

1 Objectives

- Learn how to use Eclipse.
- Practice using Eclipse.

2 Introduction

In this lab you will learn how to use Eclipse. Eclipse is an integrated development environment (IDE) for developing Java software. It also has many other uses which we will not cover. Eclipse is widely used in industry because it makes most software development tasks easier.

3 Getting Started With Eclipse

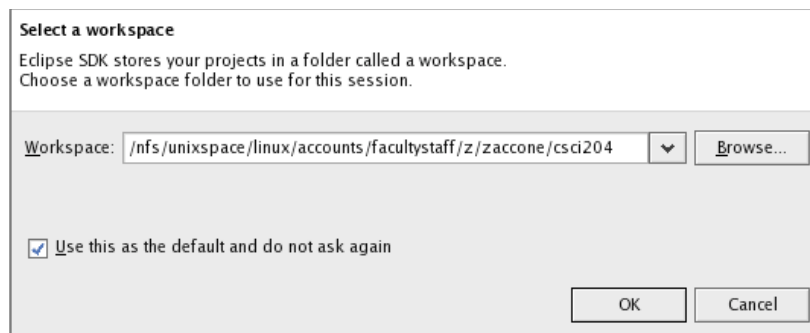
Begin by opening a terminal window and making a directory that you will use for this course.

```
mkdir csci204
```

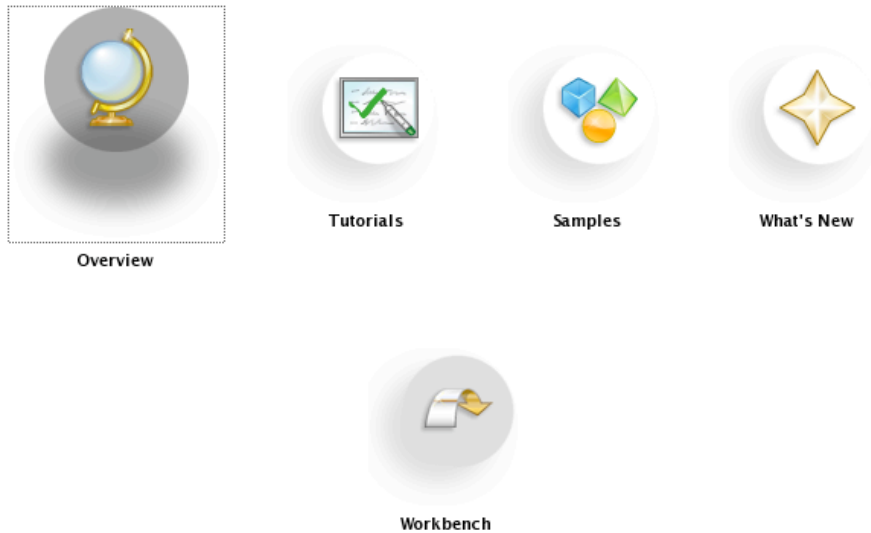
Then, start Eclipse by either typing

```
eclipse &
```

in your terminal window, or by selecting Eclipse from the Programming menu found within the Red Hat menu. Eclipse will present you with a dialog asking you for the location of your workspace. Indicate that it is your `csci204` directory and check the box so it won't ask again.



When you start Eclipse you will be presented with a welcome screen similar to the following screen.



Click on the **Workbench** icon to begin using Eclipse.

3.1 Change Eclipse Settings

Follow the instructions in this section to make sure Eclipse is configured properly.

Open Eclipse preferences by selecting **Window** → **Preferences...**

- In **Java** → **Editor** → **Folding**, uncheck Header Comments and Imports. Enable folding should remain checked. Click the **Apply** button.
- In **Java** → **Editor** → **Save Actions**, check the box that says “Perform the selected actions on save”. Then check the box that says “Format source code.” Make sure the [Format all lines] radio button is selected. Click the **Apply** button.
- In **Java** → **Code Style** → **Code Templates**, click the disclosure triangle next to Comments. Click on the word Files and then press the **Edit** button. Enter a comment similar to the one that follows, using your name.

```
/**
 * CSCI 204, Rick Zaccone
 * Assignment: ${project_name}
 * Created: ${date}, ${time}
 */
```

Click **OK**, then click the **Apply** button.

- In **General** → **Editors** → **Text Editors**, check the box that says “Show print margin” and then click the **Apply** button.
- Click the **OK** button to exit the preferences.

4 Create a Java Program

You are now ready to create your first Java program in Eclipse. It is an old computer science tradition to write a “Hello World” program as your first program. Since this is your first Eclipse program, that’s what we will do now. Follow the following steps closely.

- Select **File** → **New** → **Java Project**.
- Enter lab01 as the project name and press the **Finish** button.
- Select **File** → **New** → **Class**.
- Enter Hello as the name.
- Check the box that says to create a method stub for public static void main.
- Check the box that says to “Generate comments”.
- Click the **Finish** button.

You should be looking at a new class named Hello which has a main() method. There are comments too!

The comment

```
// TODO Auto-generated method stub
```

is there to remind you to fill in the method stub that Eclipse generated for you. You may remove it. Type the following text in its place

```
System.out.println("Hello World!");
```

and save (Ctrl-S). To run the program, click the run button in the tool bar. It’s a green circle with a white arrow inside. You will get a **Run As** dialog. Select **Java Application** and press **OK**. Your output should appear in the console window at the bottom of the Eclipse screen.


4.1 Add Javadoc Comments

When Eclipse generated your `main()` method, it inserted Javadoc comments for you. Complete these comments now. (The content of your comments is not very important for this file.) Click on the Javadoc tab at the bottom of the screen and then start editing the comment for `main()`. Eclipse will show you the formatted comment in the Javadoc pane. Edit the comment with the `@author` tag and add a description of your `Hello` class. Again, you should see the formatted comment in the Javadoc pane.

5 Scanner Class

In this section we will review how to use the `Scanner` class from the `java.util` package. While we learn about the `Scanner` class, we will also learn some features of Eclipse.

5.1 Create a New Class

As we saw before, you can create a new class by selecting **File** → **New** → **Class**. There is a toolbar icon for this too. It is a green circle with a white C in it . Click on this icon now to create a new class. Name your class `ScannerTest`, ask for comments and a `main()` method, then press the **Finish** button.


Augment the Javadoc comment with the `@author` tag with a description of the class saying that it is an exercise in lab 1 to practice using the `Scanner` class.

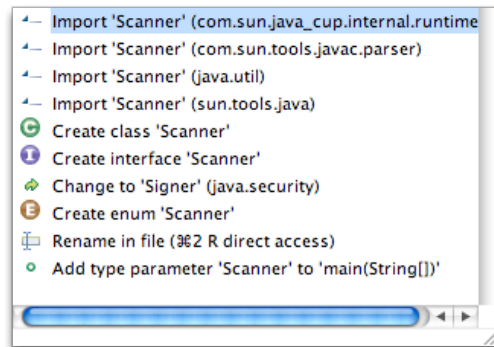
5.2 Ask Eclipse to Create an `import` Statements

A common problem is remembering the name of the `import` statement that you need to import a particular class. Why spend precious minutes searching through textbooks when you can have Eclipse do it for you?

Enter the following line into your `main()` method.

```
Scanner keyboard = new Scanner(System.in);
```

Eclipse interprets your program as you type. It immediately notices there is a problem because it doesn't know the `Scanner` class. It will underline both occurrences of the word `Scanner` in red and place a red rectangle with an X in the margin . Hover the mouse over this symbol or the underlined words, and Eclipse will tell you what's wrong. In this case it will tell you that `Scanner` is not a recognized type. Click on the red symbol in the margin, and it will offer suggestions on how to correct the problem.



Use the up and down arrow keys on your keyboard to move through this list. As each item is highlighted, Eclipse will show you what will happen if you select that item. In this case, you want to import `Scanner` from `java.util`. Select that item and press Enter. Eclipse will add the necessary `import` statement to your code. Alternatively, you can use the keyboard shortcut `Ctrl-Shift-M` to add an import statement for the currently highlighted class or `Ctrl-Shift-O` to both add import statements for *all* unknown classes and remove all unused import statements.

Use Eclipse to enter the following code into `main()`. You can use the tab key to indent.

```
Scanner keyboard = new Scanner(System.in);
System.out.println("Enter an int, double, word, and line: ");
int anInt = keyboard.nextInt();
double aDouble = keyboard.nextDouble();
String aWord = keyboard.next();
String line = keyboard.nextLine();
System.out.println(anInt + ":" + aDouble + ":" + aWord + ":" + line);
```

Run the program. Please note that the input expects to be on one line. For example:

```
Enter an int, double, word, and line:
3 4.5 hello more words
```

Try typing in different values to see how the `Scanner` class works.

6 Other Helpful Editing Shortcuts

This section describes some other useful editing techniques that will save you a lot of time.

6.1 Adding Javadoc Comments

Eclipse has a handy facility for adding Javadoc comments to a method you have already written. For example, enter the following method to your `ScannerTest`

class.

```
public void printXY(int x, double y) {
    System.out.println("x = " + x + " y = " + y);
}
```

Place the cursor anywhere within the method and select **Source** → **Generate Element Comment**.

```
/**
 * @param x
 * @param y
 */
public void printXY(int x, double y) {
    System.out.println("x = " + x + " y = " + y);
}
```

Note that it has inserted the names of the parameters. All you need to do is add descriptions. Change the comments so that they are meaningful. Here is a suggestion for `main()`.

```
/**
 * Begins program execution.
 *
 * @param args
 *         contains command line arguments
 */
```

6.2 Method Name Completion

Suppose you are typing the name of a method in Java's API but you can't remember how to spell it. Eclipse will help you with this. Add a new line to `main()` as follows

```
System.out.prin
```

and stop typing after the "n". Eclipse will offer a list of suggestions. Use the arrow keys to move through the list and select

```
System.out.println(String x)
```

and press Enter. Eclipse has highlighted the `x`, so whatever you type will replace it. Type any string you would like to print.

6.3 Content Assist Example: Printing Shortcut

Eclipse has a nice feature that allows you to quickly insert patterns into your program called **Content Assist**. The kind of patterns available to use will depend on the element of the program you are currently editing (or have highlighted). For instance, Content Assist can be used to transform any Java expression into code that

prints the expression to the console. Suppose you wanted to print “Hello World!”. Type the following string on a line by itself in `main()`.

```
"Hello World!"
```

Now highlight this string with your mouse and select content assist (**Edit** → **Content Assist** → **Default**). Use the arrow keys or your mouse to select “`sysout` — print to standard out” and press Enter (or double click). Content assist is so useful that you will want to invoke it with its keyboard shortcut (Ctrl-Space).

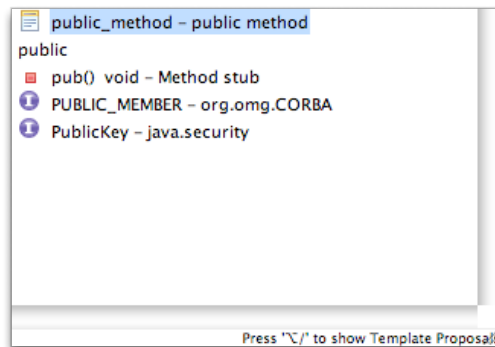
6.4 Content Assist Example: Creating a New Method

Here is another example of what you can do with Content Assist. Suppose you would like to create the following method.

```
public boolean isZero(int count) {  
    return count == 0;  
}
```

Complete the following steps.

1. Type just the letters `pub` and invoke content assist.
2. The first choice in the list is to create a public method. Accept it.



3. Eclipse has highlighted the return type. Type the word `boolean` to replace what is there.
4. Press tab and Eclipse will highlight the method name. Type `isZero`.
5. Press tab and Eclipse will move the cursor between the parentheses so you can type the parameter to the method. Enter `int count`.
6. Press tab again and Eclipse will position the cursor so that you are ready to enter the method body. Complete the method and save.

7 Additional Information on Eclipse

We have explored only a small number of the features of Eclipse. We will do more in later labs. For additional information about Eclipse, look in the **Help** menu or visit the Eclipse web page <http://www.eclipse.org/>.

8 Exercise to Hand In

Use the features of Eclipse to write a class `Purse` that represents a coin purse. It will have three integer instance variables named `numNickels`, `numDimes`, and `numQuarters` that contain the number of nickels, dimes and quarters in the purse. The constructor for the class should initialize these to zero.

Create a method called `addNickels()` that takes an integer parameter and does not return a value. It will add that many nickels to the purse. Create similar methods called `addDimes()` and `addQuarters()`. Finally, create a method called `getTotal()` that returns the value of the coins in the purse. It will have no parameters and it returns a `double`. Document each method using Javadoc comments. (Although you might not normally document such short and obvious methods, it is good practice for this lab).

Add the following `main()` method to your `Purse` class.

```
public static void main(String[] args) {
    Purse myPurse = new Purse();
    myPurse.addNickels(3);
    myPurse.addDimes(1);
    myPurse.addQuarters(2);
    double totalValue = myPurse.getTotal();
    System.out.print("The total is ");
    System.out.println(totalValue);
}
```

Run the program and make sure it is working.

9 What To Hand In

Run your program using the Run button in the toolbar or use the **Run** menu. Open a text editor, copy and paste the output and save into a file called `output.txt`. You can use emacs or the text editor found in **Red Hat** → **Accessories** → **Text Editor**.

Hand in a copy of `Purse.java` and `output.txt`.