Kotlin Programming Exercise

CSC315 Programming Language Concepts

11 October 2021

Base code.

Get this program working on your computer. Work with your classmates.

- How can you divide the work amongst yourselves?
- How can you reduce the amount of work that you must do?
- Where can you take shortcuts?
- Can you get the IDE to do some of the work for you?
- Which pieces of the program will you write first?
- Can you test your program at each stage of its development?

```
import java.awt.Color
import java.awt.Graphics
import java.awt.Graphics2D
import java.awt.geom.AffineTransform
import java.awt.geom.Ellipse2D
import javax.swing.JFrame
import javax.swing.JPanel

object ClusterConstants {
    // specify the size of the window
    // on the computer's screen
    const val WIDTH = 512
    const val HEIGHT = 512
    const val TITLE = "Clusters"
    val BG_COLOR = Color(248, 224, 112)
} // ClusterConstants

data class Rectangle(
    // use this class to specify the
    // bounds on the virtual world
    // in which the program draws
    // a picture
    val xMin: Double, val yMin: Double,
    val xMax: Double, val yMax: Double
) // Rectangle

class PictureCanvas(
    private val bounds: Rectangle,
    private val bg: Color
) : JPanel() {
    init {
        this.background = bg
    } // init

    override fun paintComponent(g: Graphics): Unit {
        super.paintComponent(g)
        val g2D = g as Graphics2D

        val w = this.width
        val h = this.height

        // AffineTransforms can rotate, scale,
        // and rotate shapes——use to make shapes
        // defined in world coordinates fit in
        // this panel (whose location and dimensions
        // differ from the world in which another
        // part of the program defined shapes)
        val translate = AffineTransform()
        translate.translate(−bounds.xMin, −bounds.yMin)

        val sx = w / (bounds.xMax − bounds.xMin)
        val sy = h / (bounds.yMax − bounds.yMin)
        val scale = AffineTransform()
        scale.scale(sx, sy)

        val transform = AffineTransform()
        transform.concatenate(scale)
        transform.concatenate(translate)

        val centerX = (bounds.xMax + bounds.xMin) / 2
        val centerY = (bounds.yMax + bounds.yMin) / 2
        val radius = minOf(
            bounds.xMax − bounds.xMin,
            bounds.yMax − bounds.yMin
        ) / 20
        // make a circle with a specified center
        // and radius——the constructor's arguments
        // are the coordinates of the upper left corner
        // of the smallest rectangle that can hold the ellipse
        // and the width and length of that rectangle
        val dot = Ellipse2D.Double(
            centerX - radius, centerY - radius,
            2 * radius, 2 * radius
        )

        g2D.color = Color.RED // or any other color
        // fill() draws the outline of the shape and its
        // interior
        // must transform shape before drawing it
        g2D.fill(transform.createTransformedShape(dot))
    } // paintComponent()

    override fun repaint() {
        // if there is a need to redraw the picture
        // after changing colors, shapes, and so on,
        // call this method (not paintComponent)
    } // repaint()
} // PictureCanvas

class PictureFrame(
    private val w: Int, private val h: Int,
    private val bg: Color, title: String
) : JFrame(title) {
    init {
        this.setSize(w, h)
        this.defaultCloseOperation = JFrame.EXIT_ON_CLOSE
        this.isVisible = true
        val bounds = Rectangle(
            -1.0, -1.0,
            1.0, 1.0
        )
        val panel = PictureCanvas(bounds, bg)
        this.contentPane = panel
    } // init
} // PictureFrame

fun main() {
    println("Good morning!")
    val picture = PictureFrame(
        ClusterConstants.WIDTH,
        ClusterConstants.HEIGHT,
        ClusterConstants.BG_COLOR,
        ClusterConstants.TITLE
    )
} // main()
```

Extending the program.

1. Define a class that models a point.

   A point has...

   - a read-only floating point property that specifies the point's x coordinate
   - a read-only floating point property that specifies the point's y coordinate
   - an integer property that specifies the cluster to which the point belongs
     - this property has a default value of -1 (to signify no cluster)
     - client code will be able to read and write the values of this property

2. Define a function that returns a function to its caller.

   This require defining two functions: an outer function and (within the outer function) an inner function. The outer function returns to its caller the inner function.

   The outer function has 5 parameters...

   - a random number generator
   - a floating point parameter that specifies the left boundary of a rectangle
   - a floating point parameter that specifies the right boundary of a rectangle
   - a floating point parameter that specifies the bottom boundary of a rectangle
   - a floating point parameter that specifies the top boundary of a rectangle

   This outer function has 5 local variables. Each local variable corresponds to one of the function's parameters. The function assigns to these local variables the values of the corresponding parameters.

   The outer function defines an inner function. The outer function returns to its caller this inner function.

   The inner function has no parameters. It will return to its caller a point that is equally likely to be anywhere within the rectangle specified by the outer function's parameters.

3. Define another function that returns a function to its caller.

   Again, this require defining two functions: an outer function and (within the outer function) an inner function. The outer function returns to its caller the inner function.

   The outer function has 3 parameters...

   - a random number generator
   - a point that specifies the center of a cluster
   - a positive floating point value that specifies a mean distance from the center of the cluster

   The outer function will have 3 local variables. The local variables correspond to the parameters. The outer function assigns to its local variables the values of the corresponding parameters.

   The outer function defines and returns to its caller an inner function. The inner function has no parameters. It returns to its caller a point...

   - whose distance from the given center is a value drawn from an exponential distribution with the given mean

     A distance drawn from an exponential distribution is more likely to be small than large.
   - that lies on a ray that extends from the center in a direction that is equally likely to have any value between 0 and 2π radians

4. Define the boundaries of a rectangle with 4 numbers.

   (Each of the edges of this rectangle is parallel to either the x or y axis.)

5. Call the previously defined function to create a point that is equally likely to be anywhere within a rectangle. This point will be the center of a cluster of points. Choose a non-negative integer to identify this cluster. Store that value in the point.

6. Call the previously defined function to create a collection of points that is scattered around the center.

   - Decide how many points will go into this cluster
   - Decide how far on average these points will be from the center of the cluster.

   These points belong to the same cluster. Store the identifying index of this cluster in the points.

7. Decide on a means of creating several clusters.

   Define a new class or a new function.

8. Define a function that, given 2 points, returns the distance between the 2 points.

   Should this be a function or a method?