COM1300 Lab #2

To complete this lab it may help to review Chapter 2 of your text, as well as the BlueJ tutorial on Angel. Please work on your lab results individually, but collaborate to resolve operational difficulties with BlueJ or Codingbat.

Objectives of this Lab:

- Writing constructors and methods
- Calling object methods from the BlueJ menu
- Evaluating Java expressions and debugging using Code Pad
- Printing to the Terminal Window
- Elementary conditional logic

Part 1: Codingbat problems (graded)

For this week, we move on to the Logic-1 section. Here are the daily problems. Some of these have a very easy, short solution if you can re-phrase the problem requirements into a slightly different order. Watch out for "except when…", "unless …" and "However…" requirements in the problems which express special exceptions to a general rule. Handle these similarly to how you did sumDouble last week. There are also videos for some of these, which I will publish as the week progresses.

Tuesday: SquirrelPlay

Wednesday: in1to10.

Thursday: TeenSum

Weekend: CaughtSpeeding

Monday: teaParty

Remember, at the bottom of almost every Codingbat page is a list of 4 help documents. Skim the first one on "If boolean logic" if you haven't already. Don't expect everything on that page to make sense at once. I will be explaining more of this in class.

Part 2: Another Simple BlueJ Lab Exercise

All you need to turn in for Part 2 is a lab project jar file with your code, just as you practiced back in Lab zero. There are no additional questions or essays involved. See how clean and stylish (see Appendix J) you can make this code, given that you have plenty of time.

To complete this assignment, make sure you review page 55 of your text for details on how to use the String class's *length* method. For instance, "abc".length() returns 3. That is the only String method you should need for this lab.

For this lab only, I have copied the lab steps out of the book. From this point on, I will assume that everybody has their 5th-edition texts, and I will assign book problems

HINT: Test your work *after each step* by manually running tests from the Object Bench or CodePad. You may record the tests if you want to, but that is not a requirement this week.

1. Download the *book-exercise* project from Angel.

2. (2.83) Add two accessor methods – **getAuthor** and **getTitle** – to the Book class that return the values stored in the author and title fields as their respective results.
   Write these methods after the class constructor. Test your class by compiling it, creating some instances, and calling these methods. These are just simple “getters”.

3. (2.84) Add two methods, **printAuthor** and **printTitle**, to the Book class. These should print the author and title fields, respectively to the terminal window. They should not return any values to their client.

4. (2.85) Add a further field **pages** to the Book class to store the number of pages in a book. This should be of type **int,** and its initial value should be passed to the constructor in a parameter called *bookPages*, along with the author and title information. (This means you have to change the constructor). Add an appropriate **getPages** accessor method to the Book class for this field.

5. (2.86, modified) Add a method **printDetails** to the Book class. This should print details of the author, title, and pages to the terminal window. Print out each book’s details on a single line with captions like "Title:" and "Author:", as in the format used in the text for *Robinson Crusoe.* To get all that information in one line, you can either do some string concatenation, or else use System.out.print. Good practice either way!

6. (2.87) Add a further field r**efNumber** to the Book class. This field can store a reference number for a library, for example. This field should be of type String. I should be initialized to a zero-length string (two double quotes with nothing between them) in the constructor. The constructor should not have a parameter specifying an initial value for this field since the value is constant. Instead, to set its value, define a mutator for it with the following header:

   public void setRefNumber (String ref)

   The body of this method should assign the value of the parameter **ref** to the **refNumber** field. Add a corresponding **getRefNumber** accessor to help you check that the mutator works correctly.

7. (2.88) Modify your **printDetails** method to include printing the reference number with an appropriate caption. However, the method should print the reference number only if it has been set by the mutator – that is, the reference number has non-zero length. If it has not been set, then **printDetails** should print the string “ZZZ” for the reference number instead. *Hint:* use a conditional statement whose test calls the **length** method on the **refNumber** string.

8. (2.89) Modify your **setRefNumber** mutator so that it sets the **refNumber** field only if the parameter is a string of at least three characters. If it is less than three, then print an error message and leave the field unchanged. Hint: re-read these instructions very carefully and think about them before you jump into coding.

9. (2.90) Add a further integer field **borrowed** to the Book class. This keeps a count of the number of times a book has been borrowed. Add a mutator, **borrow**, to the class. This should update the field by 1 each time it is called. Include an accessor, **getBorrowed,** which returns the value of this new field as its result. Modify **printDetails** so that includes the value of this field with an explanatory piece of text, such as "borrowed <n> times".

10. (2.91) Add a further **boolean** field **courseText,** to the Book class. This records whether or not a book is being used as a textbook on a course. The field should be set through a new constructor parameter named *courseBook*, and the field itself should be *immutable.* Provide an accessor method for it called i**sCourseText**.

11. Make sure your code is correctly formatted and indented before turning in your work. Use the control-A control-shift-I key combo to auto-indent your code. (command-A, command-shift I on Mac.)

12. Jar your modified project and upload to the drop box.

Part 3: A Short Challenge exercise:

Finishing this challenge exercise is required to have a good chance for a check-plus grade. For this exercise, you need to create a project from scratch from within BlueJ, and then a new class within that project. We haven't talked about how to do this yet, but the menu commands to use should be pretty self-explanatory. Remember there is also a BlueJ tutorial file on Angel under "General Course Info"

1. Do exercises 2.92 and 2.93. Note that the author has jammed a lot of steps together in each exercise, so be careful to note each step separately so you don't miss one.

2. When you are finished, jar up your new project and upload to the drop box.