CIS 255 • Java Programming
Laboratory Assignment 2: Variables, Operators, and Input

In this laboratory assignment you will work with Java variables, arithmetic operators, and console input to allow the user to determine the best way to divide a group of soccer players among a number of teams.

Example Run (assume the user enters the values at the ends of the first two lines):

```
How many players are available in this soccer league?  25
How many teams are there in this league?  4
If you divide 25 players among 4 teams,
you will have 6 players per team with 1 left over.
```

Another Example Run:

```
How many players are available in this soccer league?  30
How many teams are there in this league?  5
If you divide 30 players among 5 teams,
you will have 6 players per team with 0 left over.
```

Step 1: Configure Your Computer

If you want to be able to compile and run your Java programs on your own computer, you'll need to install the Java Development Kit (JDK). Additionally, you'll probably want to use a text editor that highlights Java syntax; TextPad is one such program. Refer to the configuration instructions for details of what needs to be done.

Step 2: Launch TextPad and Use Java Highlighting

Launch TextPad from the task bar, the Start Menu, or the CIS folder on the Desktop. Immediately select **Save As…** from the **File** menu to save the file as a Java file and receive the benefit of syntax highlighting.

In the **Save As** dialog, navigate to the location where you want your code to be saved (use your flash drive if you have it with you), enter the file name **FLast_Lab02.java** (where **F** is your first initial and **Last** is your last name), and select **Java (*.java)** in the **Save as type:** combo box.

Step 3: Begin Your Code

The first element you should provide in your code is a set of comments identifying yourself and the purpose of your program. You may enter this as a single block comment or as a set of line comments. (See section 2.11 in your textbook, or the outline from Lecture 2, for more details on the syntax of comments.)

Comments for the code you submit for this class should include your name, the course number, the laboratory assignment or project number, and a brief description of the program's contents. You may also wish to include the date (perhaps after the laboratory assignment number). Place each of these items on a separate line, either as individual line comments or in a block comment:

```
// Student Name
// CIS 255 (Semester and Year)
// Laboratory Assignment 2
// Current Date
// In this assignment, I am practicing basic Java statements such as
// declaring variables and constants, console input and output, and
// assignment using arithmetic operators.
```

As you continue your code, you may go ahead and add comments to the statements or groups of statements that you believe deserve additional explanation. You may also wait until after you have written your entire program to add comments. On each assignment you submit, you will be required to provide a minimum number of comments (usually listed at the end of the assignment description).

It's a good idea to go ahead and set up the basic structure of your code as well, including `import` statements, the `class` header and brackets, and the `main` method header and brackets. Since the program will obtain input from the user via the console, it needs the `import` statement for the `Scanner` class:

```
import java.util.Scanner;
```

After the `import` statement (and a blank line for readability), add the `class` header:

```
public class FLast_Lab02
```

Remember that the class name must match the name of your Java source code file (use your first initial and last name). Enter the opening curly bracket for the class on the next line.

Next, you need to enter the header for the `main` method:

```
public static void main(String[] args)
```

Enter another opening curly bracket for the `main` method. Leave a blank line for the space where you'll enter the statements for your program; then, type the closing curly bracket for the `main` method and the closing curly bracket for the `class`.

Keep the indentation in your program consistent. The opening and closing curly brackets for the `class` are usually flush with the left side of the document window; the curly brackets for the `main` method are indented one level, usually with a tab; and the contents of `main` are indented one level further.

Step 4: Declare Your Variables

In this program, you will deal with four numeric quantities: the number of players available for teams in a soccer league, the number of teams into which to divide the players, the number of players to be placed on each team, and the number of players left over. Each of these quantities is a whole number, so the data type `int` is most appropriate.

Return to the blank line between the inner set of curly brackets for the `main` method. Insert **declarations** for all four variables in the `main` method. You may declare these individually (one variable per statement) or all at once (all four variables in a single declaration statement). Make sure that you use appropriate punctuation.

Don't put a space in the middle of a variable name. Use an underscore (_) or camelCase to include multiple words in a variable name (e.g., `my_variable`, `myVariable`).

Step 5: Declare a `Scanner` Object

The user will be allowed to enter the number of players available and the number of teams desired in the console. In order to obtain this input using simple syntax, you should declare an object of the class `Scanner`. Below the declarations for the `int` variables, add a **declaration** for a `Scanner` object based on `System.in`:

```
Scanner keyboard = new Scanner(System.in);
```

Although the textbook uses the name `keyboard` for each `Scanner` object declared for console input, any legal identifier can be used for this object.

Step 6: Obtain Input from the User

The program uses a `Scanner` method to read each value from the user; however, it should display a prompt to let the user know what value is expected for each input statement. Write an **output** statement that asks the user for the total number of players available (e.g., `"How many players are available in this soccer league?"`). Follow this with an **input** statement that reads the next available `int` value using the `Scanner` object and assigns this value to the appropriate variable. Repeat these two statements (**output** and **input**) for the number of teams.

Step 7: Perform the Calculations

There are two operators that you'll need to use in determining the results. The number of players per team is the number of players available divided by the number of teams (the division operator). The number of players left over is the remainder when performing this division (the modulus operator). Add two **assignment** statements to perform the arithmetic and store these quantities in their respective variables.

Don't forget that arithmetic involving integers always produces an integer result! In this assignment, you want the number of players per team to be a whole number, but in other programs, you may need to use at least one floating-point value to ensure that the quotient (the value provided by the division operator) is not truncated.

Step 8: Display the Results

Without output statements, the user is unaware of anything happening in our program. Use a set of **output** statements to display the results in the following format (words in italics represent variables):

`If you divide` *number of players* `players among` *number of teams* `teams,`
`you will have` *players per team* `players per team with` *players left over* `left over.`

Don't put variable names in quotation marks!

Use plus signs, not commas, between the items in an output statement. Also, don't forget to include spaces inside the quotation marks before and after variables.

Step 9: Compile Your Code

To determine if there are any syntax errors in your source code, and to generate the byte code once your source code is free of syntax errors, select the item **Compile Java** from the **External Tools** submenu of the **Tools** menu. The results of the build process will appear in the **Tool Output** area below your code. If the Tool Output area indicates that there are syntax errors in your code, you may double-click the line that describes each syntax error to see the line that contains the error (or, in some cases, the line after the error).

Don't try to correct all of your syntax errors at once. Correcting one error listed in the Tool Output area and compiling your code again may resolve or reveal additional errors.

Step 10: Execute the Byte Code

Once you receive a message in the Tool Output area that the tool completed successfully, you're ready to test your program. In the **External Tools** submenu of the **Tools** menu, select **Run Java Application**. Observe the values that appear and verify that they match what you expect from the calculations.

Step 11: Add Inline Comments and Submit Your Code

Make sure that your code has **three or more inline comments** near the statements that you believe warrant the additional description. These can be simple line comments at the end of a line of code that needs explanation, or block comments where each precedes a series of statements described in the comment.

You are not required to attach your compiled byte code (the file that ends in `.class`) for any assignment. Only your source code file (the file that ends in `.java`) is required.

Follow the instructions on the course website to attach and submit your source code.

Make sure you complete the submission process! If you attach your file without submitting the assignment, your file will not be received!