**Lab Sheet 2 – Intro to Version Control**

**Part 1: Import project to Eclipse and build project**

**1.1 Browsing Code in Eclipse（Optional）**

Reopen the BuddyInfo project you created in Lab 1. If the project is not yours or you could not find your project in local folder, you need to follow the instruction below to import the BuddyInfo project.

Create a new project and import the BuddyInfo class that you developed in the first lab.Use the following steps:

1. Right-click on your new project, click “Import...”

2. Select “File System”, click “Next” (on newer versions of Eclipse File System may

be under the “General” tab).

3. For the “From Directory”, click “Browse...” and find the folder containing the file,

and select it.

4. In the options on that page, make sure “Create selected folders only” is checked. The other option would use the complete file hierarchy as the package name, which is not the case in this example (which does not declare any package).

5. Success! Now go back to the Java or Java Browsing perspective and the file should

be there.

**1.2 Building on BuddyInfo**

Create an AddressBook class that contains a collection of BuddyInfo objects (the choice of the collection class is entirely up to you). Provide an addBuddy() and a removeBuddy() method. We will need this new class in subsequent assignments...

**Part 2: Gentle introduction to Version Control using Git**

**2.1 Introduction**

“Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.”1 It is an indispensible tool for modern programmers, whether working alone or in a team. There are many version control systems available: CVS, SVN, Git, Mercurial, ClearCase, etc.

**2.2 Git**

In this lab, we will use Git for source code management. Git was originally created for managing the code in the Linux kernel but has since gained widespread popularity. One major benefit of Git is that it is distributed, and so it doesn't rely on a central server like many other systems. It is also free, well-supported, and there are great code hosting sites that support it.

The versions of Eclipse in the lab should have a Git plugin installed, called eGit. If you are using your own computer you may need to download and install this plugin. Also note that many programmers prefer using the Git command lines directly, without relying on their Integrated Development Environment. For this you would have to install a separate Git program. We will mention the command line equivalent of the Eclipse dialogs when applicable.

**2.3 Creating a local repository**

**2.3.1.** Back to the BuddyInfo project in your Eclipse.

1 http://git-scm.com/book/en/Getting-Started-About-Version-Control

**2.3.2.** Highlight your project name and click File->Team->Share Project...

**2.3.3.** Select Git and click Next.

**2.3.4.** Check the box “Use or create repository in parent folder of project”.

**2.3.5.**Highlight your project and click “Create Repository”. (the command line equivalent of this is: “git init”)

**2.3.6.**Click Finish.

**2.3.7.** You now have your local Git repository created. To add your source code to the repository, so changes to it can be tracked, right-click “AddressBook.java” and go to Team->Add to Index. (the command line equivalent is “git add AddressBook.java”)

**2.3.8.** Let’s make a change to our code. Output a string such as “Address book” in the main method of AddressBook.java. An example is as follow, however you can add others as long as it compiles.

**2.3.9.**Now that we have changed our code, we will commit the change. Right-click on “HelloWorld.java” and select Team->Commit. (or “git commit” using the command line.)

**2.3.10.** You will get a popup asking you to enter a message about your changes. Enter a message and click Commit.

**2.3.11.**Make another change to the main method. Create a BuddyInfo and add it to AddressBook and then remove it from AddressBook. Commit your changes to the repository.

**2.3.12.**Normally, the original version of your code would be gone. But remember, version control systems store previous versions of your code. Right-click “AddressBook.java” and select Team->Show in History. (or “git log” using the command line)

**2.3.13.** You should see two entries (one for each time you committed code). Double- click the top one to see your current version and the second one to see the old version.

**2.3.14.** If you click the Compare Mode icon in the History window and then double- click the older version of your code you can compare the differences in the old and new versions. (“git diff” using the command line)

**2.3.15.** You now know how to create your own local Git repository, add files to the repository, commit changes and examine older versions of your code.

**Deliverables 1: Addressbook.java added to your project created in Lab 1.**

**Deliverables 2: The screenshot of the comparison of the two commit you did to AddressBook.java.**

Additional Resources on Git:

- the official Git web site: <http://git-scm.com/>

- a free online book on Git: <http://git-scm.com/book>

- a standalone Git program for Windows: <http://msysgit.github.io/>

In future labs, we will get into the creation of version branches, and how to host your project on GitHub.