CPS122 - OBJECT-ORIENTED SOFTWARE DEVELOPMENT

Programming Project #2 - Due Tuesday, March 8, at the end of lab

Purpose: To give you experience with creating and using a class

Introduction

In Labs 3 and 4, you created a `Clock` class. This project involves using several instances of it in a program, together with making significant improvements to its appearance and functioning.

What you will do for this project is to create a program that - as a bare minimum - displays several clocks that continually display the correct time in different time zones. Full credit will come from improving the appearance of the clocks and/or adding the ability to display the time in digital form as well as analog form.

Your program will consist of two classes: a main class (`Project2`) and a `Clock` class. The latter will be based on the class you completed in Lab 4, but with various improvements, as discussed below.

Because your main class will be derived from `objectdraw.WindowController`, which is, in turn a subclass of `java.Applet`, supplying a suitable .html file and a downloadable copy of the `objectdraw` library will make it viewable on other computers. After the projects are completed, your program will be posted on the department's web server so that fellow students, parents, friends, etc. can see what you've created as well. You may also enjoy looking at versions of a similar project created by students in a previous course, linked off the course home page -though you should bear in mind that the project requirements in some of the previous years were significantly different.

Evaluation

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

1. Correct, operation and neat, aesthetically pleasing appearance. (maximum 50-80 points, depending on options chosen) [Note: the maximum total with all the options done correctly is 85; but the operation score is capped at 80 to allow for the possibility of possible minor problems.]

2. Good methodology, including use of comments, appropriate variables and symbolic constants,, meaningful names, and good use of white space to aid readability (indentation and blank lines). (maximum 20 points)

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

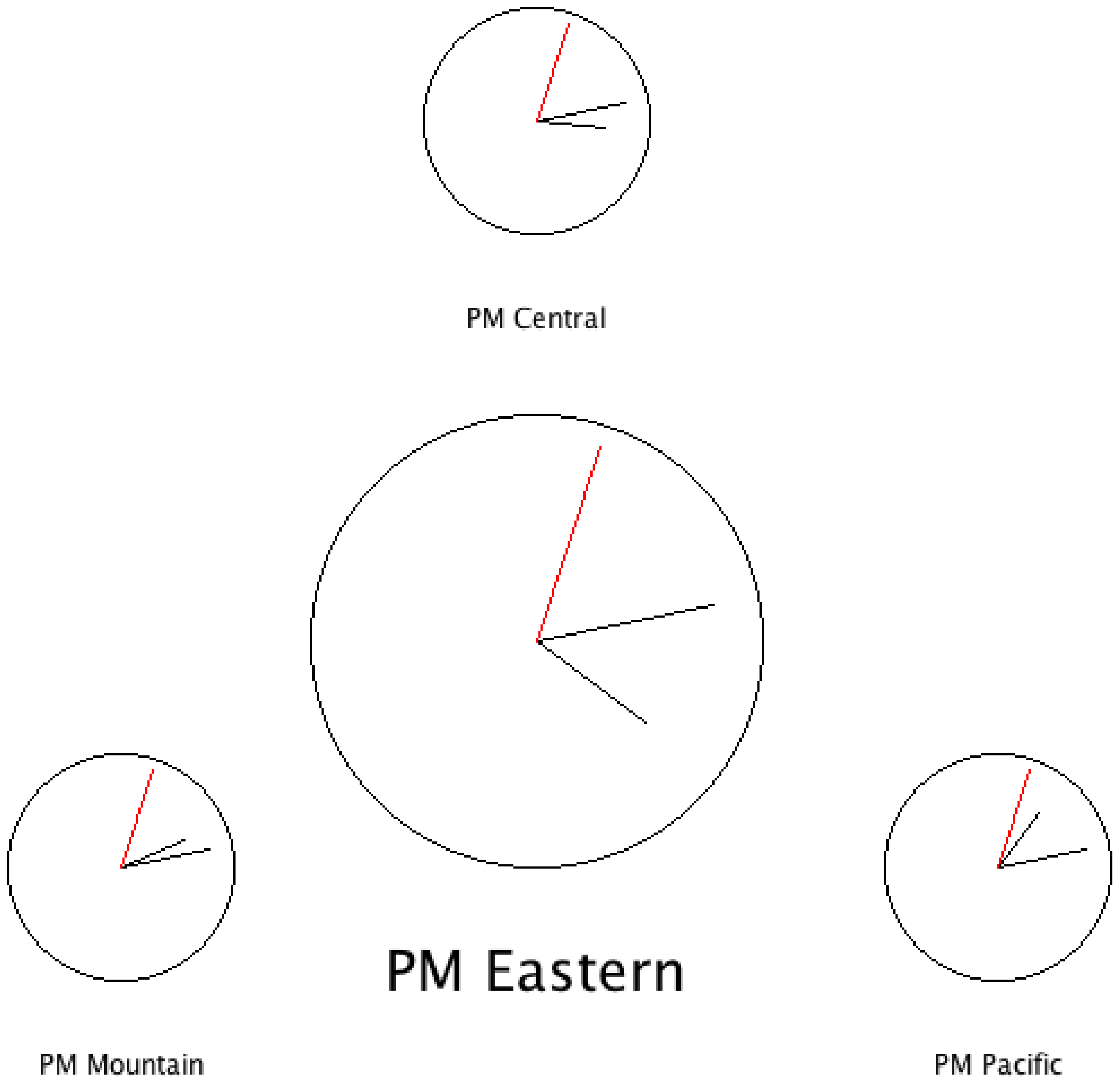
Requirements

Bare Minimum Requirements - maximum of 50 points for correct operation and aesthetic appearance. (Note: you will have to do more than just this to even earn a "C" level grade on the project!)

Fulfill the following requirements:

- Your program must display a minimum of four different clocks - though it can display more if you wish. At least one of these must be significantly bigger than the others. (E.g. perhaps twice as big, though aesthetics may dictate using a somewhat smaller or larger size).

- Each clock should display the current time in a different time zone, and should be labeled in some fashion with the name of its time zone. 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.) All clocks must update themselves continually.
  The following shows what a minimally-satisfactory fulfillment of these requirements might look like. [See discussion below under "Implementation Notes" for suggestions on how to do this.]

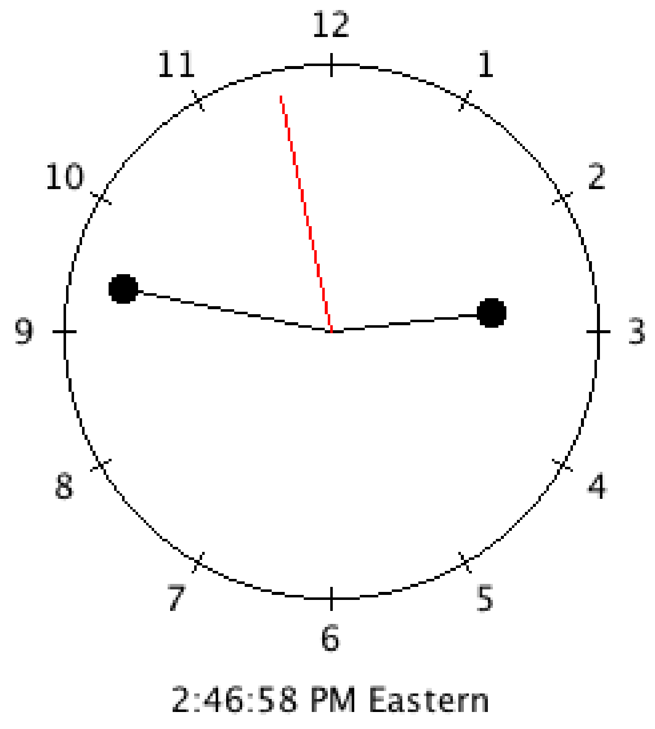


Ways of Earning Additional Points toward Full Credit

Each of the following is worth a maximum additional points as specified. Credit will be based on the aesthetics and correct operation of the finished product. It is more beneficial to do what you do well than to do lots of things sloppily. Full credit for each item will require doing it **very** well, with partial credit awarded for a reasonable effort.

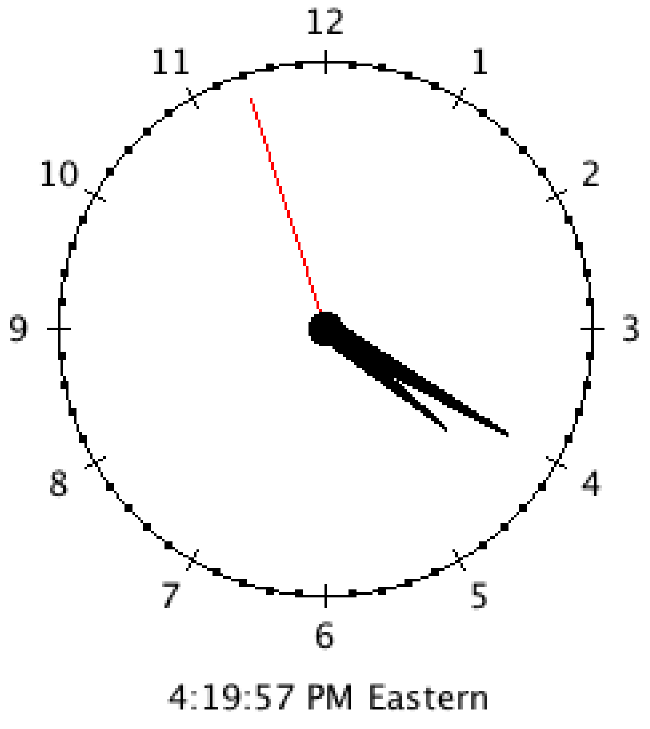
- Add "hash-marks" around the face of the clock (up to 5 or 10 points, depending on method chosen)
- Add numbers around the face of the clock (up to 5 points)
- Improve the appearance of the clock hands (up to 5 or 10 points, depending on method chosen)
- Display the time in digital form as well as analog form. This can be combined with the time zone label if you wish, or can be done on two different lines, with one above and one below the clock or both below it as you see fit. (See student examples from last year for different ways of handling this.) (up to 5 points)
- Other creative possibilities of your own devising (up to 5 points) Note that you can include material in the background behind the clocks if you wish by including appropriate code in Project2.java - as has been done in some of the examples from last year.

The following shows what possible approaches to fulfilling the first four of these might look like - but please don't view them as something to be slavishly imitated. No examples are provided for "other creative possibilities", but you may get ideas by looking at projects from previous years.



GOOD

(5 points for hashmarks and 5 for hands)



BETTER

(10 points for hashmarks and 10 for hands)

Some things to note in the above

- "Hash-marks" are shown at all twelve hour positions, in the 5 point version, with shorter intermediate marks for the minute/second positions between in the 10 point version. [The hash marks at the hour positions in the example are short lines, and the intermediate marks are dots (small filled circles), but you can use lines of differing lengths or colors or dots of different sizes or colors for both if you prefer - just so it is possible to visually distinguish the two kinds of mark.] Of course, there must <u>not</u> be an intermediate mark at the twelve hour positions. It is <u>not</u> required that the size of these scale with the clock diameter - they can be a constant size regardless of diameter. [See discussion below under "Implementation Notes" for suggestions on sizes.]

- All twelve hours positions are numbered, and the position of the numbers relative to the clock face is uniform. This is required for full credit, with lesser credit possible for fewer marks and/or for spacing that is not as uniform. [See discussion below under "Implementation Notes" for suggestions on how to do this.] Once again, it is <u>not</u> required that the size of the numbers or spacing between the numbers and the clock face scale with the clock diameter.

- Two approaches are shown for improving the appearance of the hands. The first approach just adds dots at the ends of the hands; the second draws the hands as filled arcs. The second is much harder to do, but is worth 10 points, whereas the first approach is worth 5. [See discussion below under "Implementation Notes" for suggestions on how to do the filled arcs.]

- The digital time is formatted correctly - e.g. 1:03:04 would be displayed as 1:03:04 not 1:3:4 or 01:03:04. Of course, a number less than 1 or greater than 12 will never appear as an hour!

Implementation Notes:

1. To start the project, copy the Project2 folder from the common volume to your server volume. This folder contains a NetBeans project with a "starter" version of the main class (`Project2`), You will also need to copy the `Clock` class you created in lab into this folder.

2. Your visual display will need to be bigger than the default size objectdraw uses. You should change the symbolic constants `WINDOW_WIDTH` and `WINDOW_HEIGHT`, defined in the main class, to suitable values.

3. There are many places on the web where you can find information about timezones, including differences with UTC. Remember that, for your class, the difference with UTC must be measured in milliseconds, <u>You don't need to worry about daylight savings time - after all, it's still winter!</u>

4. In your main class, after creating each clock, start it by calling its `start()` method.

5. When creating hash marks, you want the center of the line or dot to be on the circle representing the clock face. Since the underlying drawing mechanisms work with integer numbers of pixels, you should use an even number for line length or dot size so that the result is an exact integer when you divide by two for centering.

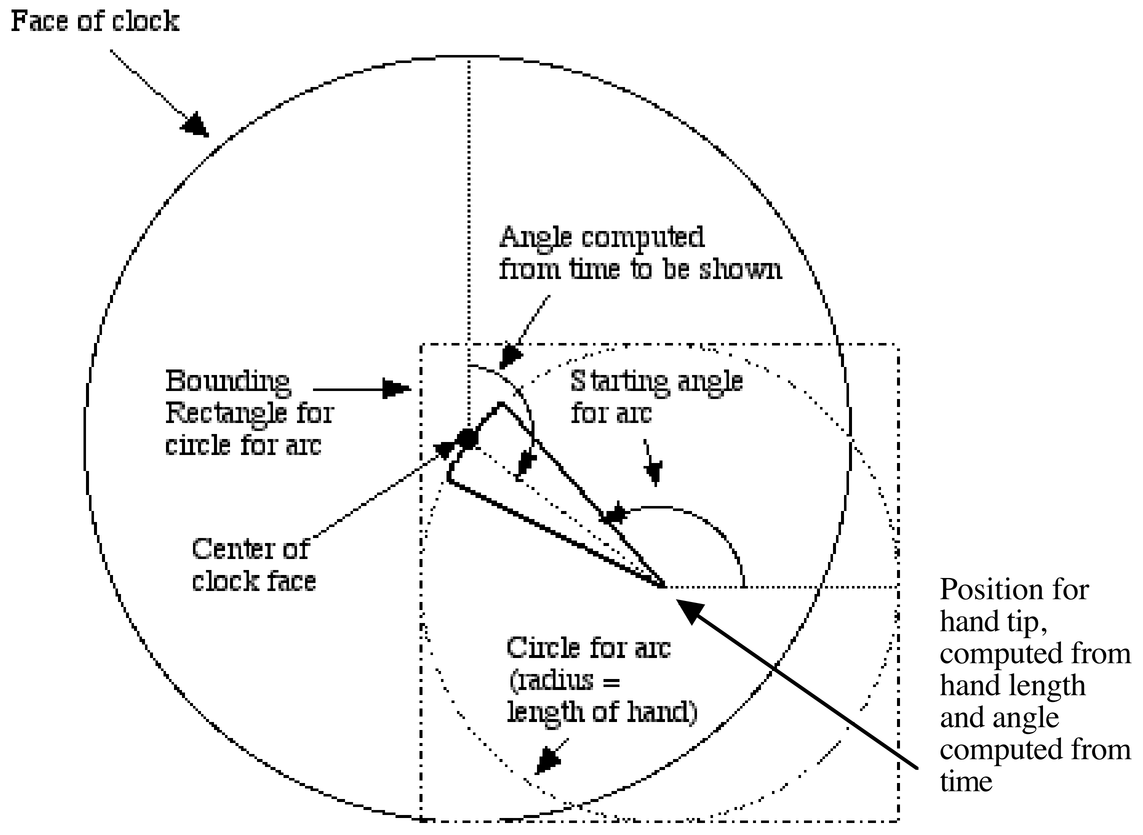
6. Uniform spacing of the numbers around the clock face can be achieved by positioning the center of each number on a circle that is concentric with the clock face. This can be done by using `getWidth()` and `getHeight()` on the `Text` object after it is created and then moving it so that its center is properly positioned.

7. The better version of the improved hands was created by using the FilledArc class. A filled arc represents a portion of a circle, and is specified by giving the coordinates of this circle as well as the starting and ending angle for the arc. (In this case, we want the center of the arc circle to be at the tip of the hand, with the arc circle passing through the clock center - so its radius is equal to the length of the hand.) [This was done for the mouth in the `SmileyFace` example.]

   This means that each time you update the time, you will need to adjust the circle position and the starting angle of the arc; the circle size and size of the arc will not need to change. Its `moveTo()` and `setStartAngle()` methods can be used for this purpose.

   • The circle position is specified by giving the upper-left coordinate of its bounding rectangle.
   • The starting angle of the arc is specified in degrees, relative to "3-oclock" hand position, measured counter-clockwise.
   • The size of the arc is also given in degrees measured counter-clockwise - a value of 6° seems to work well.

   The following drawing shows how these all relate. You will need to do a bit of geometry to figure everything out!


Face of clock
Angle computed from time to be shown
Bounding Rectangle for circle for arc
Starting angle for arc
Center of clock face
Circle for arc (radius = length of hand)
Position for hand tip, computed from hand length and angle computed from time

   If you wish to tackle this option, you can ask for a handout developing an example of the process.

Turn in the following, neatly stapled in the order listed:

1. Project Coversheet (attached)
2. Printout of the java sources for the two classes. You just need to turn in a single program, reflecting the highest option you did. However, you will probably find it wise to attempt options successively, and to save a copy of an option before moving on to the next in the case of catastrophic error. Be sure you have included a suitable prologue comment in each class, and have deleted unnecessary lines in `Project2.java`.

Put the Following in the Drop Box:

The NetBeans project folder **<u>with its name changed to *yourlastname*2</u>**

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

CPS122 - OBJECT-ORIENTED SOFTWARE DEVELOPMENT - PROJECT TWO

Author ____

I. Correct Operation / Neat and Aesthetically Pleasing Output

Minimum requirements (max 50)	______		
Hash marks - good (max 5)	______		
OR			
Hash marks - better (max 10)	______		
Numbers (max 5)	______		
Hand appearance - good (max 5)	______		
OR			
Hand appearance - better (max 10)	______		
Digital time (max 5)	______		
Creativity (max 5)	______	Total Points (capped at 80)	_____

II. 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) follows a consistent convention and makes the overall structure of the program clear by enhancing its readability.

   Definitely 4 Mostly 3 Partially 2 Not at all 0

Points _____

OVERALL TOTAL (Max 100) Points _____