

Building Java Programs

Lab 2: Ch. 2: Expressions and Variables, For Loops

Except where otherwise noted, the contents of this document are Copyright 2012 Stuart Reges
Marty Stepp.

lab document created by Marty Stepp and Stuart Reges

Today's lab

Goals for today:

- use **expressions** and **variables** for numeric computations
- trace and write **for loops** for repeating lines of code
- draw **patterned figures** of text using loops
- use **class constants** to represent common unchanging values

Expressions

Recall that Java has **expressions** to represent math and other computations. Expressions may use **operators**, which are evaluated according to rules of **precedence**. Every expression produces a value of a given **type**.

Type	Description	Example	Result
int	integers (up to $2^{31} - 1$)	<code>3 + 4 * 5</code>	23
double	real numbers (up to 10^{308})	<code>3.0 / 2.0 + 4.1</code>	5.6
String	text characters	<code>"hi" + (1 + 1) + "u"</code>	"hi2u"

Exercise 1: Expressions (2.1)

Write the results of each of the following expressions. If you're stuck, ask a TA or neighbor.

$$12 / 5 + 8 / 4$$

$$3 * 4 + 15 / 2$$

$$-(1 + 2 * 3 + (1 + 2) * 3)$$

$$42 \% 5 + 16 \% 3$$

$$2.5 * 2 + 17 / 4$$

$$4.5 / 3 / 2 + 1$$

Exercise 2: More expressions (2.1)

Write the results of each of the following expressions.

$$5 * 6 / 4 \% 3 - 23 / (14 \% 6)$$

$$30 \% 9 + 5 \% 8 - 11 \% 4 \% 2$$

$$1 + 9 / 2 * 2.0$$

$$46 / 3 / 2.0 / 3 * 4/5$$

$$50 / 9 / 2.0 + 200 / 10 / (5.0 / 2)$$

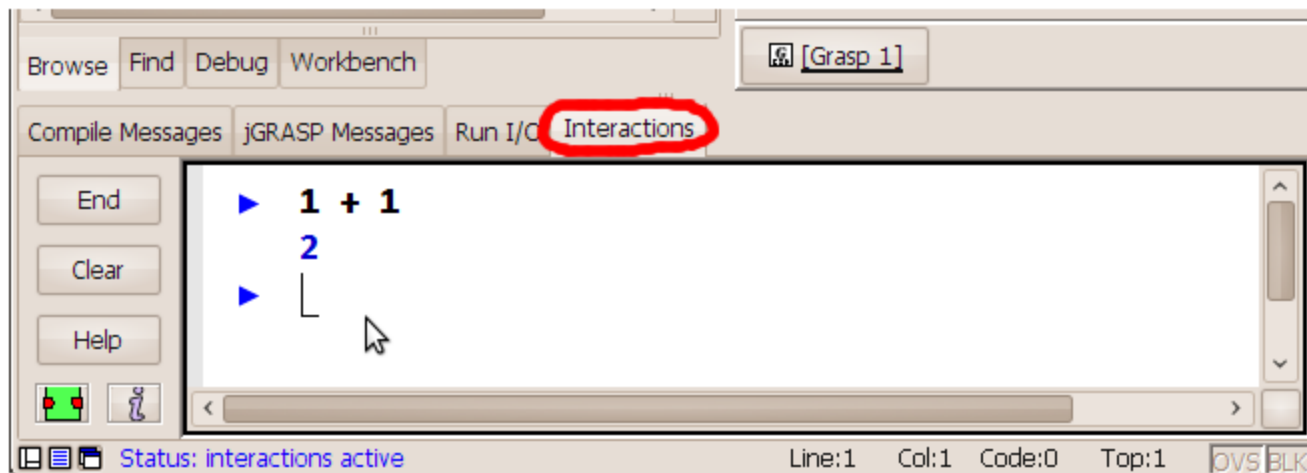
Exercise 3: Expressions

For each expression in the left-hand column, indicate its value in the right-hand column. Be sure to list a constant of appropriate type (e.g., 7.0 rather than 7 for a double, Strings in quotes).

<code>4 * (2 + 4) - 3 * 5</code>	
<code>54 % 10 + 8 * 3 % 9</code>	
<code>3 * 2 + 4 + "+" + 2 + 3 * 4</code>	
<code>2.3 * 3 + 19 / 5 / 2 + 6.0 / 5</code>	
<code>108 / 20 * 3 / 4 / 2.0 + 1.0 / 2</code>	

jGRASP Interactions Pane

- jGRASP has a useful feature called the **Interactions Pane** that allows you to type Java expressions or statements one at a time and instantly see their results.
- To use it, run jGRASP and then click the "Interactions" tab near the bottom.



continued on the next slide...

Exercise 4: Using jGRASP Interactions Pane

In this exercise, you'll use the Interactions Pane to quickly discover the result of some expressions that would be difficult to evaluate by hand. **Copy/paste** each expression below into the Interactions Pane to evaluate it, then input the answer into this slide.

<code>123 * 456 - 789</code>	
<code>3.14 + 1.59 * 2.65</code>	
<code>2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2</code>	
<code>2 + 2 + "xyz" + 3 + 3</code>	

(For the last expression, the Interactions Pane doesn't put "" quotes around Strings when displaying results, so you must add those yourself if needed. For example, if the Interactions Pane gives you result of `abc123`, it should be written here as `"abc123"`.)

Variables

Recall that you can use a **variable** to store the results of an expression in memory and use them later in the program.

```
type name;           // declare
name = value or expression; // assign a value
...
type name = value or expression; // declare-and-initialize together
```

Examples:

```
double iPhonePrice;
iPhonePrice = 499.95;

int siblings = 3;
System.out.println("I have " + siblings + " brothers/sisters.");
```

Exercise 5: Variable declaration syntax

Which of the following choices is the correct syntax for declaring a real number variable named `grade` and initializing its value to `4.0`?

- a. ☐ `grade = 4.0;`
- b. ☐ `4.0 = grade;`
- c. ☐ `int grade = 4.0;`
- d. ☐ `grade : 4.0;`
- e. ☐ `grade = double 4.0;`
- f. ☐ `double grade = 4.0;`

Exercise 6: Variable assignment syntax

Suppose you have a variable named `grade`, set to 1.6:

```
double grade = 1.6; // uh-oh
```

Suppose later in the program's code, we want to change the value of `grade` to 4.0. Which is the correct syntax to do this?

- a. ☐ `set grade = 4.0;`
- b. ☐ `double grade = 4.0;`
- c. ☐ `grade : 4.0;`
- d. ☐ `4.0 = grade;`
- e. ☐ `grade = 4.0;`

Exercise 7: a, b, and c

What are the values of a, b, and c after the following statements? Write your answer in the boxes on the right.

```
int a = 5;
int b = 10;
int c = b;

a = a + 1;           // a?
b = b - 1;           // b?
c = c + a;           // c?
```

Exercise 8: i, j, and k

What are the values of *i*, *j*, and *k* after the following statements?

```
int i = 2;  
int j = 3;  
int k = 4;  
int x = i + j + k;
```

<code>i = x - i - j;</code>	<code>// i?</code>	
<code>j = x - j - k;</code>	<code>// j?</code>	
<code>k = x - i - k;</code>	<code>// k?</code>	

Exercise 9: ComputePay

The following program redundantly repeats the same expressions many times. Download it and open it in jGRASP, then modify the program to remove the redundancy using **variables**. Use an appropriate type for each variable.

- [!\[\]\(5338ce82ce7b6718f3db173c378c93da_img.jpg\) ComputePay.java](#)

The program's output should be the same after your modifications. No expression should be computed more than once in the code.

Exercise 10: Syntax errors

- The following program contains 9 mistakes! What are they? Copy and paste the following code into jGRASP and correct the various mistakes. The answer is on the next 2 slides if you need some help.

```
1 public class Oops {  
2     public static void main(String[] args) {  
3         int x;  
4         System.out.println("x is" x);  
5  
6         int x = 15.2;    // set x to 15.2  
7         System.out.println("x is now + x");  
8  
9         int y;           // set y to 1 more than x  
10        y = int x + 1;  
11        System.out.println("x and y are " + x + and + y);  
12    }  
13 }
```

answer on next slide...

Exercise 10 - answer

1. line 4: missing `+` between `"x is"` and `x`
2. line 4: cannot print the value of `x` before assigning it a value
3. line 6: cannot assign `15.2` into a variable of type `int`
4. line 6: should not redeclare the variable's type
5. line 7: `"` mark should be between `now` and `+`
6. line 10: should not write the word `int` here
7. line 10: variable `y` should be same type as `x`
8. line 10: does not properly set `y` to be 1 more than `x` (should not write the word `int` here)
9. line 11: `and` should be in quotes with surrounding spaces

Exercise 10 - corrected version

- Here is a corrected version of the program:

```
public class Oops {  
    public static void main(String[] args) {  
        double x = 0.0;  
        System.out.println("x is" + x);  
  
        x = 15.2;           // set x to 15.2  
        System.out.println("x is now " + x);  
  
        double y;          // set y to 1 more than x  
        y = x + 1;  
        System.out.println("x and y are " + x + " and " + y);  
    }  
}
```

for loops

A **for loop** repeats a group of statements a given number of times.

```
for (initialization; test; update) {  
    statement(s) to repeat;  
}
```

Example:

```
for (int i = 1; i <= 10; i++) {  
    System.out.println("We're number one!");  
}
```

Exercise 11: simple for loop

- **Copy/paste** the following code into jGRASP.

```
public class Count2 {  
    public static void main(String[] args) {  
        for ( fill me in! ) {  
            System.out.println( fill me in! );  
        }  
    }  
}
```

- **Modify** the code to produce the following output:

```
2 times 1 = 2  
2 times 2 = 4  
2 times 3 = 6  
2 times 4 = 8
```

Exercise 12: Verify solution in Practice-It!

Our **Practice-It!** system lets you solve Java problems online.

- Go to the  [Practice-It!](#) web site.
- **Create an account** if you don't have one, and log in.
- Under the "CS1 Labs", "Lab 2" category, select the "simple for loop" problem. ([direct link](#))
- Copy/paste your program from jGRASP into Practice-It, and **submit**. If it does not pass the test, modify your code and re-submit it.

Lab 2 (expression)

Write a complete Java program in a

```
2 times 1 = 2
2 times 2 = 4
2 times 3 = 6
2 times 4 = 8
```

```
1 public class Count2 {
2     public static void main() {
3         for ( ... ) {
            System.out.println("2 times " + i + " = " + 2*i);
        }
    }
}
```



#	name	expected output
1	output	2 times 1 2 times 2 2 times 3 2 times 4

You passed 1

Exercise 13: What's the output?



What output is produced by the following Java program? Write the output in the box on the right side.

```
public class OddStuff {  
    public static void main(String[] args) {  
        int number = 32;  
        for (int count = 1; count <= number; count++) {  
            System.out.println(number);  
            number = number / 2;  
        }  
    }  
}
```

Output:

Exercise 14: Bottles of beer

- Write a for loop that produces the song *Bottles of Beer on the Wall*:

```
10 bottles of beer on the wall, 10 bottles of beer  
Take one down, pass it around, 9 bottles of beer on the wall  
  
9 bottles of beer on the wall, 9 bottles of beer  
Take one down, pass it around, 8 bottles of beer on the wall  
  
... (output continues in the same pattern) ...  
  
1 bottles of beer on the wall, 1 bottles of beer  
Take one down, pass it around, 0 bottles of beer on the wall
```

- Write and test your code in **Practice-It!** by clicking the checkmark icon  at the bottom of the slide.

Exercise 15: jGRASP Debugger

- Breakpoints help you **debug** programs ([tutorial](#)). Copy/paste this code into jGRA

```
1 public class Numbers {  
2     public static void main(String[] args) {  
3         int number = 42;  
4         for (int i = 1; i <= 1000; i++) {  
5             number = number * 37 % 103;  
6         }  
7         int number2 = number * number;  
8         System.out.println("result = " + number2);  
9     }  
10 }
```

- Set a breakpoint** on the statement inside the for loop (line 5). Do this by moving your cursor to the beginning of that line until you see a **stop-sign icon**  and the clicking.

continued on the next slide...

Exercise 15 - jGRASP Debugger

- Now you can debug the program by clicking on the debug icon  (looks like a ladybug).
- It will stop every time it hits line 5. Then you can see the values of variables `i` and `number`.
- To continue running, press the top-left **play**  or **step**  buttons.
- Using this approach, **fill in** the table below indicating what value `number` has when `i` has the given value. Keep in mind that you are figuring out what value `number` has just before it executes this line of code.

<code>i = 1, number =</code>	
<code>i = 2, number =</code>	
<code>i = 3, number =</code>	
<code>i = 4, number =</code>	

continued on the next slide...

Exercise 15 - jGRASP Debugger

- Click on the stop-sign  a second time to get rid of it and then set a new stop point on line number 7. This will allow you to find out the value of `number` after the loop is done executing.
- Again press "play"  to start the programming running again.
- At line 7, what is the value of `number`?

Exercise 16: Sequence of numbers

- **Write** a complete Java program that produces this sequence of numbers using a for loop:

```
8
11
14
17
20
23
```

- Hint: You want to come up with an expression with a multiplier and a constant. Pick a multiplier (if your loop variable goes up by 1 each time, how much do these numbers go up by each time?). Then pick a constant (when your loop variable is this is supposed to be 8, what do you have to add to get this value?).

Exercise 17: Sequence of characters

- **Modify** the code from the previous exercise to produce the following output:

```
*****
*****
*****
*****
*****
*****
*****
```

- Use the [Output Comparison Tool](#) to check your work.
- (Hint: Note the number of stars in each row. Instead of printing the number 8, now you're printing 8 stars. You will now need a nested loop.)

Exercise 18: for loop table practice

Create a **table** of the number of stars on each line:

```
*****  
*****  
***  
*
```

line stars

1	
2	
3	
4	

multiplier: How much does the # of stars change between lines?

shift: Given your multiplier, what must be added to get that many stars on line 1?

Test your loop expression in **Practice-It!** using the checkmark icon above. Use the form:

```
for (int stars = 1; stars <= multiplier * line + shift; stars++) {
```

Exercise 19: What's the output?

- What output is produced by the following Java method?

```
public static void stars() {  
    for (int i = 1; i <= 10; i++) {  
        for (int j = 1; j <= i; j++) {  
            System.out.print("*");  
        }  
        for (int j = 1; j <= 20 - 2 * i; j++) {  
            System.out.print(" ");  
        }  
        for (int j = 1; j <= i; j++) {  
            System.out.print("*");  
        }  
        System.out.println();  
    }  
}
```

- (Try to figure it out without running the code...
If you give up, copy and paste it into jGrasp and run it!)

• Hint: Think of each inner loop from 1 to X as a loop that prints X copies of the given

Exercise 20: What's the output?

- What's the output of the following Java method?

```
public static void mystery() {  
    for (int i = 1; i <= 10; i++) {  
        for (int j = 1; j <= 10; j++) {  
            System.out.print(i * j + "\t");  
        }  
        System.out.println();  
    }  
}
```

- (Try to figure it out without running the code...
If you give up, copy and paste it into jGrasp and run it!)
- Recall: `\t` represents a tab character.

Exercise 20 - answer

Answer:

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81
10	20	30	40	50	60	70	80	90

Exercise 21: printing a design

Write a program to produce the following output using nested `for` loops. Use a table to help you figure out the patterns of characters on each line.

```
-----1-----  
-----333-----  
---55555---  
--7777777--  
-999999999-
```

	Line	Dashes	Numbers
--	------	--------	---------

1		
2		
3		
4		
5		

dashes expression

* line +

numbers expression

* line +

Test your loop expressions in **Practice-It!** using the checkmark icon above. Use your expressions in the loop tests of the inner loops of your code.

Exercise 22: SlashFigure

Write a Java program in a class named `SlashFigure` to produce the following output with nested `for` loops. Use a loop table if necessary to figure out the expressions.

```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!
\\!!!!!!!!!!!!!!!!!!!!!!//
\\\\!!!!!!!!!!!!!!!!!!!!//
\\\\\\\\!!!!!!!!!!!!//
\\\\\\\\\\\\!!!!!!!!//
\\\\\\\\\\\\\\\\!!//
\\\\\\\\\\\\\\\\\\//

```

Line	\	!	/
1			
2			
3			
4			
5			
6			
multiplier			
shift			

Test your code in **Practice-It!** or the [Output Comparison Tool](#).

Exercise 23: Equation

Suppose you have a real number variable x . Write a Java expression that computes variable named y storing the following value:

$$y = 12.3x^4 - 9.1x^3 + 19.3x^2 - 4.6x + 34.2$$

(We haven't learned a way to do exponents yet, but you can simulate them using several multiplications.)

Use the example program on the next slide to test your code.

Exercise 23 - Example code

Copy/paste this program into jGRASP to test your solution.

```
// expected output:  
// y is 7043.7  
  
public class EquationY {  
    public static void main(String[] args) {  
        double x = 5;  
  
        double y = put your expression for y here ;  
  
        System.out.println("y is " + y);  
    }  
}
```

(answer on next slide)

Exercise 23 - answer

```
double y = 12.3*x*x*x*x - 9.1*x*x*x + 19.3*x*x - 4.6*x + 34.2;
```

If you want an added challenge, try to come up with a way to compute the above value while using the `*` operator no more than 4 times.

(click Next → for answer)

```
double y = (((12.3 * x - 9.1) * x + 19.3) * x - 4.6) * x + 34.2;
```

Exercise 23 - answer

```
double y = 12.3*x*x*x*x - 9.1*x*x*x + 19.3*x*x - 4.6*x + 34.2;
```

If you want an added challenge, try to come up with a way to compute the above value while using the `*` operator no more than 4 times.

(click Next → for answer)

```
double y = (((12.3 * x - 9.1) * x + 19.3) * x - 4.6) * x + 34.2;
```

Exercise 24: Birthday variables

- Create a complete Java program in a class named `Bday` that declares four variables and assigns appropriate values to them.
 - your birthday month (1-12)
 - your birthday day (1-31)
 - the birthday month of another student sitting near you today (1-12)
 - the birthday day of that same student near you (1-31)
- Ask your neighbor for their name and for the proper numbers to store in the variables for his/her birthday. Then produce output in this format using your four variables:

My birthday is 9/19, and Suzy's is 6/14.

Class constants

A **class constant** is a global value that cannot be changed.

```
public static final type name = expression;
```

Example:

```
public static final int DAYS_PER_WEEK = 7;  
public static final double TAX_RATE = 0.10;
```

for loop expressions w/ constant

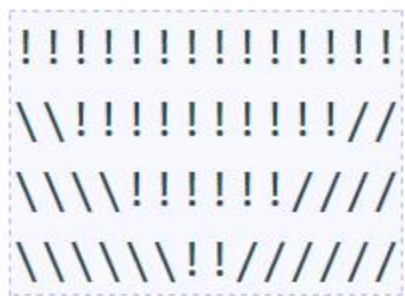
When adding a **class constant** to a loop expression, it affects the constant that must be added in the expression. Suppose we have the two loop expressions below for figure sizes of 5 and 9. The third line of the table shows the general formula that would be used if we turned our figure's size into a constant named `SIZE`.

size	expression	relationship
5	<code>8 * line + 16</code>	<code>16 = 3 * 5 + 1</code>
9	<code>8 * line + 28</code>	<code>28 = 3 * 9 + 1</code>
SIZE	<code>8 * line + (3 * SIZE + 1)</code>	

continued on the next slide ...

Exercise 25: for loop table w/ constant

You already found loop expressions for the slash figure at size 6. Now make a **table** size 4 and use the two to generalize the loop expression in terms of a **constant** for figure size.



Line \	!	/
1		
2		
3		
4		

\ and /	
size 6	2 * line + -2
size 4	2 * line +
size SIZE	2 * line +
!	
size 6	-4 * 26
	line +
size 4	-4 *
	line +
size	-4 * (* SIZE
SIZE	line + +)

Exercise 26: SlashFigure2

- **Add a class constant** to your slash figure program so that it can be resized from its default of 6:

size 4	size 7
	

Test your code in the [Output Comparison Tool](#).

Exercise 27: Number lines, part 1

- **Write** some nested for loops to produce the following output:

```
000111222333444555666777888999
000111222333444555666777888999
000111222333444555666777888999
```

- You can use the [Output Comparison Tool](#) to check your work.

Exercise 28: Number lines, part 2

- **Modify** the previous code to have it do something a little different:

```
99999888887777766666555554444433333222221111100000
99999888887777766666555554444433333222221111100000
99999888887777766666555554444433333222221111100000
99999888887777766666555554444433333222221111100000
99999888887777766666555554444433333222221111100000
```

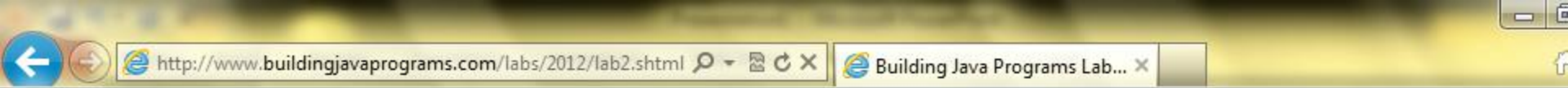
- You can use the [Output Comparison Tool](#) to check your work.

Exercise 29: Number lines, part 3

- **Modify** the code to have it produce this output:

```
9999999999888888887777777666666555554444333221
9999999999888888887777777666666555554444333221
9999999999888888887777777666666555554444333221
9999999999888888887777777666666555554444333221
```

- You can use the [Output Comparison Tool](#) to check your work.



If you finish them all...

If you finish all the exercises, try out our [Practice-It](#) web tool. It lets you solve Java problems from our *Building Java Programs* textbook.

You can view an exercise, type a solution, and submit it to see if you have solved it correctly.

Choose some problems from the book and try to solve them!