Using IntelliJ, Git and Chime

Last year you used the command-line tools `javac` and `java`. This coming year you will learn to use Git to store and manage your source code and IntelliJ[1] to write, compile, and test your Java programs. Chime can then be used to submit hour code for testing.

In this guide you will learn how to create a new Java project in IntelliJ, how to clone a Git repository, and how to compile and test your code. You will also learn how to submit your tick for automated assessment.

The instructions below work on the MCS machines in the Intel Lab, however you can carry out the same task by installing IntelliJ (https://www.jetbrains.com/idea/) and Git (https://git-scm.com/) on your own laptop; the instructions should be similar, but there may be minor differences in screen layout and text descriptions.

Setting up Git

We will use Git to manage all source code you write for this course. This is a useful skill to learn in general, but also good preparation for your Part IB Group Project where you will need to use a source code management tool to support collaboration between multiple team members.

You should now setup Git using the following instructions. You only need to follow these instructions once at the very start of the course.

1. Run `ssh-keygen` in a terminal to generate a new public-private key pair. By default, on MCS machines, `ssh-keygen` will generate a new private key and place it in `~/.ssh/id_rsa` and public key in `~/.ssh/id_rsa.pub`. Please protect your private key as it can be used to submit code for this course on behalf of you (as well as other actions if you use SSH more generally). On MCS machines, we recommend entering a passphrase when prompted to provide additional protection for your private key.

2. Copy your *public* key into Chime so you can access Chime using the SSH protocol. To do so, go to https://chime.cl.cam.ac.uk/, entering your Raven credentials as necessary.

3. Click on the SSH keys menu icon at the very top of the screen.

4. Click on **Register new key** (blue button).

5. Copy and paste the public key into the **SSH Key** field.

6. Click the **Add new key** (blue button).

You should repeat the above steps for each additional computer you wish to associate with your Chime account. You can do this at any point.

Clone, edit, stage, commit and push with Git

There is a separate Git repository associated with each Ticklet for this course. You will need to carry out the steps described in this section once for each Ticklet. In brief, you create your own copy of the starting point into your own Git repository, then make a local copy of your version of the repository (a *clone*) on your MCS machine or laptop. You can then work on this local copy (without Internet connectivity, if required). You should regularly *stage* and *commit* changes you make to files in your local repository (more details to come); this allows you to look back in time at the history of your changes if you, say, accidentally delete a file by mistake. Once you are ready to test your code, you should *push* your changes from your local repository to the remote repository hosted on Chime. From there you will be able to test your work via the web interface.

[1] You can use an alternative IDE, or indeed a command line editor if you prefer.

First, let's copy the official repository for Ticklet 0, and check that you have setup SSH properly, as described in the previous section.

1. Go to https://www.cl.cam.ac.uk/teaching/1819/FJava/ticklet0, edit the repository name if you wish to change the default, and then click on the **Start task** button.

2. You should now see an overview page for your repository. You can browse the files and directories in the repository using the web interface. Under the task description is a URI (of the form `crsid@chime.cl.cam.ac.uk:crsid/further_java_ticklet_0`, where `crsid` is your CRSID). You can use the URI to *clone* this repository, in other words, make a local copy, by transferring the data over the SSH protocol to your computer. On your computer check you have correctly configured everything by using the `git` command line tool to clone the repository in a terminal (making sure you use your own URI for the repository name!):

   ```
   bash$ git clone crsid@chime.cl.cam.ac.uk:crsid/further_java_ticklet_0
   Cloning into 'further_java_ticklet_0'...
   The authenticity of host 'chime.cl.cam.ac.uk (128.232.21.23)' can't be established.
   RSA key fingerprint is SHA256:nEryDfQuwSCP9HxY/aC8wTjyPVBGBZQNMvYHxy+hiWI.
   No matching host key fingerprint found in DNS.
   Are you sure you want to continue connecting (yes/no)? yes
   Warning: Added 'chime.cl.cam.ac.uk,128.232.21.23' (RSA) to the list of known hosts.
   remote: Counting objects: 26, done.
   remote: Compressing objects: 100% (14/14), done.
   remote: Total 26 (delta 2), reused 0 (delta 0)
   Receiving objects: 100% (26/26), done.
   Resolving deltas: 100% (2/2), done.
   Checking connectivity... done.
   ```

   Note that the first time you connect to the Chime server, SSH does not know the identity (public key) of the Chime server. You will therefore need to type `yes` to confirm that you would like to connect. The security conscious may wish to check that the SHA256 hash is the same as that shown above.

You now have a copy (called a *clone*) of the source code repository on your machine inside the directory `further_java_ticklet_0`. In the next section, we will move to using IntelliJ to interact with the repository and consequently we will make no further use of this directory or its contents, so you can safely delete it.

Git is a large piece of software, with the potential to support complex software development workflows and an extensive command line interface which can be used to interact with software repositories. We strongly recommend you take the time to read the introductory chapters of the Pro Git book, which is available for free [https://git-scm.com/book/en/v2], to understand what is going on at a more fundamental level. This will help you in your career generally, but also might get you out of a mess in this course or in the Part IB Group Project next term. You may also wish to install a GUI to manage Git repositories on your laptop or MCS machine. We recommend the *SourceTree* app for Mac; *GitKraken* is also used by students and works on Windows, Mac and Linux.

In summary, in this course, we will restrict ourselves to the following workflow using IntelliJ to provide a GUI on top of the basic git commands:

1. On Chime (http://chime.cl.cam.ac.uk/) fork the repository which contains the template for the ticklet for the current week.

2. Clone the forked repository on Chime into a local repository on the MCS machine or your laptop.

3. Renaming existing files in the repository as well as creating new files or editing existing files.

4. Explicitly add (or *stage* in git parlance) one or more created, modified, moved or renamed files and/or directories in preparation for a *commit*

5. Perform a *commit* to the *local* repository.

6. *Push* the changes committed in your local repository to the Chime server.

If you decide to use more than one machine to write your code on, we recommend you *push* changes from one machine to the Chime server, and *pull* those changes onto the other machines to ensure all your copies stay in sync. Git will require you to pull any outstanding commits from the Chime server before you can push local changes to the Chime server. With more than one machine it is possible to get *merge conflicts* which you will need to learn how to resolve. See the Pro Git book [https://git-scm.com/book/en/v2] for details.

Setup IntelliJ

These instructions assume you are using IntelliJ on the MCS machines. They should work, perhaps with minor modifications, on your own laptop.

1. Start the IntelliJ program by using the application menu on the left-hand side of the screen and selecting the top icon (this icon is has the tool tip **Search your computer**). Inside the Dash Home panel type **IntelliJ** into the search box and then click on the IntelliJ IDEA Community Edition icon.

2. A setup wizard will appear asking you to configure IntelliJ. Continue with the defaults across all configuration screens.

3. A splash screen will be visible for ten seconds or so, after which the welcome screen of IntelliJ will load. Select **check out from version control** and then choose Git in the pop-up menu.

4. Enter the URI to the git repository you want to clone into IntelliJ. You should enter the path that was created when you forked the repository on Chime. Then select **Clone**. When prompted to create an IDEA project from the sources, select **Yes**. In the next menu select "Create project from existing sources", press **Next**, enter a memorable project name, and press **Next** four times until you are asked to select a project SDK. For the project, hit the **+** button to add a new SDK choose a **JDK**. Please select a recent version of the Java JDK as found on your system (java-8-openjdk-amd64 is fine for MCS Machines). Select **Next** and then **Finish** to complete the setup of IntelliJ.

5. You should now see a new development window with the name of the project in the top left. Expand the file lists in the left-hand Project Pane by clicking on the triangle icon which should rotate into a downwards arrow (▼) and reveal a sub-directory called "src" associated with the project. Further expansions should reveal a package called `uk.ac.cam.crsid.fjava.tick0`, where `crsid` is your CRSID.

6. You can now provide your implementation of external sort in the method `ExternalSort.sort`.

7. You can build a jar file containing your compiled code as follows in IntelliJ. First, go to **File → Project Structure...**. In the dialog box select **Artifact** from the left pane and then **+ → JAR → Empty**. Enter a name for the artifact and select an output directory of your choice. Next click **Create Manifest** and select the default place to save the manifest file and choose the Main Class to be `uk.ac.cam.crsid.fjava.tick0.ExternalSort` (of course changing `crsid` to your CRSID). Add the code to the Jar file by expanding the **ticklet0** icon from the **Available Elements** panel and double-click on **'ticklet0' compile output** to add this into the Jar. Then select **OK** on the Project Structure window and close the window. Finally, to build the Jar file using the structure you have just created, go to the main IntelliJ menu and select **Build → Build Artifacts...** and then select **build** from the pop-up menu for the artifact you wish to build. The Jar file should now be saved in the output directory you previously configured.

8. It is often useful to test programs with command line arguments inside IntelliJ. Test your program in this way now by editing the settings needed to run your program. To do so, go to the main IntelliJ menu and select **Run → Edit configurations...**. This opens a new dialog window which allows you to create a custom mechanism for running the `ExternalSort` program. Create a new runtime configuation by selecting **+ → Application** from the top-left corner, enter a name for it (e.g. "External Sort") in the **Name** field. Make sure the **Main class** is `uk.ac.cam.crsid.fjava.tick0.ExternalSort`

and add two filenames as arguments to the **Program arguments** field — these will be used to test your sorting routine; you might like to use fines from the starter pack. Finally click the **Okay** button to execute your program.

Testing on Chime

After you have completed your implementation of external sort *and tested it with the starter pack*, you should commit your changes to your local repository, push them to Chime and ask Chime to test your code for you. You can do so as follows:

1. Go to the main IntelliJ menu and select **VCS → Commit Changes...**.

2. At the top of the dialog box you can add (or remove) files you wish to stage for a commit. The default (all files that have changed) is okay if you have not added additional files; otherwise you should check that you have added in new files that you have created.

3. Type in a short, but meaningful, commit message describing the changes you have made. Something like "External sort, working as tested with starter pack." is suitable. Select **Commit and push...** to commit the changes to your local repository and push them to the Chime server.

4. At this point you may be prompted with a dialog to ask whether you wish to review a couple of warnings. In general, there should be no warnings on any code you believe is ready for assessment, so please fix these before testing.

5. Another dialog should then appear inviting you to push your commit to Chime. Press the Push button. The dialog button should disappear, and you should see a small message at the bottom of the IntelliJ screen to let you know that the push was successful. If this did not work, then once you have fixed the underlying issue, you can push without a commit by going to the main IntelliJ menu and selecting **VCS → Git → Push...**.

6. Now visit your Chime webpage for the repository associated with this ticklet (e.g. http://chime.cl.cam.ac.uk/page/repos/crsid/further_java_ticklet_0) and check with the file browser (click on the </> icon) that the updated version of your ticklet has appeared.

7. You can now test your implementation by clicking on the **New submission** link at the top of the file browser web page. Depending on the task and the number of other students completing their work, it may take some time for the server to test your work. Results will be available on this page once testing is complete.