CPS122 - OBJECT-ORIENTED SOFTWARE DEVELOPMENT

Individual Programming Project

Reading Quiz: Wednesday, February 15, at the start of class
First Preliminary Milestone Due: Wednesday, February 22, at the start of class
Second Preliminary Milestone Due: Wednesday, February 29, at the start of class
Complete Project Due: Tuesday, March 6, at the end of lab

Purpose: To give you experience developing a complete system that uses multiple classes

Introduction

It is a common sight in locations like hotel lobbies to see several clocks displayed showing the time in different places - like in this picture taken in the lobby of a hotel in Lanzhou, China:

For this project, you will be using the clock classes you developed in Lab 3 to produce a similar display. Unless you request otherwise, your project will be posted on the department web server, so that you can show off your work to friends and family during spring break. (You will also want to look at similar projects from previous years to get ideas - though you should bear in mind that the project requirements in some of the previous years were significantly different!)

Requirements

1. Create a display of multiple clocks showing the current time in different locations. The number and placement of the clocks is up to you (use your creativity, and look at what others have done for ideas). As a minimum, the following must be true.

   • The display must include 4 or more clocks, each displaying the current time in a different time zone.
   • Each clock must be labelled appropriately (e.g the name of its time zone or the name of a prominent city in that time zone). (The label can be below the clock or at some other position.)

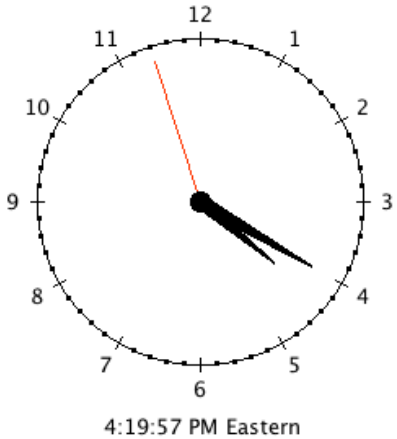
• The clocks must be of two or more different sizes. The largest clock must be at least twice as big as the smallest. (Of course, all the features of the clocks must be scaled appropriately to their size.) You will probably want the larger clock to display the local time zone (Eastern), though you could use it to display some other zone if you prefer (e.g. the zone of your home town.)

2. Make the overall appearance of the clocks aesthetically pleasing, by incorporating some or all of the following improvements.

- "hash-marks" around the face of the clock:
  - minimum - at 12, 3, 6, and 9
  - better: at every hour
  - best: more prominent marks at the hour locations, with 4 smaller marks at the minute locations in between
- numbers correctly positioned around the face of the clock:
  - minimum - at 12, 3, 6, and 9
  - better: at every hour

  (Note: correct positioning that scales with clock size is important!)
- hour and minute hands that are not simply straight lines - the second hand can continue to be a line.
  - minimum: balls at clock center and ends of the hands
  - better: draw the hand as a polygon or use arcs to produce a curved efffect
- time displayed in both digital and analog form - perhaps by adding the digital time to the label for the clock
- improvement of the clock's face appearance
  - use of multiple colors in a single face (not just one as in the lab)
  - lots of other possibilities - see previous projects by other students for ideas

The following illustrates what several (but by no means all!) of these improvements might look like - but don't feel you have to make things look just like this!



Reading Quiz and Milestones

Waiting until the last minute to start this project is definitely a very bad idea1 For this reason, there will be a reading quiz and there will be two preliminary milestones due as noted above to help you get working on the project early. The reading quiz and each milestone will be worth 5 points out of the final grade of 100 points.

1. Reading quiz: **based on the implementation notes below**.

2. First preliminary milestone - place a version of the project in the Drop Box that displays at least four labelled clocks correctly displaying the time in different time zones. (Aesthetic appeal is not important at this point)

3. Second preliminary milestone - place a version of the project in the Drop Box that exhibits some improvements to clock appearance. (You certainly don't need to do everything you intend to do for the project - just enough to indicate that you have been thinking about improvements you might make.)

Evaluation

Your grade on this project will be based on four criteria:

1. The reading quiz (5 points)
2. Fulfilling the preliminary milestones on time (5 points each = 10 total)
3. Correct operation and neat, aesthetically pleasing appearance. (maximum 65 points)
4. Good methodology, including use of comments, appropriate variables and symbolic constants, meaningful names, and good use of white space (indentation and blank lines) to aid readability - see cover sheet (maximum 20 points)

A blank project cover sheet is attached and should be stapled to the front of the final submission of your project.

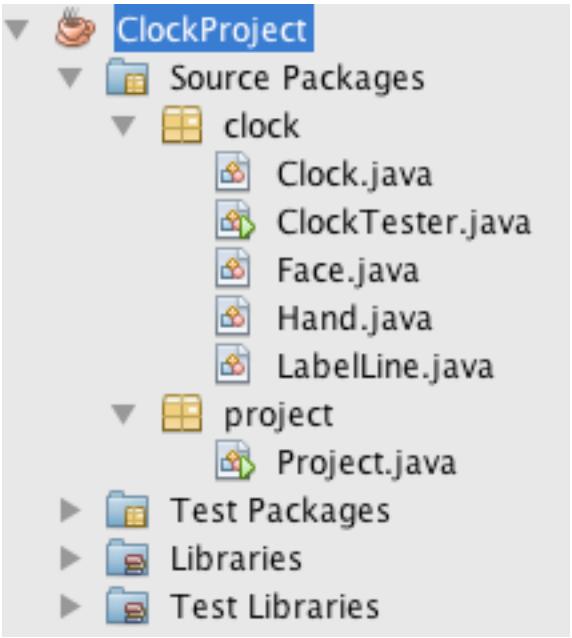
Implementation Notes:

1. To start the project:

    a. Copy the `ClockProject` folder from the common volume to your server volume. This folder contains a NetBeans project with a "starter" version of the main class (`Project`).

    b. Change the name of this folder to incorporate your last name - e.g. `aardvarkClockProject`. Having each student use a different name for the folder is essential when placing work in the Drop Box. (Note: changing the folder name will not change the name of the project that NetBeans displays when you open the project. This is not a problem.)

    c. Copy the `src/clock` folder from the `Clock` folder you used for lab 3 to the `src` folder of the project folder. (You can copy a folder from one place to another on a Macintosh by holding down the option key while dragging the folder to its destination.) When you have done this correctly, the project window will look like this when you open the project in NetBeans and click the disclosure triangles.



d. At this point, it should be possible to run the project. You should see a clock like the one you created in lab, correctly displaying the time in the Eastern time zone.

2. The main class (`Project`) contains code that will allow your project to run either in NetBeans or as an applet on the web. You will need to add code to this class to create your clocks

   a. All the code you add to the `Project` class will go in one of two places:

      • At the point indicated in the `start()` method.
      • Symbolic constant declarations at the end of the class

   b. In addition, you will may need to change the declarations of the symbolic constants `APPLICATION_HEIGHT` and `APPLICATION_WIDTH` if you need more screen space to work with.

   c. You may also want to change the value of the constant `BACKGROUND_COLOR` to better fit the way you have set up your clocks. Feel free to do this.

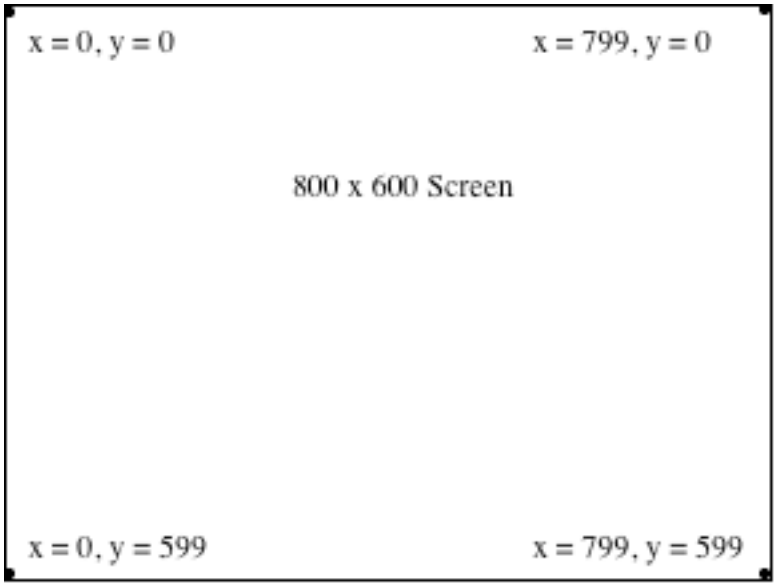
   d. Be sure to delete the comments and examples you are told to delete! Leaving any of these in the project you turn in will result in a point reduction!

   e. Do not change anything else in the `Project` class.

   f. Of course, may also need to make changes to the classes representing the clock and its face, hands, and label line - but see 8b below before changing the `Clock` class.

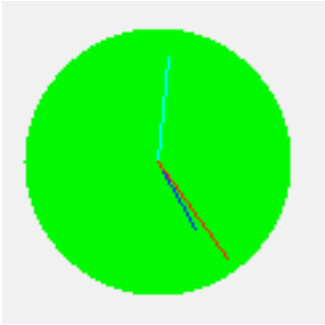
3. When creating a clock, its time zone is specified by giving an offset relative to Greenwich Mean time (GMT). This will be a string having the form `GMT+hh:mm` or `GMT-hh:mm`. For example, Eastern Standard Time is `GMT-05:00`, meaning that Eastern time is 5 hours earlier than Greenwich Mean Time. (In most cases, the minutes part of the specifier is not needed and can be omitted, but some time zones such as Newfoundland need it.)

4. The coordinate system used for graphics systems differs in one significant way from coordinate systems you may have used in mathematics courses. The (0,0) point is the upper left corner of the screen. Therefore, to go up the screen you need to subtract from the y coordinate, while going down requires adding. For example, suppose you use an 800 pixel wide by 600 pixel high screen. Then the coordinate system would look like this:



5. The `java.awt.Graphics` class has a number of methods that you will need when drawing improved versions of the parts of your clock.

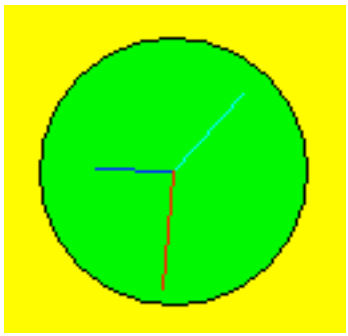
   a. For most shapes, there is a method whose name is of the form `draw____` draw and a method whose name is of the form `fill____`. The "draw" method draws the outline of the shape, while the "fill" method draws a filled in shape. For example, suppose you used `fillOval()` instead of `drawOval()` in the `Face` class. Then your clock would look like this:



   b. You can use both of these methods together to draw both a frame and an outline (using different colors for each, of course.) For example, if you used code like this in your `Face` class:

```
graphics.setColor(color);
graphics.fillOval(leftX, upperY, faceDiameter, faceDiameter);
graphics.setColor(Color.BLACK);
graphics.drawOval(leftX, upperY, faceDiameter, faceDiameter);
```

Your clock would look like this:



c. It is also possible to draw other shapes on top of each other - e.g. a ring can be drawn by drawing a larger circle and then a smaller circle.

d. **Important**: if you draw two things at the same place, the one you draw last will be on top. Since drawing is opaque, that means it will hide anything underneath it. So, in the example above, if you drew the filled oval after you drew the outline, the filled oval would be the only thing you would see!

6. To improve the appearance of the hour and minute hands, you will probably want to create a new `FancyHand` class, which you use for the hour and minute hands, while continuing to use the `Hand` class you developed in lab for the second hand. Of course, you will also need to change the constructor of the `Clock` class to use your new class for the hour and minute hands.

7. There are several things you need to keep in mind to produce aesthetically pleasing results:

   a. If you display 12 hour markers around the face of the clock, the centers of the numerals should be located at 30 degree intervals around an invisible circle perhaps 10%-15% larger than the face circle itself. The `drawString()` method of the `Graphics` class requires you to specify the coordinate of the lower left corner of the string - which will require subtracting off half the width of the string from the calculated x coordinate of the center and adding half the height of the string to the calculated y coordinate of the center. See the `draw()` method of class `LabelLine` for how to get the width and height of a displayed string.

   b. When showing digital time, recall that the convention is to use two digits for minutes and seconds even if only one is necessary - e.g. at time like 1:05:02 PM is represented this way, rather than 1:5:2.

   c. Also, recall that in digital time, while the internal representation of the hour ranges from 0 .. 23, the hours displayed go 12, 1, 2, 3 ... AM, 12, 1, 2, 3 ... PM.

8. The `clock` package that you copy from the lab will also contain the `ClockTester` class. You can use this to test correct operation of your clocks, since the project only displays the current time and you may want to be able to test displaying specific times. (And, in any case, I will want to do this when testing your final submission!)

a. You can use this class for testing your clock by making it the main class for your project. Here's how:
   • Control-click on the Project itself, and choose `Properties` at the bottom of the pop-up menu.
   • Select the `Run` panel in the window that is displayed.
   • Click the `Browse` button next to the Main class field
   • Choose `clock.ClockTester`
   • To return to running your project as the main class, follow the same process but choose `project.Project`.

b. Therefore, be sure to not change the parameters of the `Clock` class constructor or its `setTime()`, `setCurrentTime()` and `start()` methods, since the tester makes use of these. (The `ClockTester` class must still compile correctly).

9. Your project is configured to let you run your program as an application, but you should also test it when run as an applet by opening the file `Project.html` in a web browser. (It should launch Safari if double-clicked.) Note that while both Safari and Firefox seem to handle this applet well, Chrome seem to have a problem with flicker when drawing this applet.

Turn in the following on the final due date, neatly stapled in the order listed:

1. Project Coversheet (attached)
2. Printout of the java sources for the classes you wrote or modified. You just need to turn in a single program, reflecting your finished product. However, you will probably find it wise to develop your improvements incrementally, and to save a copy of working code before moving on to the next feature in the case of catastrophic error. Be sure you have included a suitable prologue comment in each class, and have deleted unnecessary lines in `Project.java`.

Put the Following in the Drop Box for each Milestone and the Final Submission

The NetBeans project folder **with its name changed to *yourlastname*ClockProject**

Unless you specifically request otherwise, a copy of your finished project will be posted on the department server for others to enjoy!

(This page intentionally left blank)

CPS122 - OBJECT-ORIENTED SOFTWARE DEVELOPMENT - PROJECT TWO

Author _____

I. **Preliminaries**

Quiz on implementation notes _____ Milestone 1 _____ Milestone 2_____

Total (max 15) _____

II. Correct Operation / Neat and Aesthetically Pleasing Output

Letter Grade for this operation/aesthetics _____ Points (max 65) _____

III. Methodology

1. Prologue comments for classes and methods and parameter tags make the purpose of each item clear.

   Definitely 4 Mostly 3 Partially 2 Not at all 0

2 Identifiers (class, method, and variable names) clearly describe the item they name and follow OO naming conventions.

   Definitely 4 Mostly 3 Partially 2 Not at all 0

3. Symbolic constants are used where needed.

   Definitely 4 Mostly 3 Partially 2 Not at all 0

4. Local and instance variables are used appropriately and where needed.

   Definitely 4 Mostly 3 Partially 2 Not at all 0

5. Whitespace (indentation and blank lines) and internal comments follow a consistent convention that makes the overall structure of the program clear by enhancing its readability. Unnecessary comments in the original skeleton are deleted.

   Definitely 4 Mostly 3 Partially 2 Not at all 0

Points (max 20) _____

OVERALL TOTAL (Max 100) _____