Debugging

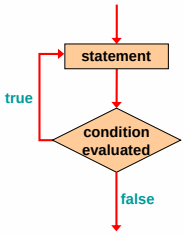
October 4, 2006

ComS 207: Programming I (in Java)
Iowa State University, FALL 2006
Instructor: Alexander Stoytchev

HW Questions?

Quick review of last lecture

Logic of a do Loop



The do Statement

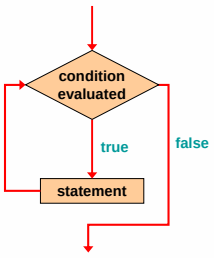
- A *do statement* has the following syntax:

```
do
{
   statement;
}
while ( condition )
```

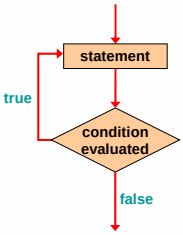
- The `statement` is executed once initially, and then the `condition` is evaluated
- The statement is executed repeatedly until the condition becomes false

Comparing while and do

The while Loop



The do Loop



The do Statement

- An example of a do loop:

```
int count = 0;
do
{
   count++;
   System.out.println (count);
} while (count < 5);
```

- The body of a do loop executes at least once
- See ReverseNumber.java (page 244)

The for Statement

- A *for statement* has the following syntax:

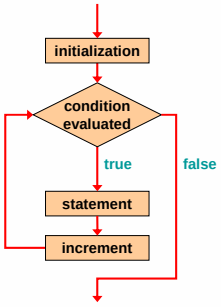
The *initialization* is executed once before the loop begins

The *statement* is executed until the *condition* becomes false

```
for ( initialization ; condition ; increment )
   statement;
```

The *increment* portion is executed at the end of each iteration

Logic of a for loop



The for Statement

- A `for` loop is functionally equivalent to the following `while` loop structure:

```
initialization;
while ( condition )
{
   statement;
   increment;
}
```


The for Statement

- An example of a `for` loop:

```
for (int count=1; count <= 5; count++)
   System.out.println (count);
```

- The initialization section can be used to declare a variable
- Like a `while` loop, the condition of a `for` loop is tested prior to executing the loop body
- Therefore, the body of a `for` loop will execute zero or more times

The for Statement

- The increment section can perform any calculation

```
for (int num=100; num > 0; num -= 5)
   System.out.println (num);
```

- A `for` loop is well suited for executing statements a specific number of times that can be calculated or determined in advance
- See Multiples.java (page 248)
- See Stars.java (page 250)

Example: Multiples.java (page 248)

Example: Stars.java (page 250)

The for Statement

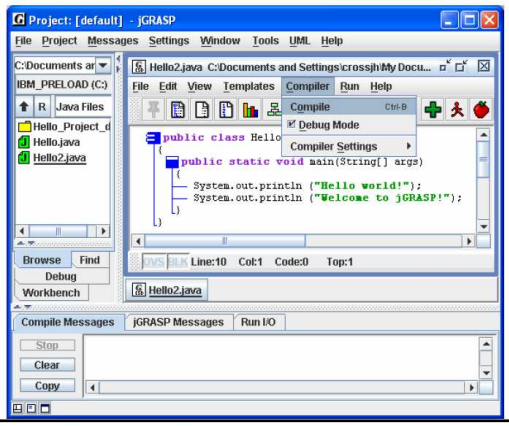
- Each expression in the header of a `for` loop is optional
- If the initialization is left out, no initialization is performed
- If the condition is left out, it is always considered to be true, and therefore creates an infinite loop
- If the increment is left out, no increment operation is performed

Note: This material is not from the textbook.

It is from the jGRASP manual.

How to use the jGRASP Debugger

The jGRASP Tutorials

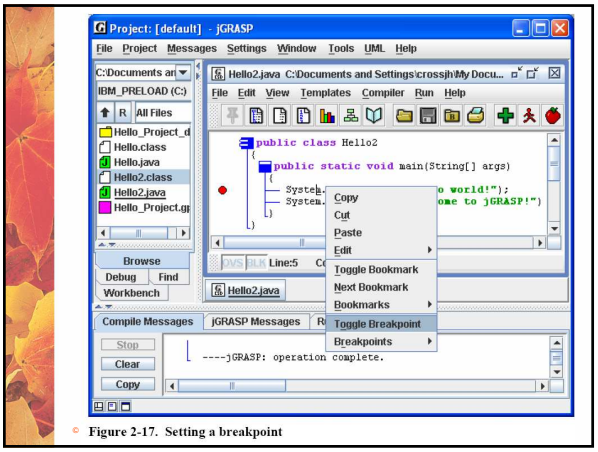


2.10 Using the Debugger

jGRASP provides an easy-to-use visual Debugger that allows you to set one or more breakpoints, then step through a program statement by statement. To set a breakpoint, left-click on the statement where you want your program to stop, then right-click and select **Toggle Breakpoint** (Figure 2-17). You should see the red octagonal breakpoint symbol appear to the left of the line. The statement you select must be an executable statement (i.e., one that causes the program to do something). You can also set a breakpoint by hovering the mouse over the leftmost column of the line where you want to set the breakpoint. When you see the red breakpoint symbol, left-click the mouse to set the breakpoint.

In the Hello2 program below, a breakpoint has been set on the first of the two *System.out.println* statements, which are the only statements in this program that allow a breakpoint.

After setting the breakpoint, click **Run – Debug** (Figure 2-18). This should raise the Debug tab pane (in place of the Browse tab pane), and your program should stop at the breakpoint. The highlighted statement is the one about to be executed.



© Figure 2-17. Setting a breakpoint

To *step* the program, click on the "down-arrow" at the top of the Debug pane. Each time you click on the "down-arrow", your program should advance to the next statement. After stepping all the way through your program, the Debug tab pane will go blank to signal the debug session has ended.

In the example below, the program has stopped at the first output statement. When the *step button* (down-arrow) is clicked, this statement will be executed and "Hello world!" will be printed standard out and shown in the Run I/O tab pane. Clicking the step button again will output "Welcome to jGRASP!" on the next line. The third click on the step button will end the program, and the Debug tab pane should go blank as indicated above.

If you want to step through your program automatically, turn on *AutoStep* by clicking the button with multiple "down-arrowheads." With this feature turned ON, after your program stops at the first breakpoint and you click the *Step* button the first time, the debugger will step through your program at the rate indicated. For more details see the tutorial on the *Integrated Debugger*.

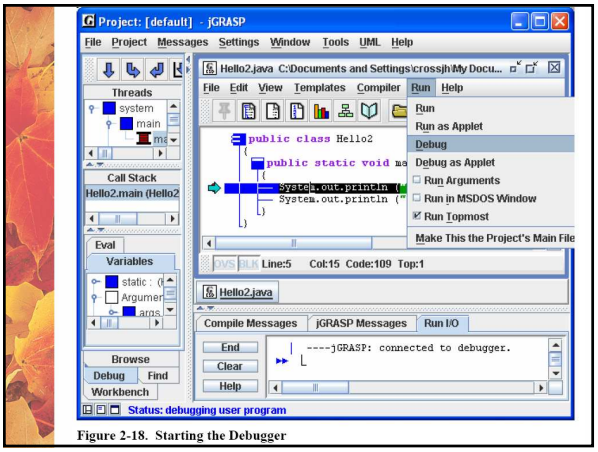
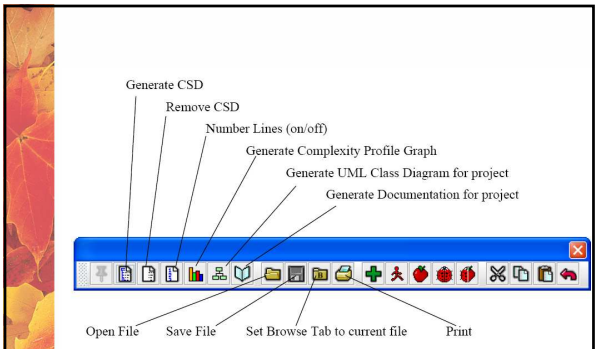
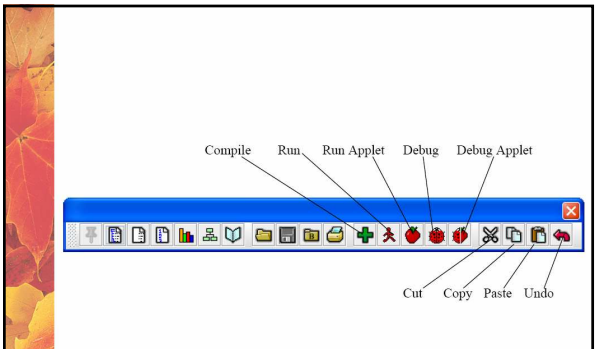


Figure 2-18. Starting the Debugger





Debugging Examples and Demos

Example: Multiples.java (page 248)

Example: Stars.java (page 250)

Other Things About Loops

- 'Break' Statement
- 'Continue' Statement
- Empty Statement - ';'

The for Statement

- A *for statement* has the following syntax:

The *initialization* is executed once before the loop begins

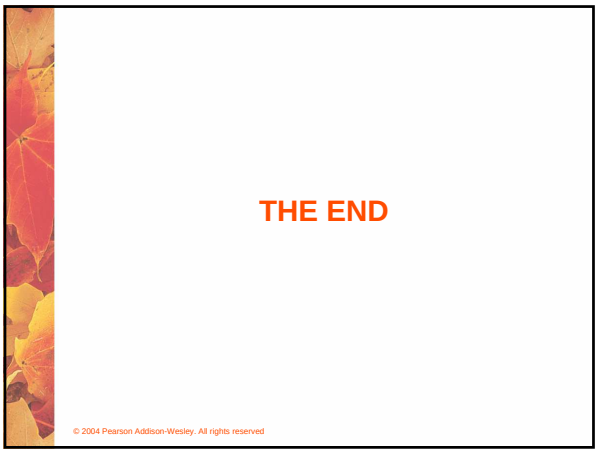
The *statement* is executed until the *condition* becomes false

```
for ( initialization ; condition ; increment )
   statement;
```

The *increment* portion is executed at the end of each iteration

The for Statement

- Each expression in the header of a `for` loop is optional
- If the initialization is left out, no initialization is performed
- If the condition is left out, it is always considered to be true, and therefore creates an infinite loop
- If the increment is left out, no increment operation is performed


THE END
© 2004 Pearson Addison-Wesley. All rights reserved