



Using an Integrated Development Environment



Integrated Development Environments

- An Integrated Development Environment, or IDE, permits you to edit, compile, test, and debug a program using a single program.
 - There are several IDEs for Java.
 - We will look at one of them, jGRASP, today.



Objectives

- You will be able to use the jGRASP IDE to edit, compile, test, and debug Java programs.

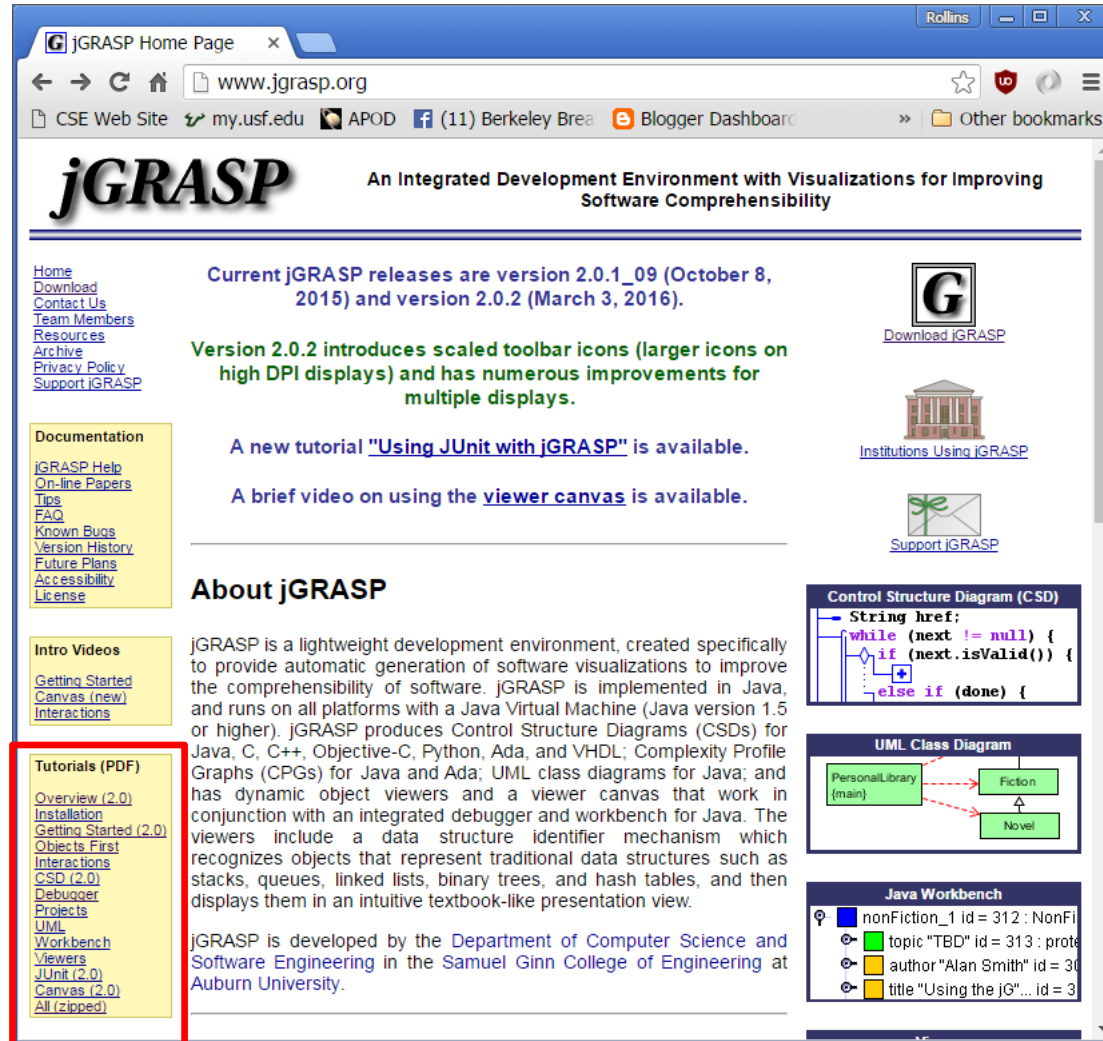


Installing jGRASP

- jGRASP is available on lab computers.
- To install it on your own computer, see
 - http://www.csee.usf.edu/~turnerr/Programming_Concepts/120_Installing_jGRASP.pdf
- Versions are available for
 - Windows
 - Mac
 - Linux

jGRASP Tutorials

There is an extensive set of jGRASP tutorials, with links on the start page:



The screenshot shows the jGRASP Home Page in a web browser. The page features a navigation menu on the left with links to Home, Download, Contact Us, Team Members, Resources, Archive, Privacy Policy, and Support jGRASP. Below this is a 'Documentation' section with links to jGRASP Help, On-line Papers, Tips, FAQ, Known Bugs, Version History, Future Plans, Accessibility, and License. There is also an 'Intro Videos' section with links to Getting Started, Canvas (new), and Interactions. A 'Tutorials (PDF)' section is highlighted with a red box, containing links to Overview (2.0), Installation, Getting Started (2.0), Objects First, Interactions, CSD (2.0), Debugger, Projects, UML, Workbench, Viewers, JUnit (2.0), Canvas (2.0), and All (zipped). The main content area includes the jGRASP logo, the tagline 'An Integrated Development Environment with Visualizations for Improving Software Comprehensibility', and information about current releases (2.0.1_09 and 2.0.2). It also mentions that Version 2.0.2 introduces scaled toolbar icons and has numerous improvements for multiple displays. A new tutorial 'Using JUnit with jGRASP' is available, along with a brief video on using the viewer canvas. The 'About jGRASP' section describes it as a lightweight development environment for software visualization. The right sidebar contains links to Download jGRASP, Institutions Using jGRASP, and Support jGRASP. At the bottom, there are three diagrams: a Control Structure Diagram (CSD) showing a while loop with an if statement, a UML Class Diagram showing relationships between PersonalLibrary, Fiction, and Novel, and a Java Workbench showing a list of objects with their IDs and names.

jGRASP Home Page

www.jgrasp.org

CSE Web Site my.usf.edu APOD (11) Berkeley Brea Blogger Dashboard Other bookmarks

jGRASP

An Integrated Development Environment with Visualizations for Improving Software Comprehensibility

Home
Download
Contact Us
Team Members
Resources
Archive
Privacy Policy
Support jGRASP

Current jGRASP releases are version 2.0.1_09 (October 8, 2015) and version 2.0.2 (March 3, 2016).

Version 2.0.2 introduces scaled toolbar icons (larger icons on high DPI displays) and has numerous improvements for multiple displays.

A new tutorial "Using JUnit with jGRASP" is available.

A brief video on using the viewer canvas is available.

About jGRASP

jGRASP is a lightweight development environment, created specifically to provide automatic generation of software visualizations to improve the comprehensibility of software. jGRASP is implemented in Java, and runs on all platforms with a Java Virtual Machine (Java version 1.5 or higher). jGRASP produces Control Structure Diagrams (CSDs) for Java, C, C++, Objective-C, Python, Ada, and VHDL; Complexity Profile Graphs (CPGs) for Java and Ada; UML class diagrams for Java; and has dynamic object viewers and a viewer canvas that work in conjunction with an integrated debugger and workbench for Java. The viewers include a data structure identifier mechanism which recognizes objects that represent traditional data structures such as stacks, queues, linked lists, binary trees, and hash tables, and then displays them in an intuitive textbook-like presentation view.

jGRASP is developed by the Department of Computer Science and Software Engineering in the Samuel Ginn College of Engineering at Auburn University.

Download jGRASP

Institutions Using jGRASP

Support jGRASP

Control Structure Diagram (CSD)

```
String href;  
while (next != null) {  
    if (next.isValid()) {  
        // ...  
    }  
    else if (done) {  
        // ...  
    }  
}
```

UML Class Diagram

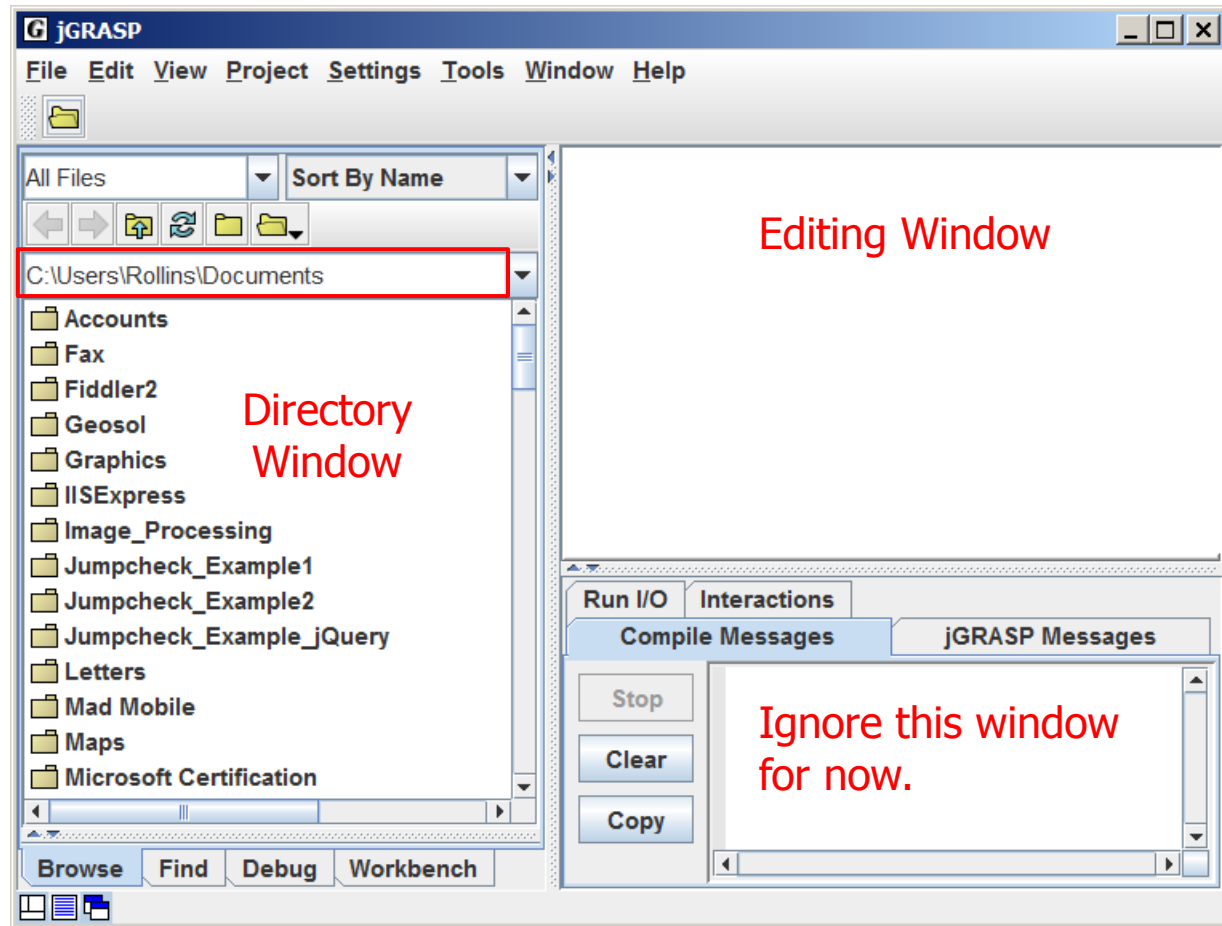
```
classDiagram  
    class PersonalLibrary {  
        main  
    }  
    class Fiction  
    class Novel  
    PersonalLibrary --> Fiction  
    Fiction --|> Novel
```

Java Workbench

- nonFiction_1 id = 312 : NonFi
- topic "TBD" id = 313 : prot
- author "Alan Smith" id = 3
- title "Using the jG" id = 3

Using jGRASP

Navigation Buttons
Initial Directory



Directory
Window

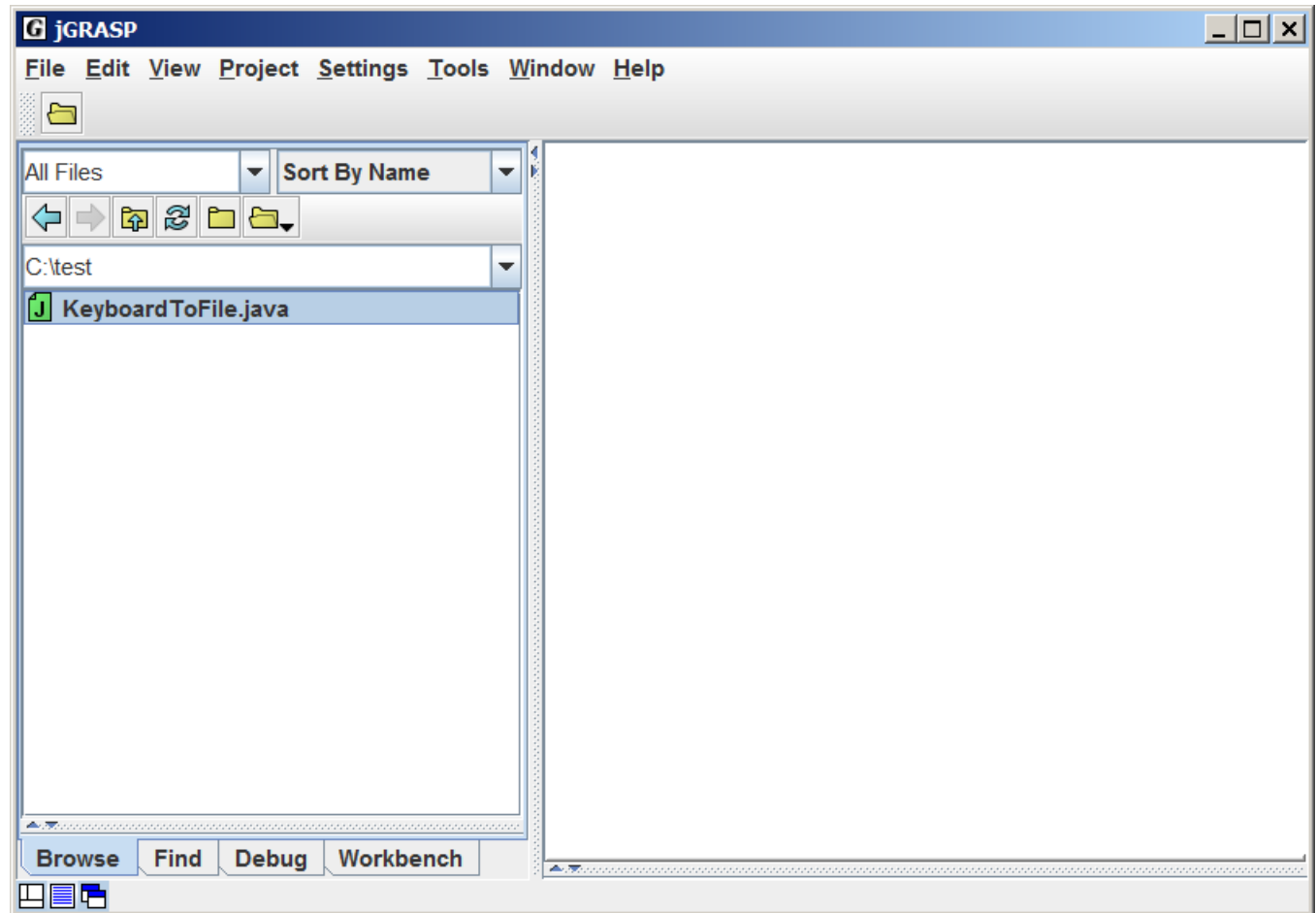
Editing Window

Ignore this window
for now.

Start page.

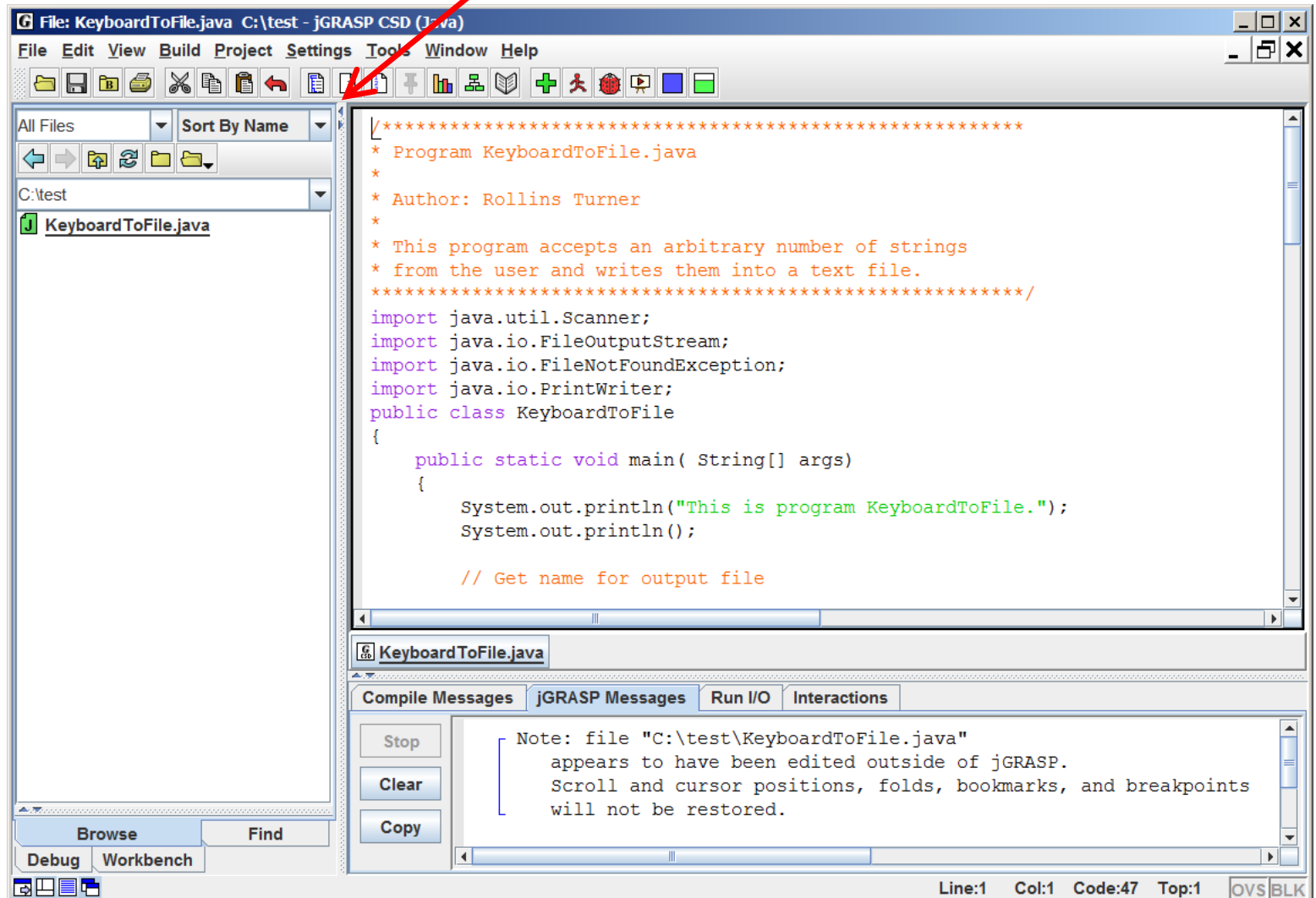
Navigate to Your Test Directory

Double click on file name to open file in editing window.

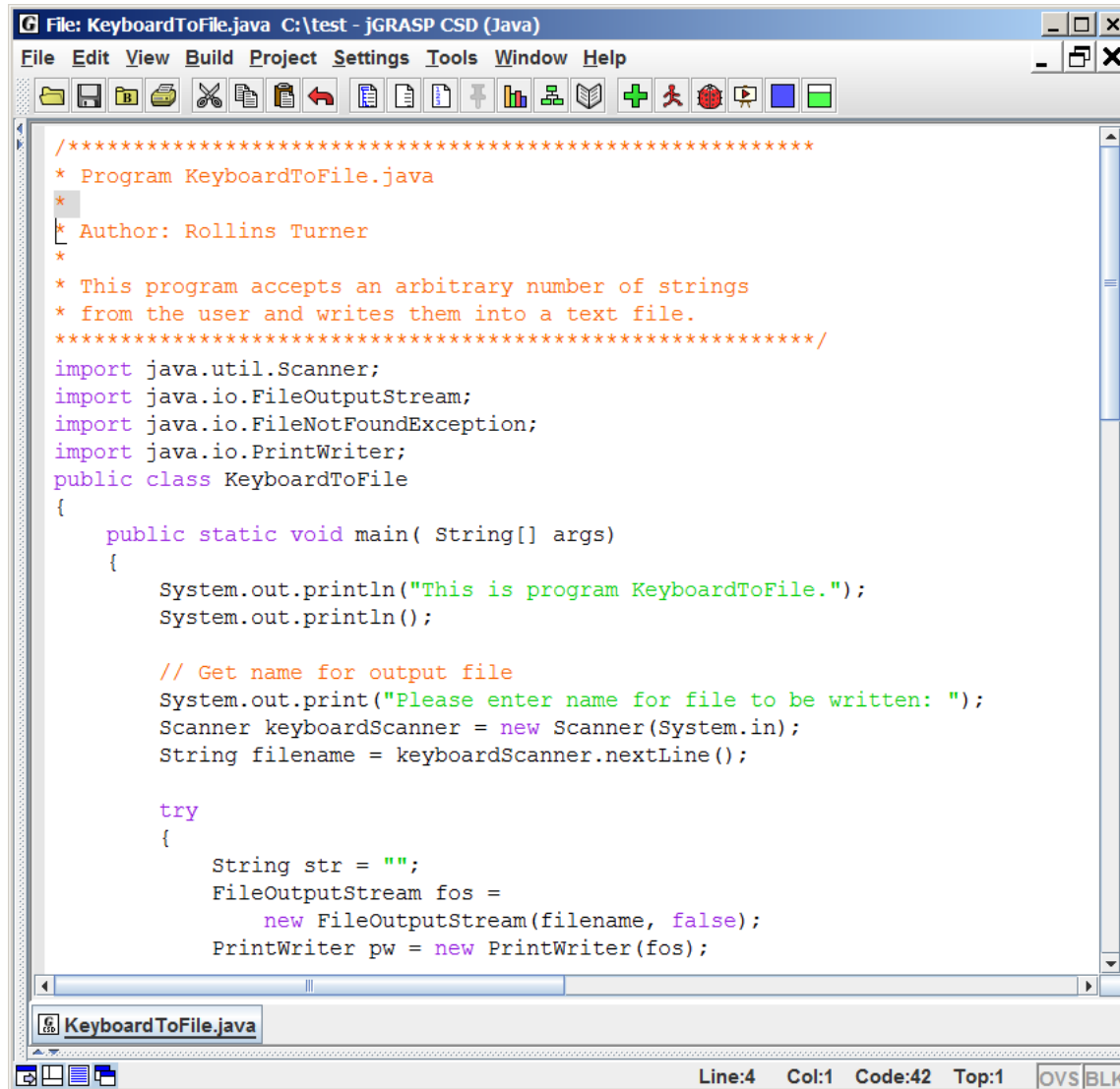


Source File Open in Editing Window

Click here to maximize editing window.



Editing Window



The screenshot shows a Java IDE window titled "File: KeyboardToFile.java C:\test - jGRASP CSD (Java)". The window contains a Java program named "KeyboardToFile.java". The code is as follows:

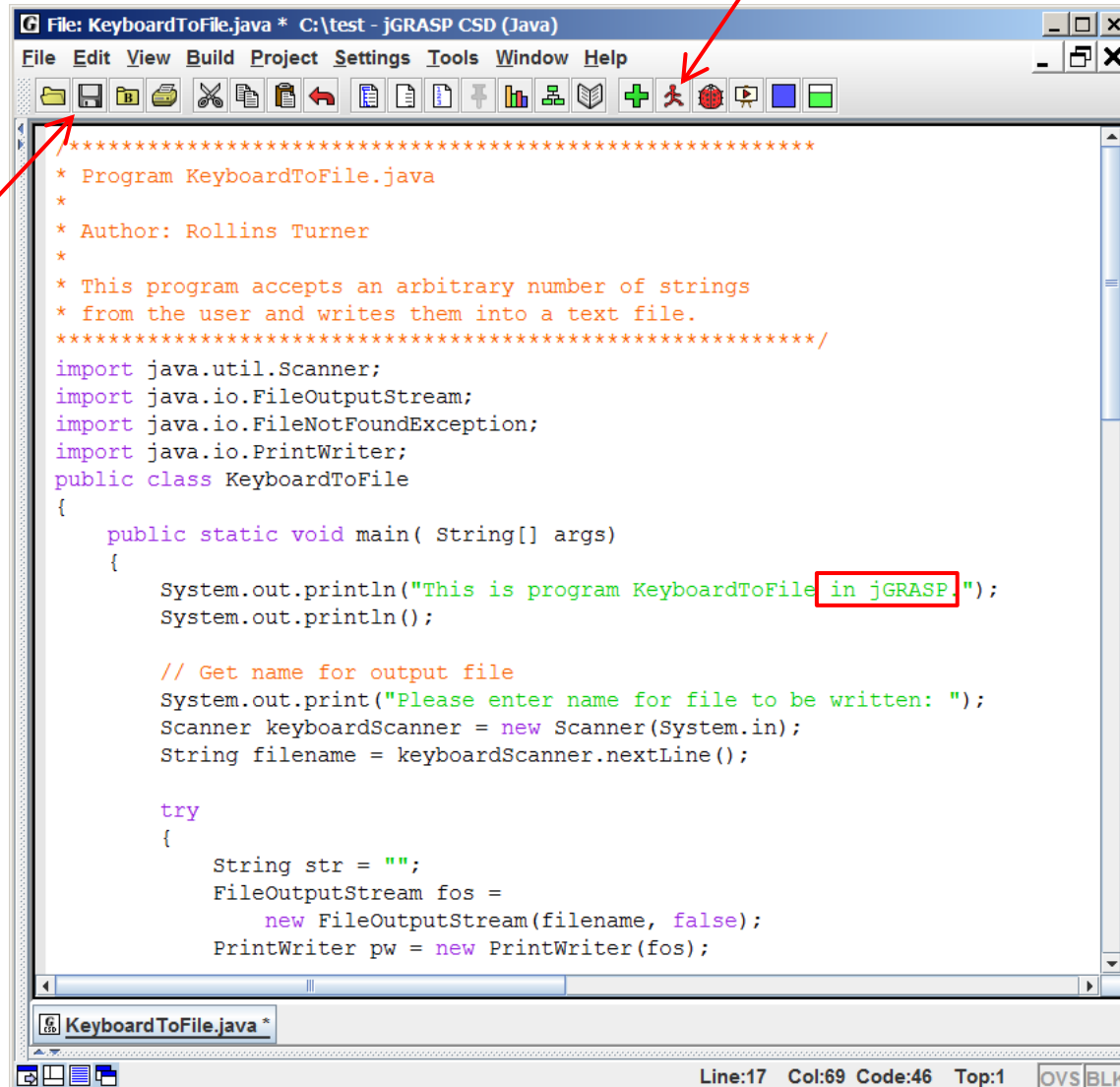
```
/* *****  
 * Program KeyboardToFile.java  
 *  
 * Author: Rollins Turner  
 *  
 * This program accepts an arbitrary number of strings  
 * from the user and writes them into a text file.  
 * ***** */  
import java.util.Scanner;  
import java.io.FileOutputStream;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
public class KeyboardToFile  
{  
    public static void main( String[] args)  
    {  
        System.out.println("This is program KeyboardToFile.");  
        System.out.println();  
  
        // Get name for output file  
        System.out.print("Please enter name for file to be written: ");  
        Scanner keyboardScanner = new Scanner(System.in);  
        String filename = keyboardScanner.nextLine();  
  
        try  
        {  
            String str = "";  
            FileOutputStream fos =  
                new FileOutputStream(filename, false);  
            PrintWriter pw = new PrintWriter(fos);
```

The status bar at the bottom indicates "Line:4 Col:1 Code:42 Top:1" and "OVS|BLK".

Editing Window

Click here to run program.

Click here to save.



```
*****
* Program KeyboardToFile.java
*
* Author: Rollins Turner
*
* This program accepts an arbitrary number of strings
* from the user and writes them into a text file.
*****/
import java.util.Scanner;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
public class KeyboardToFile
{
    public static void main( String[] args)
    {
        System.out.println("This is program KeyboardToFile in jGRASP.");
        System.out.println();

        // Get name for output file
        System.out.print("Please enter name for file to be written: ");
        Scanner keyboardScanner = new Scanner(System.in);
        String filename = keyboardScanner.nextLine();

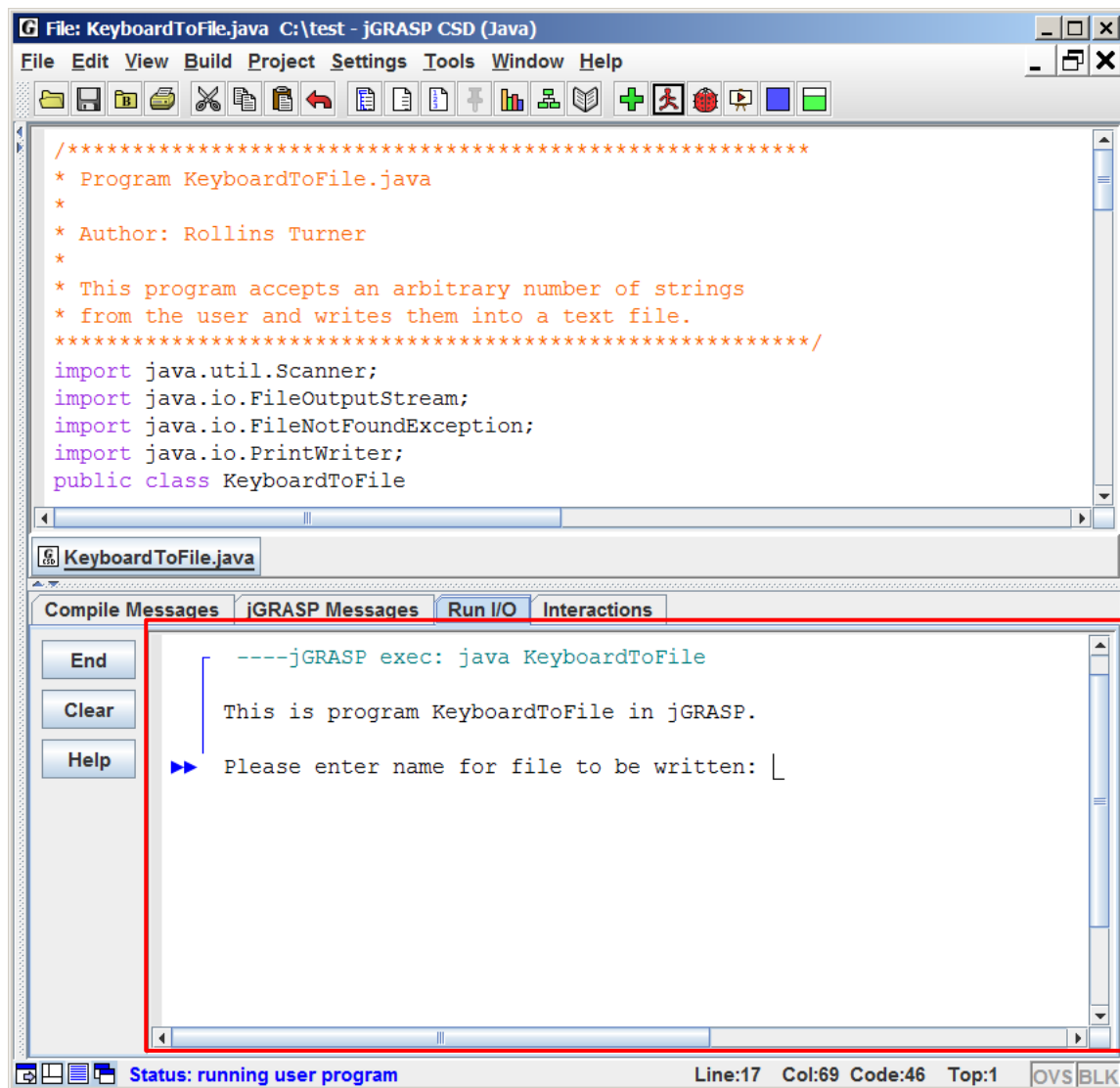
        try
        {
            String str = "";
            FileOutputStream fos =
                new FileOutputStream(filename, false);
            PrintWriter pw = new PrintWriter(fos);
        }
    }
}
```

KeyboardToFile.java *

Line:17 Col:69 Code:46 Top:1 OVS BLK

Make a change.

Program Running



The screenshot shows the jGRASP IDE interface. The main editor window displays the source code for `KeyboardToFile.java`. The code includes a multi-line comment describing the program's purpose and author, followed by imports for `Scanner`, `FileOutputStream`, `FileNotFoundException`, and `PrintWriter`, and the start of the `KeyboardToFile` class. Below the editor, the `Run I/O` tab is selected in the console area. The console output shows the command `----jGRASP exec: java KeyboardToFile` and the program's output: `This is program KeyboardToFile in jGRASP.` followed by a prompt `Please enter name for file to be written:` with a cursor. The status bar at the bottom indicates `Status: running user program`, `Line:17 Col:69 Code:46 Top:1`, and the theme is `OVS/BLK`.

```
File: KeyboardToFile.java C:\test - jGRASP CSD (Java)
File Edit View Build Project Settings Tools Window Help

/*****
 * Program KeyboardToFile.java
 *
 * Author: Rollins Turner
 *
 * This program accepts an arbitrary number of strings
 * from the user and writes them into a text file.
 *****/
import java.util.Scanner;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
public class KeyboardToFile

KeyboardToFile.java
Compile Messages jGRASP Messages Run I/O Interactions

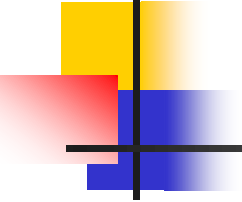
End
Clear
Help

----jGRASP exec: java KeyboardToFile

This is program KeyboardToFile in jGRASP.
▶ Please enter name for file to be written: |

Status: running user program Line:17 Col:69 Code:46 Top:1 OVS/BLK
```

Console
Window



File: KeyboardToFile.java C:\test - jGRASP CSD (Java)

File Edit View Build Project Settings Tools Window Help

```
/*  
 * Program KeyboardToFile.java  
 *  
 * Author: Rollins Turner  
 *  
 * This program accepts an arbitrary number of strings  
 * from the user and writes them into a text file.  
 */  
import java.util.Scanner;  
import java.io.FileOutputStream;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
public class KeyboardToFile
```

KeyboardToFile.java

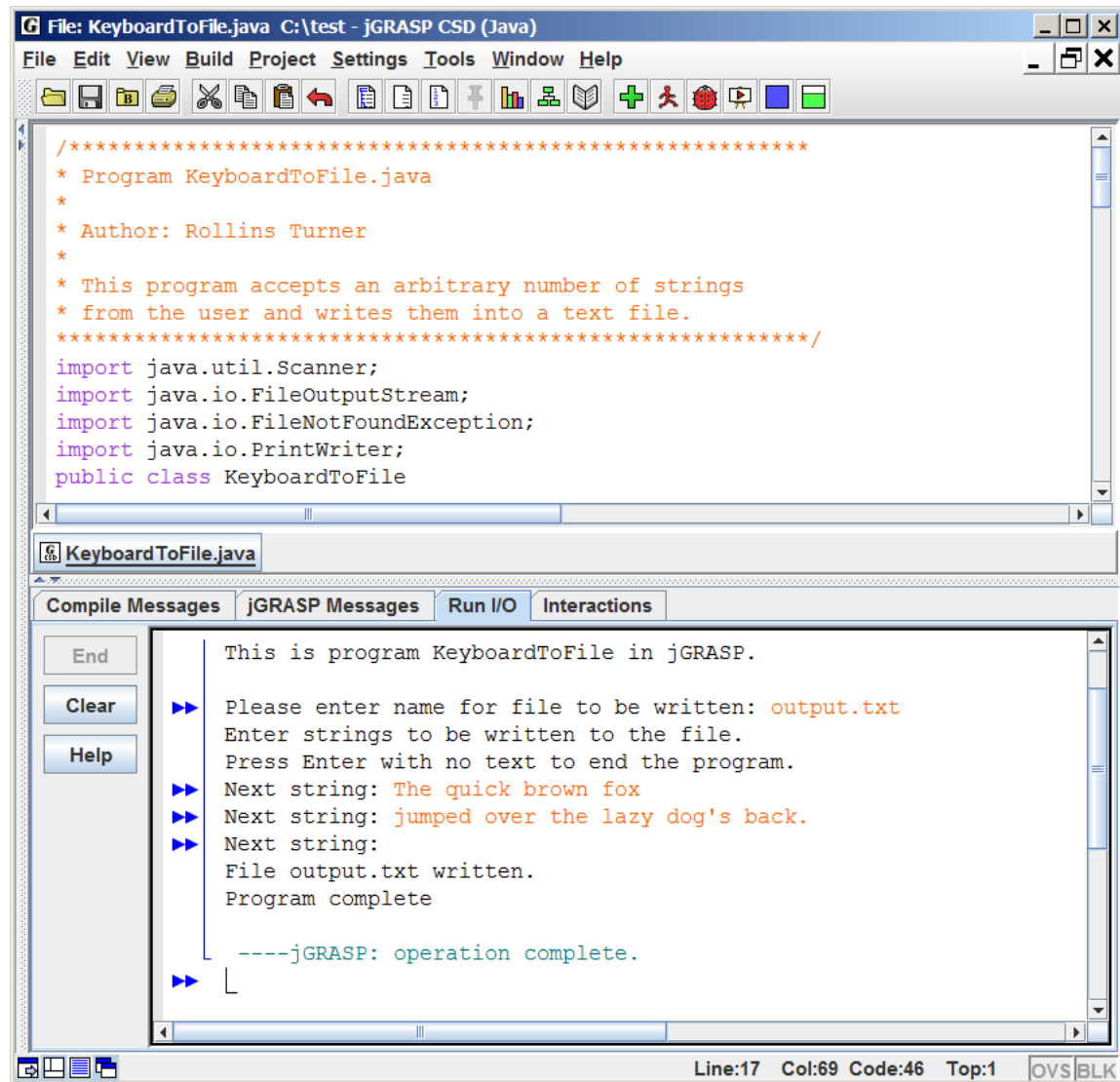
Compile Messages jGRASP Messages Run I/O Interactions

End
Clear
Help

```
----jGRASP exec: java KeyboardToFile  
This is program KeyboardToFile in jGRASP.  
Please enter name for file to be written: output.txt
```

Status: running user program Line:17 Col:69 Code:46 Top:1 OVS BLK

Program Running



The screenshot displays the jGRASP IDE interface. The top pane shows the source code for `KeyboardToFile.java`. The bottom pane, titled "Run I/O", shows the program's execution. The output indicates that the program successfully writes user input to `output.txt` and completes its operation.

```
File: KeyboardToFile.java C:\test - jGRASP CSD (Java)
File Edit View Build Project Settings Tools Window Help

/*****
 * Program KeyboardToFile.java
 *
 * Author: Rollins Turner
 *
 * This program accepts an arbitrary number of strings
 * from the user and writes them into a text file.
 *****/
import java.util.Scanner;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
public class KeyboardToFile

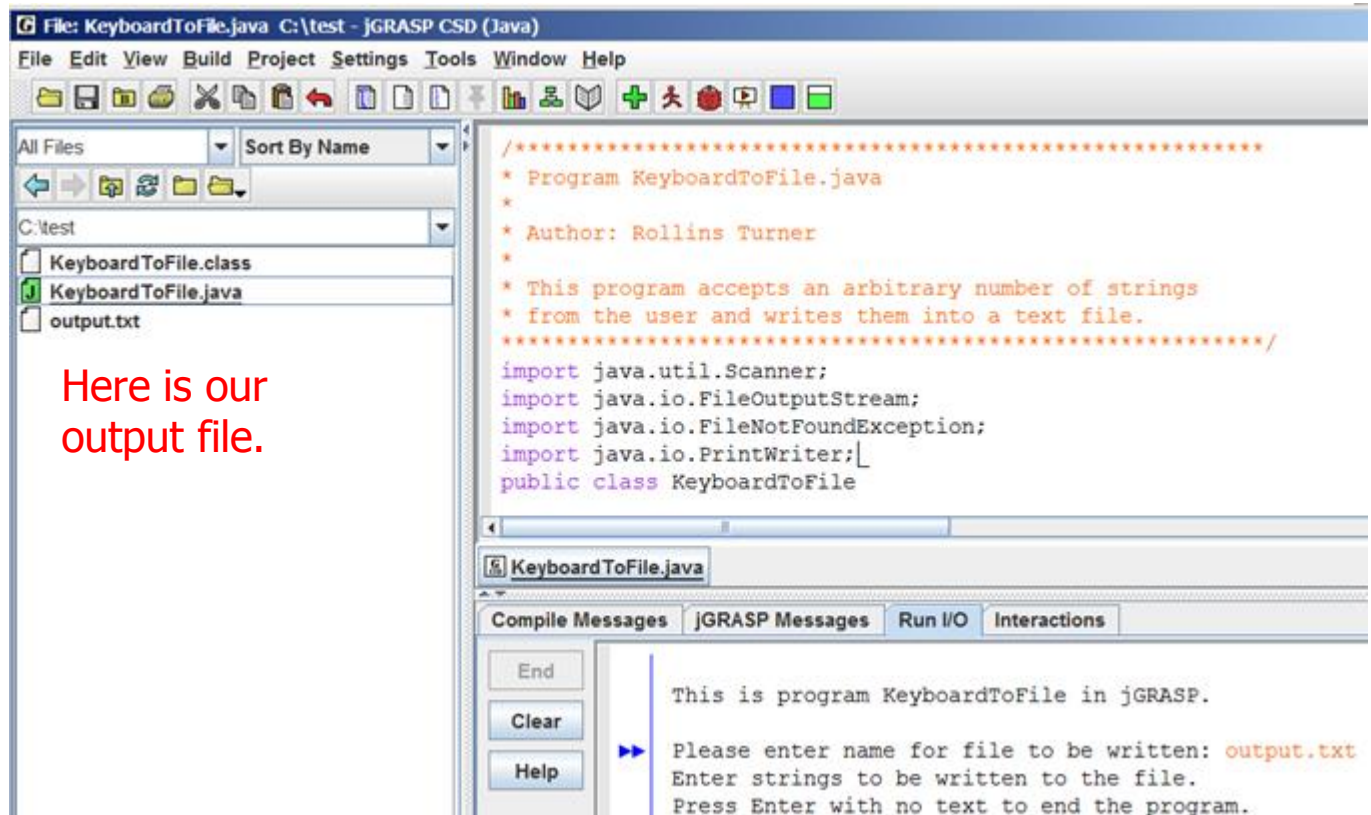
This is program KeyboardToFile in jGRASP.

>> Please enter name for file to be written: output.txt
Enter strings to be written to the file.
Press Enter with no text to end the program.
>> Next string: The quick brown fox
>> Next string: jumped over the lazy dog's back.
>> Next string:
File output.txt written.
Program complete

----jGRASP: operation complete.
>> L
```

Line:17 Col:69 Code:46 Top:1 OVS BLK

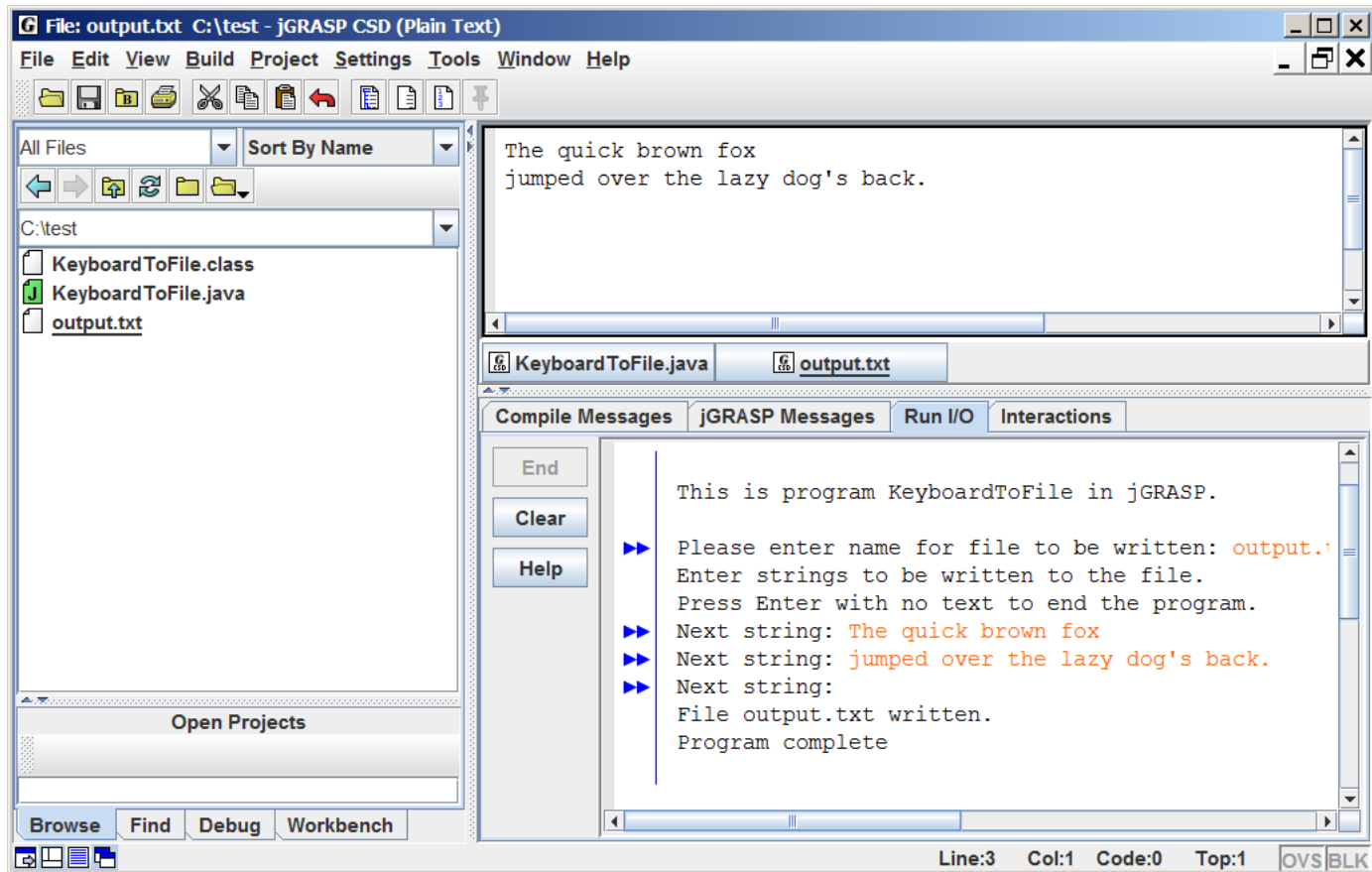
Expand and Refresh Directory Window



Here is our
output file.

Double click file
name to display
the file.

Output File Contents



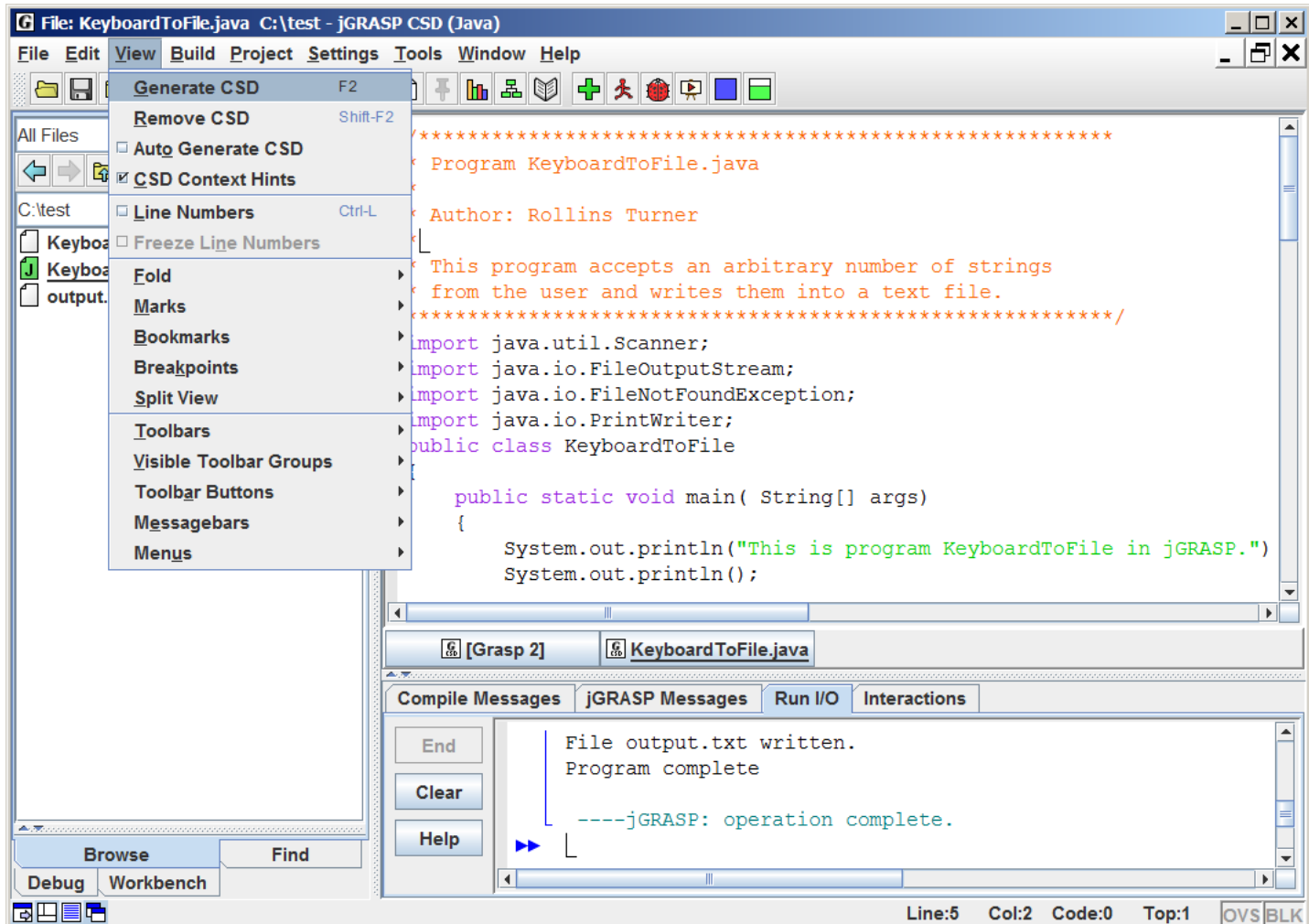


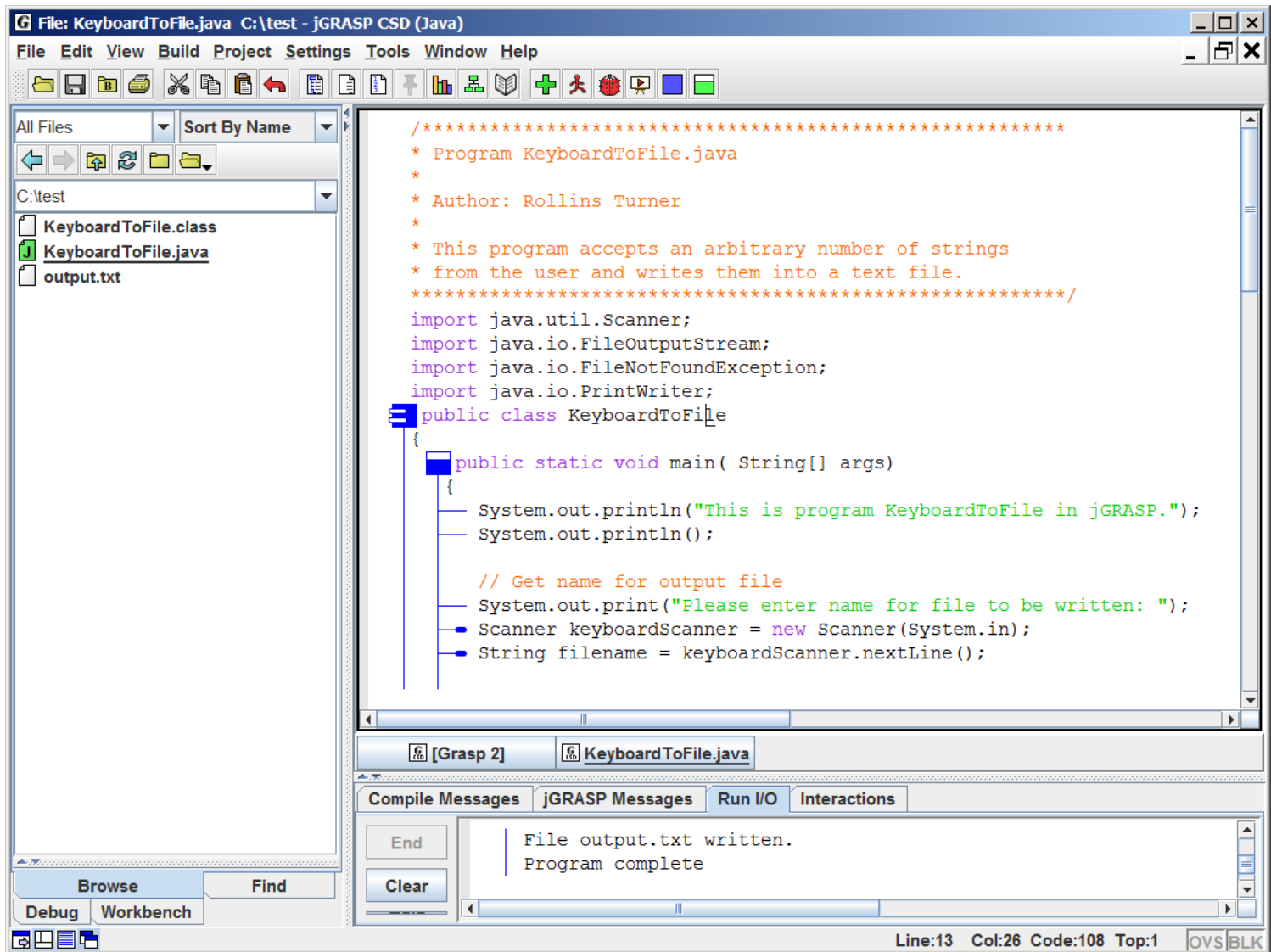
Control Structure Diagrams

From the Overview tutorial:

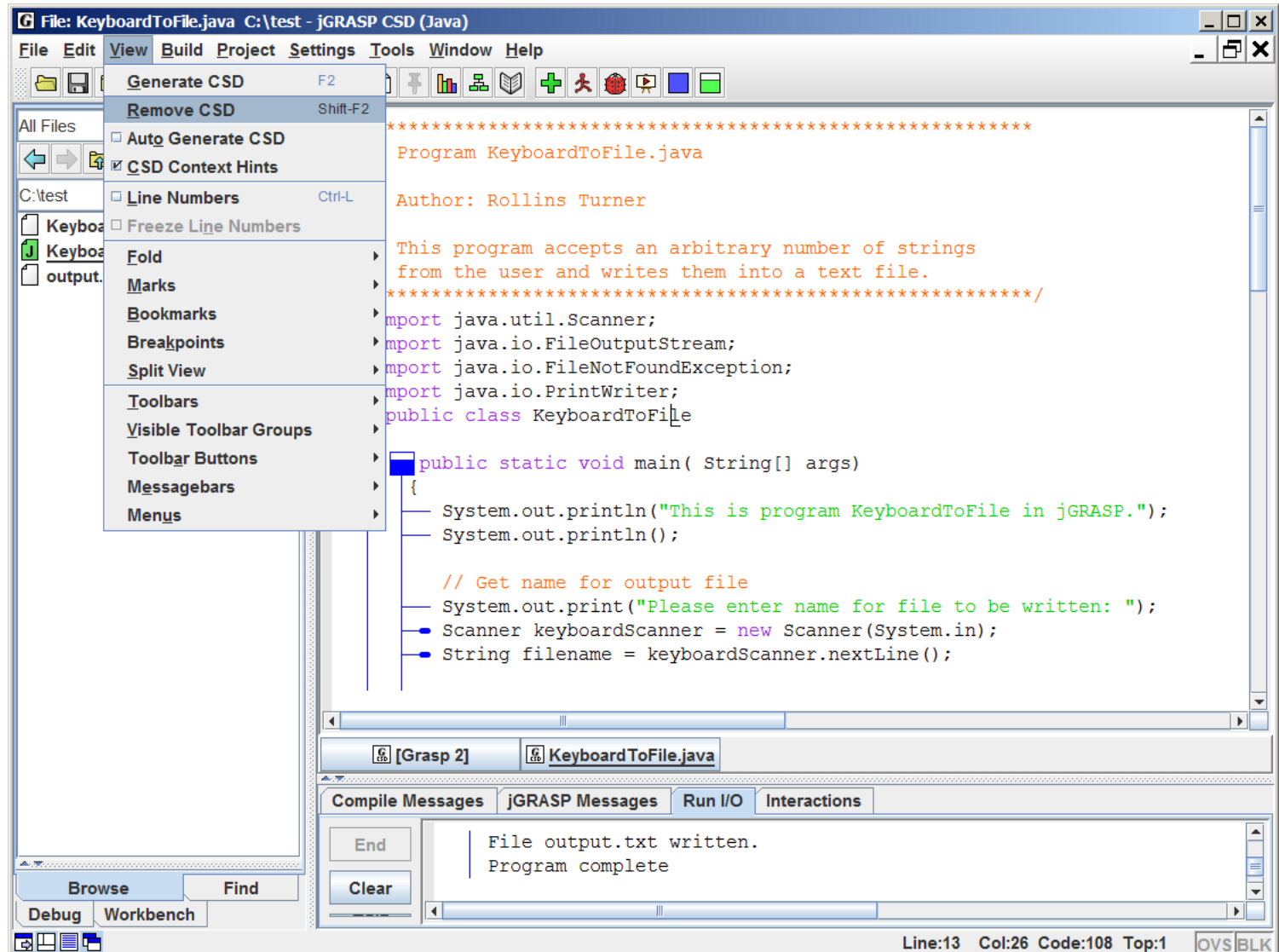
The **Control Structure Diagram (CSD)** is an algorithmic level diagram which is generated for Ada, C, C++, Objective-C, Java, VHDL, and Python. The CSD is intended to improve the comprehensibility of source code by clearly depicting control constructs, control paths, and the overall structure of each program unit. The CSD, designed to fit into the space that is normally taken by indentation in source code, is an alternative to flow charts and other graphical representations of algorithms. The CSD is a natural extension to architectural diagrams such as UML class diagrams.

Generating a Control Structure Diagram

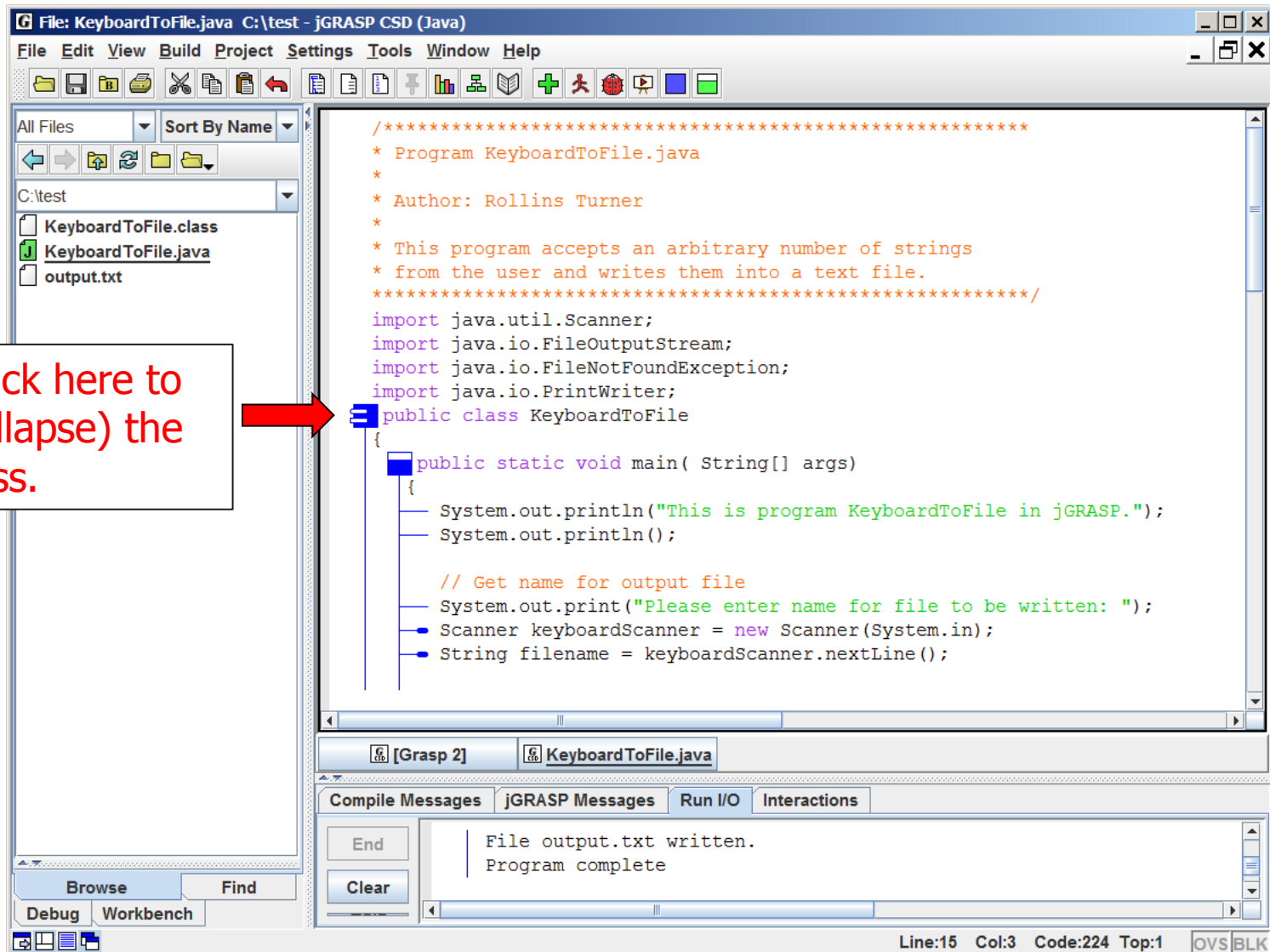


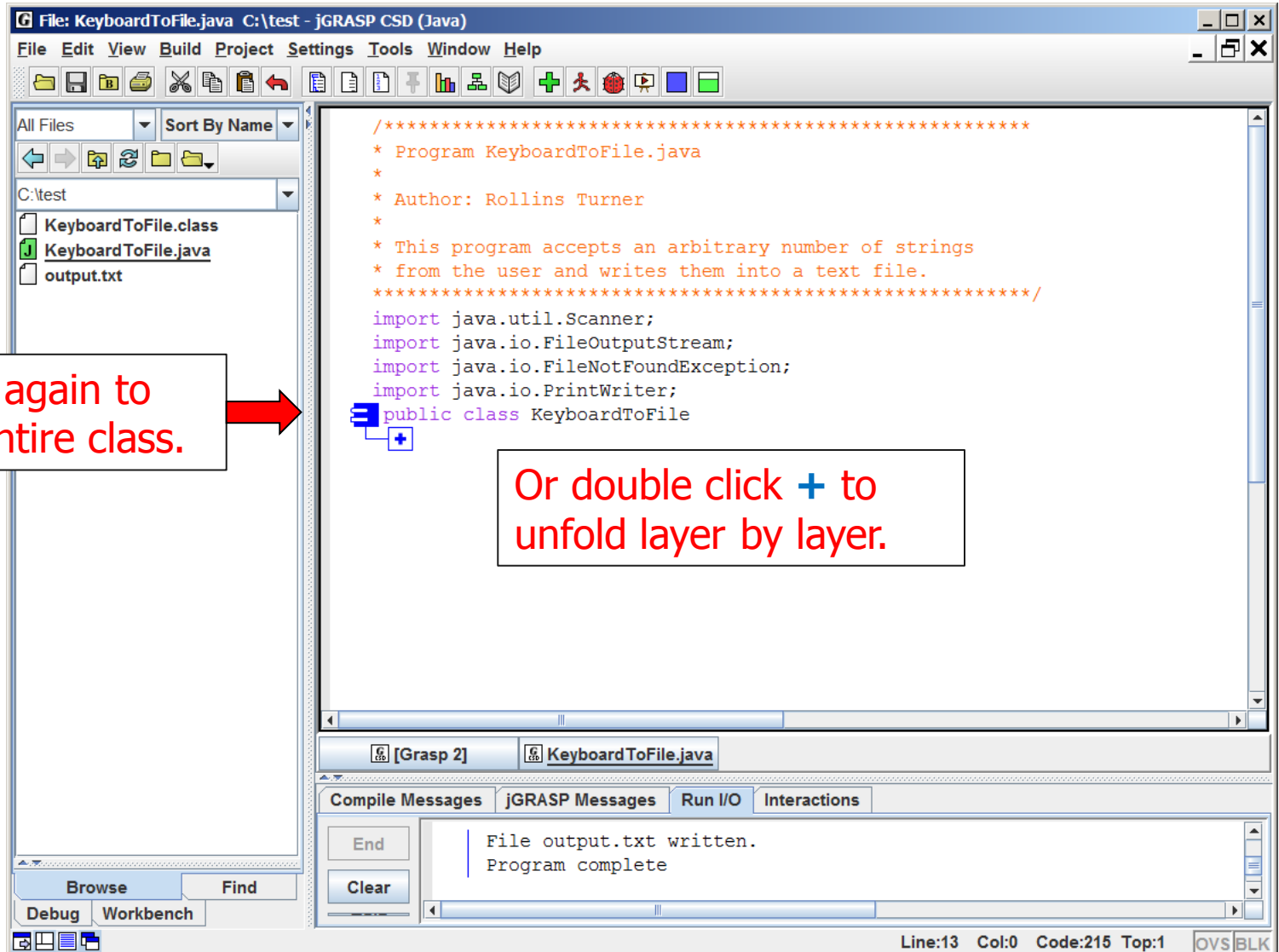


If you want to remove the CSD

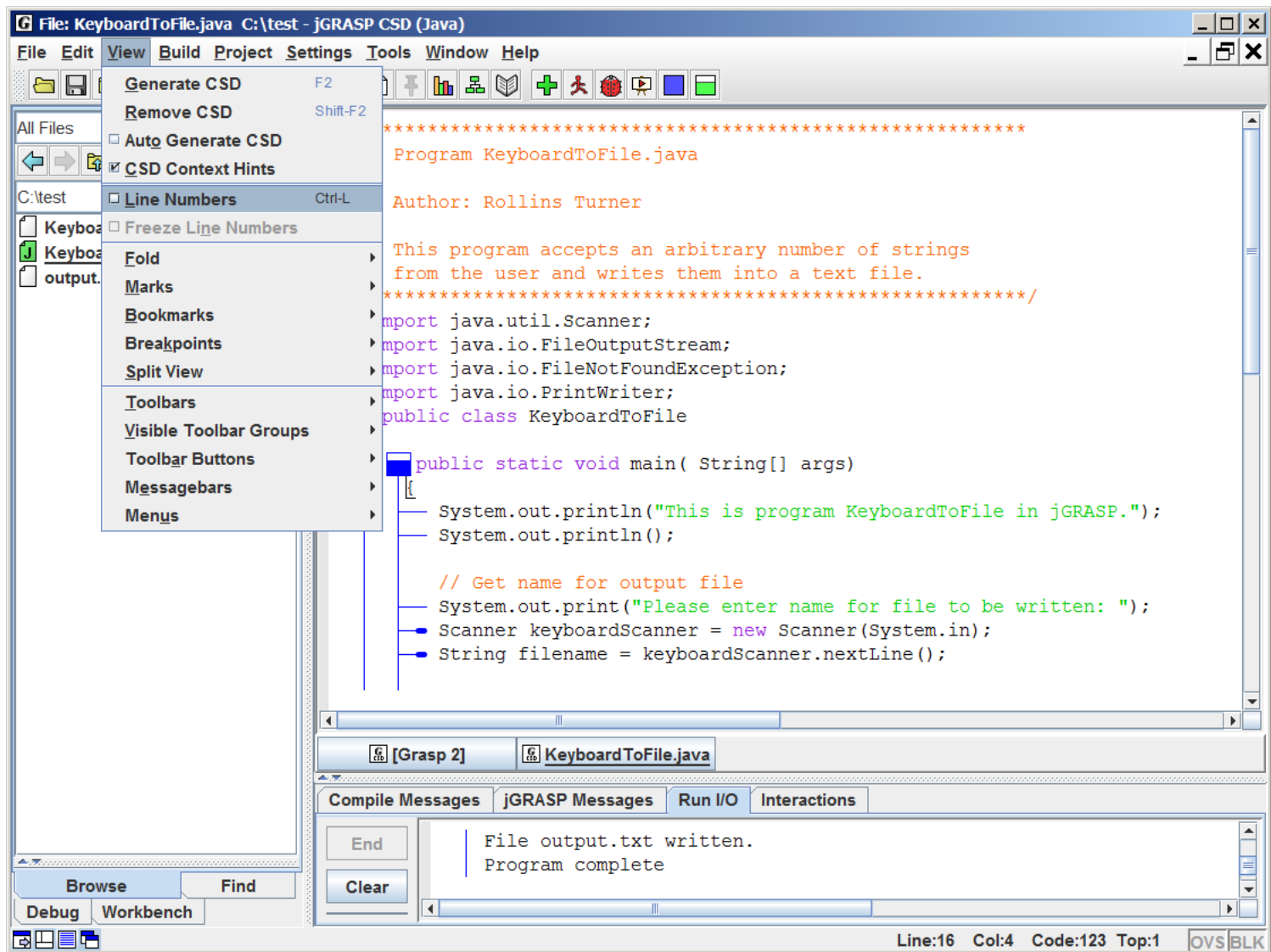


Folding a Program Element

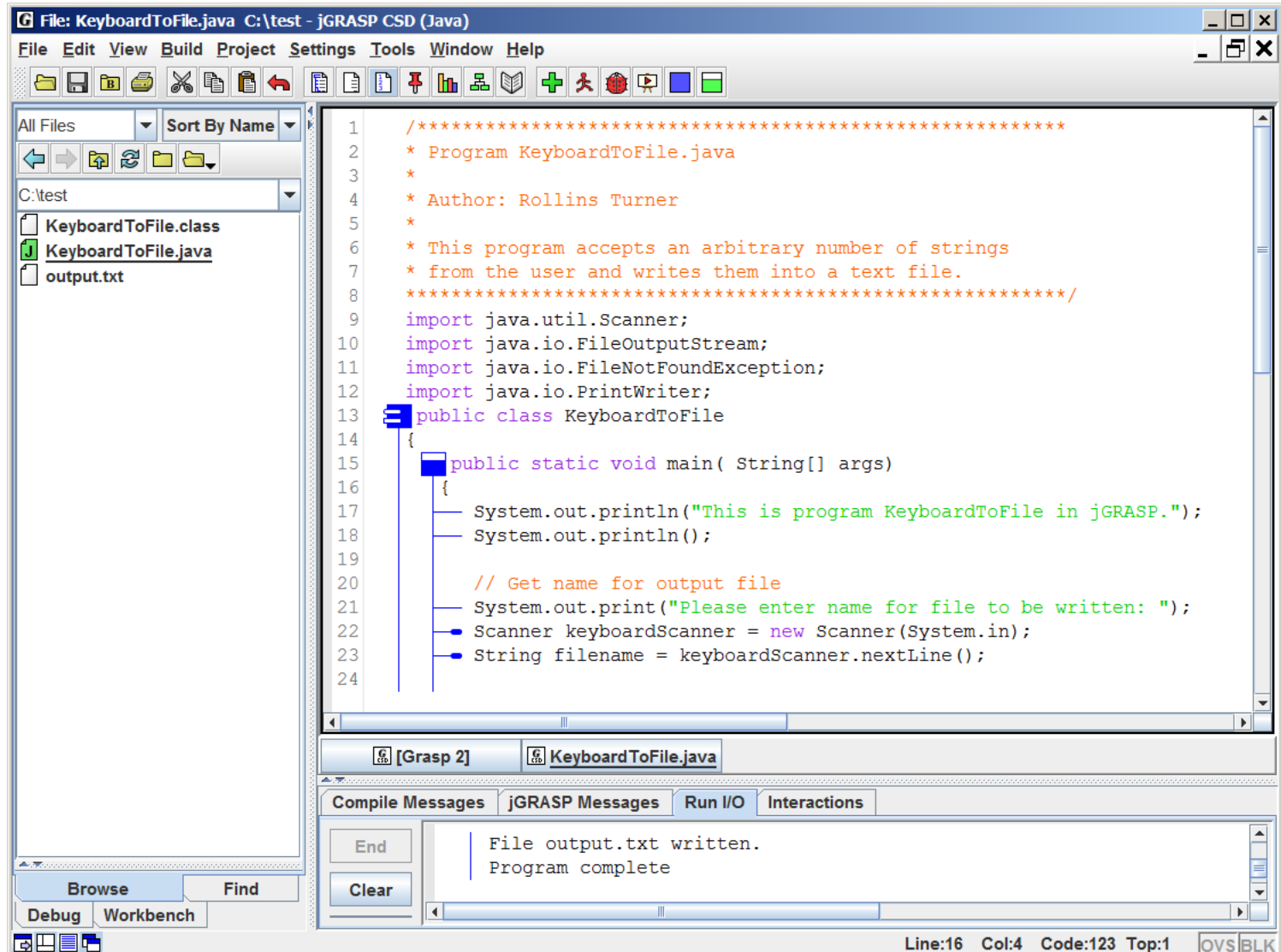




Showing Line Numbers



Line Numbers Shown



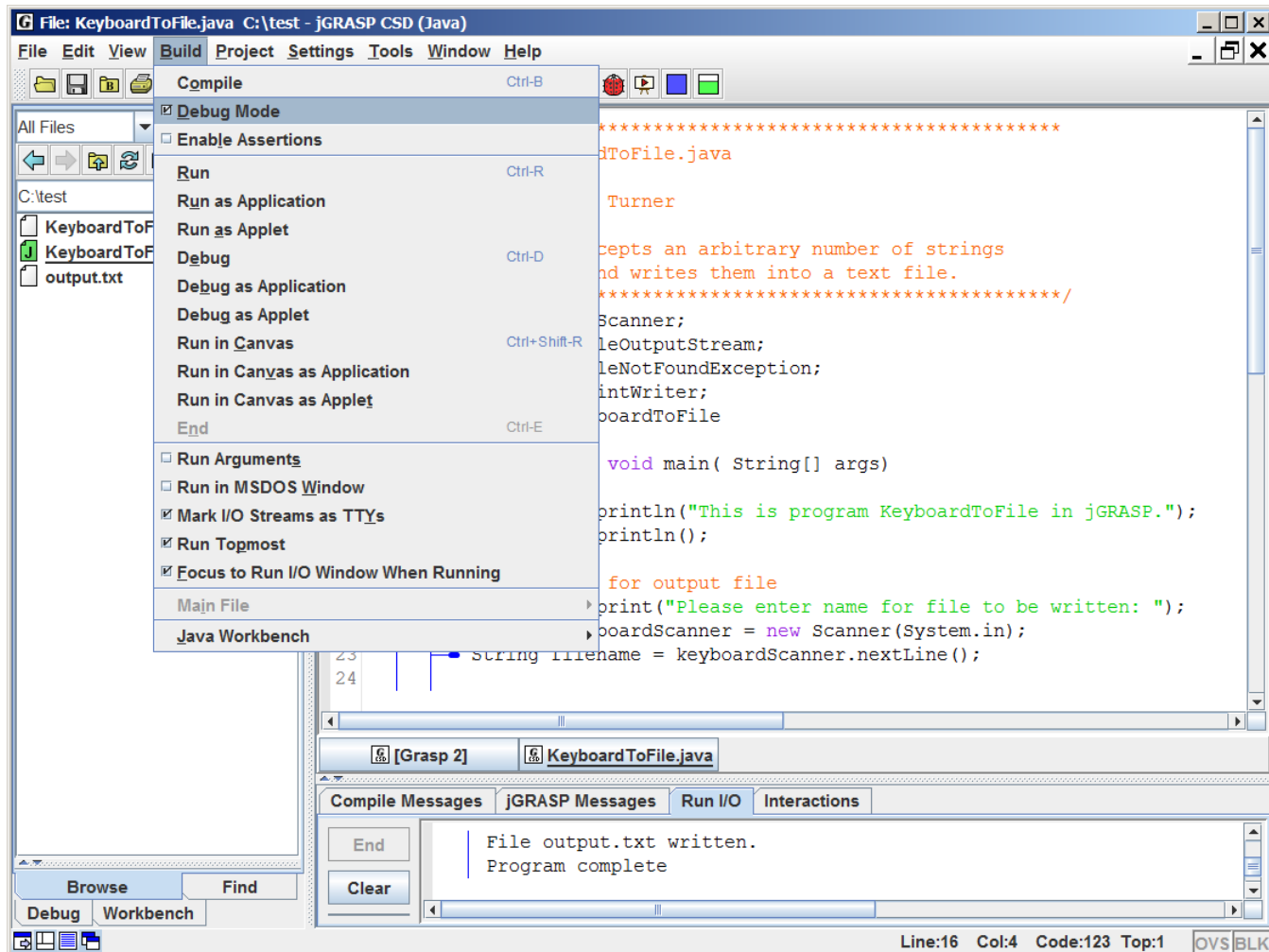


Using the Debugger

- The debugger permits us to set *breakpoints* in our source code.
 - One of the most useful features of an IDE.
- When a running program reaches a breakpoint it will stop (prior to executing the statement).
- We can examine variables.
- We can single-step, or continue from the breakpoint.

Using the Debugger

Be sure Debug Mode is enabled in the Build menu.



Setting a Breakpoint

File: KeyboardToFile.java C:\test - jGRASP CSD (Java)

File Edit View Build Project Settings Tools Window Help

All Files Sort By Name

C:\test

- KeyboardToFile.class
- KeyboardToFile.java
- output.txt

```
10 import java.io.FileOutputStream;
11 import java.io.FileNotFoundException;
12 import java.io.PrintWriter;
13 public class KeyboardToFile
14 {
15     public static void main( String[] args)
16     {
17         System.out.println("This is program KeyboardToFile in jGRASP.");
18         System.out.println();
19
20         // Get name for output file
21         System.out.print("Please enter name for file to be written: ");
22         Scanner keyboardScanner = new Scanner(System.in);
23         String filename = keyboardScanner.nextLine();
24
25         try
26         {
27             String str = "";
28             FileOutputStream fos =
29                 new FileOutputStream(filename, false);
30             PrintWriter pw = new PrintWriter(fos);
31
32             System.out.println("Enter strings to be written to the file.");
33             System.out.println("Press Enter with no text to end the program.");
```

Click in the gray margin column (inside the edit window) to set a breakpoint.

[Grasp 2] KeyboardToFile.java

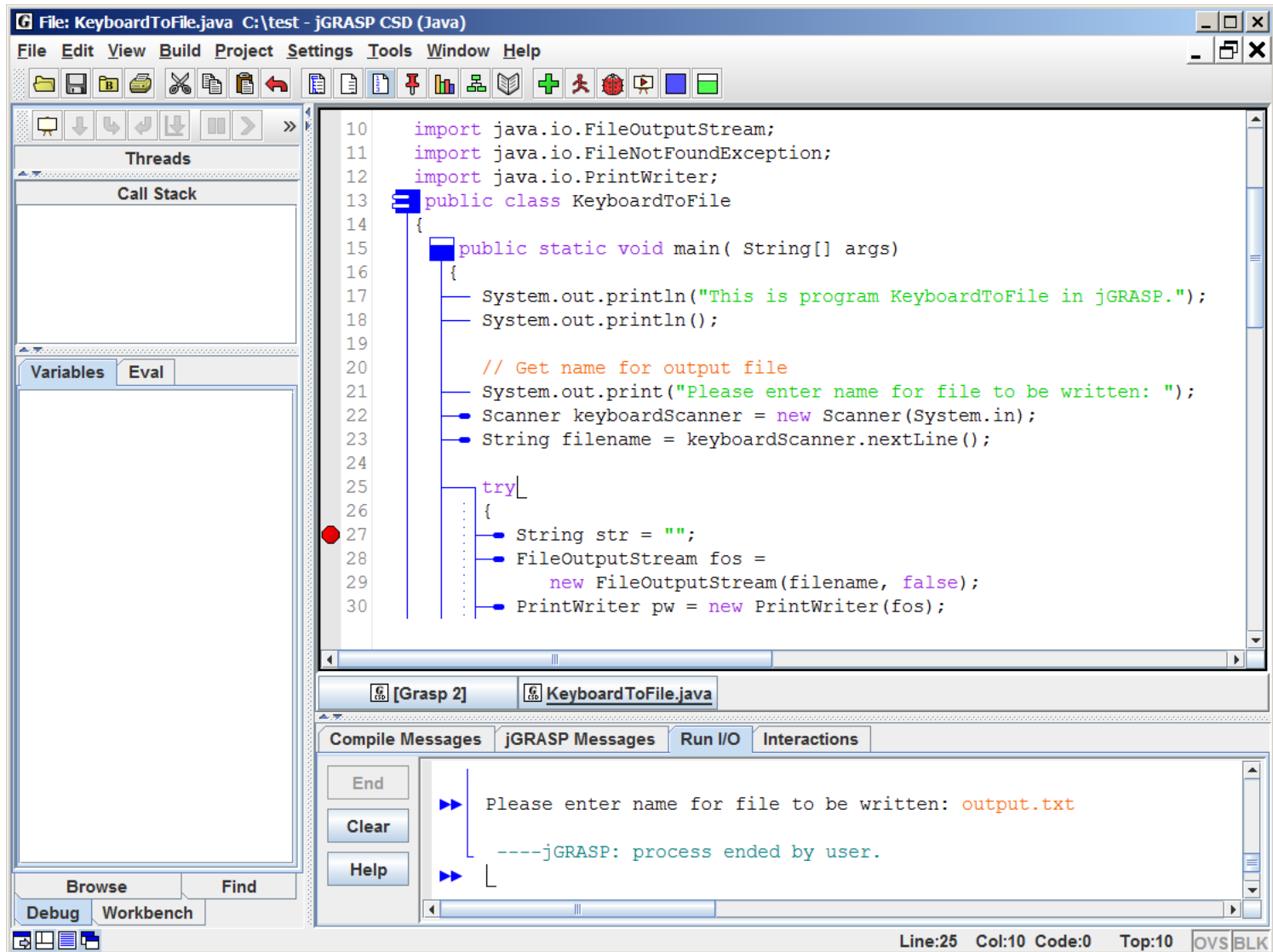
Compile Messages jGRASP Messages Run I/O Interactions

End Clear

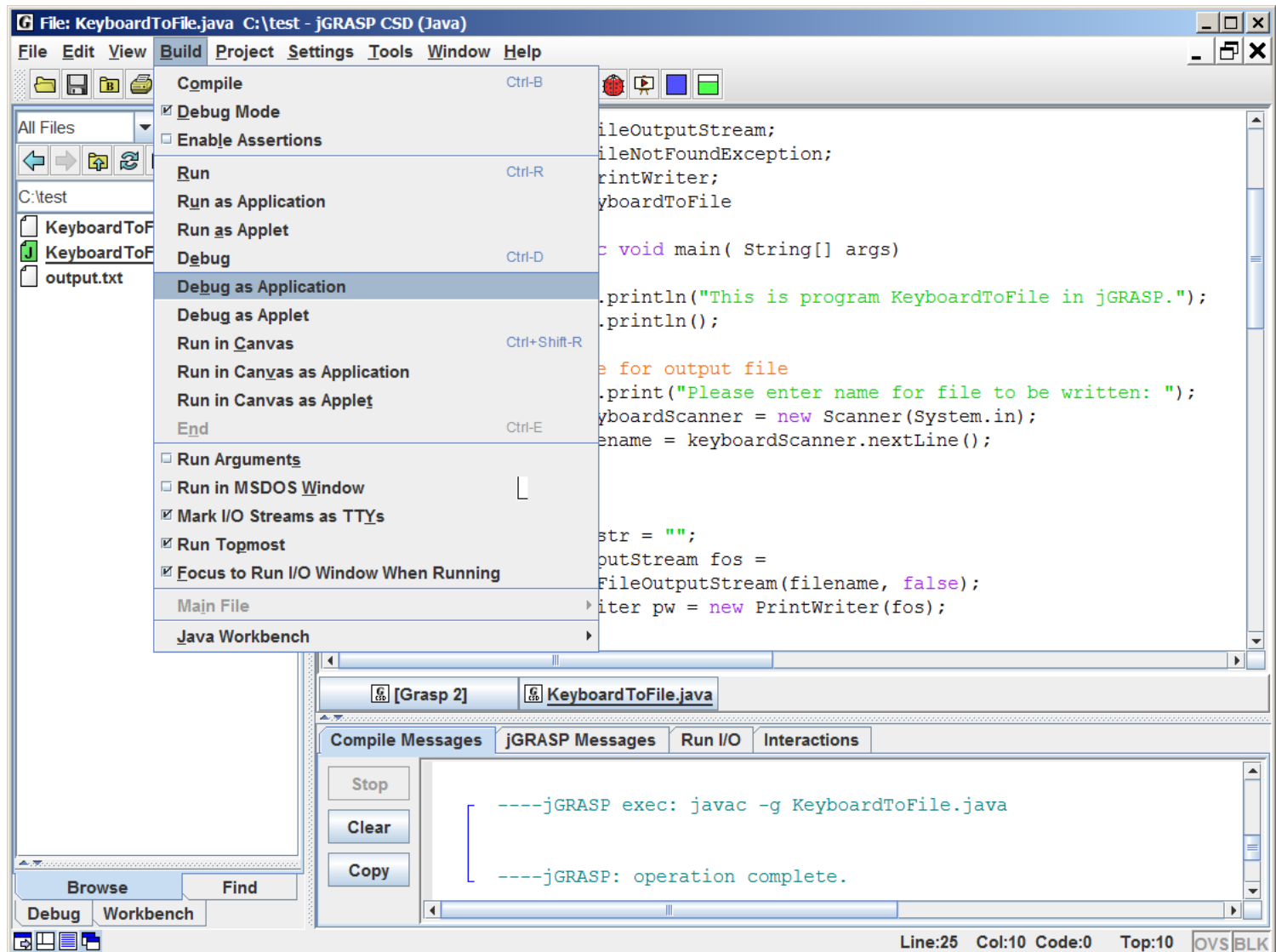
File output.txt written.
Program complete

Line:26 Col:-2 Code:207 Top:10 OVS BLK

Breakpoint Set



Start Debugging



Variables in the Debug Tab

The screenshot displays the jGRASP IDE interface during a debug session. The main window shows the source code of `KeyboardToFile.java`. The left sidebar contains the **Variables** and **Eval** tabs. The **Variables** tab is active, showing a tree structure of variables. The **Call Stack** tab shows the current method `KeyboardToFile.main`. The **Threads** tab is also visible. The **Compile Messages** tab at the bottom shows the output of the compilation process.

Variables Tab:

- static : KeyboardToFile
- Arguments
 - args --> (obj 583 : java.lang.String[0]) java.lang.String[]
- Locals
 - keyboardScanner --> (obj 584 : java.util.Scanner) java.util.Scanner
 - filename --> "output.txt" (obj 610 : java.lang.String) java.lang.String

Call Stack:

- [1] KeyboardToFile.main (KeyboardToFile.java : 27) pc = 38

Source Code:

```
10 import java.io.FileOutputStream;
11 import java.io.FileNotFoundException;
12 import java.io.PrintWriter;
13 public class KeyboardToFile
14 {
15     public static void main( String[] args)
16     {
17         System.out.println("This is program K
18         System.out.println();
19
20         // Get name for output file
21         System.out.print("Please enter name f
22         Scanner keyboardScanner = new Scanner
23         String filename = keyboardScanner.nex
24
25         try
26         {
27             String str = "";
28             FileOutputStream fos =
29                 new FileOutputStream(filename,
30                 PrintWriter pw = new PrintWriter(f
```

Compile Messages:

```
----jGRASP exec: javac -g KeyboardTo
----jGRASP: operation complete.
```

Status: debugging user program

Line/Col/Code/Top: Line:28 Col:30 Code:32 Top:10

Continuing from Breakpoint

Single Step

Resume

The screenshot displays the jGRASP IDE interface for debugging a Java program named 'KeyboardToFile.java'. The main editor window shows the source code with a breakpoint set at line 27. The 'Variables' pane on the left shows the state of the program, including the 'args' array and the 'filename' variable. The 'Threads' and 'Call Stack' panes are also visible. The 'Compile Messages' pane at the bottom shows the output of the compilation process.

File: keyboardToFile.java C:\test - jGRASP CSD (Java)

File Edit View Build Debug Project Settings Tools Window Help

Threads

Call Stack

[1] KeyboardToFile.main (KeyboardToFile.java : 27) pc = 38

Variables **Eval**

static : KeyboardToFile

Arguments

args --> (obj 583 : java.lang.String[0]) java.lang.String[]

Locals

keyboardScanner --> (obj 584 : java.util.Scanner) java.util.Scanner

filename --> "output.txt" (obj 610 : java.lang.String) java.lang.String

```
10 import java.io.FileOutputStream;
11 import java.io.FileNotFoundException;
12 import java.io.PrintWriter;
13 public class KeyboardToFile
14 {
15     public static void main( String[] args)
16     {
17         System.out.println("This is program Ke
18         System.out.println();
19
20         // Get name for output file
21         System.out.print("Please enter name fo
22         Scanner keyboardScanner = new Scanner
23         String filename = keyboardScanner.next
24
25         try
26         {
27             String str = "";
28             FileOutputStream fos =
29                 new FileOutputStream(filename, 1
30             PrintWriter pw = new PrintWriter(f
```

Compile Messages **jGRASP Messages** **Run I/O** **Interactions**

Stop

Clear

Copy

----jGRASP exec: javac -g KeyboardTo

----jGRASP: operation complete.

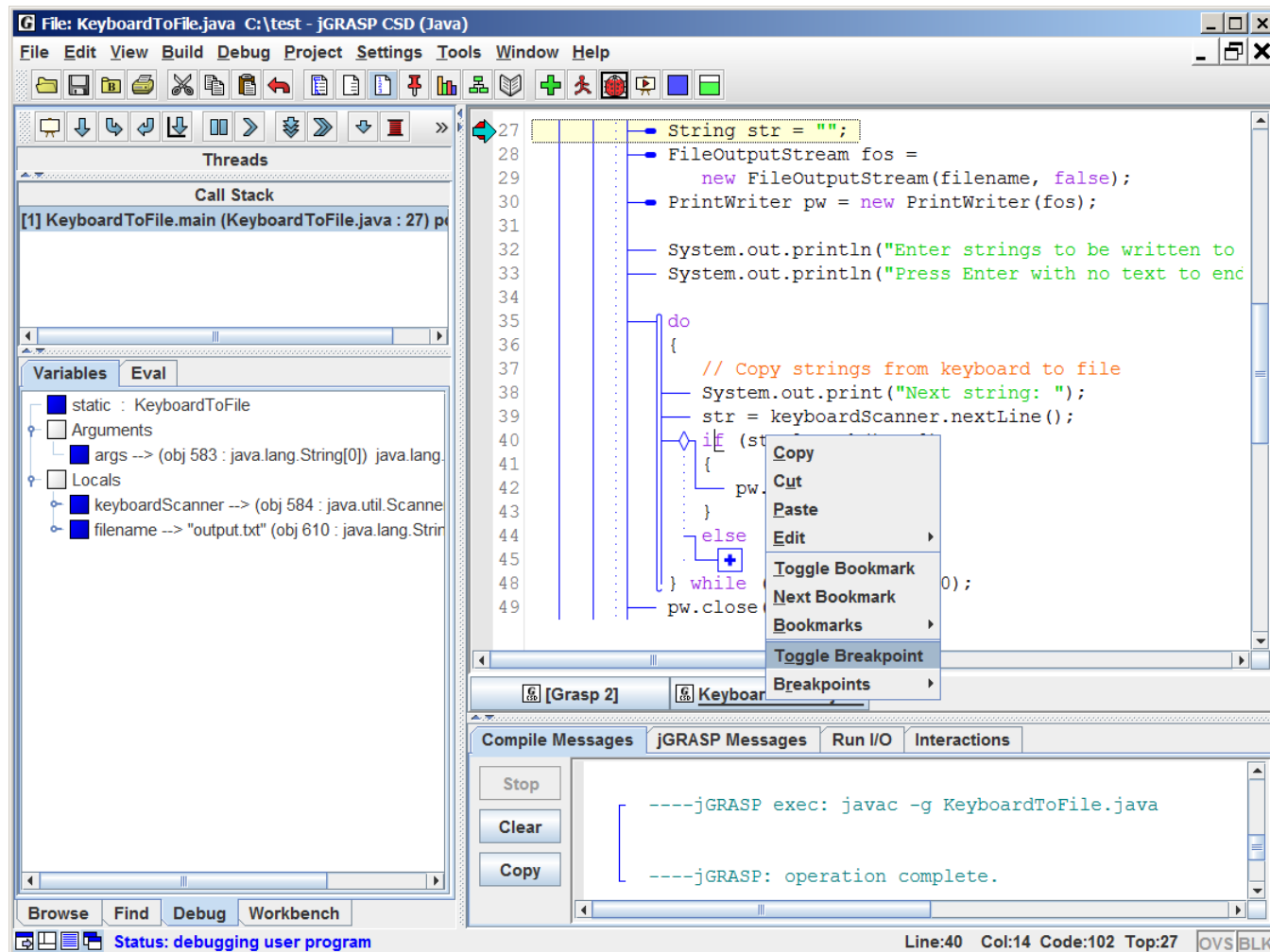
Browse Find Debug Workbench

Status: debugging user program

Line:28 Col:30 Code:32 Top:10 OVS|BLK

Another Way to Set a Breakpoint

Left click on an executable statement, then right click and select Toggle Breakpoint.



Breakpoint Set

The screenshot displays the jGRASP IDE interface with the following components:

- File:** KeyboardToFile.java C:\test - jGRASP CSD (Java)
- Menu Bar:** File, Edit, View, Build, Debug, Project, Settings, Tools, Window, Help
- Toolbars:** Standard IDE toolbars for file operations, editing, and debugging.
- Left Panel:**
 - Threads:** Empty.
 - Call Stack:** [1] KeyboardToFile.main (KeyboardToFile.java : 27) p...
 - Variables/Eval:**
 - static : KeyboardToFile
 - Arguments
 - args --> (obj 583 : java.lang.String[0]) java.lang.
 - Locals
 - keyboardScanner --> (obj 584 : java.util.Scanner
 - filename --> "output.txt" (obj 610 : java.lang.Strin
- Main Editor:** Displays the source code of KeyboardToFile.java. A red dot indicates a breakpoint is set at line 40.

```
27 String str = "";
28 FileOutputStream fos =
29     new FileOutputStream(filename, false);
30 PrintWriter pw = new PrintWriter(fos);
31
32 System.out.println("Enter strings to be written to
33 System.out.println("Press Enter with no text to enc
34
35 do
36 {
37     // Copy strings from keyboard to file
38     System.out.print("Next string: ");
39     str = keyboardScanner.nextLine();
40     if (str.length() > 0)
41     {
42         pw.println(str);
43     }
44     else
45     {
46         +
47     }
48     while (str.length() > 0);
49     pw.close();
```
- Bottom Panel:**
 - Compile Messages:** Empty.
 - jGRASP Messages:** Empty.
 - Run I/O:** Empty.
 - Interactions:** Empty.
 - Buttons:** Stop, Clear, Copy.
 - Log:**

```
----jGRASP exec: javac -g KeyboardToFile.java
----jGRASP: operation complete.
```
- Bottom Bar:** Browse, Find, Debug, Workbench. Status: debugging user program. Line:40 Col:14 Code:102 Top:27 OVS|BLK

Stopped at Breakpoint

The screenshot shows the jGRASP IDE with the following components:

- File:** KeyboardToFile.java C:\test - jGRASP CSD (Java)
- Menu Bar:** File, Edit, View, Build, Debug, Project, Settings, Tools, Window, Help
- Toolbars:** Standard IDE toolbars for file operations, editing, and debugging.
- Threads:** A single thread is running.
- Call Stack:** [1] KeyboardToFile.main (KeyboardToFile.java : 40) pc = 92
- Variables:**
 - static : KeyboardToFile
 - Arguments: args --> (obj 583 : java.lang.String[0]) java.lang.String[]
 - Locals:
 - keyboardScanner --> (obj 584 : java.util.Scanner) java.util.Scanner
 - filename --> "output.txt" (obj 610 : java.lang.String) java.lang.String
 - str --> "The quick br..." (obj 623 : java.lang.String) java.lang.String
 - fos --> (obj 616 : java.io.FileOutputStream) java.io.FileOutputStream
 - pw --> (obj 617 : java.io.PrintWriter) java.io.PrintWriter
- Code Editor:** The code is stopped at line 40, which has a breakpoint. The code is:

```
27 String str = "";
28 FileOutputStream fos =
29     new FileOutputStream(filename, false);
30 PrintWriter pw = new PrintWriter(fos);
31
32 System.out.println("Enter strings to be w
33 System.out.println("Press Enter with no t
34
35 do
36 {
37     // Copy strings from keyboard to file
38     System.out.print("Next string: ");
39     str = keyboardScanner.nextLine();
40     if (str.length() > 0)
41     {
42         pw.println(str);
43     }
44     else
45     {
46         +
47     } while (str.length() > 0);
48
49 pw.close();
```
- Interactions:**
 - End
 - Clear
 - Help
 - Input: Please enter name for file to be written: o
 - Input: Enter strings to be written to the file.
 - Input: Press Enter with no text to end the program.
 - Input: Next string: The quick brown fox
- Status Bar:** Status: debugging user program
- Footer:** Line:31 Col:10 Code:0 Top:27 OVS BLK

Program Complete

The screenshot displays the jGRASP IDE interface with the following components:

- Title Bar:** File: KeyboardToFile.java C:\test - jGRASP CSD (Java)
- Menu Bar:** File Edit View Build Project Settings Tools Window Help
- Toolbars:** Standard Java IDE toolbars for file operations, editing, and debugging.
- Left Panel:**
 - Threads:** Empty panel.
 - Call Stack:** Empty panel.
 - Variables/Eval:** Empty panel with tabs for Variables and Eval.
- Editor:** Displays the source code of KeyboardToFile.java with line numbers 27 to 49. The code includes variable declarations, file opening, printing instructions, a loop for copying strings, and file closing. Red circular markers are present at line 27 and line 40.
- Bottom Panel:**
 - Tab Bar:** Shows [Grasp 2] and KeyboardToFile.java.
 - Message Tabs:** Compile Messages, jGRASP Messages, Run I/O, Interactions.
 - Run I/O Tab:** Contains a text area with the output: "Next string: File output.txt written. Program complete" and "----jGRASP: operation complete." Below the text area are buttons for End, Clear, and Help.
- Footer:** Line:31 Col:10 Code:0 Top:27 OVS|BLK

Back to Browse Tab

The screenshot displays the jGRASP IDE interface. The top menu bar includes File, Edit, View, Build, Project, Settings, Tools, Window, and Help. Below the menu is a toolbar with various icons. The left sidebar shows a file explorer with the following structure:

- All Files (Sort By Name)
- C:\test
 - KeyboardToFile.class
 - KeyboardToFile.java** (selected)
 - output.txt

A red text overlay on the left sidebar reads: "Double click on output.txt to open in edit window".

The main editor window displays the source code for KeyboardToFile.java. The code is as follows:

```
27 String str = "";
28
29 FileOutputStream fos =
30     new FileOutputStream(filename, false);
31
32 PrintWriter pw = new PrintWriter(fos);
33
34 System.out.println("Enter strings to be written to file");
35 System.out.println("Press Enter with no text to end");
36
37 do
38 {
39     // Copy strings from keyboard to file
40     System.out.print("Next string: ");
41     str = keyboardScanner.nextLine();
42     if (str.length() > 0)
43     {
44         pw.println(str);
45     }
46     else
47     {
48         break;
49     }
50 } while (str.length() > 0);
51 pw.close();
```

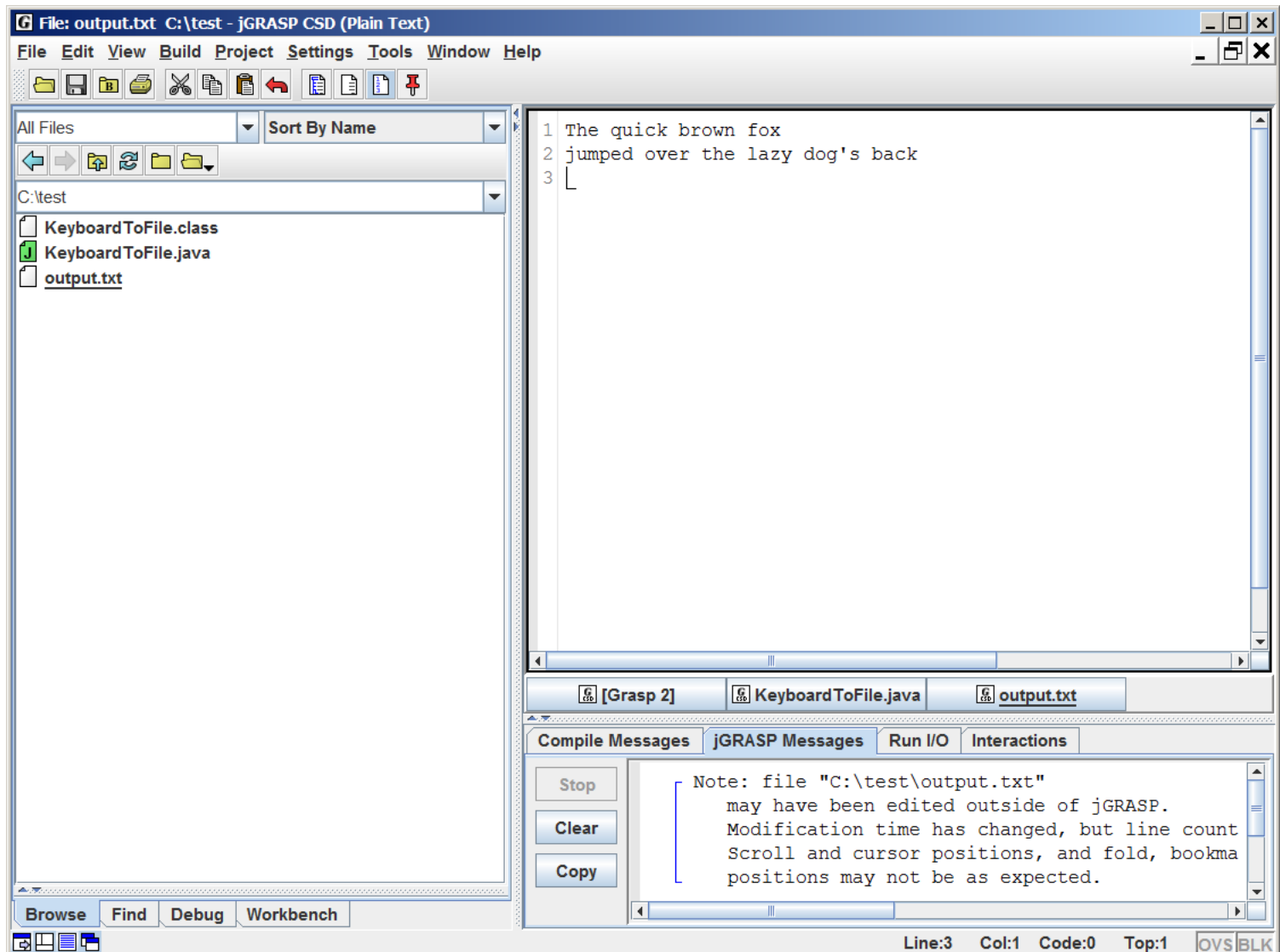
The bottom of the IDE features a tab bar with "Grasp 2" and "KeyboardToFile.java". Below this is a panel with tabs for "Compile Messages", "jGRASP Messages", "Run I/O", and "Interactions". The "Run I/O" tab is active, showing the following output:

```
Next string: File output.txt written.
Program complete

----jGRASP: operation complete.
```

The status bar at the bottom indicates "Line:31 Col:10 Code:0 Top:27" and includes a small "OVS|BLK" indicator.

Output File



Assignment

- Reading

- Go to the jGRASP web site
- <http://www.jgrasp.org/>
- Read tutorials:
 - Overview (2.0)
 - Getting Started (2.0)
 - CSD (2.0)
 - Debugger

Tutorials (PDF)

[Overview \(2.0\)](#)

[Installation](#)

[Getting Started \(2.0\)](#)

[Objects First](#)

[Interactions](#)

[CSD \(2.0\)](#)

[Debugger](#)

[Projects](#)

[UML](#)

[Workbench](#)

[Viewers](#)

[JUnit \(2.0\)](#)

[Canvas \(2.0\)](#)

[All \(zipped\)](#)

- Lab

- Project 12 Cars from Files