Lab 11
BlueJ, Classes and jUnit Revisited I

Fall 2017

1 Introduction

Grading: This lab requires the use of the grading sheet for responses that must be checked by your instructor (marked as Question) AND the submission of some programs to WebCAT (marked as Lab).

In an earlier lab, you learned how to debug a class, specifically a `PersonName` class, and how use jUnit to test the class.

In this lab you will extend your understanding of classes and objects by

- building a new `PersonName` class (you will not reuse the old one),
- incorporating the `PersonName` class into a `Person` class (next week),
- doing some testing with jUnit, and
- investigating what `static` means.

2 Exercises

Fork and clone the `cpsc150l-lab11` repository from the 150 students group on Gitlab.

2.1 `PersonName`

Create a class named `PersonName` with the private fields

- `int numberOfNames`,
- `String firstName`,
- `String middleName`,
- `String lastName`,
- `String suffixName`.

Implement the following methods.

1. `public PersonName()`

   Initializes each name to `""` and sets the number of names to zero.

2. `public PersonName(String wholeName)`

   Counts the number of names in `wholeName` and stores it in `numberOfNames` and sets each name appropriately.

   - If there is 1 name, then there is only a first name in `wholeName`.
   - If there are 2 names, then there is a first and last name in `wholeName` in that order.
   - If there are 3 names, then there is a first, middle, and last name in `wholeName` in that order.
   - If there are 4 names, then there is a first, middle, last name, and suffix in `wholeName` in that order.

   You may assume that `wholeName` will never have more than 4 names and that `wholeName` has no extraneous white space anywhere in it. That is, `wholeName` will have one of the following formats exactly.

- `first`
- `first last`
- `first middle last`
- `first middle last suffix`

Hint: Use `String.split(" ")` to get an array of names.

3. Getter and setter methods for each of the names above. Each setter should appropriately adjust the `numberOfNames` field each time a name is modified. That is, if the name equals `""` before setting, increment `numberOfNames` and if the name equals `""` after setting, decrement `numberOfNames`. For example, consider what should happen if we call `setFirstName("")` when `firstName` equals `""`. Additionally, suppose the first name doesn't exist. If we call `setMiddleName("Bob")`, then this should set the middle name to `"Bob"`, adjust the number of names appropriately, and leave the first name as empty.

   (a) `public void setFirstName(String name)`

   (b) `public String getFirstName()`

   (c) `public void setMiddleName(String name)`

   (d) `public String getMiddleName()`

   (e) `public void setLastName(String name)`

   (f) `public String getLastName()`

   (g) `public void setSuffixName(String name)`

   (h) `public String getSuffixName()`

4. `getNumberOfNames()`

   Returns the number of names in this `PersonName` object.

5. `public String getEntireName()` This method returns

   Code:

   ```
   firstName + " " + middleName + " " + lastName + " " + suffixName
   ```

with appropriate spacing (one space between each name ex: if `middleName` is `""`, then it returns `firstName+" "+lastName+" "+suffixName`).

Hint: Use `String.trim()` liberally. We provide a tutorial on `String.trim()` in Section 4.2.

6. `public String getInitials()`

   This returns `firstInitial + middleInitial + lastInitial` if each respective name exists. For example if there is no middle name, then it should return the first and last initials.

Exercise 1 Complete

Run:

```
git add .
git commit -m "Completed exercise 1"
git push origin master
```

2.2 Testing the Person Class with jUnit

Create a jUnit test class which minimally creates three different people, one under 16, one equal to 16, and one over 16. This type of testing does boundary condition testing as it investigates how your program functions as the data transitions from one state to another; here from being too young to drive to being eligible to drive.

Question 1: *When you have finished coding the jUnit class for testing* `Person`*, show your instructor.*

Exercise 2 Complete

Run:

```
git add .
git commit -m "Completed exercise 2"
git push origin master
```

---

3 Common Mistakes

The solutions to some common mistakes are as follows.

1. Pay close attention to when you need to increment and decrement the name count in your setter methods.
2. Pay attention to spacing in the `getEntireName` method. Specifically, ensure that there is no extraneous whitespace in your output. That is if the first name is `"Matt"`, the middle name is empty, and the last name is `"Smith"`, the output should be `"Matt Smith"`, not `"Matt  Smith"`.
3. When writing jUnit tests, be sure to put the `@Test` annotation before each test declaration.
4. Do **not** delete the original `canDrive` method when you overload it in the last exercise.

4 Tutorial

4.1 Using `String.split(" ")`

Look up the JavaDoc for the `String` class and look at the `String[] split(String regex)` method. We note that the method processes a `String` object based on a rule called a regular expression and returns an array of `String` objects. Furthermore, if we pass `" "` to `String.split` it will create a new array of each word in the original `String` separated by at least one space. An example of its use is as follows.

Code:

```
String yolo = "You only live once";
String[] words = yolo.split(" ");
for (int i = 0 ; i < words.length ; i++)
     System.out.println(words[i]);
```

The previous example will output the following.

Code:

```
You
only
live
once
```

4.2 Using `String.trim()`

Consider the `String`, `"    you only   live  once "`. We see that it has far more whitespace than necessary. One way to get rid of some of this whitespace is to use `String.trim()`. The `String.trim()` method removes all whitespace from the beginning and end of a `String` object. Consider the following code.

Code:

```
String s = "    you only   live  once ".trim();
```

Since `trim()` removes whitespace from the beginning and end of a `String`, `s` is equal to `"you only   live  once"`. Note how `trim()` did not touch the extra whitespace inside of the string.