Objective After completing this activity you should:

- Know how to declare and initialize `String` and `int` variables
- Be able to choose appropriate data types for data in the program specifications
- Know how to use `print` or `println`, together with escape characters

Activities

In this part of the lab you will write a Java program according to the specifications below.

1. Create a new BlueJ project named `Lab1`.
2. Click on the `Lab1 Tests` link on AsULearn. Download the zip file to your desktop and unzip it. Do not put these test files into the BlueJ project until told to do so by this document.
3. Create a class named `Address`. Delete everything in this class except for the class declaration and the open and closed braces associated with the `Address` class. Your file should look like this.

```
public class Address
{

}
```

4. Create a `main` method in the class `Address`. Make sure `main` is specified properly, as

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

}
```

5. In `main`, create the following variables, using integers whenever you are able:
   - Create a variable named `number` and initialize it to `224`.
   - Create a variable named `street` and initialize it to `"Joyce Lawrence Lane"`.
   - Create a variable named `city` and initialize it to `"Boone"`.
   - Create a variable named `state` and initialize it to `"NC"`.
   - Create a variable named `zip` to store the zip code and initialize it to `28608`.
6. Print the address exactly as follows:

```
224 Joyce Lawrence Lane
Boone, NC 28608
```

   For full credit
   - you *must* use the variables you created above.
   - you *must* use a single `println` statement.
7. Complete the following tasks, changing only the values of the variables you created above. If you used variables in your `println` statements properly, your calls to `println` should not need to be changed.

- Change the initialization of `number` to 123.
- Change the initialization of street to "Main Street".
- Change the initialization of `city` to "Hickory".
- Change the initialization of `zip` to 28601.

8. When you run `main` this time you should see exactly the following:

```
123 Main Street
Hickory, NC 28601
```

9. If you do not see *exactly* the above, correct your `println` statements so that changing *only* the values of the variables is necessary to switch back and forth between the two output examples given above.

10. Copy the `TestAddress.java` class into the `Lab1` project directory.

11. Import the project directory (`Project` → `Import`) or restart BlueJ so that you see `TestAddress` in the BlueJ workspace.

12. Compile `TestAddress`. If you get an error, it is in your `Address` class even if BlueJ says it is in `TestAddress`. The error is either because `TestAddress` cannot find your `Address` class, or because it cannot find a properly defined `main` method inside your `Address` class.

13. Right-click on `TestAddress` and select `Test All`. Make sure you get a green check before continuing. Your grade to this point should appear at the bottom of the terminal window.

    YOU MUST PASS THIS TEST BEFORE CONTINUING. THE TESTS ARE DESIGNED SO THAT LATER ACTIVITIES WILL NOT BE GRADED UNTIL ALL OF THE PREVIOUS ACTIVITIES ARE COMPLETED WITH A GREEN CHECK MARK.

14. Convert the project on your Desktop to a compressed (zipped) folder. In order for us to grade your program, *YOUR PROGRAM MUST BE IN* `zip` *FORMAT*. Note carefully that if you use *any* other format (for example, `RAR` or `7z`) you will earn a grade of 0 for this lab.

    Here's how to compress (zip) your file:

    - In Windows:
      - Right-click on the `Lab1` BlueJ project folder on your desktop. Hover your mouse over `Send to`. A submenu should appear. Click the menu item labeled `Compressed (zipped) folder`.
      - You should now see a `Lab1.zip` file on your desktop.
    - To compress on a Mac, just choose `Compress` from the menu that appears from right-clicking on the `Lab1` project folder.

15. Zip your `Lab1` project and upload on AsULearn as described above. This will overwrite your last upload. In order for us to grade your program, *YOUR PROGRAM MUST BE IN* `zip` *FORMAT*. Note carefully that if you use *any* other format (for example, `RAR` or `7z`) you will earn a grade of 0 for this lab.

16. Make sure you *always* submit the test files with your project. If a test file is not present it will be assumed that portion of the lab was not tested because it does not function properly. In this case you will earn a grade of 0 for that portion of the lab.

17. *NEVER UPLOAD A PROJECT THAT CONTAINS A CLASS THAT DOES NOT COMPILE. YOU WILL EARN A GRADE OF 0 FOR THE ENTIRE LAB IF YOU DO.*

Objective After completing this activity you should know how to:

- Use `String` methods
- Extract a character from a `String` and store it

Activities

In this part of the lab you will write a new Java program.

1. In your `Lab1` project create a class named `FunWithNames`. Make sure the `'F'`, `'W'`, and `'N'` are all capitalized. Delete any extra text inside the body of the class.
2. Create a `main` method.
3. Create the following variables in `main`:
   - A variable named `firstName`, initialized to "John".
   - A variable named `middleName`, initialized to "Clarence".
   - A variable named `lastName`, initialized to "Doe".
4. Below the variables you created in the previous step, create the following variables whose values will *use* the values of the above variables:
   - A variable named `fullName` that concatenates `firstName`, `middleName`, and `lastName` (using the `+` operator) with a single space between each of the parts of the full name.
   - A variable called `characterCount` that calculates the number of characters in a full name, not counting spaces. Do this by first counting the characters in `firstName`, then counting the characters in `middleName`, then counting the characters in `lastName,` and then combining these counts appropriately. *Do not* just count the characters in `fullName`. Remember that strings have a `length` method.
   - A variable called `login` that creates an all lowercase login ID from a full name. To do this, extract the first character of `firstName` and the first character of `middleName`, and concatenate them onto the end of `lastName`. Remember that strings have `charAt` and `toLowerCase` methods.
5. Display the following output using the the `fullName`, `characterCount`, and `login` variables. Use a single `println` statement for each line of output.

```
Name: John Clarence Doe
Number of characters in full name: 15
Login id: doejc
```

6. Complete the following tasks, changing only the values of the variables you created above. If you used your variables properly, neither your calls to `println` nor the lines where you computed values for `fullname`, `characterCount`, and `login` should not need to be changed.
   - Change `firstName` so it is initialized to "Donald".
   - Change `middleName` so it is initialized to "Richard".
   - Change `lastName` so it is initialized to "Dirka".
7. When you run `main` this time you should see exactly the following:

```
Name: Donald Richard Dirka
Number of characters in full name: 18
Login id: dirkadr
```

8. If you do not see *exactly* the above, correct your `println` statements so that changing *only* the values of the variables is necessary to switch back and forth between the two output examples given above.

9. Copy the `TestName.java` class into the `Lab1` project directory.

10. Right-click on `TestName` and select `Test All`. Make sure you get a green check before continuing. Your grade to this point should appear at the bottom of the terminal window.

    YOU MUST PASS THIS TEST BEFORE CONTINUING. THE TESTS ARE DESIGNED SO THAT LATER ACTIVITIES WILL NOT BE GRADED UNTIL ALL OF THE PREVIOUS ACTIVITIES ARE COMPLETED WITH A GREEN CHECK MARK.

11. Zip your `Lab1` project and upload on AsULearn as described previously. This will overwrite your last upload. In order for us to grade your program, *YOUR PROGRAM MUST BE IN* `zip` *FORMAT*. Note carefully that if you use *any* other format (for example, `RAR` or `7z`) you will earn a grade of 0 for this lab.

12. Make sure you *always* submit the test files with your project. If a test file is not present it will be assumed that portion of the lab was not tested because it does not function properly. In this case you will earn a grade of 0 for that portion of the lab.

13. *NEVER UPLOAD A PROJECT THAT CONTAINS A CLASS THAT DOES NOT COMPILE. YOU WILL EARN A GRADE OF 0 FOR THE ENTIRE LAB IF YOU DO.*

Objective After completing this activity you should know how to:

- Declare, initialize, and use `double` variables
- Perform computations with `doubles`
- Display `doubles`
- Use tabs in a `println` statement

Activities

In this part of the lab you will write a new Java program.

1. In your `Lab1` project create a class named `Sales` and remove any extra text.
2. Create a `main` method.
3. Create a variable named `purchaseAmount` and initialize it to `32.0`.
4. Below `purchaseAmount`, create variables named `stateTax`, `countyTax`, `totalTax`, and `totalPrice`. What type should these variables be, given that they all deal with monetary amounts?
5. Calculate the state sales tax assuming a tax rate of 5% and store that value in the appropriate variable.
6. Calculate the county sales tax assuming a tax rate of 3%, and store the resulting value in the appropriate variable.
7. Calculate the total tax paid on the purchase and store the resulting value in the appropriate variable.
8. Calculate the total amount paid for the item including all taxes and store the resulting value in the appropriate variable.
9. Display the data as shown below:

```
Amount of Purchase:        $32.0
State Sales Tax Paid:      $1.6
County Sales Tax Paid:     $0.96
Total Sales Tax Paid:      $2.56
Total Sales Price:         $34.56
```

   Note that the dollar amounts are neatly aligned in a second column (use `'\t'`), and that they are printed with leading dollar signs. Also, your answers may have one, two, or more digits to the right of the decimal point.

10. If you change `purchaseAmount` to `55.0` (changing *only* the value of `purchaseAmount`) and rerun `main`, you should see the following output instead:

```
Amount of Purchase:        $55.0
State Sales Tax Paid:      $2.75
County Sales Tax Paid:     $1.65
Total Sales Tax Paid:      $4.4
Total Sales Price:         $59.4
```

11. Make sure you get the proper output by changing only the value of `purchaseAmount` before continuing.

12. Copy the file `Given.java` from the `Lab1Tests` folder into your project directory.

13. You should have a line in your `Sales` class that looks like the one below. (You may have a 32.0 instead of 55.0 if you changed it back to its original value).

```
double purchaseAmount = 55.0;
```

    Or you may have done it in two lines like this:

```
double purchaseAmount;
purchaseAmount = 55.0;
```

    Delete that line or those two lines. Make sure that you have absolutely no lines anywhere in `main` that assign a value to `purchaseAmount`.

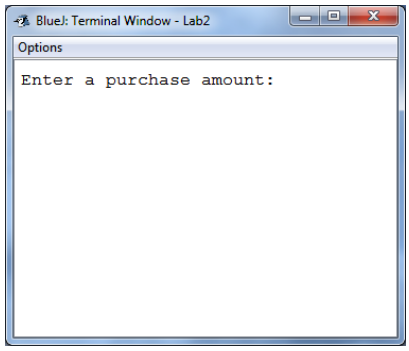
14. As the very first thing in `main`, copy and paste the following two lines:

```
System.out.println("Enter a purchase amount: ");
double purchaseAmount = Given.getDouble();
```

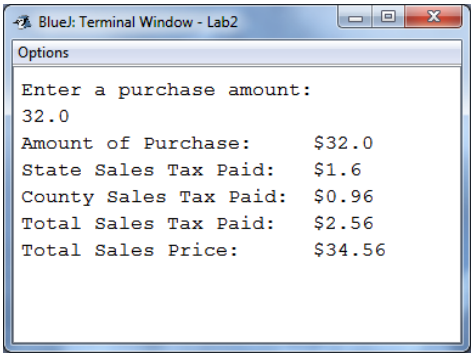
15. Make sure your terminal window is open in BlueJ. Clear all text from the window.

16. Run `main` from the `Sales` class.

17. The terminal window should look like the image below:



18. Enter `32.0` and press enter. Your terminal window should display the output from as shown in the image below:



19. Run main again and enter `55.0`. You should see the data shown. Try some different numbers. Use a calculator to verify the accuracy of the output. Some of the output numbers have a bunch of digits to the right of the decimal. That is OK for now. You will learn how format numbers later.

20. Copy the `TestSales.java` class into the `Lab1` project directory.

21. Right-click on `TestSales` and select `Test All`. Make sure you get a green check before continuing. Your grade to this point should appear at the bottom of the terminal window.

    YOU MUST PASS THIS TEST BEFORE CONTINUING. THE TESTS ARE DESIGNED SO THAT LATER ACTIVITIES WILL NOT BE GRADED UNTIL ALL OF THE PREVIOUS ACTIVITIES ARE COMPLETED WITH A GREEN CHECK MARK.

22. Zip your `Lab1` project and upload on AsULearn as described previously. This will overwrite your last upload. In order for us to grade your program, *YOUR PROGRAM MUST BE IN* `zip` *FORMAT*. Note carefully that if you use *any* other format (for example, `RAR` or `7z`) you will earn a grade of 0 for this lab.

23. Make sure you ALWAYS submit the test files with your project. If a test file is not present it will be assumed that portion of the lab was not tested because it does not function properly. In this case you will earn a grade of 0 for that portion of the lab.

24. *NEVER UPLOAD A PROJECT THAT CONTAINS A CLASS THAT DOES NOT COMPILE. YOU WILL EARN A GRADE OF 0 FOR THE ENTIRE LAB IF YOU DO.*

Objective After completing this activity you should know how to:

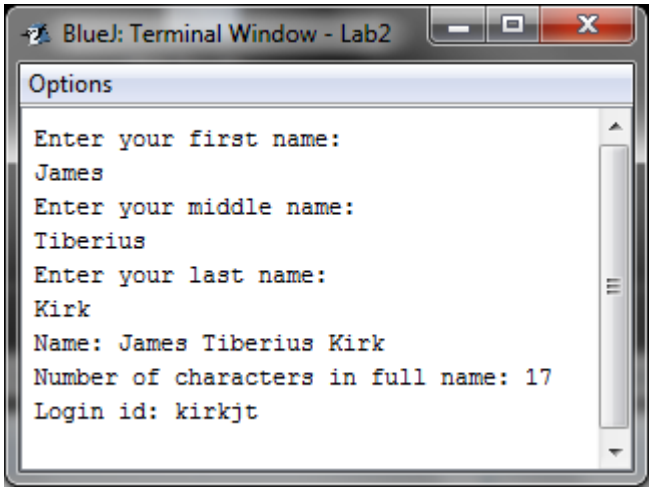
- Declare, initialize, and use double variables
- Perform computations with doubles
- Display doubles
- Use tabs in a `println` statement

Activities

1. Create a new class in `Lab1` named `FunWithNamesV2`, clear out unneeded text, and create a `main` method.
2. Copy all of the code from `main` in `FunWithNames` and paste it into the main in `FunWithNamesV2`.
3. You can read strings from the terminal like this:

```
System.out.println("Enter your first name: ");
String firstName = Given.getString();
```

4. Change `main` in `FunWithNamesV2` so that first, middle, and last names are read from the terminal as shown in the previous step.
5. Run `main`. If you enter `James Tiberius Kirk` you should see exactly this:



6. Test your program with `TestNamesV2.java`. Note that you won't see the actual input values in your terminal window when your program is run from the test class.
7. Zip your `Lab1` project and upload on AsULearn as described previously. This will overwrite your last upload. In order for us to grade your program, *YOUR PROGRAM MUST BE IN* `zip` *FORMAT*. Note carefully that if you use *any* other format (for example, `RAR` or `7z`) you will earn a grade of 0 for this lab.
8. Make sure you *always* submit the test files with your project. If a test file is not present it will be assumed that portion of the lab was not tested because it does not function properly. In this case you will earn a grade of 0 for that portion of the lab.

9. *NEVER UPLOAD A PROJECT THAT CONTAINS A CLASS THAT DOES NOT COMPILE. YOU WILL EARN A GRADE OF 0 FOR THE ENTIRE LAB IF YOU DO.*

Grading:

- Activities: 40 points (6 points for `Address`, 6 points for `Name`, 15 points for `Sales`, 15 points for `NameV2`), graded as reported by the test cases. Your activity grade will be 0 if the class does not compile with its associated test class.
- Formatting: 8 points if your code is indented properly.

Make sure you *always* submit the test files with your project. If a test file is not present it will be assumed that portion of the lab was not tested because it does not function properly. You will earn a grad of 0 for that portion of the lab.

Do not upload a project that contains a class that does not compile. You will earn a grade of 0 if your entire project does not compile. Remove any classes that do not compile before you upload your project to AsULearn for grading.