CS 455 Lab 2: Using objects

Spring 2019 [Bono]

Goals and Background

This lab will give you practice working with some classes from the Java API. One aspect of working with classes is being able to read the class documentation. You will also get some practice using simple console I/O.

The first few sections below are some general information about labs that apply to all labs.

Recommended advanced preparation

Before coming to lab this week we recommend you read over the lab at least through Exercise 1, and to complete that exercise. It's a pencil and paper exercise. When you come to lab you can compare your results with your lab partner.

Organizing your labs and saving your work

First, unless otherwise stated on the lab, all labs will be completed on Vocareum under an assignment corresponding to the lab number (e.g., this one is called `Lab 2`).

General note on saving your work: You should save all of your completed assignments, because sometimes we will build on what we have done in one assignment in a later assignment. Also, if there are any grading issues later, you will have evidence of the work you have done. As long as you don't delete your files in Vocareum, they will still be available in the workspace for the given lab throughout the semester. We recommend doing this even for different parts of a lab when you have already gotten the previous part checked off: thus the different file names for the different parts of this lab. Make sure you put the assignment number (e.g., CS 455 Lab2), and the names and roles of the lab partnership (described further in the next section) in a comment at the top of all the files for the lab (generally source code and README).

Lab pairs

Labs will generally be done working in partnerships (2 people). We will be assigning lab partners today, grouping students with similar programming experience. The pair of you will sit at one workstation. One person will be the designated "driver", doing the typing and mousing to use the tools, and the other will be the scribe, writing down the answers to lab questions. In subsequent labs you will switch off roles. This does *not* mean only one person is doing the work. In *pair programming* both members of the team are active participants in solving the problem, discussing as you go, but with each with a designated role. If one of you has more programming experience, we recommend the lesser experienced of you is the driver this time.

The questions to be answered by the scribe are in bold and labeled Question 1.1, 1.2, etc. You could put the answers into a `README` text file that you edit (which would be easier at a neighboring computer), but if it's more convenient, you could, alternatively, write down the answers on paper. To keep your work organized, it may be helpful to use a lab notebook for this purpose. Wherever the lab mentions the README, we mean a text file or on-paper lab write-up.

DEN students

DEN students will generally not be working in pairs since they are working remotely and away from other students in the class. DEN students will need to electronically submit their labs, so their answers to the lab

questions will go in the README file, rather than on paper. Each lab includes instructions about what DEN students need to submit, and how (for example, see the end of this lab).

Reading and reference material

This section, which will appear in every lab, gives you a guideline to what you'll need to know before doing the lab, or materials to refer to during the lab. The API Documentation link below is definitely in the latter category.

- Horstmann, Ch. 2, Using objects.
- Horstmann, Section 2.6, API Documentation. Introduction to navigating in the Java API documentation pages.
- Java 1.8 API Documentation
- Horstmann, P2.8 (Programming project 2.8) Shows some examples of using `GregorianCalendar`. (Note: Programming projects are at the end of each chapter. P2.* are at the end of Chapter 2.)
- Horstmann, 4.5.2 Concatenation. How to use String concatenation to print multiple items in one `println` statement.
- Horstmann, 4.3.1 Reading Input.
- Horstmann, 5.1 The `if` statement

Exercise 1 (1 checkoff point)

Look up `GregorianCalendar` in Java API (linked above). Note: You do not need to read all the documentation on that page. Use it to help you answer the following questions about the `GregorianCalendar` class.

Question 1.0 In the README file (or lab write-up), put the first and last names and the roles (driver/scribe) of each of the partners, the course name and lab number.

Question 1.1 What `import` statement is necessary to use the `GregorianCalendar` class?

Question 1.2 Use this documentation to figure out how to write a statement to create a `GregorianCalendar` for the date `January 20, 1995`. Write down the statement in your README. Hint: look at the constructors.

Question 1.3 Write a statement (or statements) to use the `get` method to print out the the date (using a month number rather than a name); don't worry too much about formatting. The documentation shows examples of using the `get` method.

Expressions such as `Calendar.MONTH` bear a little explanation here. They are constants related to calendars. They use the dot notation, because in Java, named constants are normally nested within classes. (And for constants it's the name of a class, not an object, that is placed before the dot.) The class in this case is `Calendar` which is the "superclass" of the `GregorianCalendar` class -- we'll be discussing more about such class relationships later in the semester.

The `get` method itself is also from the `Calendar` class. We can use it on a `GregorianCalendar`, because a subclass inherits all the stuff defined in its superclass. That's why `get`'s documentation doesn't appear on the `GregorianCalendar` page: it's on the `Calendar` page instead.

Question 1.4 Look up the detailed documentation for `Calendar.MONTH`. What is the the number we associate with the MONTH field code to indicate January?

Hint: you can get to the detailed documentation for this constant by following the appropriate link under the "Fields inherited from java.util.Calendar" heading (right after the "Field Summary").

Question 1.5 Once you've created your date object, if you wanted to change the date to one 20 days later, what method would you use? Write down the method call to accomplish this. (We're looking for something that would work no matter what the old value of the date was.)

Exercise 2 (1 checkoff point)

Read over the whole exercise before starting. Using the results of your investigations above write a program `Date.java` that creates the date January 20, 1995, prints it out, and then prints the day 20 days later. Here is what the run will look like:

```
1/20/1995
2/9/1995
```

How to copy a file from another lab. If you want to start from the Hello program we started with in the first lab, and then modify it to create this program, this is a good opportunity to try copying a file in Vocareum. (You could also do this by just cutting and pasting the contents of a file, but that becomes more unwieldy with larger files later.) Here's how you copy the file itself:

1. if you are starting from inside the Lab 2 workspace, you can get out of that by clicking "Lab 2" at the upper left. That will get you back to the list of assignments, and can switch to a different lab.
2. go into the workspace for "Lab 1". Select `Hello.java` from the list of files in your `work` folder.
3. click `Copy` (upper left).
4. go back to Lab 2. (technique explained above in step 1.)
5. click `Paste` (upper left).
6. Now you can rename the file. There are two way to do this. *Either:*

    - Select the the file in the file list. Then click the `Rename` button (upper left; you type the new name in the textbox that appears right below that button); *Or:*

    - in the Terminal window use the linux `mv` command:

      ```
      mv Hello.java Date.java
      ```

Hint: Remember that the name of the main class has to match the file prefix.

Common compile error. One common compile error is "cannot find symbol". When you get this message pay close attention to what part of your code it is flagging and what it is complaining about. Some possible causes of this error:

- you are using an undefined variable.
- you didn't import the right class or package that has this symbol.
- you changed the name of your variable, method, or class but you are still using the old name in a few places.
- you are calling a method that doesn't exist for the class you are using (e.g., `myString.sqrt(100)`).
- you are calling a method with the wrong number or type of arguments

---

Question 2.1 Describe two compile errors that you got while developing this code and their fixes. (It's ok if both errors had the same high-level message, e.g., "cannot find symbol".)

For Exercise 2 check-off, show the lab TA your program running, your source code, and your answer to the question above.

Exercise 3 (1 checkoff point)

Make a copy of your code for Exercise 2 in a file called `Birthday.java`. Hint: the linux `cp` command is probably the easiest way to do this. Read over the whole exercise before beginning to write code.

Modify Birthday.java so it prompts the user for their birthday, and then prints out whether their birthday has happened yet this year. If their birthday is the same date the program is run, it should report that their birthday has already happened.

Your output must look exactly like the example shown. Here is what it should look like running (user input shown in italics):

```
Enter your birth month [1..12]: 1
Enter your birth day of month: 5
Enter your birth year [4-digit year]: 1991
Your birthday has already happened this year.
```

and if their birthday hadn't happened yet, the last line of output would be:

```
Your birthday has not yet happened this year.
```

Note: the output of the program depends on what day you run the program. Your program should work correctly, unchanged, any day you run it.

For the purposes of this lab, you do not have to error-check input. E.g., if the user inputs negative numbers, a letter, or the month 5000 at the prompt, your program does not have to handle that condition.

Hints:

- The `GregorianCalendar` class stores a date including an exact time of day. Any of the methods that compare two such objects (e.g., `equals`) are comparing *all* of that.
- we recommend you look up the relevant documentation and plan out the logic involved in solving this problem before you jump into coding. You can use your lab notebook to write down your ideas.

Question 3.1 Describe two compile errors that you got and their fixes while developing this code.

Question 3.2 Devise three test cases (not the one above) for which you and the TA already know the answer, to help verify that your code is working correctly. Describe these test cases and the expected results in the README. Then try out your test cases to help verify your program works properly.

For Exercise 3 check-off, show your program running, your source code, and your answers to the question above to the lab TA. When you run the program include the test cases mentioned above in your runs.

Exercise 4 (1 checkoff point)

Back up your code for Exercise 3 to a file called `Birthday3.java`. One way to do this is with the following command (the `-p` means that the copy retains the permissions and last-changed date of the original);

```
cp -p Birthday.java Birthday3.java
```

This is useful if you mess up and want to revert to back to a subset you know works. Once you copy the file continue to work with `Birthday.java` for this part of the lab.

Keep all the old functionality of the program, but now it should also print out the current age of the user. See example below:

Your output must look exactly like the example shown. Here is what it should look like running (user input shown in italics):

```
Enter your birth month [1..12]: 1
Enter your birth day of month: 30
Enter your birth year [4-digit year]: 1991
Your birthday has not yet happened this year.
You're 27 years old.
```

Note: as in the last problem, the exact output of the program depends on what day you run the program. Your program should work correctly, unchanged, any day you run it.

For Exercise 4 check-off, show your program running and your source code to the lab TA. When you run the program include the test cases from Exercise 3 in your runs.

If you want an extra challenge

This isn't part of the lab proper, but if you have some extra time, you can also do this exercise. Change `Birthday.java` so that it prints out a different message if the day the program is run is on the user's birthday. Here is the message to print:

```
Your birthday is today!
```

Hint: the hints under Exercise 3 may be helpful.

Checkoff for DEN students

When you click the `Submit` button, it will be looking for and compiling the files `README`, `Date.java`, and `Birthday.java` in your home (aka, `work` directory). The lab us due by 11:59pm on Sunday Pacific Time at the end of the week for this lab.
