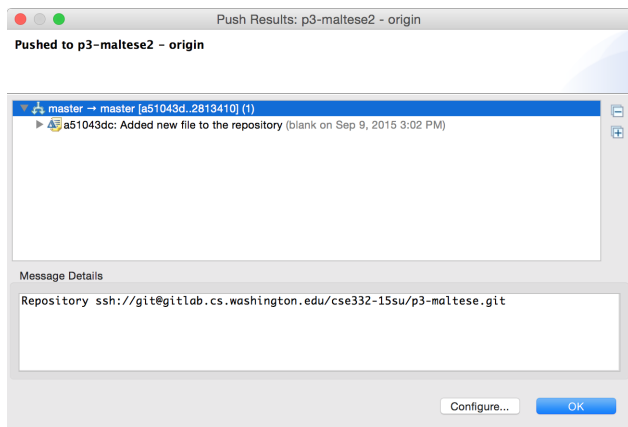
## How Staging/Committing Works



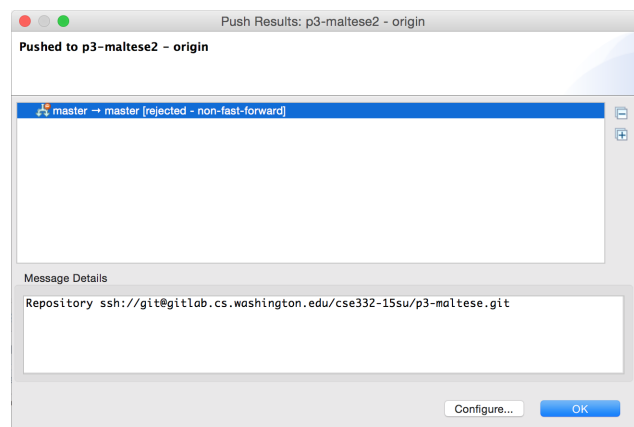
## After Pushing

After you "Commit and Push", you will get one of several dialog messages:

### Success!



### Rejected!

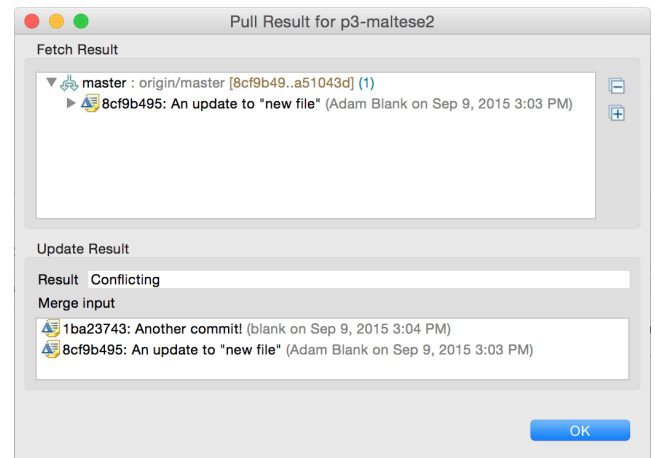
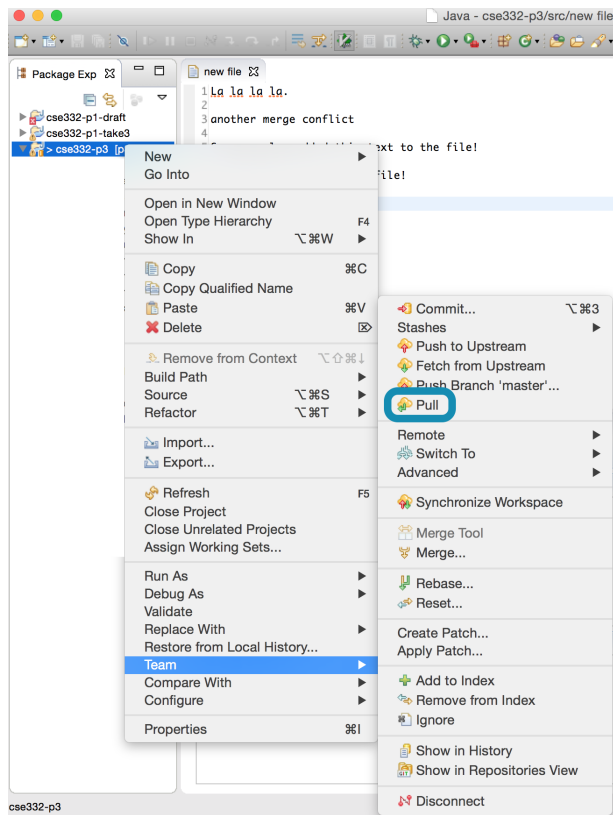


If you get a "success" message, then your code has been pushed, and you're good to go. If you get a "rejected" message, it means your partner pushed code; so, you will need to *pull* before pushing will work. Note that when you pull, you *might* have a merge conflict which you will have to fix before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

## Pulling

There are two reasons to pull: (1) you want to get the changes your partner made, and (2) you want to push your changes, but they were rejected because of a conflict.

To pull in Eclipse, click the following menu option:

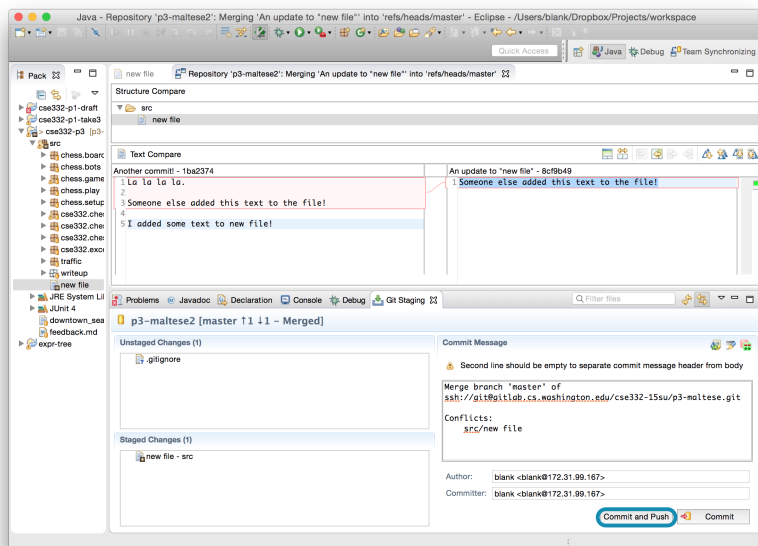
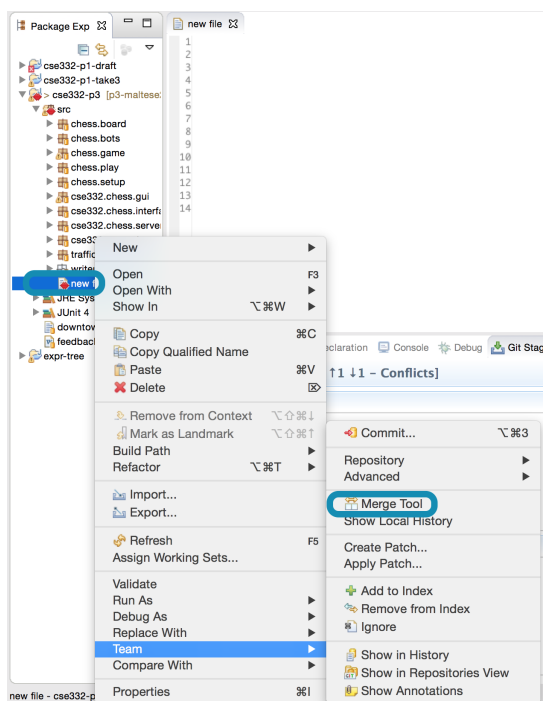


The result will either indicate that you pulled cleanly or that there is a conflict.

The best way to tell if you currently have a conflict is to look for the red diamond icon in the list of files.

## Merging a Conflict

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



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 the result you actually want. Once you've done this, drag it 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!

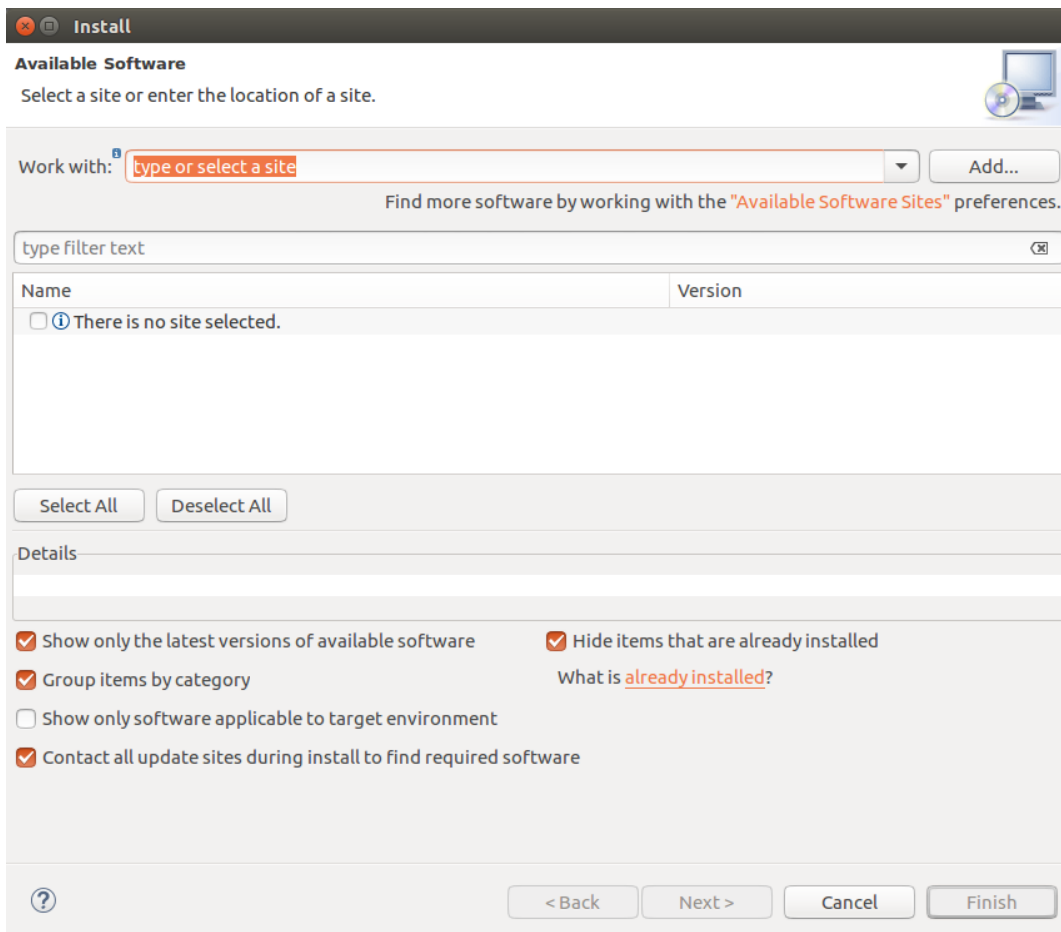
## Submitting Your Final Version

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 will grade.

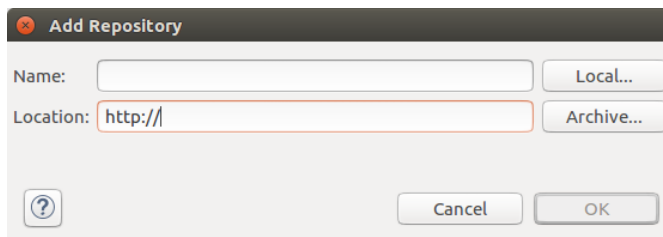
## Part 2: Setting up Viz for Eclipse

This section will help you get started with Viz for Eclipse and using it visualize various data structures and help with debugging your projects in CSE 332.

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



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

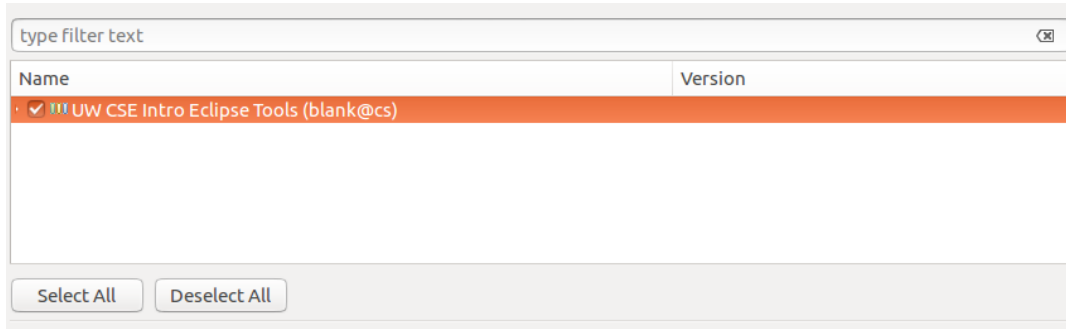


- (b) In the input labeled Location, enter

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

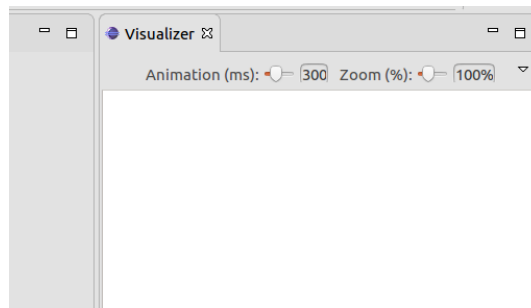
as the url and click OK.

You should then see the following in the Install window:

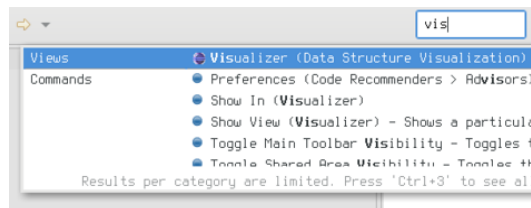


Make sure that the checkbox is checked, and then click **Finish** to close the pop up. Eclipse should then prompt you to confirm license agreements and finish the installation process, after which you should be prompted to restart Eclipse.

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



This is the “Viz” visualization tool that will allow you to see the state of your program and data structures during a debugging session. If you don’t see it, type in “vis” into the Quick Access bar and wait for a suggestion to popup. 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 can post your issue on Piazza or go to office hours for us to help):



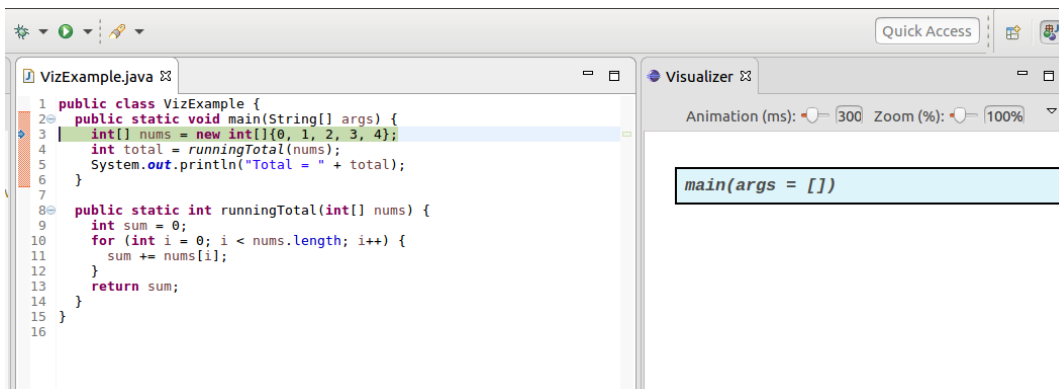
(d) Viz is now ready to go! To test it out, try writing a small program involving a data structure (e.g., an array of integers) and starting a debugging session. We provide an example walkthrough of a Viz debugging session in the following section.



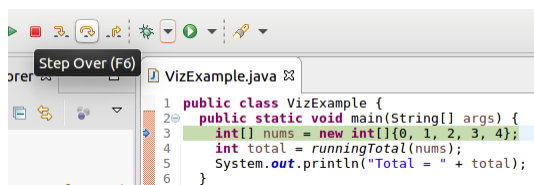
## Example Viz Walkthrough

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 visualizations (for an array in this case), and a loop iteration.

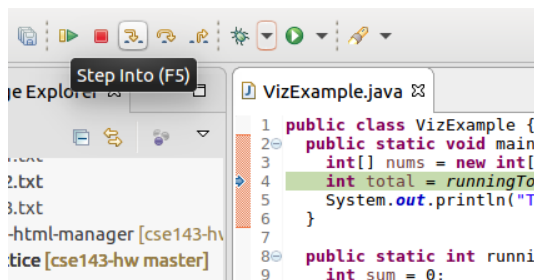
- (a) To start a debugging session, click the small bug icon at the top bar of your Eclipse environment (shown in the top-left corner of the screenshot below) and you should then see the following in the Viz pane:



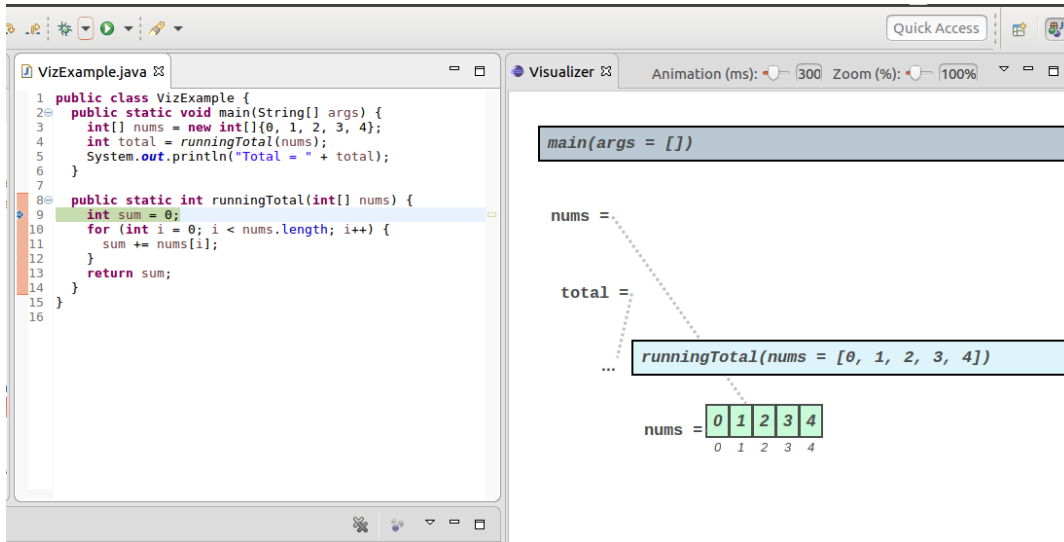
- (b) To move through a program, you can use the “Step over” command to step to the next line in the current method, and the “Step into” command to step into a method at the current line. Here, we want to step over until we get to line 4 where the runningTotal method is called.



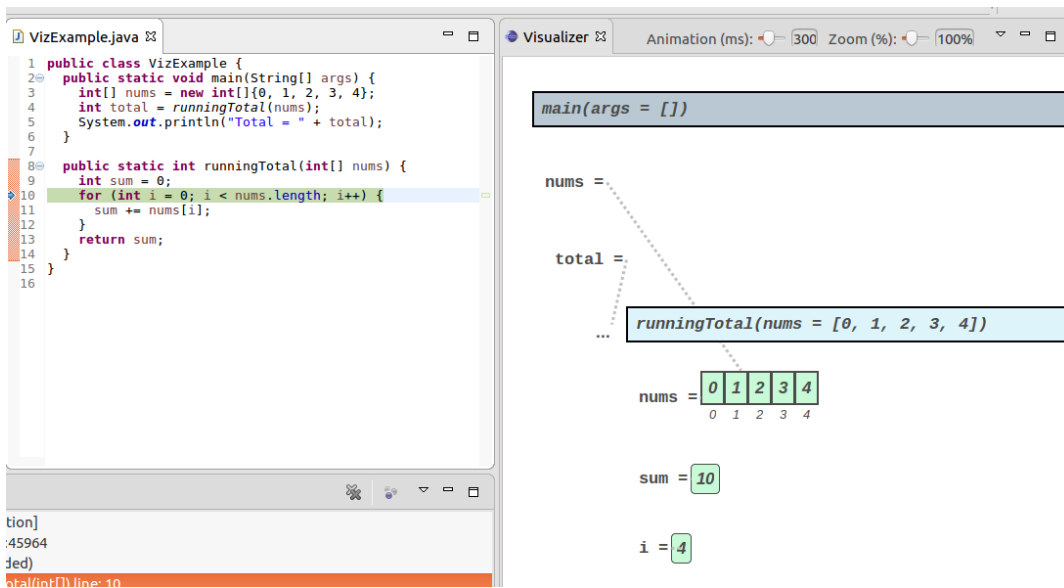
Once you see the blue arrow (indicating your current line) at line 4, click on the “Step into” command:



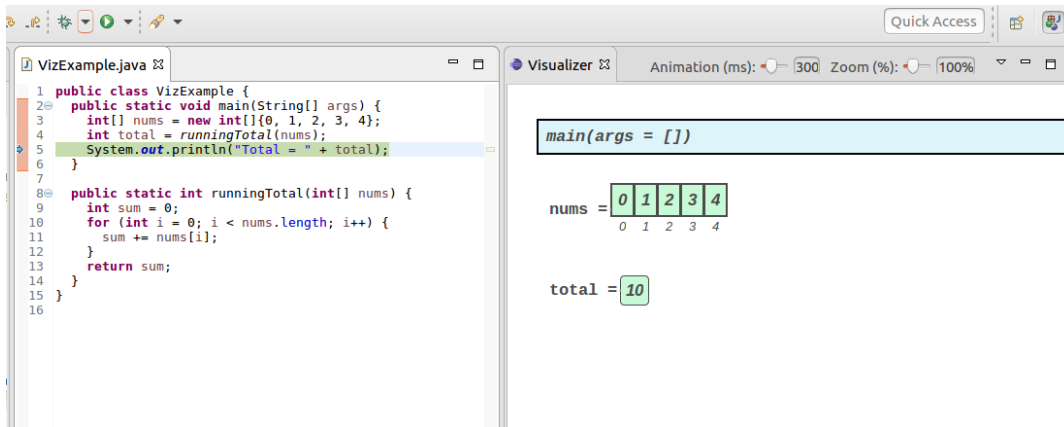
You should then see the following in Viz, indicating that you are now in the `runningTotal` method:



- (c) Now is where you'll really get to see Viz in action! Simply 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:

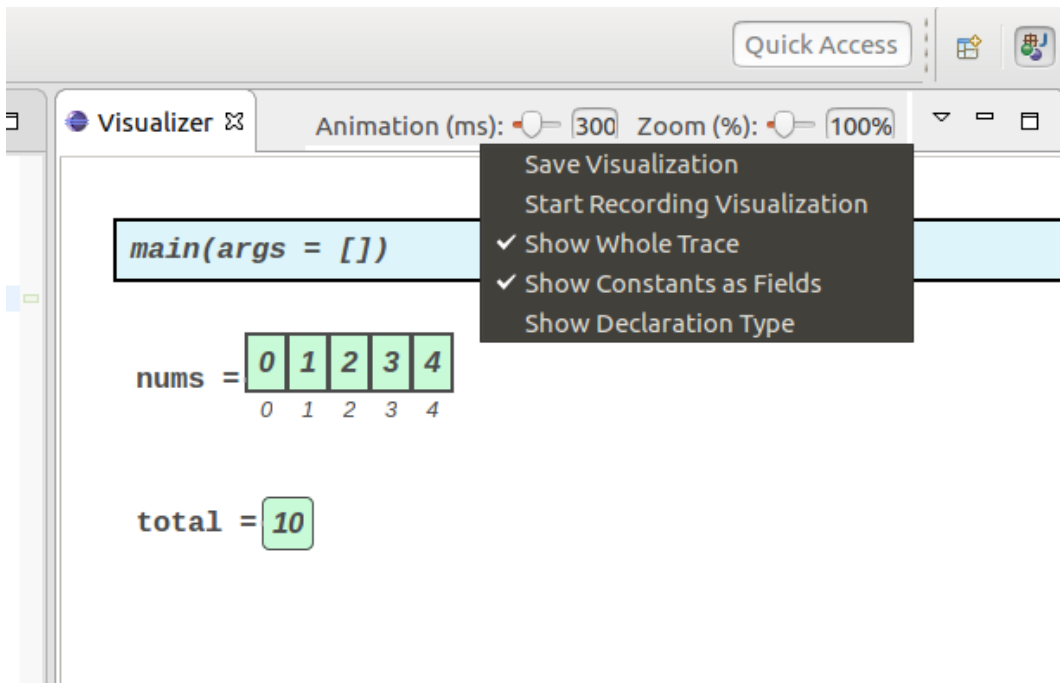


(d) Once you click “Step over” from here, the method and its variables will be removed from the Viz pane and you will return focus to the method that had called it. In this example, this looks like the following:



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

We’ve demonstrated a very simple program using Viz, but its value will hopefully be fully appreciated as you work with nested methods and complex data structures. In addition to helping you visualize the different programs you will be writing in CSE 332, Viz also lets you save an image or gif of your visualization at any time when you select the associated option 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

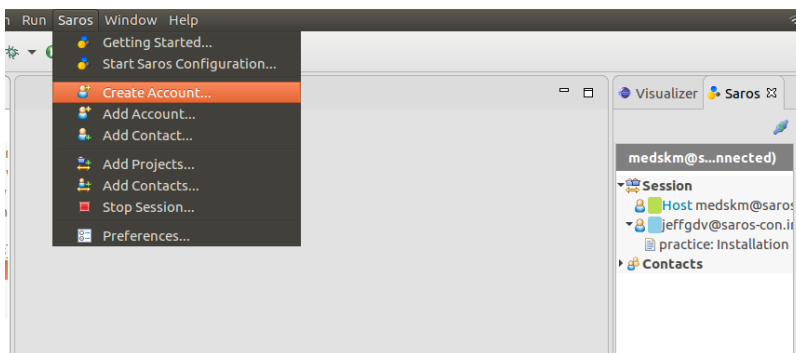
This guide will help you get started with Saros and using it to peer program with your CSE 332 project partner.

### Installing Saros

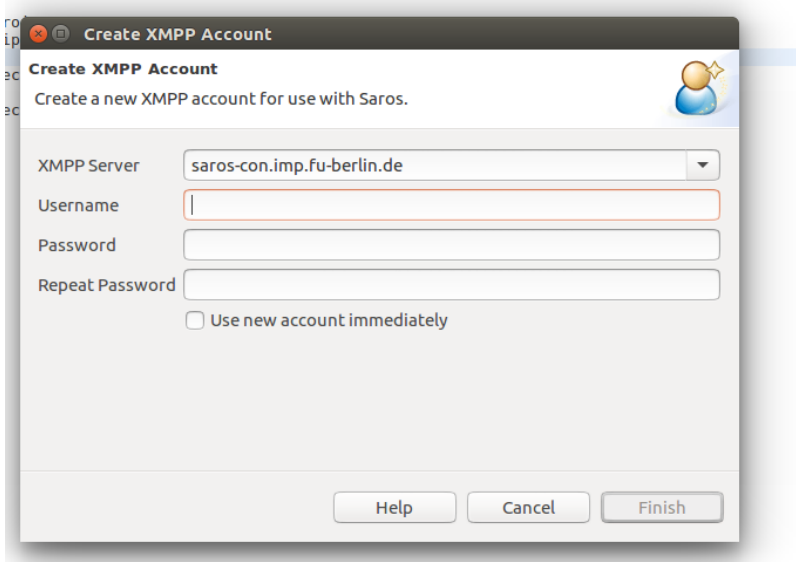
- (a) Download Saros from [Sourceforge](#).
- (b) Uncompress the downloaded folder, and then use your file viewer to find your Eclipse app. If you see an Eclipse folder, enter it, otherwise if you only see the Eclipse application icon, right click on the Eclipse icon, and select “open package contents” 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

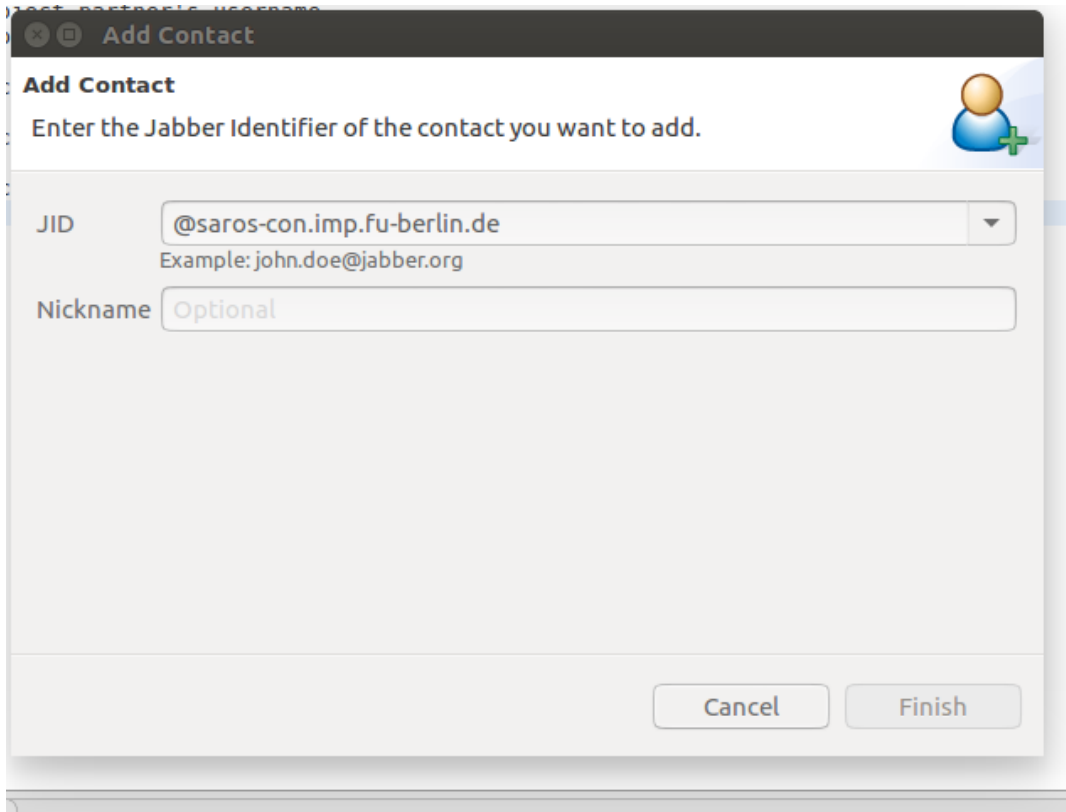
- (a) After restarting Eclipse, open Saros (either auto-populated in an Eclipse tab or by typing “Saros” in the Quick Access bar). Once you have the Saros tab open, you will need to create an account to use Saros. On the Saros option in the top menu bar, select “Create Account”:



You should see the following window appear. Fill in the information to create your account, leaving the given server name and providing your student username and a password of your choice. Make sure to select the “Use new account immediately” option to prevent having to log in again, and click “Finish”.

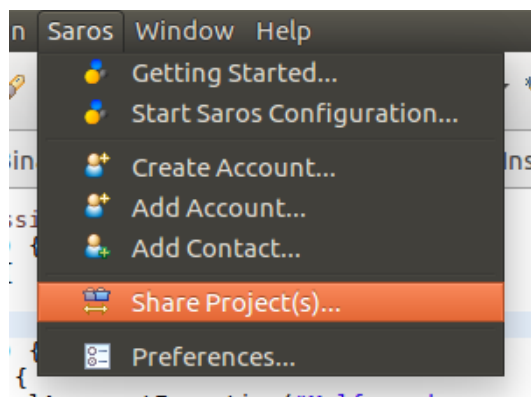


- (b) Now you will need to add your project partner as a Saros contact to enable real-time collaboration. On the same Saros menu bar from Step a, select “Add contact”. In the “Add Contact” window that appears (see below), add your project partner’s username to the left of the @-symbol in the “JID” text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

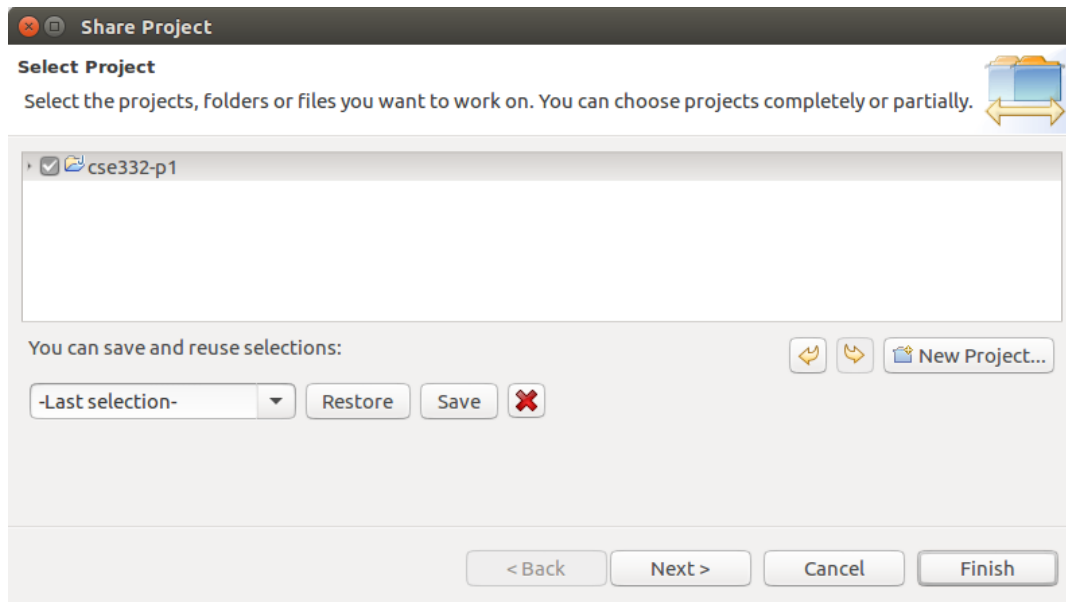


Once you select “Finish”, they should receive a contact request on their Eclipse app, which they should accept (only one of you needs to send a contact request for you both to be contacts).

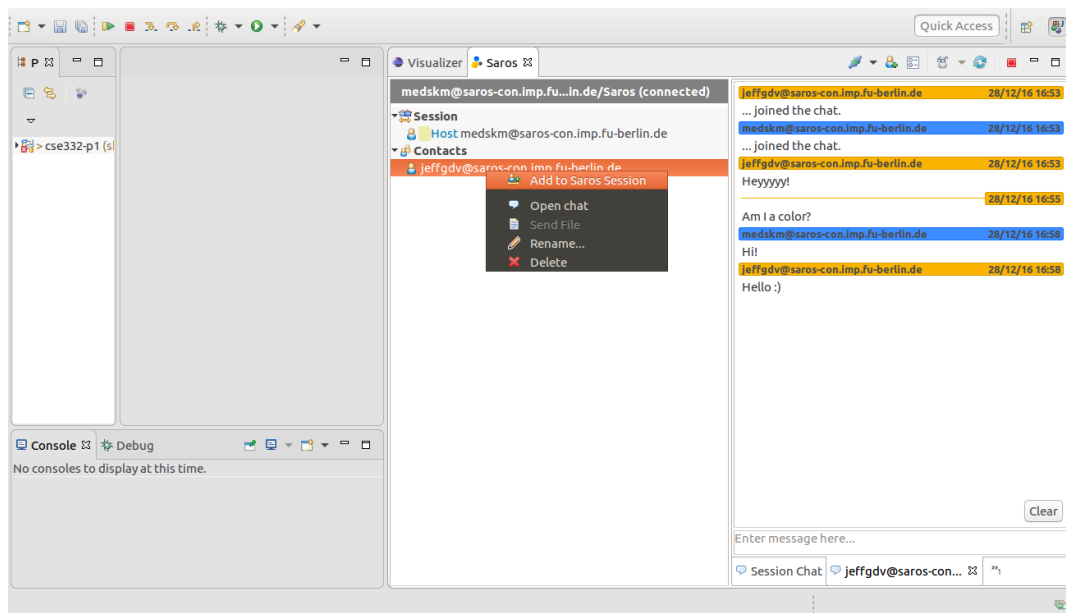
- (c) Once you have added your partner as a contact, you will want to add a project to Saros so that you can peer-code with your partner. To do this, first open up the Saros menu on the top menu bar and select “Share Project...”.

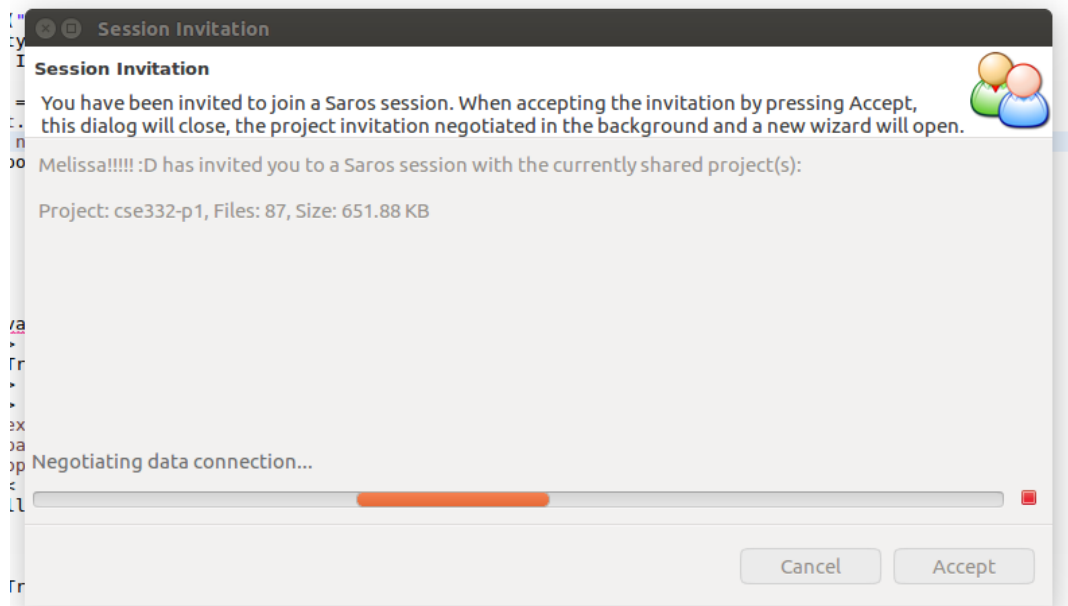


All projects in your current Eclipse workspace will appear, and you should select the one that corresponds to your project repo.

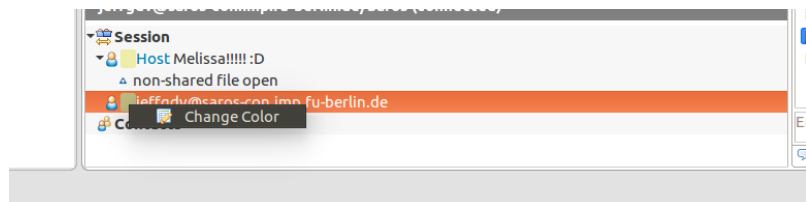


- (d) Once you've shared your project, you will still need to add contacts to that project. To do so, right-click on your partner's name in the Saros window and click "Add to Saros Session" (see below). Your partner will have to accept your invitation on their eclipse screen. If your partner is already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





- (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!](#)