Welcome to the Primitives and Expressions Lab!

Learning Outcomes

By the end of this lab:
1. Be able to define chapter 2 terms.
2. Describe declarations, variables, literals and constants for primitive numerical data types.
3. Describe String literals, concatenation and constants.
4. Evaluate a sequence of primitive data type expressions using binary operators (+, -, *, /, %, =) including conversions between numeric primitive types.
5. Use class (Math) methods in expressions.
6. Use a precedence and associativity table.
7. Write a method call and definition including parameter, and return value.

Preparation

Be prepared to review and discuss with others in the lab. The lab is a helpful study group environment. Grading is based on attendance and participation. When working through the activities, trade off who goes first. Try to help each other. Use tools such as Java Visualizer to verify code traces. Discuss with other students and group leaders to verify Explain summaries. Each TA plans to have a discussion on these topics with each student in their group every lab. Please be prepared for these discussions.

Exercise A: Terms

variable, variable declaration, data type, primitive data type, byte vs int vs short vs long vs float vs double, assignment vs equality, expression evaluation, identifier, identifier naming convention, camel casing, keyword, literal, 8 vs 8.0 vs 8.0F vs 8L, implicit type conversion vs explicit type conversion, constant vs final, method call vs. method definition, argument vs parameter

Exercise B: Trace and Explain

Trace and verify with Java Visualizer and Explain in plain English in a sentence or two.

Below is a Java Operator Precedence table that may be useful when tracing and explaining the following expressions. The Java language has rules of precedence so there is no ambiguity on the order operators in an expression are evaluated. Higher precedence operators will be evaluated before those of lower precedence. Associativity tells us how operators with the same precedence are grouped when parentheses are absent. For example, the expression: 5 + 2 - 4 is evaluated as if it were actually (5 + 2) - 4 because the operators + and - are left-to-right associative. Most operators in Java are left-to-right associative.
Operator Precedence Table:

<table>
<thead>
<tr>
<th>Level</th>
<th>Operator</th>
<th>Description</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>higher</td>
<td>( &lt;expression&gt; )</td>
<td>grouping with parentheses array index, method call, member access (dot operator)</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>[ ] ( ) .</td>
<td>post-increment, post-decrement pre-increment, unary plus/minus, logical negation</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>++ --</td>
<td>casting and creating object multiplication, division, modulus addition, subtraction, concatenation relational and Java’s instanceof operator</td>
<td>right to left</td>
</tr>
<tr>
<td></td>
<td>++ -- + --</td>
<td>equality conditional AND (short-circuits) conditional OR (short-circuits) ternary conditional assignment</td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>(type) new</td>
<td></td>
<td>right to left</td>
</tr>
<tr>
<td></td>
<td>* / %</td>
<td></td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>+ - +</td>
<td></td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>&lt;= =&gt;</td>
<td></td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>instanceof</td>
<td></td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>== !=</td>
<td></td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td>&amp;&amp;</td>
<td></td>
<td>left to right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>? :</td>
<td></td>
<td>right to left</td>
</tr>
<tr>
<td>lower</td>
<td>= += -= *= /= %=</td>
<td></td>
<td>right to left</td>
</tr>
</tbody>
</table>

a. int i;
   i = 4 + 3 * 2;
   i = 4 + (3 * 2);
   i = (4 + 3) * 2;

b. double shoeSize = 9.5;
   shoeSize = 10;

c. int mySize = 8.5;

d. int fraction = 4/8;

e. double fraction = 4 / 8;
   fraction = 4.0 / 8;
   fraction = (double) 4 / 8;

f. double answer = 5.0 / 0.0;
   int answerInt = 5 / 0;

g. final double PI = 3.14;
   PI = 3.1415926; //Does this work? Why or why not?

h. int num = 10 % 3;

i. int cents = 1097;
   int dollars = cents / 100;
   cents = cents % 100;
   int quarters = cents / 25;
cents %= 25;

j. int a = 1;
    int b = 2;
    int c = 3;
    int d = 4;
    a = b = c = d;

k. String str = "howdy";
    str = str + str;

    String str2 = 1 + 2 + str + 3 + 4;

l. String exprStr1 = 4 + 1 + " = 4 + 1";
    String exprStr2 = "4 + 1 = " + 4 + 1;
    String exprStr3 = "4 + 1 = " + (4 + 1);

m. int value = 36;
    double squareRoot = Math.sqrt( value);

n. double result = Math.pow(2, 32);

o. int lowest = Math.min(8, 9);
    lowest = Math.min(8 + 2, 9);
    lowest = Math.min(Math.min(5, 6), Math.min(19, 20));
    lowest = Math.min(lowest, value);

p. int result = Math.min(Math.max(1, Math.min(3, 5)),
                        Math.min(3, Math.min(2, 4)));

q. int x = -4;
    int abs = Math.abs(x);

r. answer = a + b * (c - d) * e;

s. int a = 1;
    System.out.println((a = 2) + a);
    int b = 1;
    System.out.println(b + (b = 2));

Exercise C: Change

Extend the following to also calculate dimes, nickels and pennies.

int cents = 1097;
int dollars = cents / 100;
cents = cents % 100;
int quarters = cents / 25;
cents %= 25;

Create a program that prompts the user for a starting amount and then calculates the number of dollars, quarters, dimes, nickels and pennies that sum to that amount.

Exercise D: Math documentation
Where can you find the documentation for all the methods available in the java.lang.Math class?

Exercise E: Surface Area of a Cube
These lines, when organized and indented correctly, calculates the Surface Area of a Cube. Can you fix them?

Exercise F: Example for Area of Triangle
Read through this exercise. The following exercises ask you to implement additional methods.

We want a program to calculate the area of a triangle. We found the following equation:

\[
\text{Area} = \frac{ba}{2}
\]

where

- \(b\) is the length of the base
- \(a\) is the length of the corresponding altitude

http://www.mathopenref.com/trianglearea.html

We implement this equation with the following Java statements:
double base; //length of base
double altitude; //length of altitude
double area = base * altitude / 2.0;

Then, we define a method for these statements:

```java
public static double areaOfTriangle( double base, double altitude) {
    //equation from: http://www.mathopenref.com/trianglearea.html
    double area = base * altitude / 2.0;
    return area;
}
```

Next, we call the method from another part of our program, such as the main method:

```java
public class Triangle {
    public static double areaOfTriangle(double base, double altitude) {
        //equation from: http://www.mathopenref.com/trianglearea.html
        double area = base * altitude / 2.0;
        return area;
    } 

    public static void main(String[] args) {
        double baseLength = 6.0;
        double altitudeHeight = 4.5;
        double triangleArea = areaOfTriangle(baseLength, altitudeHeight);
        System.out.println("Triangle with base " + baseLength
            + " and altitude " + altitudeHeight
            + " has area " + triangleArea);
    }
}
```

Finally, test the method.
How can you determine that it works for a variety of values?
What might the terms boundary case or corner case mean?
Would it be better to write code to test a fixed set of cases or write code to prompt the user?

**Exercise G: Celsius to Fahrenheit Temperature Conversion**

Find the formula, convert to Java, make a method and test the program.

**Exercise H: Area of Cylinder**

Find the formula, convert to Java, make a method and test the program.

**Exercise I: Volume of a Cone**

Find the formula, convert to Java, make a method and test the program.
Exercise J: Distance between Points - Logic Errors

The following program has some logic errors. Can you fix them?

```java
import java.util.Scanner;

public class DistanceBetweenPoints {
    public static void distanceBetweenPoints(double x1, double y1,
        double x2, double y2) {
        double d = Math.sqrt( (x2 - x1) + (y2 - y1));
        return;
    }

    public static void main(String [] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("This program finds the distance between "
            + "2 points (x1,y1) and (x2,y2).");
        System.out.print("Enter x1: ");
        int x1 = input.nextInt();
        System.out.print("Enter y1: ");
        int y1 = input.nextInt();
        System.out.print("Enter x2: ");
        int x2 = input.nextInt();
        System.out.print("Enter y2: ");
        int y2 = input.nextInt();

        double distance = 5.656854249492381;
        distanceBetweenPoints( x1, y1, x2, x2);

        System.out.println("Distance between points " + x1 + "," + y1
            + ") and (" + x2 + "," + y2 +") is " + distance);
        input.close();
    }
}
```

Exercise K: Pick an Equation

When you have tested your methods then review with a TA. Format your programs following the course Style Guide.

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 zyBooks being individual work, it is Not appropriate to work on them during the Team Lab.

Lab designed by Jim Williams with some problems from Sean McClanahan.