

## Chapter 2 Lab

### Java Fundamentals

#### Lab Objectives

- Write arithmetic expressions to accomplish a task
- Use casting to convert between primitive types
- Use a value-returning library method and a library constant
- Use `String` methods to manipulate string data
- Communicate with the user by using the `Scanner` class or dialog boxes
- Create a program from scratch by translating a pseudocode algorithm
- Be able to document a program

#### Introduction

This lab is designed to give you practice with some of the basics in Java. We will continue ideas from lab 1 by correcting logic errors while looking at mathematical formulas in Java. We will explore the difference between integer division and division on your calculator as well as reviewing the order of operations.

We will also learn how to use mathematical formulas that are preprogrammed in Java. On your calculator there are buttons to be able to do certain operations, such as raise a number to a power or use the number pi. Similarly, in Java, we will have programs that are available for our use that will also do these operations. Mathematical operations that can be performed with the touch of a button on a calculator are also available in the `Math` class. We will learn how to use a `Math` class method to cube the radius in the formula for finding the volume of a sphere.

This lab also introduces communicating with the user. We have already seen how console input and output work in lab 1. We will now need to learn how to program user input, by investigating the lines of code that we need to add in order to use the `Scanner` class. We will also learn the method call needed for output.

Alternately, you may use dialog boxes for communicating with the user. An introduction to graphical user interface (GUI) programming is explored using the `JOptionPane` class.

The `String` class is introduced and we will use some of the available methods to prepare you for string processing.

We will bring everything we have learned together by creating a program from an algorithm. Finally, you will document the program by adding comments. Comments are not read by the computer, they are for use by the programmer. They are to help a

programmer document what the program does and how it accomplishes it. This is very important when a programmer needs to modify code that is written by another person.

## Task #1 Correcting Logic Errors in Formulas

1. Copy the file *NumericTypes.java* (see Code Listing 2.1) from the Student CD or as directed by your instructor.
2. Compile the source file, run the program, and observe the output. Some of the output is **incorrect**. You need to **correct logic errors** in the average formula and the temperature conversion formula. The logic errors could be due to conversion between data types, order of operations, or formula problems. The necessary formulas are:

$$\text{average} = \frac{\text{score1} + \text{score2}}{\text{numberOfScores}} \qquad C = \frac{5}{9}(F - 32)$$

3. Each time you make changes to the program code, you must compile again for the changes to take effect before running the program again.
4. Make sure that the output makes sense before you continue. The average of 95 and 100 should be 97.5 and the temperature that water boils is 100 degrees Celsius.

## Task #2 Using the Scanner Class for User Input

1. Add an `import` statement above the class declaration to make the `Scanner` class available to your program.
2. In the `main` method, create a `Scanner` object and connect it to the `System.in` object.
3. Prompt the user to enter his or her first name.
4. Read the name from the keyboard using the `nextLine` method, and store it into a variable called `firstName` (you will need to declare any variables you use).
5. Prompt the user to enter his or her last name.
6. Read the name from the keyboard and store it in a variable called `lastName`.
7. Concatenate the `firstName` and `lastName` with a space between them and store the result in a variable called `fullName`.
8. Print out the `fullName`.
9. Compile, debug, and run, using your name as test data.
10. Since we are adding on to the same program, each time we run the program we will get the output from the previous tasks before the output of the current task.

## Task #2 (Alternate) Using Dialog Boxes for User Input

1. Add an `import` statement above the class declaration to make the `JOptionPane` class available to your program.

2. In the `main` method, prompt the user to enter his or her first name by displaying an input dialog box and storing the user input in a variable called `firstName` (you will need to declare any variables you use).
3. Prompt the user to enter his or her last name by displaying an input dialog box and storing the user input in a variable called `lastName`.
4. Concatenate the `firstName` and `lastName` with a space between them and store the result in a variable called `fullName`.
5. Display the `fullName` using a message dialog box.
6. Compile, debug, and run, using your name as test data.
7. Since we are adding on to the same program, each time we run the program we will get the output from the previous tasks before the output of the current task.

### Task #3 Working with Strings

1. Use the `charAt` method to get the first character in `firstName` and store it in a variable called `firstInitial` (you will need to declare any variables that you use).
2. Print out the user's first initial.
3. Use the `toUpperCase` method to change the `fullName` to uppercase and store it back into the `fullName` variable.
4. Add a line that prints out the value of `fullName` and how many characters (including the space) are in the string stored in `fullName` (use the `length` method to obtain that information).
5. Compile, debug, and run. The new output added on after the output from the previous tasks should have your initials and your full name in uppercase characters.

### Task #4 Using Predefined Math Functions

1. Add a line that prompts the user to enter the diameter of a sphere.
2. Read in and store the number into a variable called `diameter` (you will need to declare any variables that you use).
3. The diameter is twice as long as the radius, so calculate and store the radius in an appropriately named variable.
4. The formula for the volume of a sphere is:

$$V = \frac{4}{3} \pi r^3$$

Convert the formula to Java code and add a line which calculates and stores the value of volume in an appropriately named variable. Use `Math.PI` for  $\pi$  and `Math.pow` to cube the radius.

5. Print your results to the screen with an appropriate message.
6. Compile, debug, and run using the following test data and record the results.

Diameter	Volume (hand calculated)	Volume (resulting output)
2		
25.4		
875,000		

## Task #5 – Create a program from scratch

In this task the student will create a new program that calculates gas mileage in miles per gallon. The student will use string expressions, assignment statements, input and output statements to communicate with the user.

1. Create a new file in your IDE or text editor.
2. Create the shell for your first program by entering:  

```
public class Mileage
{
    public static void main(String[] args)
    {
        // Add your declaration and code here.
    }
}
```
3. Save the file as *Mileage.java*.
4. Translate the algorithm below into Java code. Don't forget to declare variables before they are used. Each variable must be one word only (no spaces).  
*Print a line indicating this program will calculate mileage*  
*Print prompt to user asking for miles driven*  
*Read in miles driven*  
*Print prompt to user asking for gallons used*  
*Read in gallons used*  
*Calculate miles per gallon by dividing miles driven by gallons used*  
*Print miles per gallon along with appropriate labels*
5. Compile the program and debug, repeating until it compiles successfully.
6. Run the program and test it using the following sets of data and record the results:

Miles driven	Gallons used	Miles per gallon (hand calculated)	Miles per gallon (resulting output)
2000	100		
500	25.5		
241.5	10		
100	0		

7. The last set of data caused the computer to divide 100 by 0, which resulted in what is called a **runtime error**. Notice that runtime can occur on programs which compile and run on many other sets of data. This emphasizes the need to thoroughly test your program with all possible kinds of data.

## Task #4 Documenting a Java Program

1. Compare the code listings of `NumericTypes.java` with `Mileage.java`. You will see that `NumericTypes.java` has lines which have information about what the program is doing. These lines are called comments and are designated by the `//` at the beginning of the line. Any comment that starts with `/**` and ends with `*/` is considered a documentation comment. These are typically written just before a class header, giving a brief description of the class. They are also used for documenting methods in the same way.
2. Write a documentation comment at the top of the program which indicates the purpose of the program, your name, and today's date.
3. Add comment lines after each variable declaration, indicating what each variable represents.
4. Add comment lines for each section of the program, indicating what is done in that section.
5. Finally, add a comment line indicating the purpose of the calculation.

### Code Listing 2.1 (NumericTypes.java)

```
// TASK #2 Add an import statement for the Scanner class
// TASK #2(Alternate)
// Add an import statement for the JOptionPane class

/**
    This program demonstrates how numeric types and
    operators behave in Java.
 */

public class NumericTypes
{
    public static void main (String [] args)
    {
        // TASK #2 Create a Scanner object here
        // (not used for alternate)

        // Identifier declarations
        final int NUMBER = 2 ;           // Number of scores
        final int SCORE1 = 100;          // First test score
        final int SCORE2 = 95;           // Second test score
        final int BOILING_IN_F = 212;    // Boiling temperature
        int fToC;                        // Temperature Celsius
        double average;                  // Arithmetic average
        String output;                   // Line of output
    }
}
```

```

// TASK #2 declare variables used here
// TASK #3 declare variables used here
// TASK #4 declare variables used here

// Find an arithmetic average.
average = SCORE1 + SCORE2 / NUMBER;
output = SCORE1 + " and " + SCORE2 +
        " have an average of " + average;
System.out.println(output);

// Convert Fahrenheit temperature to Celsius.
fToC = 5/9 * (BOILING_IN_F - 32);
output = BOILING_IN_F + " in Fahrenheit is " +
        fToC + " in Celsius.";
System.out.println(output);
System.out.println();          // To leave a blank line

// ADD LINES FOR TASK #2 HERE
// Prompt the user for first name
// Read the user's first name
// Prompt the user for last name
// Read the user's last name
// Concatenate the user's first and last names
// Print out the user's full name

System.out.println();          // To leave a blank line

// ADD LINES FOR TASK #3 HERE
// Get the first character from the user's first name
// Print out the user's first initial
// Convert the user's full name to uppercase
// Print out the user's full name in uppercase

System.out.println();          // To leave a blank line

// ADD LINES FOR TASK #4 HERE
// Prompt the user for a diameter of a sphere
// Read the diameter
// Calculate the radius
// Calculate the volume
// Print out the volume
}
}

```