BSM 261: Object Oriented Programming
LAB 4: Static Methods and Conditionals

October 27/28, 2022

Date Performed: October 27/28, 2022
Instructor: Orhan özgüner
Teaching Assistant: Yunus Emre Avci

1 Getting Started

Before you perform the lab please check the following information:

- Make sure that the lab computer you are using has DrJava downloaded into the desktop. Here is the jar file to check:

  drjava-beta-20190813-220051.jar

- If you can not locate DrJava, please download the DrJava from the following link and place it to your desktop.

  https://sourceforge.net/projects/drjava/files/latest/download

- If you have the DrJava jar file in your desktop, you are ready to perform your lab. In order to use jar files in linux, you need to run it from terminal. Open your terminal and use the following commend to start DrJava

  java -jar /Desktop/drjava-beta-20190813-220051.jar

2 Lab Rules

The purpose of the lab is to get you used to using DrJava, to start experimenting with Java, and to digest the topics you learn over the week.

Here are the lab rules:

- The teaching assistant will make a short recap of the week. Then you will perform the lab in the recitation hour.
- Your grade will be determined by your participation and finishing the assigned work during the recitation. When you get stuck on a question, please ask for help.
- Once you finish all of the tasks, show your work to the TA to receive full credit.
- **If you did not join the lab, or joined more than 10 minutes late or did not show your work to the Teaching Assistant, or did not finish all of the questions in class time you will receive 0.**

3 Lab Instructions and Questions

In this lab, you will create a class, add several static methods, and practice the conditional statements.

3.1 Programming Some Static Methods and Conditional Statements

Write a class called **Lab4**. Since Lab4 is not going to extend another class (other than Object), you begin the class with

```
public class Lab4 {
```

Save the class in a file called Lab4.java.
Remember to put a comment above the class declaration that includes author names and the class description.
Method 1: Write the following public static method. (Put "public" and "static" before the return type and method name.) The method should have two inputs, each of type **double**.

```
/** Input two doubles and return the larger of the two doubles.  */
double maxDouble
```

You will need an conditional statement in the body of the method. Try to write your method so that the body is just of the form:

```
if (do some test) {
what you should do if the test passes
} else {
what you should do if the test fails
}
```

Compile it, and fix any errors.
Test Lab4.maxDouble in the Interactions pane.

Method 2: Add the following class method to class Lab 4:

```
/** Input three doubles and return the middle value of the three.  */
double middleValue
```

So, if the values entered are 1.5, 0.2, and 9.3, the middle value would be 1.5.
Hint: You are going to need more than one conditional to complete this task.
Compile it and fix and errors. Test Lab4.middleValue in the Interactions pane.

Method 3: We will end the lab with an even larger challenge. This routine requires a fairly complicated conditional statement(s). To help you see how to reason through the logic, we will create this one in stages.
Step 1: Writing down the comment and header:
Add the following method header, including the comment:

```
/**
* Return true of the given date/time is daylight savings.
* Daylight savings time begins 2am the second Sunday of March and ends 2am the first
Sunday of November.
*
* @param month - represents the month with 1 = January, 12 = December
* @param date - represents the day of the month, between 1 and 31
* @param day - represents the day of the week with 1 = Sunday, 7 = Saturday
* @param hour - represents the hour of the day with 0 = midnight, 12 = noon
*
* Precondition:  the month is between 1 and 12, the date is between 1 and 31, the
day is between 1 and 7 and the hour is between 0 and 23.
*/
public static boolean isDayLightSavings(int month, int date, int day, int hour) {
```

Step 2: Getting rid of the easy cases: When faced with a complicated logical statement, it is often best to do things in pieces. First, write an if statement that returns false if the month is January, February, or December, and it returns true if the month is April, May, June, July, August, September, or October. You can return either true or false if the month is March or November.

Put a little thought in it because you can accomplish this with both a very long and a rather short if statement.
Important: Test your code. Call Lab4.isDayLightSavings(...) with different values for the month from 1 to 12 and make sure it returns the correct value. If it does not, fix your code. **Hint:** Use the up arrow on the interactions pane to avoid lots of typing.

Step 3. Handle one of the tricky cases: Now add an additional part to your conditional statement to handle the month of November. (I suggest making a test to see if the month is November, and make your then-statement a compound statement to handle all the possibilities for that month.)

First, try to tell from the day and date parameters if this is the first Sunday of the month. If it is, the method should return true if the hour is less than 2 and false if the hour is 2 or greater.

Second, modify your if statement so that if the day/date combination falls before the first Sunday, your method returns true, and if it falls after the first Sunday, the method returns false.

Take a little time thinking about this part. You may be able to figure out a short mathematical way to test this.

Important: Test your code. First rerun the tests you did in the previous step for the different months (except for March and November) to make sure the code still works correctly. Then, test Sunday for dates 1 through 14 and hours 1 and 2 to make sure it recognizes the first Sunday from other Sundays and that it switches answers at hour 2. Finally, test non-Sunday days and various dates to make sure it recognizes day/date combinations that fall before the first Sunday and those that do not.

Step 4. Handle the other tricky case: Now try adding the cases for March. The code will be very similar to the code for November, but there will need to be some adjustment because you now need to deal with the change happening on the second Sunday instead of the first. In particular, make sure your code distinguishes the second Sunday from the first and later Sundays, and it can tell if a day/date combination falls before the second Sunday.

Demonstrate your class to your lab instructor to receive full credit