CSC 161 Java Snippets File

Created: Jan 2015 by Prof Bill

This file is a repository of Java information and examples for our class, CSC 161 Winter 2015. This is an experiment… I'm trying to improve communication in class.

Here's my email: wtkrieger@noctrl.edu. Please hit me with any questions, errors, requests, or (best of all) new submissions.
thanks… yow, bill

0. Index

Search down to find the snippet you desire. So far, the snippets are:

1. Using BufferedReader to read a file

*Submitted by: **Reilly R***

Here's a snippet using BufferedReader to read lines of a file.

```
try {
   FileReader freader = new FileReader( "secret_message.txt");
   BufferedReader breader = new BufferedReader(freader);

   String line = breader.readLine();
   while (line != null) {
      // do things with each line of file input

      line= breader.readLine();
   }
} catch( Exception exc) {
        System.out.println( exc);   // print error message
    }
```

We haven't covered exceptions yet, so I have tried to simplify this as much as possible. On a file error (like a missing file), an error message will be printed.

2. Using Scanner to get console input

Submitted by: Prof Bill
A Scanner snippet from our text, page 85:

```
Scanner keyboard = new Scanner( System.in);
System.out.println( "Enter your name:");
String name = keyboard.nextLine();
```

Like BufferedReader, you can use Scanner on files as well. The two are quite similar. Here's a stackoverflow discussion on the differences (mostly efficiency) between Scanner and BufferedReader: stackoverflow.com/questions/2231369/scanner-vs-bufferedreader

3. Copying existing Java files into NetBeans

Submitted by: Prof Bill
I'll show two ways to do this. The first way is slow, but never fails. Go!

- Create a new file in NetBeans (File/New file)
- Open the existing file in NotePad or something
- Copy/Paste the file text into NetBeans
- File/Save

I like the second way. Sometimes you'll bump into file protection settings, but not usually.

- Create a new NetBeans project or use an existing project
- Quit NetBeans
- Copy your Java files to the src folder in your NetBeans project folder
- Start NetBeans. Your files should now be there, ready to compile and edit

4. Rotating letters in a String

Submitted by: ***Samantha G***
This snippet rotates the letters in a String, ala Lab 01.

```
//this loop runs through character array and rotates any characters that
are letters by rotation amount
char[] charArray = line.toCharArray();
for (int i = 0; i < charArray.length; i++) {
    if (Character.isLetter(charArray[i])) {
        charArray[i] = (char) (charArray[i] + rot);

        if (!Character.isLetter(charArray[i])) {
            charArray[i] = (char) (charArray[i] - 26);
        }
    }
}
```

5. Creating a JFrame

Submitted by: Prof Bill

Almost all JFrames that you create will have this code as part of it. You can find most of this code at various places in our text.

```
JFrame f = new JFrame();
f.setTitle( "The Frame’s Title");
f.setPreferredSize( new Dimension( 800, 600));   // width, then height
f.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);
f.setResizable( false);

// some optional things for you too
Color c = new Color(210,105,30);        // chocolate color
this.selectFrame.setBackground( c);   // set the background color
f.selectFrame.setResizable( false);   // don’t allow resizing
f.setLocation( 200, 300);   // set location of frame’s upper left corner
```

Of course, there are many other methods available for JFrame. You can browse them at the official JavaDoc page: docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html

Reminder - When you change something within a JFrame, call pack() to repack the contents and display it properly. A change without packing may not be shown on the screen.

6. GUI shortcuts with JOptionPane

Submitted by: ***Andrew F***

Here is a way to get simple user input using `JOptionPane`.

```
// shows Input Box and returns a string that is stored in yourName
String yourName;
yourName = JOptionPane.showInputDialog("What is your first name?");
```

You can easily send a nice GUI message to the user like this:

```
// shows Message Box and displays the name you entered
JOptionPane.showMessageDialog(null, "Your name is: " + yourName);
```

JOptionPane is discussed on page 92 of our textbook.

7. AudioClip workaround

Submitted by Prof Bill

Our textbook shows how to read a sound file from an applet on page 917.
Page 919 shows an example creating an AudioClip using a local file.
Here's a snippet to read an audio file from the internet.

```
String webAddress =
"http://www.pacdv.com/sounds/machine_sound_effects/dial-up-modem-1.wav";

// get an audio clip from the internet
AudioClip audio = null;
try {
   URL url = new URL( webAddress);
   audio = Applet.newAudioClip( url);
} catch( Exception exc) {
   System.out.println( exc);
}
```

It's fun to google for new audio files and then play them in your Java program. Sites like www.wavsource.com/ are fun. Alas, some audio files you find won't play because (I think) they include an encryption algorithm that Java doesn't support. Try it and if it doesn't play, then move along.

8. Centering a JLabel

Submitted by: ***Gerardo P***

You can center a JLabel with a CENTER parameter to its ctor.

```
// Displays random quote that is centered on the JFrame
JLabel messageLabel = new JLabel("Example", SwingConstants.CENTER);
messageLabel.setPreferredSize(new Dimension(LABEL_WIDTH, LABEL_HEIGHT));
```

Please note that you must give the label a size to center it. Without knowing its size, the label won't be centered.

Also, note that the method setPreferredSize() is almost always better than setSize(). Most Java GUI layout managers prefer setPreferredSize().

9. Using ArrayList

Submitted by: ***Ricky M***

You'll find ArrayLists on page 515 of our textbook.

Creating an ArrayList of strings.

```
ArrayList<String> words = new ArrayList();
```

Adding a string to the end of the list.

```
words.add(line);
```

Foreach loop with an ArrayList.

```
//This is used to go through each word in the ArrayList
for (String s: words) {
   if (s.equalsIgnoreCase(word)) {
      found = true;
      break;
   }
}
```

Get a specific string in the list.

```
int num = 7;
String s = return words.get(num);
```

ArrayLists work with any class, not just String. There are many more ArrayList methods. Check out the standard library Javadoc to see these:
http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

10. Comparing Strings

*Submitted by: **Prof Bill***

Use the equals() method to compare 2 strings, not ==. Like this:

```
String s1 = "Bill";
String line = br.readLine();
if( s1.equals( s2)) {
   //  do something
}
```

You can also compare strings without regard for case using equalsIgnoreCase().

The == comparison check if the strings are the exact same object… not whether they have the same value.

```
String s1 = "snip";
String s2 = "snip";
```

```
if( s1 == s2) {
   // no way, jose
}
```

11. Reading and resizing images

Submitted by: ***Jack B***

This snippet show you how to read a JPG image file and then resize the image in your program.

First, reading the file using the ImageIO class…

```
String FILE_NAME = “test.jpg”;
BufferedImage img = null;   // stores the image you want to use
try {
      img = ImageIO.read(new File(FILE_NAME);
} catch (IOException e) {
      System.out.println( e);
}
```

Now, resize the image to be a specific height and width:

```
final int IMAGE_WIDTH = 200;
final int IMAGE_HEIGHT = 150;

// scale the image and then convert to ImageIcon for use later
BufferedImage scaledImg;
scaledImg = img.getScaledInstance(IMAGE_WIDTH, IMAGE_HEIGHT,
                                     Image.SCALE_SMOOTH);
ImageIcon tempImage = new ImageIcon( scaledImg);
```

As an ImageIcon, the image can now be used in a JLabel or JButton, ala page 812 of our text.

12. Opaque JLabels

Submitted by: ***Prof Bill***

It’s easy to set the background color of a JLabel.

```
// create a label test with an orange bg
JLabel lab = new JLabel( “Test”);
lab.setBackground(Color.ORANGE);
pan.add( lab);      // add it to panel
```

When you do this, however, your background doesn’t change. Labels are transparent by default. So, their background color is whatever the panel is. To see the background color of a lable, make it opaque.

```
lab.setOpaque( true);
```

This operation is true of all GUI Components, like labels and buttons and more!

13. NetBeans tricks

Submitted by ***Julian C***

Some NetBeans productivity tricks for you.

1. Automatically formatting to nicer-looking code
   - ➔ Select all of your code (Ctrl+A)
   - ➔ Click Source/Format

2. Selecting Main - if your project has multiple main() methods, then you can choose which one to run by:
   - ➔ Run/Set Project Configuration/Customize.../Run/Main Class
   - ➔ Use Browse… click the main class you would like to run and compile with

3. Passing arguments to main()
   - ➔ Use the menu selections as if you were selecting your main class…Run/Set Project Configuration/Customize.../Run
   - ➔ Use the Arguments field underneath
   - ➔ Put in arguments, separating by using spaces (e.g. "bill rules" or "bill")

In the main class later on, you can detect for arguments by doing something like:

```
public static void main( String[] args) {

   // handle main() arguments
   if (args.length > 0) {
      // code dealing with args String values
   }
```

14. Using Collections.shuffle()

Submitted by: ***Phil K***

Here is my code for shuffling the letters of a word.
I found the code below at:
www.stackoverflow.com/questions/3316674/how-to-shuffle-characters-in-a-string

Choose the second response because it's easy to read/understand
Here we go… using shuffle() to randomize letters in a string:

```
// step 1: turn String (word) into a list of chars
List<String> letters = Arrays.asList(word.split(""));

// step 2: shuffle the items (chars) in the list
Collections.shuffle(letters);

// step 3: rebuild our String from the randomized chars
String shuffled = "";
for (String letter : letters) {
   shuffled += letter;
}
```

15. Timing a method

Submitted by: ***Luke F***
The method System.currentTimeMillis() returns the current time in milliseconds.
You can measure the elapsed time of a piece of code with two calls

```
long startMillis = System.currentTimeMillis();

   aMethodToTime();   // some code or methods to time

long endMillis = System.currentTimeMillis();

System.out.println( "Elapsed time in millis = " + (endMillis -
startMillis));
```

Remember: this is elapsed time, clock time. It is not a measure of CPU cycles or some other performance metric.

16. Integer to String

Submitted by: ***Shaun R***

Note: There are many ways to do this. Here's one.
If you have an `int` and you need a `String`... use the `Integer` class.

```
// Must change the INTEGER into a STRING for the JLabel
String name = Integer.toString( index);
matrixLabels[index] = new JLabel( name, SwingConstants.CENTER);
```

Notice that this `toString()` method is static to the `Integer` class. That means it is (usually) called using the class name (`Integer`) rather than a specific object.

How about going from a `String` to an `int`? Try `Integer.parseInt()`.

17. Pausing your program

Submitted by: Prof Bill
This snippet pauses the execution of your program N milliseconds. I used it to pause between moves of matrix simulation in Program #2.

```
/**
 * Pause the program for N milliseconds.
 * I got help from this web page: www.mindprod.com/jgloss/sleep.html
 * @param millis Num milliseconds to pause.
 */
public static void pause( int millis) {
    try {
        Thread.sleep(millis);
    } catch (java.lang.InterruptedException ie) {
}
```

18. Using KeyListener

*Submitted by: **Shaun R***
This snippet shows how to drive a GUI panel using keystrokes, rather than just the mouse. It's a two-step process.

Step 1: Add a `KeyListener` to your `JPanel`.

```
JPanel p = new JPanel();
p.setFocusable(true);          // panel must be focusable to work!

// create a key listener and add it to the panel
KeyListener kl = new MoveListener();
p.addKeyListener(kl);
```

Step 2: Now, we add a private `KeyListener` class and we're ready to rock!

```
private class MoveListener implements KeyListener {
      @Override
      public void keyPressed(KeyEvent e) {
            int keyPressed = e.getKeyCode();
            System.out.println("KEY PRESSED");
      }

      @Override
      public void keyTyped(KeyEvent ke) {
            System.out.println("Key Typed");
      }

      @Override
      public void keyReleased(KeyEvent ke) {
            System.out.println("Key Released");
            }
    }
```

Finally, here's a nifty tutorial for more information:
docs.oracle.com/javase/tutorial/uiswing/events/keylistener.html

19. Fixing jGrasp, sometimes

Submitted by: ***Prof Bill***

In our lab, sometimes jGrasp isn't properly installed. You may get this error trying to compile:

```
----jGRASP exec: javac -g Squid.java

----jGRASP wedge2 error: command "javac" not found.
----   This command must be in the current working directory
----   or on the current PATH to use this function.
----   PATH is "C:\Program
Files(x86)\Java\jre1.8.0_25;C:\ProgramData\Oracle\Jav…
----
----   Make sure you have the full JDK, not just the JRE, installed.
----   The JDK is available from
http://www.oracle.com/technetwork/java/index...
----jGRASP: operation complete.
```

This error means that jGrasp cannot find the Java compiler, javac.
So, two questions: 1) Where is javac? 2) How do I tell jGrasp where it is?

For Windows 7 in our lab, the Java installation usually happens on the local disk, C:. After that, you may have to hunt and peck. On this machine in the lab, I found javac at:

```
C:\Program Files (x86)\Java\jdk1.8.0_25\bin
```

OK, how do I communicate this to jGrasp, so he can find javac.

- Select the menu Settings/PATH CLASSPATH/Workspace
- Select New
- Then use Browse to find the bin folder holding javac

Hit OK and Apply a few times and that should do it. Now try compiling.

```
----jGRASP exec: javac -g Squid.java


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

Huzzah!
If you weren't successful… than as a last gasp, try googling: "jgrasp wedge2 error".

20. Reading CSV

*Submitted by: **Prof Bill***

This method parses a line of comma-separated values (CSV).

```
/**
 * Parses one line of Employee File input.
 * @param line The input line.
 * @return Returns a list of 3 strings in the input line.
 */
public static ArrayList<String> parseLine( String line) {
    ArrayList<String> tokens = new ArrayList();
    Scanner s = new Scanner( line);
    s.useDelimiter( ",");      // use comma for CSV parsing

    while( s.hasNext()) {
        tokens.add( s.next());
    }
    return tokens;
}
```

An example: If line is "Prof,Bill,rules baby!", then the list returned will have three tokens (Strings): [Prof, Bill, rules baby!].

The magic sauce here is the method `useDelimiter()` in `Scanner`. By default, Scanner uses any whitespace to separate tokens using the `next()` method. By calling `useDelimiter()`, we're changing this from whitespace to a comma character.

There's an example of a CSV file on **page 639 of our textbook**.

21. String.split()

*Submitted by: **Gerardo P***

This method `String.split()` is very handy to read CSV input.

```
String line = "bill,is,nice";
String[] csvTokens= line.split(",");
System.out.println( Arrays.toString( csvTokens));
```

This will print out the 3 tokens (bill, is and nice) in the original line.
The `split()` method also works with space or other delimiters as well.
The `String.split()` method is discussed on **page 633 of our textbook**.

22. Centering your JFrame on the screen

Submitted by: ***Luke F***

You can center a frame on your screen by passing `null` to the `setLocationRelativeTo()` method for a `JFrame`.
Like this:

```
//Centers the GUI to the center of the screen
JFrame frame = new JFrame();
frame.setLocationRelativeTo(null);
```

EZ.

23. StringBuilder

Submitted by: ***Nathan M***

The StringBuilder class is very helpful in building large strings. The usual paradigm is 1) create a new `StringBuilder`, 2) use the `append()` method to build up a large string, and 3) get the `String` using `toString()`.

Here's a snippet:

```
StringBuilder str = new StringBuilder();

// append values to the builder
str.append(99.56);
str.append(" dollars were spent over ");
str.append(3);
str.append(" days");

// returns the appended integer and double valued
// StringBuilder object entirely as a String.
System.out.println( str.toString());
```

`StringBuilder` is covered on **page 620 of our textbook**.

24. PrintWriter

Submitted by: ***Suby***

`PrintWriter` is an output stream class that provides some convenient write methods, like `println()`, for a file.
Here's the snippet:

```
// NOTE - PrintWriter will overwrite an existing file
PrintWriter outputFile = new PrintWriter("fileName.txt");

outputFile.println("There are many methods you can use with
PrintWriter");
outputFile.println(30); // Can print numbers
outputFile.println('A'); // Can print chars


outputFile.close();   // remember to close the file
```

The PrintWriter class is discussed on **page 214 of our textbook**.

25. Creating a random color

Submitted by: ***Zach M***

A random color is created by generating random RGB values. Liek this:

```
public static Color randomColor(){
            Random rand = new Random();

            int r = rand.nextInt(256);
            int g = rand.nextInt(256);
            int b = rand.nextInt(256);

            Color randColor = new Color(r, g, b);

            return randColor;
      }
```

Nice.

26. Fixing JGrasp compile woes

Submitted by: ***Julian C***

If JGrasp won't compile, then you need to tell it where the Java compiler, javac, is located. To fix JGrasp's javac errors when compiling with an easier way, try these steps:

1. Click Start, All Programs, JGrasp, and JGrasp Startup Settings

2. In the lowermost dropdown menu where it says [Default], select the first option underneath [Default] (Should look like C:\Program Files\...)
3. Click OK and restart JGrasp and try compiling again

27. Adding a background image to a panel

Submitted by: ***Prof Bill*** *and* ***Ricky M***

The source for this idea is:
www.stackoverflow.com/questions/7092067/adding-a-background-image-to-a-panel-containing-other-components

This is a little longer than our usual snippet, but follow along. Add a background image to your JPanel in 3 steps: 1) pass image to the ctor, 2) save it as a class variable, 3) draw it in paint().

```
public class WorkPanel extends JPanel {
    private final Image bgImage;    // the background image!

    public WorkPanel( Image img) {
        this.bgImage = img;    // store the background image
        setOpaque(true);

        // optional - make your panel size match your image
        if( this.bgImage != null) {
            Dimension dim = new Dimension( this.bgImage.getWidth(null),
                    this.bgImage.getHeight(null));
            setPreferredSize( dim);
        }
    }

    @Override
    public void paint( Graphics g) {
        super.paint(g);     // always - paint the super class

        // paint the background image before work
        if( this.bgImage != null) {
            g.drawImage(this.bgImage, 0, 0, null);
        }

        // now paint our "work"
        g.setColor( Color.PINK);        // a pink square
        g.fillRect( 100, 100, 100, 100);
    }
}
```

There’s a snippet on reading image files: 11. Reading and resizing images (by Jack B)

Notice that, in `paint()`, if the image is missing (null), then it isn't drawn. Setting your panel to be the size of the image is optional. They don't have to match.
This solution won't work if you have Swing objects like buttons or labels. You can do this with 2 panels: a base with the background image, and then a panel with Swing objects added to it.

28. (Preferred) Size your Panel

Submitted by: ***Nick D***

It's usually better to 1) set the size of your panel, rather than your frame and 2) use setPreferredSize(), not setSize().
Like this:

```
// create and size the panel
JPanel p = new JPanel();
p.setPreferredSize(new Dimension (500,500));

// create the frame, set panel in frame
JFrame f = new JFrame();
f.setContentPane(p);

// display frame
f.pack();
f.setVisible(true);
```

Java tries to intelligently size and position GUI objects. If you size your panel, then the frame will grow to accommodate it. If you ask for a preferred size, then Java will try to meet your request. Java will sometimes ignore a setSize() request as too limiting.
Yes… some of this stuff is pretty weird.

29. Collections and Arrays classes

Submitted by: ***Jason K***

The `Collections` class in the Java library provides many power and easy-to-use methods for use with collections of objects. These methods are all static and simply require the class name, rather than an object to be called.

Here are some of these methods:

- **Reverse** – `Collections.reverse(Collection c)`: reverses the selected collection
- **Sort** – `Collections.sort(Collection c, Comparator cmp)`: sorts the selected collection, using the comparator to determine the order.
  `Collections.sort( Collection c)`: this sort requires `Comparable` objects to work

- ➔ **Shuffle** – `Collections.shuffle(List l)`: shuffles the elements in a `List` into a random order
- ➔ **Max** – `Collections.max (List l, Comparator cmp)`: returns the max value from the collection, using the comparator. `Collections.max( List l)` requires `Comparable` objects to work
- ➔ **Min** – `Collections.min(List l,cmp)`: same as Max, but returns the min value. instead. Same for `Collections.min( List l)`

The `Arrays` class is similar to `Collections`, but for arrays of objects. Arrays can also handle arrays of primitive values like int and double. I'll show method signatures for int[] arrays.

Read on!

- ➢ **Binary search** – `int binarySearch( int[] array, int value)`: performs a binary search of the specified value. The array must previously be sorted before making this call
- ➢ **Sort** – `Arrays.sort(int[] array)`: sorts the array in natural order
- ➢ **toString** – `Arrays.toString( int[] array)` - converts an array into a nicely-formatted string. We have done this many times to convert an array into a readable string, instead of something like "ArrayName@684038903082"

30. Thank you

*Submitted by: **Prof Bill***

I like these Snippets. Everyone in class contributed and, I think, everyone benefited.
Great job, 161!
thanks… yow, bill

.