CS102 – Algorithms and Programming II
Lab Programming Assignment 5
Fall 2022

ATTENTION:

- Compress all of the Java program source files (.java) files into a single zip file.
- The name of the zip file should follow the below convention:
  CS102_Sec1_Asgn5_YourSurname_YourName.zip
- Replace the variables “YourSurname” and “YourName” with your actual surname and name.
- You may ask questions on Moodle.
- Upload the above zip file to Moodle by the deadline (if not significant points will be taken off). You will get a chance to update and improve your solution by consulting to the TA during the lab. You will resubmit your code once you demo your work to the TA.

GRADING WARNING:

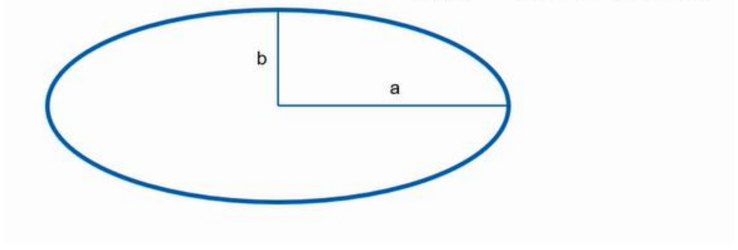
- Please read the grading criteria provided on Moodle. The work must be done individually. Code sharing is strictly forbidden. We are using sophisticated tools to check the code similarities. The Honor Code specifies what you can and cannot do. Breaking the rules will result in disciplinary action.

Space: The Final Frontier

In this assignment, you are going to implement a Java program with a Graphical User Interface to animate the Solar System. You will draw the planets relative to the Sun, so the Sun should be stationary and at the center of your screen. Each planet is going to revolve around the Sun following an elliptical orbit. Try to capture the likeness of the planets by adding color and detail. For example, on top of a filled circle, you may add vertical lines of differing width and color to mimic the planet’s texture. You should also draw the elliptical orbit of each planet.

Implement a **Planet** class that includes *void setPosition(double angle)* and *void draw(Graphics g)* methods. The method *setPosition* should determine the draw location of the planet based on the given angle. In order to calculate the x and y coordinates of the planet, you may use the following equations of the ellipse:

$$(x, y) = (a\cos\theta, b\sin\theta)$$

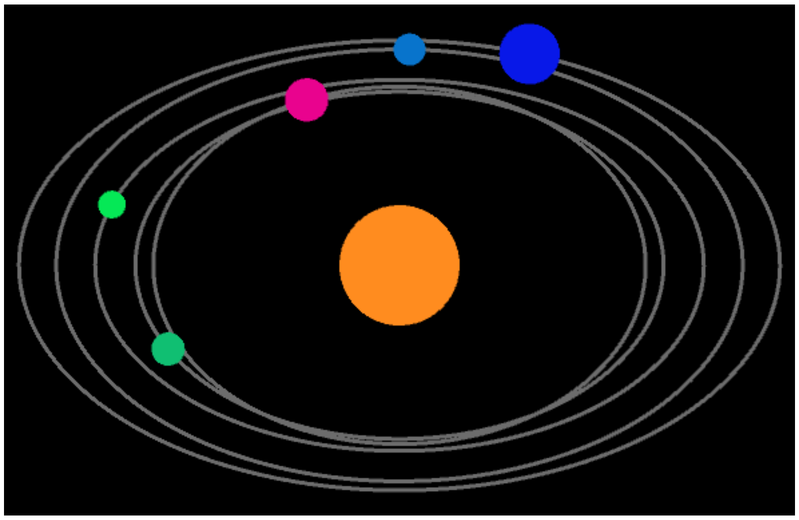


In the equation, *a* and *b* are the lengths of the major radius and the minor radius of the ellipse. Use different a and b values for each planet so that you get different orbits. Also keep a *double* variable to represent the current angle (θ in the equation) of the planet. Additionally, we will keep the *x* and *y* coordinates, and the planet's radius to use in the *draw* method.

The *draw* method should draw the planet on the given graphics context, based on the *x* and *y* coordinates of each planet object. Note that, we may not animate the system all the time, therefore we do not need to calculate the planet positions for each *draw* call, we will simply use the existing *x* and *y* coordinates. The *draw* method should also draw the planet's orbit as an ellipse shape. Note that the width of the orbit would be *2*a* and the height would be *2*b*.

For the preliminary submission, implement the **Planet** class and create a test method where you place the Sun and at least 4 planets around the Sun with different orbits. Test your *setPosition* method to see that it places the planets on the correct positions. Note that Java assumes the top left corner as the origin, so you may need to shift the planets by half of their radiuses while drawing.

For the final submission, you are going to animate your system. Include **Play**, **Pause** and **Rewind** buttons to your user interface. **Play** button should start a timer that increments the angle of each planet by a predefined amount of angle, you are free to introduce new variables to your **Planet** class to resolve this. For example, each planet can have a different angular speed. The timer should calculate the new planet positions using *setPosition* method and then repaint the screen at each frame of animation. **Pause** button should stop the animation. **Rewind** button should animate the system in reverse, decrementing the angles before updating planet positions.



Include a mechanism to insert new planets while the system is running. To this end, you may include text fields for the user to enter orbit radiuses, planet radius, starting angle, and angular speed. In your main method, you may use an ArrayList for keeping your planets. You are free to design your user interface as you want, experimenting with different GUI elements, panels and frames.

Preliminary Submission: You will submit an early version of your solution before the lab. This version should at least include the following:

- Planet class should be functional, including setPosition and draw methods.
- You should test your Planet class, place a Sun at the middle of the screen and a couple of planets around it. Use your setPosition method with different angles to see that it works.

You will have time to complete your solution in the lab. Do not forget to make your final submission at the end. Even if you finish the assignment in the preliminary submission, you should submit for the final submission on Moodle.

Not completing the preliminary submission on time results in 50% reduction of this assignment's final grade.