CS1130 Lab 01. Expressions, variables, declarations, assignments Spring 2012

This is webpage www.cs.cornell.edu/courses/cs1130/2011fa/handouts/labs/lab01.html

This lab deals with Java expressions and assignment statements, for the most part treating Java just as a calculator.

To open DrJava in a ACCEL-lab computer, use:

> Start --> All programs --> Class Files --> DrJava --> "the jar file that is listed there"

Below is a list of expressions. Next to each, write down the value you think it is (Write "?" if you have no idea). In the third column, write down the value that DrJava gives, by cutting and pasting each expression from the online pdf or html version of this handout (see above link) into the Interactions pane and hitting the enter key to have the expression evaluated. If your expected value was different from the calculated value, put an explanation in the fourth column of how you think the calculated value was calculated .

When finished, show this sheet to your lab instructor, who will record that you did it. If you do not finish during the lab, finish it within the next few days and show it to your lab instructor next week. This paper is yours to keep.

You may find it convenient to enlarge the Interactions pane: put the mouse on the horizontal bar running across the entire window, hold down the mouse button, and drag the bar upwards. Also, you can use the up-arrow key to obtain a previous expression; then you can edit the expression and hit the return key to have the modified expression evaluated.

Don't waste time! If you don't understand something, ask your lab instructor or a consultant immediately! You should understand HOW each expression is evaluated, so if an answer doesn't make sense, ask someone. The lab instructors and consultants are in the lab to help. They will look over your shoulder from time to time and give you advice.

INT EXPRESSIONS	Expected value	Calculated value	Reason for calculated value
5 + 2 * 5			
(5 + 2) * 5			
4 - 3 - 3			
4 - (3 - 3)			
-4 - -4 - -4			
6 / 2			
6 / 4			
7 % 2			
6 % 3			
Integer.MIN_VALUE			
Integer.MIN_VALUE + 1			
Integer.MIN_VALUE - 1			
Integer.MAX_VALUE + 1			

DOUBLE EXPRESSIONS	Expected value	Calculated value	Reason for caculated value
5.0 + 2.0			
(5 + 2.1) * 5			
4.0 - 3 - 3			
4.0 - (3 - 3)			
6.0 / 2			
6.0 / 4			
6.0 % 3			
-6.0 % 4			

Double.MIN_VALUE			
Double.MIN_VALUE + 1			
Double.MAX_VALUE			
Double.MAX_VALUE + 1			
Double.MAX_VALUE + Double.MAX_VALUE			

We want you to figure out why an addition like Double.MIN_VALUE + 1 equals 1. Here is the explanation:

In Java, the mantissa in "mantissa E exponent" can only hold a certain maximum number of digits. Let's see how this causes Double.MIN_VALUE + 1 to give 1:

Imagine a Java world where the mantissa can have only 3 digits. Both 3.24E5 and 32.4E5 can be stored in a double in this imaginary java world, but not 32.46E5 because it contains 4 digits in the mantissa.

Using this restriction in this imaginary Java world, add up the two numbers 1.0E0 and 3.20E-2. 1 + 0.032 = 1.032 = 1.032E0, right? But in our imaginary Java world where the mantissa can only hold 3 digits, the 2 would be dropped and imaginary Java will return 1.03E0. In essence, the least significant digits are simply removed.

Now, using the same restriction, YOU add up the two numbers 1.00E0 and 1.00E-4 —remembering that at all times, each mantissa is only 3 digits. What would imaginary Java return? This example should show you why Double.MIN_VALUE + 1 equals 1.

CASTING	Expected value	Calculated value	Reason for caculated value
(**double**) 4			
(**int**) 4			
(**double**) 7 / 4			
(**double**) (7 / 4)			
Which operator has higher precedence, casting or division? (rows above)			
(**int**) 5.3			
(**double**) (**int**) 5.3			
(**int**) -5.3			
7 / 4 + 5			
(**double**) 7 / 4 + 5			
7 / (**double**) 4 + 5			
(**double**) (7 / 4 + 5)			

BOOLEAN EXPRS	Expected value	Calculated value	Reason for caculated value
3 < 5			
3 < 5 && 5 < 3			
true			
true && **false**			
true && **true**			

false			
true || **false**			
true || **true**			
!**true**			
!**false**			
!!**false**			
true && **false** && **true**			
true || **false** || **true**			
true || (**false** && **true**)			
true && (**true** || **false**)			
false && (5 / 0 == 1)			
(5 / 0 == 1) && **false**			
Why does the expression above not work but the one above it does?			

STRING EXPRESSIONS	**Expected value**	**Computed value**	**Reason for caculated value**
"Truth " + "is " + "best"			
"Truth " + ("is " + "best")			
56 + "" + 56			
"" + 4 / 2			
("" + 4) / 2			
4 + 2 + ""			
4 + (2 + "")			
What does + do if at least one operand is a String?			

Variables, declarations, and assignment statements

It is important that you know the difference between a declaration and an assignment statement. A declaration like

boolean b;

simply says that in the rest of the "program", a variable named b may be used and its type is **boolean**. On the other hand an assignment like

b= 3 < 5;

is a command to do something: evaluate the expression 3<5 and store its value in variable b.

In Java, variables must be declared before they are used. In the newest version of DrJava, in the interactions pane, variables must be declared before they are used. You can change this to allow uses of variables without declarations. Do this now: Use menu item Edit->Preferences; in the window that opens, click on "Interactions pane" in the left column, and then uncheck the appropriate box.

VARIABLES, DECLARATIONS, ASSIGNMENTS			
Write the purpose of a declaration like "**int** j;".			
Explain how the assignment statement "j= j + 5;" is executed.			
Statement or Expression	**Expected value**	**Calculated value**	**Reason for caculated value**
int j;	None		
j			
j= 2;	None		
j			
j= j + 9;	None		
j			
k= 5;	None		
j + k			
double w= j + k;	None		(a combination of a declaration of a variable and an assignment to it)
w			

FUNCTION CALLS	**Expected value**	**Calculated value**	**Reason for caculated value**
Note: In the function call below, 25 and 4 are called the **arguments** of the call. In the third call below, the arguments are 25 and Math.max(27, 4)			
Math.min(25, 4)			
Math.max(25, 4)			
Math.min(25, Math.max(27, 4))			
Math.abs(25)			
Math.abs(- 25)			
Math.ceil(25.6)			
Math.floor(25.6)			
Math.ceil(- 25.6)			
Math.floor(- 25.6)			