CS 100: Programming Assignment P4

Due: Thursday, March 11, 5pm, Carpenter Lab (or in lecture)

You may work in pairs. Do not submit your assignment for grading unless you have read and understand the CS100 webpage on Academic Integrity. Follow the course rules for the submission of assignments or lose points.

Background

In what phase was the Moon when George Washington crossed the Delaware river and surprised the Hessian soldiers in the early hours of December 26, 1776? We can answer questions like this if we know the Moon's period of revolution *and* its phase at some specified *base date*. Using small, made-up numbers to illustrate the point, if

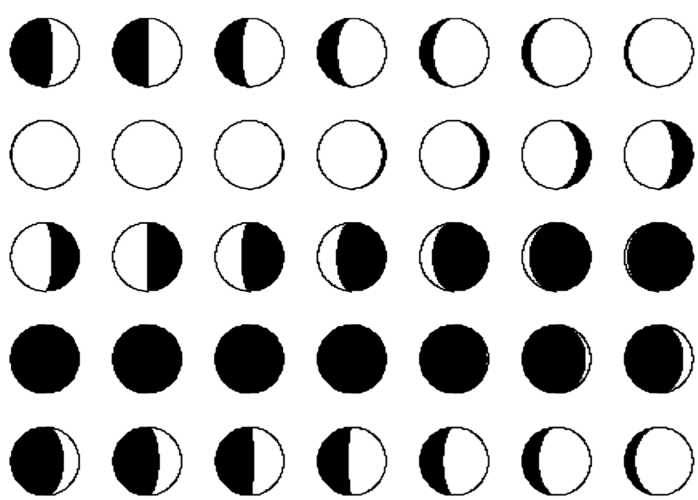
1) The base date is January 1, 1996,
2) The Moon is 4.3 days beyond the new moon phase at the start of the base date, and
3) It takes T = 29.5 days to go from one new moon phase to the next,

then at the start of March 11, 1999 the Moon will be *x* days beyond a new moon where

```
daysFromBase = (366 + 365 + 365 + 31 + 28 + 11)-1;
x = daysFromBase – Math.floor((daysFromBase+4.3)/T)*T
```

As you will see below, once *x* is known it is possible to draw the Moon with the "correct amount of crescent". For example, if $x = T/4$, then a first quarter moon could be drawn.

In this assignment you will develop a class Date with methods that can be used to compute the number of days that have elapsed since the base date. Accounting for leap years will be important. You will also write some graphical methods that display the results in the form of an "array of moons" like this:



February 21, 1999

Our plan is to depict five weeks worth of moon phases where the first moon in the first row is the Moon as it would appear on the input date. Among other things, this will involve writing a graphics method drawMoon that places a moon of prescribed radius and phase at a specified location on the screen.

Getting Set Up

Files P4A.java, P4B.java, P4C.java, and Date.java are available on the web and are required for this assignment. (Listings of the last two are attached.) In the CodeWarrior environment proceed as follows:

- Create a file P4A.java and copy into it from the web the class P4A (including the preceding import statement.)
- Create a file P4B.java and copy into it from the web the class P4B (including the preceding import statement.)
- Create a file P4C.java and copy into it from the web the class P4C (including the preceding import statement.)
- Create a file Date.java and copy into it from the web the class Date.
- Create a project using the application stationary. Remove CUCSApplication.java from the project. Add in all the above files and set the main class target to P4A.

Part A. (3 points correctness + 2 points style)

Study what we provide in the class Date. Observe that there are three instance variables: month (a String), day (an int) and year (an int). There are two class constants (T and tBase) and methods called prettyDate and relDayIndex. The latter is claimed to compute something called the *relative index* of a date. January 1 has relative index 1 and December 31 has index 365. The compound if-else construct just adds the value of day to the relative index associated with the last day of the previous month. Thus, the relative index of May 15 is computed as 120+15 since April 30 has relative index 31+28+31+30 = 120. Note that as given, relDayIndex will not work if the year in question is a leap year. To correct this situation proceed as follows:

- Implement the following static method in the class Date:

```
// Yields true if y is a leap year.
public static boolean leapYear(int y)
```

Recall that *y* is a leap year if it is a century year that is divisible by 400 or a noncentury year that is divisible by 4. Thus, 1900 and 1901 are not a leap years but 1904 and 2000 are.

- Making effective use of leapYear, modify relDayIndex so that it works for leap years *and* nonleap years. This is most easily done by incrementing (if necessary) the value of *s* that is produced by the big compund if construct. You'll want to check if the year is a leap year and that the date is March 1 or later.

Run P4A.java to make sure everything is working. Submit the output.

Part B. (5 points correctness + 3 points style)

In order to display what the Moon looks like on a specified date we have to determine how far it is beyond a new moon. To that end we define January 1, 1600 as the *base date*. The constant tBase in the class Date informs us that at the start of that particular New Years Day we are precisely 14.53206944444927 days beyond a new moon.

Identifying January 1, 1600 with "1", we count forward and assign each day an *absolute day index*. Here are some samples to clarify the concept

Date	Absolute Day Index
January 1, 1601	366 + 1
December 30, 1602	366 + 365 + 362
January 23, 1603	366 + 365 + 365 + 23

Clearly, the computation of the absolute day index involves summing whole year amounts (taking care about leap years) and adding to it the corresponding the relative day index. Note that for a particular date, tBase plus the absolute day index minus 1 is the time (in days) that have elapsed from the base new moon. Suppose for a particular date that this value is *t*. Is the moon waxing or waning? The Moon is *waxing* as it goes from the new moon phase to the full moon phase. During this time the "crescent" is on the right side and getting bigger each night. The Moon is *waning* as it goes from the full moon phase to the new moon phase. During this time the "crescent" is on the left side and is getting smaller each night. To answer the waxing or waning question, determine how many whole number multiples of T (the Moon's period of revolution) are in t. Call this number n. If

$$t - nT <= T/2,$$

then the moon is waxing. Thus, if $t = 3.8T$ then the moon is waning because $n = 3$ and $3.8T - 3T = .8T > T/2$. On the other hand, if $t = 4.2T$, then the Moon is waxing because $n = 4$ and $4.2T - 4T = .2T <= T/2$.

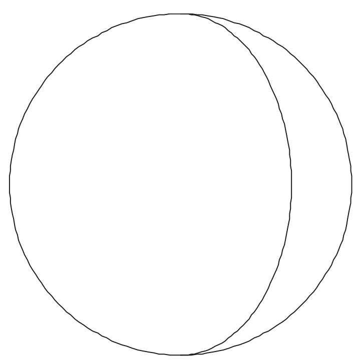
Anticipating our mathematical needs when it comes to the graphical display of a Moon with proper phase, you are required to implement these methods in the class Date:

```
// Yields the absolute index of this date.
public int absDayIndex()


// Yields the number of days since the "base" new moon.
public double sinceBaseNewMoon()


// Yields true if the Moon is waxing t days after a new Moon.
public static boolean isWaxing(double t)
```

P4B.java checks these methods for correctness. Run it and submit the output.

Part C. (5 points correctness + 2 points style)

To correctly draw a phased Moon you need to understand the *terminator*. This is the arc that separates the illuminated and non-illuminated portions of the moon face:



Assume that the equation of the circle is $(h - h_c)^2 + (v - v_c)^2 = r^2$ with v being in the vertical direction. For v values that satisfy $v_c - r <= v <= v_c + r$ define the function

$$d(v) = \sqrt{r^2 - (v - v_c)^2}$$

If t days have elapsed since (any) new moon and the moon is waxing, then the points

$$(v\ ,\ h_c + \cos(2\pi t/T)*d(v)\ ) \qquad\qquad v_c - r <= v <= v_c + r$$

define the terminator. If the moon is waxing, then the region from the terminator to the right semicircle is illuminated. If t days have elapsed since (any) new moon and the moon is waning, then the points

$$(v\ ,\ h_c - \cos(2\pi t/T)*d(v)\ ) \qquad\qquad v_c - r <= v <= v_c + r$$

define the terminator. If the Moon is waning, then the region from the terminator to the left semicircle is illuminated.

In the file P4.java you will find a template for implementing the following method:

```
// Graphics g draws a moon that is t0 days beyond a new moon.
// The center is at (hc,vc) and the radius is r.
public static void drawMoon(Graphics g, int hc, int vc, int r, double t0)
```

Proceed by setting to an appropriate dark color (e.g., Color.black) and drawing a filled circle. Then illuminate on the appropriate side of the terminator by drawing every possible horizontal line segment using a suitable light color, say Color.yellow. We do not care what colors you choose as long as we can deduce from the submitted output that the moon display is correct. You'll find test cases to run in the class MoonPhase. Do not turn in the associated output *unless* you get no further with Part C.

Once drawMoon is working implement the following method:

```
public static void drawWeekOfMoons(Graphics g, int hc, int vc, int r, double t0)
```

It should draw a horizontal row of seven moons. The leftmost moon should have center (hc,vc). All moons should have radius r and should be spaced 3r pixels apart center-to-center. The leftmost moon should be a moon that is t0 days beyond a new moon. Each successive moon should be one day later in phase. To get full credit, your implementation must use drawMoon to draw each of the seven moons. You'll find test cases to run in the class MoonPhase. Do not turn in the associated output *unless* you get no further with Part C.

Once drawWeekOfMoons is working, implement the following method:

```
public static void drawMonthOfMoons(Graphics g, int hc, int vc, int r, double t0)
```

It should draw five rows of moons. The leftmost moon in the top row should have center (hc,vc). All moons should have radius r and should be spaced 3r pixels apart center-to-center. The leftmost moons in the five rows should be(in top to bottom order) moons that are t0, t0+7, t0+14, t0+21, and t0+28 beyond a new moon. Within each row, the successive moons should be one day later in phase. To get full credit, your implementation must use drawWeekOfMoon to draw each of the five moon rows. You'll find test cases to run in the class MoonPhase. Do not turn in the associated output *unless* you get no further with Part C.

Once drawMonthOfMoons is working, use it to display a month of moons associated with one of the following dates:

December 26, 1776	Washington crosses the Deleware
June 6, 1944	D-Day
June 20, 1969	Man on the Moon
February 21, 1999	What's going on now.

Set hc = 100, vc = 100, and r = 30. (Feel free to play with these values and the window size to accommodate your computer screen.) The date should be displayed prominently at the bottom using 24-point TimesRoman Bold font. Submit output.

Submission of Listings

Submit your final version of Date.Java and P4C.java, the latter showing your final version of the class MoonPhase.

The File Date.java:

```
public class Date
{
   // An instance of this class is a particular date.

   private String month;
   private int day;
   private int year;

   // "Moon" constants.

   // T is the average time (in days) between new Moons.
   public static final double T = 29.5305879;  // 29 days + 12 hours + 44 minutes + 2.8 seconds

   // tBase is the time (in days) since the last new Moon at the (GMT) start of January 1, 1600.
   public static final double tBase    = 14.53206944444927;


   // Constructor. Creates a Date object with month s, day d, and year y.
   public Date(String s, int d, int y)
   {
      month = s;
      day   = d;
      year  = y;
   }


   // Yields the relative index of this date.
   // (Days are indexed from 1 starting with January 1.)
   public int relDayIndex()
   {
       int s;
       if (month.equals("January"))
          s = day;
       else if (month.equals("February"))
          s = day+31;
       else if (month.equals("March"))
          s = day+59;
       else if (month.equals("April"))
          s = day+90;
       else if (month.equals("May"))
          s = day+120;
       else if (month.equals("June"))
          s = day+151;
       else if (month.equals("July"))
          s = day+181;
       else if (month.equals("August"))
          s = day+212;
       else if (month.equals("September"))
          s = day+243;
       else if (month.equals("October"))
          s = day+273;
       else if (month.equals("November"))
          s = day+304;
       else
          s = day+334;

       return s;
   }

   // Yields a string that specifies this date.
   public String prettyDate()
      {return month + " " + day + ", " + year;}

}
```

The File P4C.java:

```
import java.awt.*;
public class MoonPhase extends Frame
{

    // Graphics g draws a moon that is t0 days beyond a new moon.
    // The center is at (hc,vc) and the radius is r.
    public static void drawMoon(Graphics g, int hc, int vc, int r, double t0)
    {


    }


   public void paint(Graphics g)
   {
     // Test cases for debugging drawMoon:

     // drawMoon(g,300,300,100,10);
     // drawMoon(g,600,300,100,20);

     // Test cases for drawWeekOfMoons:

     // drawWeekOfMoons(g,100,200,30,0.0);
     // drawWeekOfMoons(g,100,300,30,16.0);

     // Test case for drawMonthOfMoons:

     // drawMonthOfMoons(g,100,100,30,0.0);
   }
}

public class P4C
{
   public static void main(String args[])
   {
      MoonPhase d = new MoonPhase();
      d.resize(800,600);
      d.move(0,75);
      d.setTitle("Moons");
      d.show();
      d.toFront();
   }
}
```