Welcome to the Edit-Compile-Run Cycle!

Learning Outcomes

By the end of this lab:
1. Know where to find information about the course.
2. Be able to define chapter 1 terms.
3. Be able to trace and explain chapter 1 short code segments.
4. Describe the edit-compile-run cycle.
5. Describe what an editor, compiler, and virtual machine do.
6. Use a command-line editor, compiler and virtual machine to create, compile and run a Java program.
7. Describe errors that occur at each step.

Preparation

Prepare to discuss these with others. The lab is a helpful study group environment with grading based on attendance and participation. When working through the activities, trade off who goes first. Try to help each other. Verify traces with tools such as Java Visualizer (right side of course website). Verify summary explanations with other students and TAs. TAs plan to have a discussion on these topics with every student every lab. Please be prepared for these discussions.

Course Information

All of these can be found on the course website:

1. When are the 2 midterms and final exam?
2. If you have exam conflicts or other accommodation requests when and how do you report these to your instructor?
3. Where and when can you get help from CS 200 consultants?
4. What are five (5) different ways to get help?
5. Have you activated your CS account and received a cslogin? What is your cslogin?
6. What drive should you store your code files on in this lab?

Terms

Discuss the following terms with your partner.
input vs output, statement, code, execute, run, comments, whitespace, editor vs. compiler vs. virtual machine

Trace and Explain

What does Tracing a program mean?

Tracing means to execute the code in your mind as the computer would in order to figure out exactly what the code does. Check your tracing by putting the code in Java Visualizer (right side of course website) and stepping through it. Anticipate each step before Java Visualizer executes it and anticipate changes in memory.
before Java Visualizer shows them. Developing the knowledge and skill to trace code is very helpful for being able to explain and then write code.

What does Explain mean?
Explainning means summarizing in plain English the purpose of the code in a sentence or two.

Example
System.out.println("Hello" + "World");

Example Trace: (thinking or saying out loud) I notice that "Hello" and "World" are String literals, meaning a sequence of characters. I notice the + next to a string literal means concatenate (put together). Concatenating those strings would result in "HelloWorld". I then notice that this resulting value would be passed to the println method. A value passed to a method we call an argument. The println method writes the argument out to the screen. I also recall the println method effectively appends a newline "n" character to the end so that after HelloWorld is printed the next output would begin on the line after HelloWorld.

Example Explain: This line writes out "HelloWorld" followed by end of line.

With your partner, trace and explain each of the following. Verify with each other your understanding.
A. System.out.print("This is a short Java " + " program.
B. System.out.println("");
C. System.out.println("\n\n\n");
D. System.out.println("The title is "Programming with Java").
E. System.out.print("\the");
F. int i = 5;
G. int j = 4 + i;
H. System.out.println( i );
I. System.out.print( "i" );
J. System.out.println( i + j );
K. System.out.println("i + j = " + i + j);
L. System.out.println("This is the number + i.");
M. //This is a file header, right at the top of the file.
   //Typically containing author and reference information.

/**
 * This is the class header. Note the comment style and
 * that it is just before the class definition. This
 * class calculates information about a square.
 */
public class Square {

/**
 * This main method contains the whole program. This method
 * calculates the area and circumference of a square, when
 * provided the width of the square.
 */
public static void main(String[] args) {
    int width = 2;

    // calculate the area and circumference of a square
    int squareArea = width * width;
    int squareCircumference = width * 4;

    // write out the results
    System.out.println("square width: " + width + " area: " +
                      squareArea
                      + " circumference: " + squareCircumference);
}

Edit-Compile-Run Cycle

We recommend that you do this lab on the provided Windows lab computers. If you would like to do this lab on your own computer then you will need to have a simple text editor, similar to Windows Notepad, and Java compiler and virtual machine installed such that they work on the Command Prompt in Windows or from a Terminal window on a Mac or Linux system.

Start a Command Prompt:

A Command Prompt is a way of executing commands on a computer. The main difference is it has a textual interface rather than a graphical user interface like Windows. One way to start a Command Prompt is to open up the Start menu (press Windows key on keyboard) and type

```cmd
```

followed by [Enter] in the search box. Alternatively, you can find the command prompt application under Start > All Programs > Accessories > Command Prompt. Upon startup, you should see the prompt:

```C:\Users\your-cs-login>
```

In a command-prompt window:
This prompt lets you know what directory you are currently working with (current working directory). The C: drive is the hard drive on the computer you are sitting at and so files there will be available to anyone on that computer. Don't save files to the C: drive.

Switch to the L: drive, by typing

L:

You should see:

L:

To see more clearly what this means, open up File Explorer (should be to the right of the Start menu) and navigate to this same directory by double clicking on “L:”.
From the File Explorer, it is easy to see what files and folders are at your disposal in the current directory.

From the Command Prompt, use the command

dir

followed by [Enter] (Note that DOS commands are case insensitive, ie. DIR, dir and Dir will all work).
Like File Explorer, this outputs Date modified, Type, Size, and Name about each file and folder in the current directory.

Writing a Program

To run the Notepad program from the Command Prompt window, type:

```
notepad Hello.java
```

Notepad program should start. Click Yes to create a new file.

Next, type the following program into Notepad.

```java
public class Hello {
    public static void main(String []args) {
        System.out.println("Hello <your name>");
    }
}
```
Be sure to save the file (Ctrl-S).
Keeping Notepad open, click on the Command Prompt window and type

dir

Do you see the file Hello.java? What is the file’s size in bytes?

To see the contents of the .java file from the Command Prompt type:

type Hello.java

Note that a .java file is simply characters typed into the file following the Java Language Syntax. Within a .java file there cannot be formatting such as bold, italic, underlining, etc. (Different editors may display the syntax with bold and color as we will see when we begin using Eclipse, however that special formatting is not within the .java file itself.)

What could go wrong when editing a file?

Compiling a Program
To verify the Java compiler is able to run from the Command Prompt type:

javac -version

if the response is javac 1.8.0_144 then typing that command will run the 1.8 version of the compiler.

If the response is "javac' is not recognized as an internal or external command" then the compiler may be installed on the computer but not be easily available.
   On the lab computers the javac compiler is installed in the folder:

   C:\Program Files\Java\jdk1.8.0_144\bin

   To run the compiler type at the command prompt the full path as follows:
"C:\Program Files\Java\jdk1.8.0_144\bin\javac" -version

Hopefully, the response is:

javel 1.8.0_144

If so, then when you run the long command above you are running version 1.8 of the javac compiler. Let's try compiling our Hello.java program.

"C:\Program Files\Java\jdk1.8.0_144\bin\javac" Hello.java

Instead of retyping the long command, you can use the Up Arrow on the keyboard to recall a previously typed command and then use Backspace to edit it.

Compile the Hello.java program, whichever way is easiest and available to you.

javel Hello.java

or, if necessary, the longer form:

"C:\Program Files\Java\jdk1.8.0_144\bin\javac" Hello.java

Hopefully the response of the compiler is simply the prompt L:\> . That means success. If any errors show up, fix them (case, punctuation, some spacing, keywords and some naming all matter) and save, and recompile until no errors are found.

Running a Program
When compiling is successful, at the Command Prompt type:

dir

Look for a Hello.class file. The Hello.class file contains the compiled version of your program. The compiled version is in the form of bytecodes that are instructions for the Java virtual machine (JVM). To run your program, at the Command Prompt type:

java Hello

Hopefully, you will see:

Hello <your name>

If instead, you see an error message such as:

Error: Could not find or load main class Hello
Then the Hello.class file could not be found. Perhaps the Hello.java file wasn't compiled without error or there was a typo. When you get the program to successfully run then you have completed an Edit-Compile-Run cycle. Answer the following questions. Be prepared to discuss these with a TA.

1. What are the 3 programs used?
2. What is the purpose of each program?
3. Does the order the programs are run matter?
4. Since the programs are run separately, how is information passed between them?
5. What are some errors that could happen when using the editor, compiler, and virtual machine?
6. What do you think we are referring to when we say "compile time" and "run time"?

Create Syntax or Compile time errors

Make the following errors in order to become familiar with the messages the compiler provides. After making the error make a note of the compiler message.

- Was the message helpful and accurate or was it misleading?
- Are there key parts of the compiler message that are helpful for resolving the error?

Fix the code and then make the next error.

1. Remove a " and recompile.
2. Remove a }, and recompile.
3. Change the name of the class to Helo, and recompile.
4. Change the name of the main method to min, and recompile.
5. Remove the keyword static, and recompile.
6. Change System.out.println to println, and recompile.
7. Remove a ;, and recompile.

Exercises

Exercise A

See how many mistakes you can fix in the following example. You are welcome to use a compiler and virtual machine, or the Java Visualizer to help. (Don't look in the textbook until you have spent at least 1 hour without solving.)

```java
import java.util.Scanner;

/*
   This program calculates the amount of pasta to cook, given the
   number of people eating.

   Author: Mario Boyardee
   Date: March 9, 2014
*/

public class pastaCalculator {  
    public static void main (String [] args) {
```
scanner scnr = new Scanner(System.in);
//int numPeple = 0;  // Number of people that will be eating
int totalOuncesPasta = 0;  // Total ounces of pasta to serve numPeople

// Get number of people
System.out.println("Enter number of people: ");
numPeople = scnr.nextInt();

// Calculate and print total ounces of pasta
totalOuncesPasta = numPeople * 3;  // Typical ounces per person
System.out.println("Cook + totalOncesPasta + ounces of pasta.");

return 
}

Exercise B

In the following program, the order of lines has changed and any indenting has been removed. Please reorder all the lines and indent them correctly. You are welcome to use a compiler and virtual machine, or the Java Visualizer to help. (Don't look in the textbook until you have spent at least 1 hour without solving.)

} }
}
yetAnotherVar = myFirstVar;
thirdVar = yetAnotherVar + 1;
return;
public static void main(String[] args) {
public class WhiteSpaceEx {
myFirstVar = scnr.nextInt();
int yetAnotherVar = 0;
int thirdVar = 0;
int myFirstVar = 0;
import java.util.Scanner;
System.out.println("Final value is " + thirdVar);
System.out.print("Enter a number: ");
Scanner scnr = new Scanner(System.in);

Exercise C

In the following program, the order of lines has changed and any indenting has been removed. Please reorder all the lines and indent them correctly. You are welcome to use a compiler and virtual machine, or the Java Visualizer to help. (Don't look in the textbook until you have spent at least 1 hour without solving.)

/* Computes the total cost of leasing a car given the down payment,
* monthly rate, and number of months
*/
public class CarLeaseCost {
import java.util.Scanner;
}

public static void main (String [] args) {
    totalCost = downpayment + (paymentPerMonth * numMonths);
    return;
    paymentPerMonth = scnr.nextInt();
    numMonths = scnr.nextInt();
    int totalCost = 0; // Computed total cost to be output
    int paymentPerMonth = 0;
    int numMonths = 0;
    int downpayment = 0;
    downpayment = scnr.nextInt();
    System.out.println("Total cost: "+ totalCost);
    System.out.println("Enter number of months: ");
    System.out.println("Enter monthly payment: ");
    System.out.println("Enter down payment: ");
    Scanner scnr = new Scanner(System.in);

Exercise D

Write a program named YourName.java that will produce output similar to the following, but for your name. Use multiple print/println statements.

    J  I  M  M
    J  I     MM MM
    J  I  M  M M
    J  J  I  M  M

Now, can you write the program with only 1 print/println method call and also not exceeding a line length of 100 characters? Is it better to use 1 print/println statement or more? In other words, which way of writing the code will be easier to read for someone else?

Additional Learning Materials

When you have mastered everything in this lab, then you are welcome to learn from additional learning resources available on the web and beyond this course:
https://cs200-www.cs.wisc.edu/wp/learn-to-program-resources/

Note: Due to programs and zyLabs being individual work, it is Not appropriate to work on them during the Team Lab.

Lab designed by Jim Williams using some material from zyBooks Programming in Java.