

Chapter 2 Lab

Java Fundamentals

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 create three short programs: 1. by converting logic errors while looking at mathematical formulas in Java. We will explain the differences 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 take a number to a power or use the number pi. Similarly, in Java, we will learn programming that are available for use that will also do these operations. Mathematical operations that can be performed with the built-in `System` or a calculator are also available in the `Math` class. We will learn how to use a `Math` class method to take 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 learn to learn how to program user input, by manipulating the flow of code that we want 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 see some of the available methods to prepare you for string processing.

10. Lab Manual is Accompanying Student File with Java 5-Object Oriented Features in Objects

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 modified 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.

11. Lab Manual is Accompanying Student File with Java 5-Object Oriented Features in Objects

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 modified 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. Download the file `NumericalTypes.java` (see code listing 1.1) from www.cs.cornell.edu/~cs212 or as directed by your instructor.
2. Compile the source file, run the program, and observe the output. Some of the output is **incorrect**. We 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:
$$\text{average} = \frac{\text{temp1} + \text{temp2}}{\text{numberofItems}} \quad C = \frac{5}{9}(F - 32)$$
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.

12. Lab Manual is Accompanying Student File with Java 5-Object Oriented Features in Objects

Task #2: Using the Scanner Class for User Input

1. Add an input 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 as input.
3. Prompt the user to enter his/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/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.

13. Lab Manual is Accompanying Student File with Java 5-Object Oriented Features in Objects

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

1. Add an input 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/her first name by displaying an input dialog box and storing the user input into a variable called `firstName` (you will need to declare any variables you use).
3. Prompt the user to enter his/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.

14. Lab Manual is Accompanying Student File with Java 5-Object Oriented Features in Objects

Task #3: Working with Strings

1. Use the `charAt` method to get the first character in `fullName` and store it in a variable called `firstCharacter`. (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 all capitals 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 method `length` to obtain that information).
5. Compile, debug, and run. The new output addition after the output from the previous tasks should have your initials and your full name in all capital letters.

Diameter	Volume (hard calculation)	Volume (rounding output)
2		
20.2		
875.000		

15. Lab Manual is Accompanying Student File with Java 5-Object Oriented Features in Objects

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 using 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 MPerGallon {  
    public static void main(String[] args) {  
        // add your declarations and code here  
    }  
}
```
3. Save the file as `MPerGallon.java`.
4. Translate the algorithm below into Java. Don't forget to declare variables before they are used. Each variable must be one word only (no spaces).
Read a line indicating this program will calculate mileage.
Read in miles driven.
Read in miles per gallon.
Read in gallons used.
Calculate miles per gallon by dividing miles driven by gallons used.
Print miles per gallon using with appropriate labels.
5. Compile the program and testing, repeating until it completes 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 (hard calculation)	Miles per gallon (rounding output)
1000	100		
500	20.5		
201.2	10		
100	8		

7. The last set of data caused the computer to divide 1000 by 8, which resulted in what is called a floating error. Notice that number cut down on program, which compile and run on many other sets of data. This emphasizes the need to thoroughly test your programs with all possible kinds of data.

16. Lab Manual is Accompanying Student File with Java 5-Object Oriented Features in Objects

Task #6: Documenting a Java Program

1. Compare the code listings of `NumericalTypes.java` with `MPerGallon.java`. You will see that `NumericalTypes.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 calculations.