2 Getting Started with jGRASP 2.0

After you have successfully installed the Java JDK and jGRASP, you are ready to get started. For the examples in this section, Microsoft Windows and Java will be used. However, much of the information in sections 2.1 - 2.7, 2.11, and 2.13 - 2.16 applies to other operating systems and supported languages for which you have installed a compiler (e.g., Ada, C, C++, and Python) unless noted otherwise. For example, in the "Creating a New File" below, you may select C++ as the language instead of Java, and then enter a C++ example. Also note that for Ada, C, and C++, you'll need to "compile and link" (rather than just "compile") in order to run your program.

Objectives – When you have completed this tutorial, you should be comfortable with editing, compiling, and running Java programs in jGRASP. In addition, you should be familiar with the pedagogical features provided by jGRASP, including using interactions, generating the CSD, folding your source code, numbering the lines, stepping through the program in the integrated debugger, and using the dynamic viewers and canvas.

The details of these objectives are captured in the hyperlinked topics listed below.

2.1 Starting jGRASP

jGRASP

If you are working in a Microsoft Windows environment, you can start jGRASP by double clicking its icon on your Windows desktop. If you don't see the jGRASP icon on the desktop, try the following: click ***Start*** > ***All Programs*** > **jGRASP** (folder) > **jGRASP**.

Depending on the speed of your computer, jGRASP may take between 10 and 30 seconds to start up. The jGRASP virtual **Desktop**, shown below, is composed of a Control Panel with a menu and toolbar across the top and three resizable panes. The *left pane* has tabs for **Browse**, **Debug, Find,** and **Workbench**. The Browse tab, which is the default when jGRASP is started, lists the files in the current directory. The large *right pane* is for CSD, Canvas, and UML Windows. The *lower pane* has tabs for **jGRASP Messages**, **Compile Messages**, **Run I/O**, and **Interactions**. The panes can be resized by selecting the partition with the mouse (left-click and hold down) then dragging the partition. You can also click the arrowheads on the partition to open and close the pane.

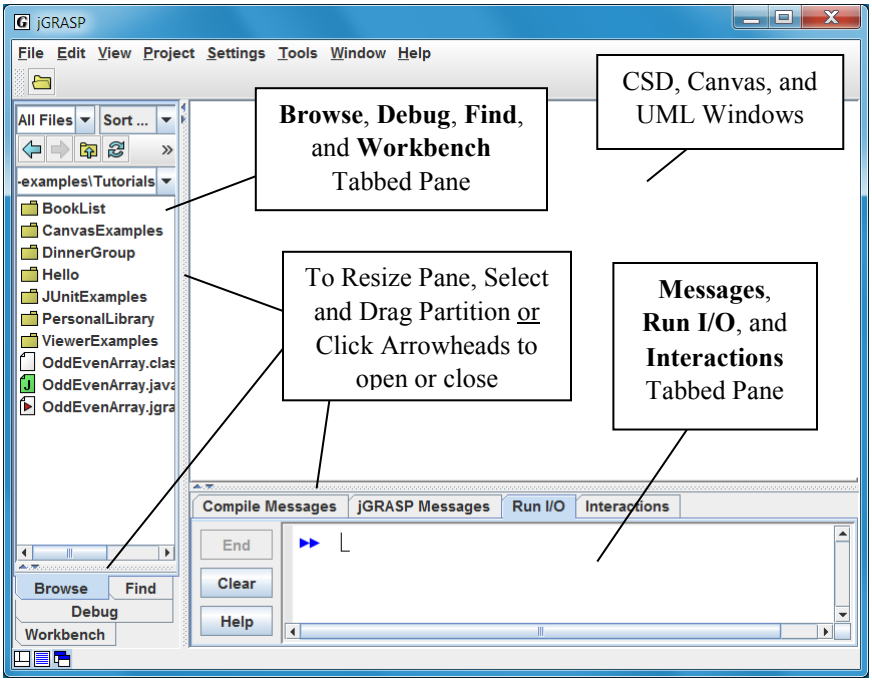


Figure 2-1. The jGRASP Virtual Desktop

2.2 Quick Start - Opening a Program, Compiling, and Running

Example programs are available in the jGRASP folder in the directory where it was installed (e.g., c:\Program Files\jgrasp\examples\Tutorials). Since the installation directory is read-only, you should copy the tutorial folder to one of your personal folders (e.g., in your *My Documents* folder) so that you will be able to edit, compile, etc. To copy the files, find the top menu, click **Tools > Copy Example Files**, then navigate to the folder where you want to save the jgrasp_examples folder.

***Note**: If you already have example programs with which you are familiar, you may* prefer *to use them rather than the ones included with jGRASP as you work through this first tutorial.*

Clicking the Open File button on the toolbar pops up the Open File dialog. However, the easiest way to open existing files is to use the **Browse** tab (below). The files shown initially in the Browse tab will most likely be in your home directory. You can navigate to the appropriate directory by double-clicking on a folder in the list of files or by clicking on as indicated in the figure below. The refresh button updates the Browse tab. Below, the Browse tab is displaying the contents of the Tutorials folder.

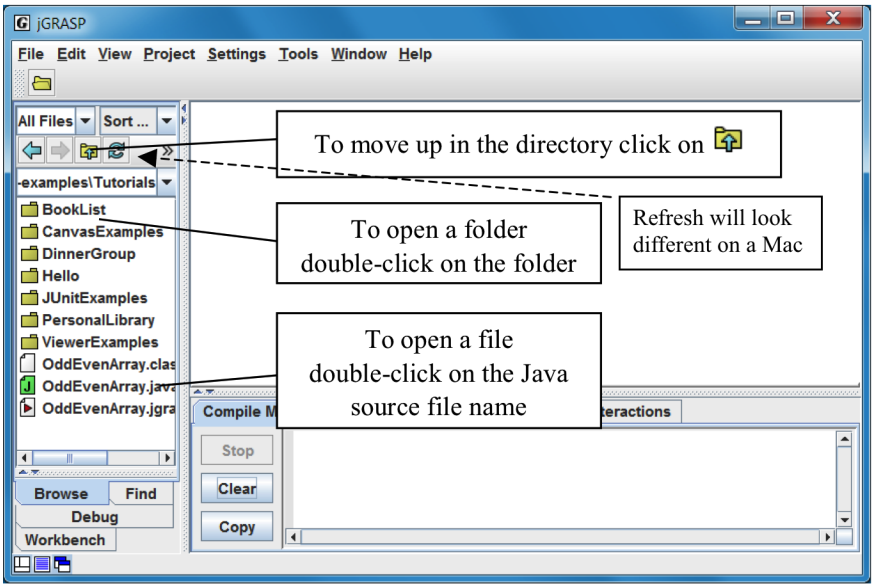


Figure 2-2. The jGRASP Virtual Desktop

Double-clicking on the Hello folder, then the Hello.java file, as shown in **Step 1** below, opens the program in a CSD window. The CSD window is a full-featured editor for entering and updating your programs. Notice that opening the CSD window places additional buttons on the toolbar. Once you have opened a program or entered a new program (**File > New File > Java**) and saved it, you are ready to compile the program and run it. To compile the program, click on the **Build** menu then select **Compile**. Alternatively, you can click on the Compile button (or press **Ctrl-B**) indicated by **Step 2** below. After a * successful compilation – no error messages in the Compile Messages tab (the lower pane), you are ready to run the program by clicking on the Run button (or press **Ctrl-R)** as shown in **Step 3** below, or you can click the **Build** menu and select **Run**. The standard input and output for your program will be in the Run I/O tab pane. Short cuts for Compile and Run are Ctrl-B and Ctrl-R. If Settings>**Auto Compile** is ON (default), **Run** will also **Compile** the file if required.

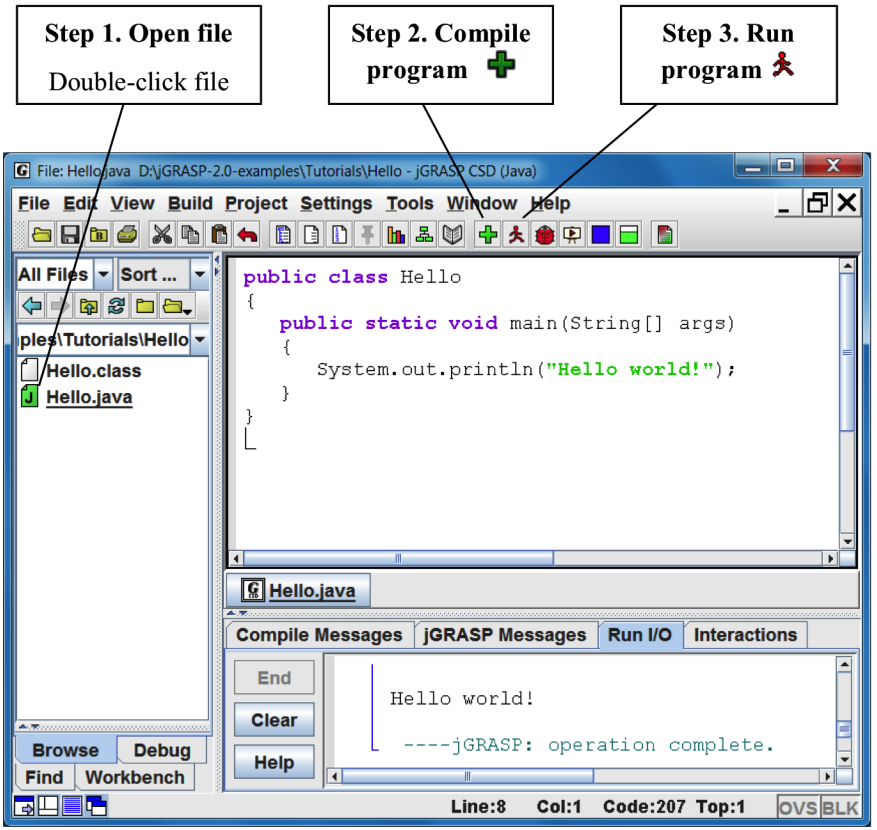


Figure 2-2. After loading file into CSD Window

* A successful compilation will show jGRASP: operation complete.

2.3 Creating a New File

On the main menu, click **File > New File > Java** to create a new Java file. Note that the list of languages displayed by **File > New File** will vary with your use of jGRASP. If the language you want is not listed, click **Other** to see the additional available languages. The languages for the last 25 files opened will be displayed in the initial list; the remaining available languages will be under **Other**.

After you click on **File > New File > Java**, a CSD window is opened in the right pane of the Desktop as shown in Figure 2-4 below. Notice the title for the frame, jGRASP CSD (Java), which indicates that the CSD window is Java specific. If Java is not the language you intend to use, you should close the window and then open a CSD window for the correct language. Notice that a *button* for each open file appears below the CSD windows in an area called the windowbar (similar to a taskbar in the Windows OS environment). Later when you have multiple files open, the windowbar will be quite useful for popping a particular window to the top. The buttons can be reordered by dragging them around on the windowbar. Figure 2-4 shows the newly opened CSD window maximized in the desktop.

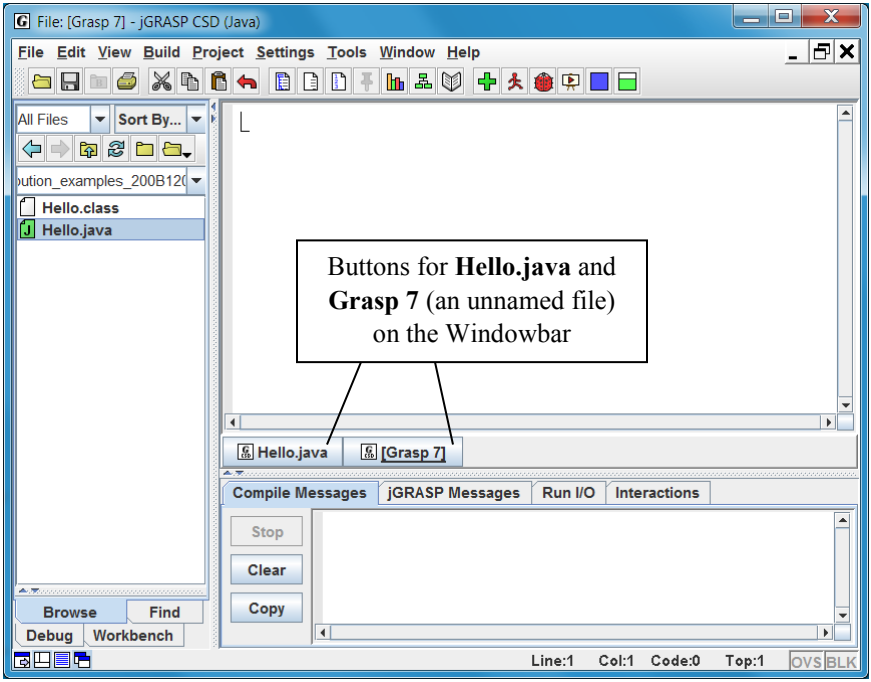


Figure 2-4. After opening a new CSD Window for Java

* When a CSD window is open, there are three buttons to the right of the menu that control its display. The first button minimizes the CSD window; the second button restores (unmaximizes) the CSD window. When the CSD window is not maximized you'll see , and the second button will maximize the CSD window. The third button closes the CSD window. You may also make the Desktop itself full screen by clicking the appropriate button in the upper corner of it.

Figure 2-5 shows the CSD window unmaximized within the virtual Desktop. The flashing "L" shaped cursor in the upper left corner of the empty window indicates where text will be entered.

TIP: You may choose whether or not you want all of your CSD windows to be maximized automatically when you open them. This is the default for new installations of jGRASP. Click **Settings > Desktop**, and then click **Open Desktop Windows Maximized** to toggle this on/off (a check mark indicates that this option is turned ON). You can also click (or) in the lower left corner of the desktop to change this setting. jGRASP will remember this setting even when you update to a new version.

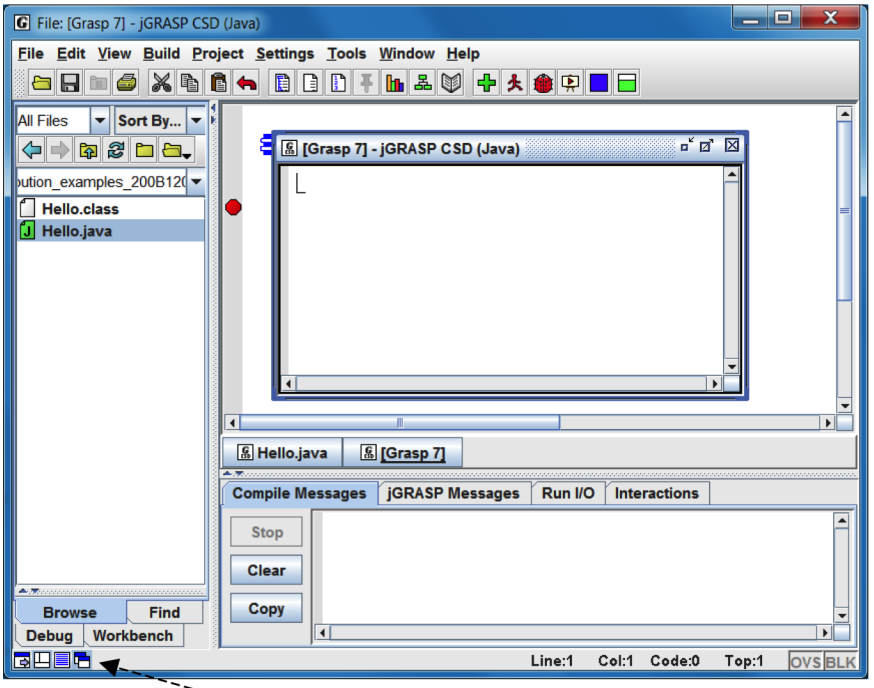


Figure 2-5. CSD Window maximized in Desktop

* if the buttons disappear, just close the file and open it again

Type the following Java program in the CSD window, exactly as it appears. Remember, Java is case sensitive. Alternatively, you may copy/paste the Hello program into this window, then change the class name to Hello2 and add the "Welcome…" line.

```
public class Hello2
{
   public static void main(String[] args)
   {
      System.out.println ("Hello world!");
      System.out.println ("Welcome to jGRASP!");
   }
}
```

After you have entered the program, your CSD window should look similar to the program shown in Figure 2-6.

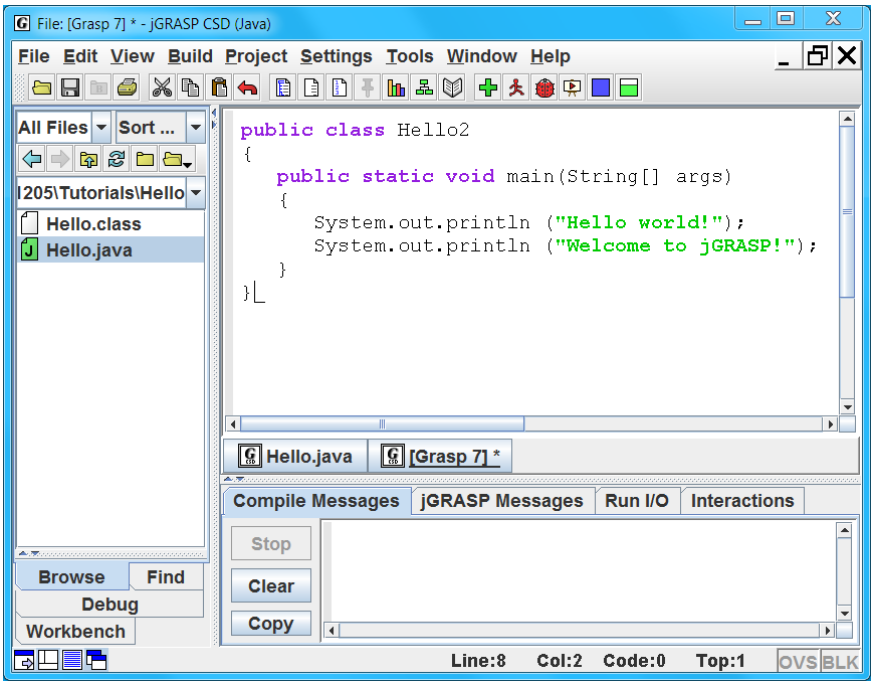


Figure 2-6. CSD Window with program entered

2.4 Saving a File

You can save the program as "Hello2.java" by doing any of the following:

(1) Click the Save button on the toolbar, or

(2) Click **File > Save** on menu, or

(3) Press **Ctrl-S** (i.e., while pressing the Ctrl key, press the "s" key).

(4) With **Auto Compile** ON (the default), attempting to Compile or Run an unsaved file will also initiate a Save operation followed by the Compile and/or Run.

If the file has not been saved previously, the Save dialog (see Figure 2-7) pops up with the name of the file set to the name of the class file. Note, in Java, the file name must match the class name (i.e., class Hello2 must be saved as Hello2.java). Be sure you are in the correct directory. If you need to create a new directory, click the folder button on the top row of the Save dialog. When you are in the proper directory and have the correct file name indicated, click the *Save* button on the dialog. After your program has been saved, it should be listed in the Browse tab (see Figure 2.10). If you do not see the program in the Browse tab, you may need to navigate to the directory where the file was saved or click on the toolbar to change the **Browse** tab to the directory of the current file. Now you are ready to compile and run Hello2.

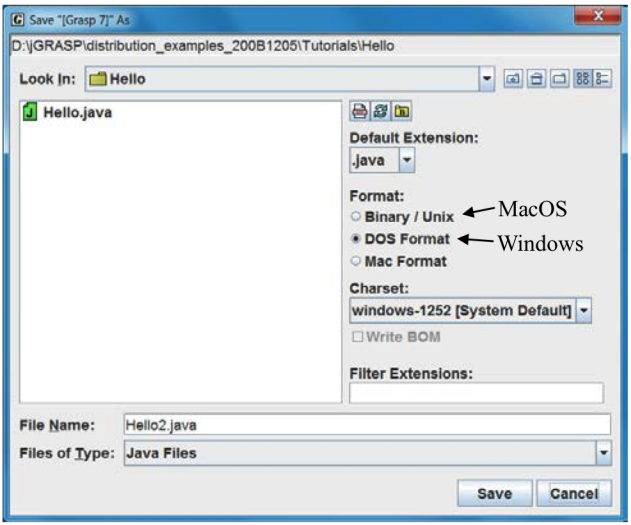


Figure 2-7. Save As dialog for previously unsaved file

2.5 Building Java Programs - - Recap

As seen in the previous sections, Java programs are written in an edit window, saved, compiled, and run. A somewhat more detailed description of steps for building software in Java is as follows.

(1) **Enter** and **Save** your source code in a CSD window. Be sure the file name matches the Java class name and has the ".java" extension (e.g., class MyProgram should be saved as MyProgram.java). You should try to enter your program in chunks so that it will always compile without errors.

(2) **Compile** the source program (e.g., MyProgram.java) to create the byte code file with a ".class" extension (e.g., MyProgram.class). After attempting to compile your program, you may need to make corrections via the edit window (step 1) based on the error messages provided by the compiler and then compile the program again. Note that the .class file is not created until your program compiles with no error messages. Keyboard shortcut: **Ctrl-B**

(3) **Run** your program (main method or applet). In this step, the byte code or .class file produced by the compiler is executed by the Java Virtual Machine. After you run your program, you should inspect the output (if any) to make sure the program did what you intended. At this point, you may need to find and correct logical errors (bugs). After making the corrections in the edit window (step 1), you will need to compile your program again (step 2). Later, we will use the debugger to step through a program so we can see what happens after each individual statement is executed. Keyboard shortcut: **Ctrl-R**

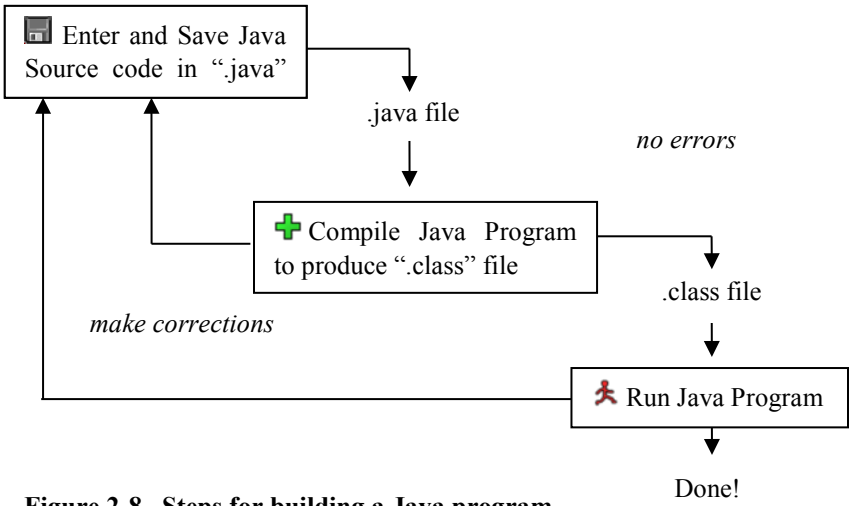


Figure 2-8. Steps for building a Java program

2.6 Generating a Control Structure Diagram

You can generate a Control Structure Diagram in the CSD window whenever you have a syntactically correct program, such as the Hello2.java program described above. Note that CSD generation checks only the structure of a program, so even though the CSD may generate successfully, the program may not compile. Generate the CSD for the program by doing any of the following:

(1) Click the Generate CSD button  , or

(2) Click **View > Generate CSD** on the menu, or

(3) Press the **F2** key. (maybe not for Mac users)

If your program is syntactically correct, the CSD will be generated as shown for the Hello2.java program in Figure 2-10. After you are able to successfully generate a CSD, go on to the next section below.

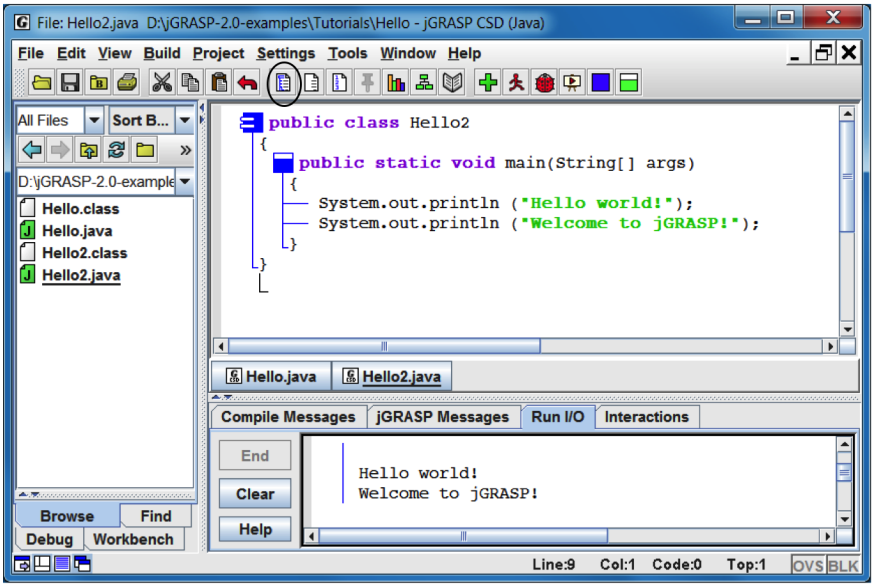


Figure 2-10. After CSD is generated

If a syntax error is detected during the CSD generation, jGRASP will highlight the vicinity of the error and describe it in the message window.

If you do not find an error in the highlighted line, be sure to look for the error in the line just above it. For example in Figure 2-11, the semi-colon was omitted at the end of the first println statement. As you gain experience, these errors will become easier to spot.

If you are unable find and correct the error, you should try compiling the program since the compiler may provide a more detailed error message (e.g., `';' expected`) as shown in Figure 2-11.

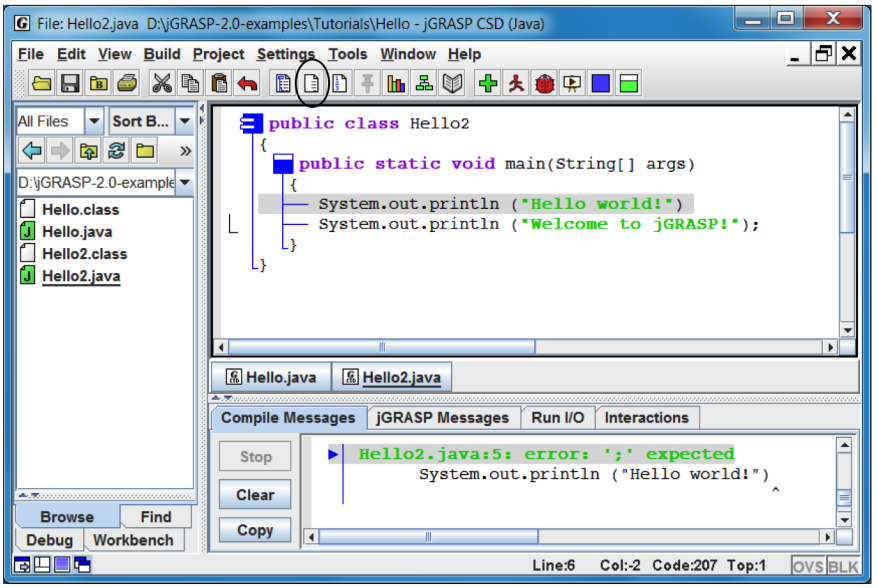


Figure 2-11. Syntax error detected

Removing the CSD leaves the code 3-space indented and removes the leading spaces that were occupied by the CSD. You can remove the CSD by doing any of the following:

(1) Click the Remove CSD button , or

(2) Click **View > Remove CSD** on the menu, or

(3) Press **Shift-F2**. (maybe not for Mac users)

Note that it is not necessary to ever remove the CSD since the CSD is never saved with your programs. Many users turn on Auto Generate (**View** > then check ON **Auto Generate CSD**) so that the CSD will be generated when a file is opened and again anytime the file is compiled.

Remember, the purpose of using the CSD is to improve the readability of your program. While this may not be obvious on a simple program like the example

above, it should become apparent as the size and complexity of your programs increase, especially when they contain nested control structures.

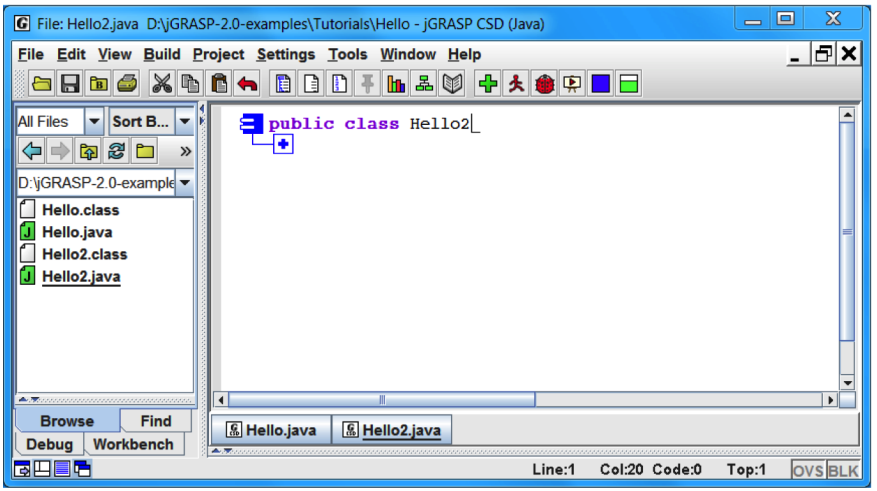
TIP: As you enter a program, try to enter it in "chunks" that are syntactically correct. For example, the following is sufficient to generate the CSD.

```
public class Hello
{
}
```

As soon as you think you have entered a syntactically correct chunk, you should generate the CSD. Not only does this update the diagram, it catches your syntax errors early. Alternatively, if you have Auto Generate (**View** > then check ON **Auto Generate CSD**) turned on, CSD will generated/updated each time you successfully compile your program.

Folding a CSD is a feature that becomes increasingly useful as programs get larger. After you have generated the CSD, you can fold your program based on its structure. For example, if you double-click on the class symbol * in the program, the entire program is folded (Figure 2-12). Double-clicking on the class symbol again will unfold the program completely.

If you double-click on the "plus" symbol, the first layer of the program will be unfolded. Large programs can be unfolded layer by layer as needed. Although the example program has no loops or conditional statements, these may be folded by double-clicking the corresponding CSD control constructs. For other folding options, see the **View > Fold** menu.



* or View > Fold > Fold

Figure 2-12. Folded CSD

2.7 Using Line Numbers

Line numbers can be very useful when referring to specific lines or regions of a program. Although not part of the actual program, they are displayed to the left of the source code as indicated in Figure 2-13.

 Line numbers can be turned on and off by clicking the line numbers toggle button on the CSD window toolbar or via the View menu where you can check
* or uncheck the check box for Line Numbers. In addition, **Ctrl-L** toggles line numbers on and off.

With Line numbers turned on, if you insert a line in the code, all line numbers below the new line are incremented.

 You may "freeze" and "unfreeze" the line numbers by clicking on the Freeze Line Numbers button. The default is for line numbers to increment when the ENTER key is pressed. When you freeze line numbers, they are not incremented when ENTER is pressed. This is useful if you are looking at several messages that reference line numbers in your program. As you make changes, you may not want the line numbers to change until you have made all of the changes. The line numbers will not be updated until you unfreeze the line numbers by clicking the button  again. This feature is also available on the View menu.

* command L on Mac

2.14 Closing a File

The open files in CSD windows can be closed in several ways.

(1)  If the CSD window is maximized, you can close the window and file by clicking the Close button at the right end of the top level Menu.

(2)  If the CSD window is not maximized, click the Close button in the upper right corner of the CSD window itself.

(3) **File Menu** – Click **File > Close** or **Close All Files**. The latter closes all CSD windows. To close "all" UML windows, you need close all projects by clicking **Projects** > **Close All**.

(4) **Window Menu** – Click **Window > Close All Windows**.

In each of the scenarios above, if the file has been modified and not saved, you will be prompted to *Save and Exit*, *Discard Edits*, or *Cancel* before continuing.

2.15 Exiting jGRASP

When you have completed your session with jGRASP, you should always close (or "exit") jGRASP rather than let your computer close it when you log out or shut down. However, you don't have to close the files you have been working on before exiting jGRASP. When you exit jGRASP, it remembers the files you have open, including their window size and scroll position, before closing them. If a file was edited during the session, jGRASP prompts you to save or discard the changes. The next time you start jGRASP, it will open your files, and you will be ready to begin where you left off. For example, open the Hello.java file and then exit jGRASP by one of the methods below. After jGRASP closes down, start it up again and you should see the Hello.java program in a CSD window. This feature is so convenient that many users tend to leave a few files open when they exit jGRASP. However, if a file is really not being used, it is best to go ahead and close the file to reduce the clutter on the windowbar.

Close jGRASP in either of the following ways:

(1) Click the Close button  in the upper right corner of the desktop; or

(2) On the File menu, click **File > Exit jGRASP**.

2.16 Review and Preview of What's Ahead

As a way of review and also to look ahead, let's take a look at the jGRASP *toolbar*. Hovering the mouse over a button on the toolbar provides a "tool hint" to help identify its function. Also, **View > Toolbar Buttons** allows you to display *text* labels on the buttons. Figure 2-27 shows the name/function of each button.

While many of these buttons were introduced in this section, some were assumed to be self-explanatory (e.g., Print, Cut, Copy, etc.), and several others will be covered in other tutorials).

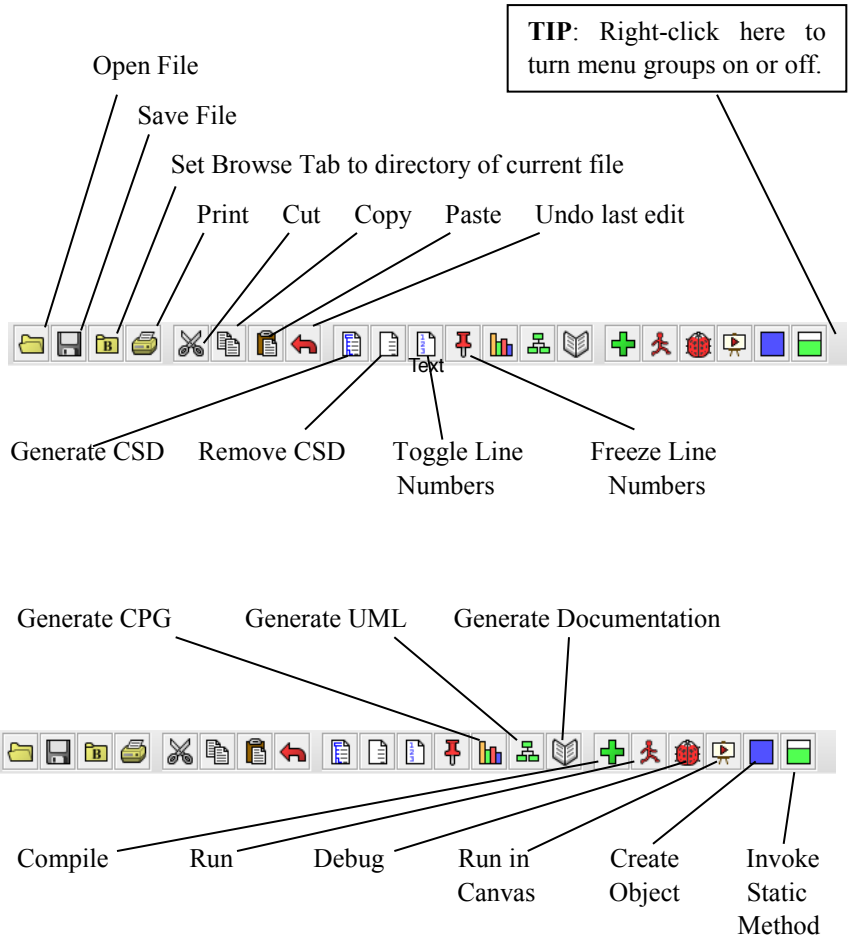


Figure 2-27. Toolbar