Object-Oriented Programming

Lab session #4

-

I. References

- Java How to program: https://github.com/nikhil-vytla/Java-How-to-Program-(Early-Objects)-10th-Edition.pdf
    - Download Code Examples: https://github.com/pdeitel/JavaHowToProgram10eEarlyObjectsVersion
    - Refer to Chapters 3, 4, 5, and 7.

- Swing Tutorial: https://www.tutorialspoint.com/swing/index.htm

II. Exercises

You are required to implement the following design as well as a main() method in another class to test your implementation:

Question 1: Using **JOptionPane** dialog called **inputdialog** to ask users to input the user's name and responds with a message dialog containing a greeting and the name that the user entered. (Refer to Chapter 3, section 3.6)

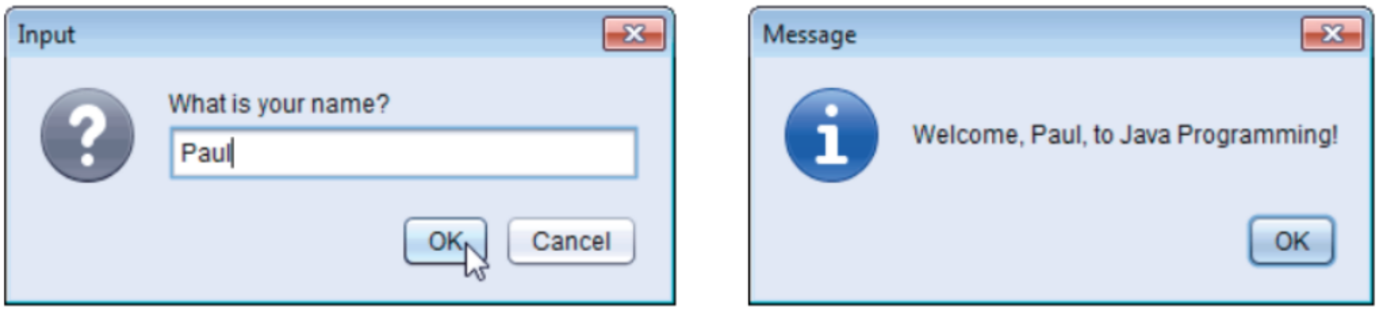


Fig. 1 | Obtaining user input from a dialog.

Question 2: Using class **Graphics** (from package **java.awt**), which provides various methods for drawing text and shapes onto the screen, and class **JPanel** (from package **javax.swing**), which provides an area on which we can draw, to create a simple application that draws four lines as shown in Figure 2. (Refer to Chapter 4, section 4.15)

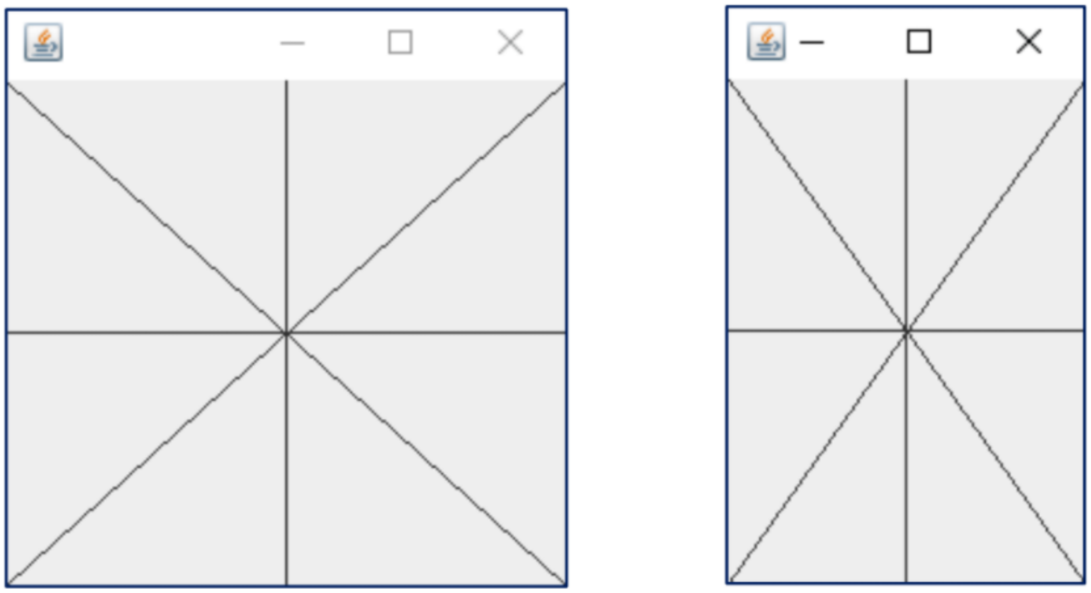


Fig. 2 | Creating JFrame to display DrawPanel.

Question 3: Using loops and control statements to draw lines can lead to many interesting designs.
a) Create the design in the left screen capture of Fig. 3. This design draws lines from the top-left corner, fanning them out until they cover the upper-left half of the panel. One approach is to divide the width and height into an equal number of steps (we found 15 steps worked well). The first endpoint of a line will always be in the top-left corner (0, 0). The second endpoint can be found by starting at the bottom-left corner and moving up one vertical step and right one horizontal step. Draw a line between the two endpoints. Continue moving up and to the right one step to find each successive endpoint. The figure should scale accordingly as you resize the window.
b) Modify part (a) to have lines fan out from all four corners, as shown in the right screen capture of Fig. 3. Lines from opposite corners should intersect along the middle.

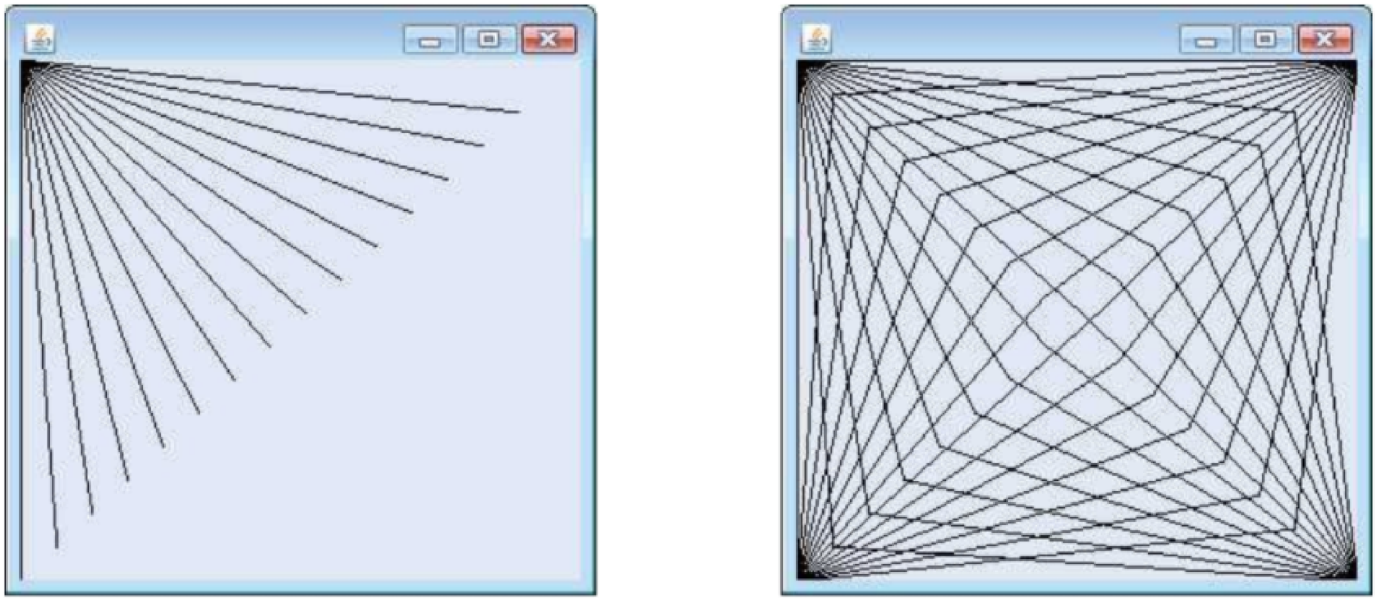
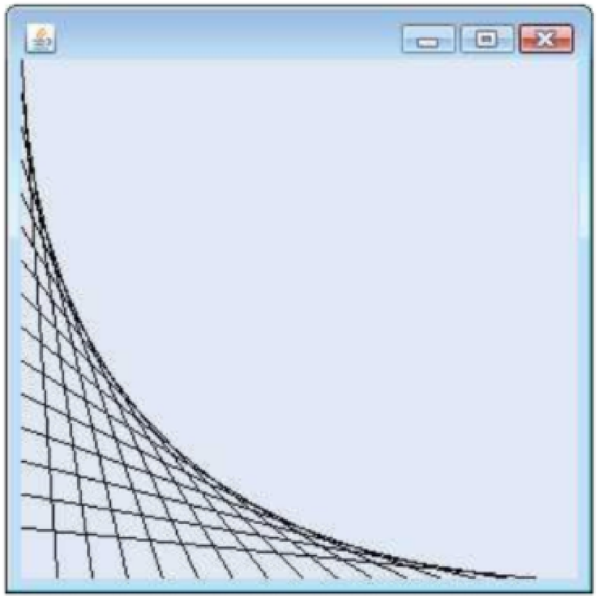


Fig. 3 | Lines fanning from a corner.

Question 4: Figure 4 displays two additional designs created using while loops and drawLine.

a) Create the design in the left screen capture of Fig. 4. Begin by dividing each edge into an equal number of increments (we chose 15 again). The first line starts in the top-left corner and ends one step right on the bottom edge. For each successive line, move down one increment on the left edge and right one increment on the bottom edge. Continue drawing lines until you reach the bottom-right corner. The figure should scale as you resize the window so that the endpoints always touch the edges.
b) Modify your answer in part (a) to mirror the design in all four corners, as shown in the right screen capture of Fig. 4.



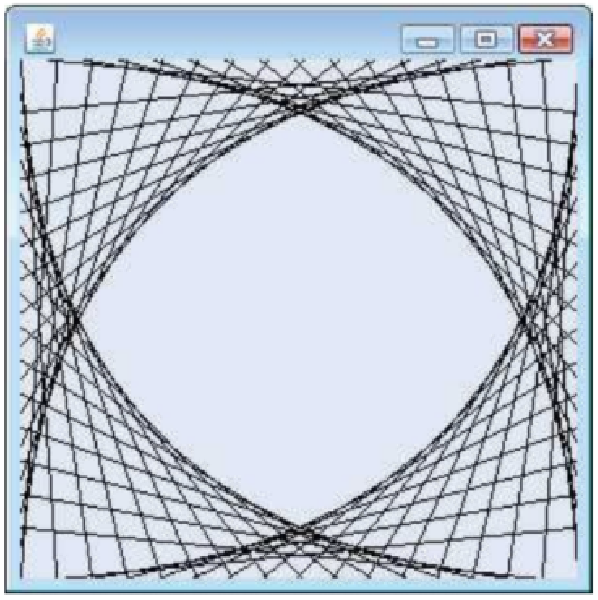


Fig. 4 | Line art with loops and drawLine.

Question 5: (Drawing Spirals) In this exercise, you will draw spirals with methods drawLine and drawArc.
a) Draw a square-shaped spiral (as in the left screen capture of Fig. 5), centered on the panel, using method drawLine. One technique is to use a loop that increases the line length after drawing every second line. The direction in which to draw the next line should follow a distinct pattern, such as down, left, up, right.
b) Draw a circular spiral (as in the right screen capture of Fig. 5), using method drawArc to draw one semicircle at a time. Each successive semicircle should have a larger radius (as specified by the bounding rectangle's width) and should continue drawing where the previous semicircle finished.

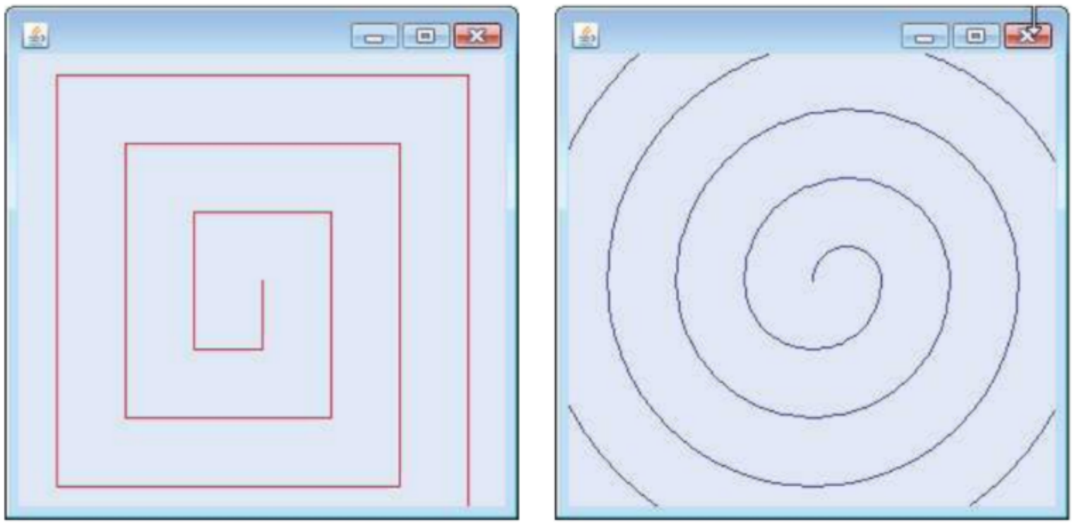
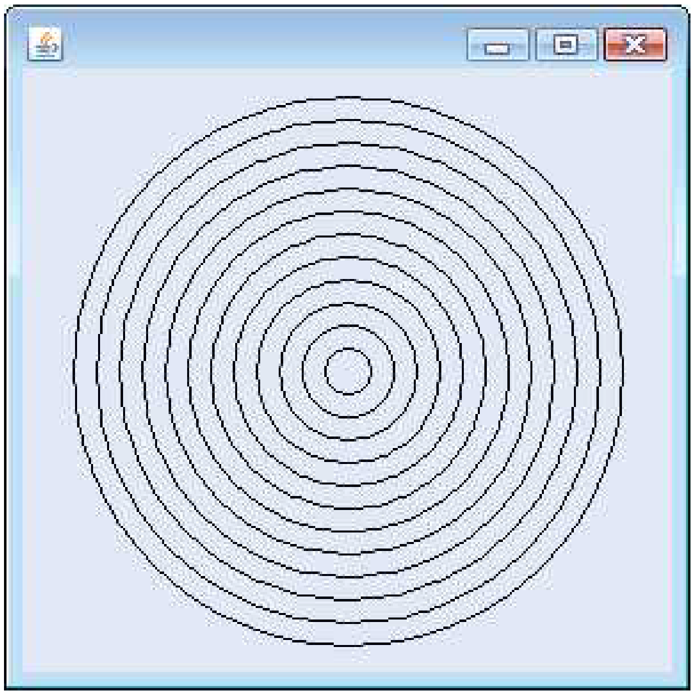


Fig. 5 | Drawing a spiral using drawLine (left) and drawArc (right).

Question 6: Draw 12 concentric circles in the center of a JPanel (Fig. 5.29). The innermost circle should have a radius of 10 pixels, and each successive circle should have a radius 10 pixels larger than the previous one. Begin by finding the center of the JPanel. To get the upper-left corner of a circle, move up one radius and to the left one radius from the center. The width and height of the bounding rectangle are both the same as the circle's diameter (i.e., twice the radius).



Question 7: Create a program that draws 10 random filled shapes in random colors, positions and sizes (Fig. 6.14). Method paintComponent should contain a loop that iterates 10 times. In each iteration, the loop should determine whether to draw a filled rectangle or an oval, create a random color and choose coordinates and dimensions at random. The coordinates should be chosen based on the panel's

width and height. Lengths of sides should be limited to half the width or height of the window.

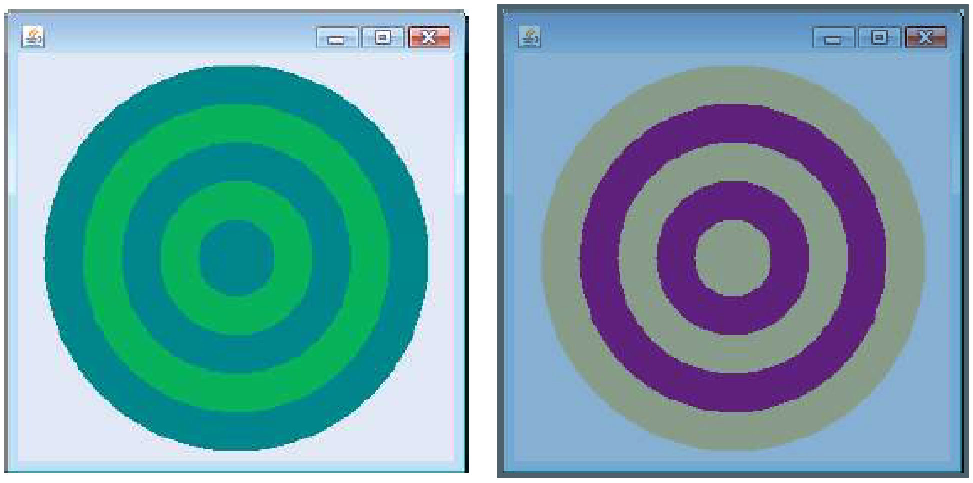


Fig. 6.13 | A bulls-eye with two alternating, random colors.

Question 8: (Bar Chart Printing Program) Write an application that reads five integer numbers using dialogs. For each number that is read, your program should display the bar chart using rectangles of varying lengths. Display the rectangles after you read all five numbers.

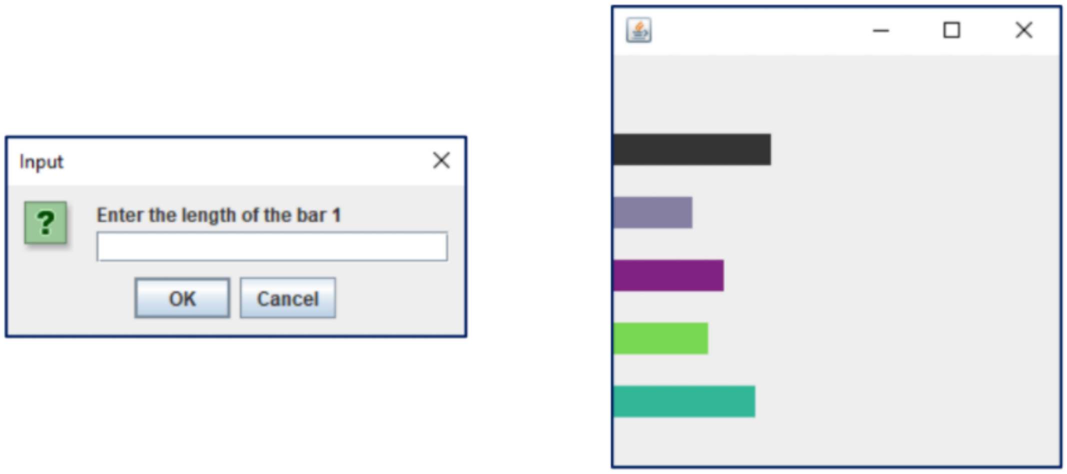


Fig. 6 | Obtaining user input and creating a JFrame to display Bar Chart.

Question 9: (Bonus) Create Captain America Shield like the image below: