Kevin Farragher

Breanna Garan

Courtney Crawford

Massimo Adhikary

Final Project Report

List all classes and modules, describe what they do, indicate what their intended function is.

System Code: 343434Z2Z2Z2SZ Manager Sign Off: Courtney Crawford

**Class: PurchaseTracker** - keeps track of each feature and whether it’s been purchased or not.

**Module:** +getters - return the field variables and their current value.

**Module:** +setters - set the field variables to new values.

**Class: Driver** - used to create and display the UI for the program. Also used to display popups to allow the user to purchase features not purchased by typing in a valid credit card number.

**Module:** +main(String args[][]):void – creates Driver object; used to run the program

**Module:** +Driver():Driver – Constructor for the Driver object. Creates the display for the program and its components

**Module:** +actionPerformed(ActionEvent e):void – called when a component that’s part of the display (button, combo box, slider) is clicked or changed and performs the appropriate action defined for that component. If the component’s feature is not bought, a pop up comes up giving the user the ability to purchase the feature by typing in a credit card number.

**Module:** +luhnCheck(string card):Boolean – checks of entered credit card number for purchasing features is valid using the Luhn algorithm.

**Module:** +calculateCheckDigit(string card): String – checks each character in the entered credit card number and makes sure each is a digit.

**Class: DrawingArea** - creates the type of Jpanel that will be added in driver and handles the mouse listeners that will be used to draw, erase, make shapes, etc. on the Panel. Used to construct the drawing area for the program and allow the user to perform certain action in the drawing area based on the selected function.

**Module:** +paintComponent(Graphics gfx):void – draws shapes, text, and images and redraws previously drawn shape, text, and images in the correct order. Also used to initialize the program.

**Module:** +mousePressed(MouseEvent me):void - does appropriate action when mouse is clicked based on current function selected by the user.

**Module:** +mouseDragged(MouseEvent me):void - does appropriate action when mouse is dragged based on current function selected by the user.

**Module:** +mouseReleased(MouseEvent me):void - does appropriate action when mouse is released based on current function selected by the user.

**Module:** +mouseClicked(MouseEvent me):void - not used, but required to have written out.

**Module:** +mouseEntered(MouseEvent me):void – not used, but required to have written out.

**Module:** +mouseExited(MouseEvent me):void - not used, but required to have written out.

**Module:** +mouseMoved(MouseEvent me):void - not used, but required to have written out.

**Module:** +drawRectangle():void – enables the user having the ability to draw a rectangle and

disables all other drawing functions.

**Module:** +drawCircle():void - enables the user having the ability to draw circles and disables all other drawing functions.

**Module:** +drawPencil():void - enables the user having the ability to drawing with pencil and disables all other drawing functions.

**Module:** +drawEraser():void - enables the user having the ability to erase pixels on the screen and disables all other drawing functions.

**Module:** +drawLine():void - enables the user having the ability to draw lines and disables all other drawing functions.

**Module:** +drawText():void - enables the user having the ability to draw and place texts and disables all other drawing functions.

**Module:** +drawFill():void - enables the user having the ability to fill clicked shapes and disables all other drawing functions.

**Module:** +drawSelect():void - enables the user having the ability select shapes and images and move them around the screen and disables all other drawing functions.

**Module:** +initialize():void – used to initialize the drawing area when the program is loaded. All drawing functions are disabled and a call is made to the paintComponent method to draw undraggable white rectangle that encompasses the entire drawing area that allows the user to change the background color of the drawing area.

**Module:** +getters - return the field variables and their current value.

**Module:** +setters - set the field variables to new values.

**Class: DrawingsClass -** used to stores variables that are common between all drawings, particularly each drawing having starting and ending x and y coordinates and having a line thickness (except for images and text).

**Module:** +getters - return the field variables and their current value.

**Module:** +setters - set the field variables to new values.

**Class: ShapeDrawings -** contains methods to set a shape’s fill color and stroke. Also has methods for the actions that occur when a shape is pressed or released.

**Module:** +setFillColor(DrawingArea drawingArea,Graphics g):void – sets the fill color for the shape

**Module:** +setStroke(DrawingArea drawingArea, Graphics gfx):void – sets the stroke and its thickness for the shape

**Module:** +mousePressed(MouseEvent me, DrawingArea drawingArea):void – sets the shape’s starting x and y coordinates

**Module:** +mouseDragged(MouseEvent me, DrawingArea drawingArea): - sets the shape’s ending x and y coordinates

**Module:** +getters - return the field variables and their current value.

**Module:** +setters - set the field variables to new values.

**Class: RectangleDrawings -** used to store each rectangle and their attributes. Also has methods for drawing a rectangle, initializing the drawing area, the actions that occur when a rectangle is released, and deleting all rectangles and their attributes.

**Module:** +drawRectangle(DrawingArea drawingArea, Graphics g):void – draws a rectangle and saves the rectangle’s color, line thickness, fill status, draggable status, and drawing number (when drawn) relative to all the other drawings if the mouse is no longer being dragged and is released.

**Module:** +initializeDrawingArea(DrawingArea drawingArea, Graphics g):void – used to initialize drawing screen when the program is started. The method is used to draw an undraggable white rectangle that encompasses the entire drawing area that allows the user to change the background color of the drawing area. The rectangle’s starting and ending x and y coordinates, color, line thickness, fill status, draggable status, and and drawing number (when drawn) relative to all the other drawings is saved

**Module:** +mouseReleased(MouseEvent me, DrawingArea drawingArea):void – sets the rectangle’s ending x and y coordinates and saves the rectangle’s ending and starting x and y coordinates.

**Module:** +deleteAll():void – deletes all previously drawn rectangles. Each rectangle’s starting and ending x and y coordinates, color, line thickness, fill status, draggable status, and drawing number (when drawn) relative to all the other drawings are deleted from the rectangle lists storing each rectangles’ attributes

**Module:** +getters - return the field variables and their current value.

**Module:** +setters - set the field variables to new values.

**Class: CircleDrawings -** used to store each circle and their attributes. Also has methods for drawing a circle, the actions that occur when a circle is released, and deleting all circles and their attributes.

**Module:** +drawCircle(DrawingArea drawingArea, Graphics g):void - draws a circle and saves the circle’s color, line thickness, fill status, and drawing number (when drawn) relative to all the other drawings if the mouse is no longer being dragged and is released.

**Module:** +mouseReleased(MouseEvent me, DrawingArea drawingArea):void - sets the circle’s ending x and y coordinates and saves the circle’s ending and starting x and y coordinates.

**Module:** +deleteAll():void - deletes all previously drawn circles. Each circle’s starting and ending x and y coordinates, color, line thickness, fill status, and drawing number (when drawn) relative to all the other drawings are deleted from the circle lists storing each circles’ attributes.

**Module:** getters - return the field variables and their current value.

**Module:** +setters - set the field variables to new values.

**Class: LineDrawings -** used to store each line and their attributes. Also has methods for drawing a line, the actions that occur when a line is released, and deleting all lines and their attributes.

**Module:** +drawLine(DrawingArea drawingArea, Graphics g):void - draws a line and saves the line’s color, line thickness, and drawing number (when drawn) relative to all the other drawings if the mouse is no longer being dragged and is released.

**Module:** +mouseReleased(MouseEvent me, DrawingArea drawingArea):void - sets the line’s ending x and y coordinates and saves the line’s ending and starting x and y coordinates.

**Module:** +deleteAll():void - deletes all previously drawn lines. Each line’s starting and ending x and y coordinates, color, line thickness, and drawing number (when drawn) relative to all the other drawings are deleted from the line lists storing each lines’ attributes.

**Module:** +getters - return the field variables and their current value.

**Module:** +setters - set the field variables to new values.

**Class: NonShapeDrawings Class - used for the inheritance hierarchy; no field variables/methods are written**

**Class: PathDrawings -** used to store each path and their attributes. Also has methods for setting the pencil and eraser color for a path, drawing a path, the actions that occur when a path is pressed, dragged, and released is released, and deleting all paths and their attributes.

**Module:**+setPointColor(DrawingArea drawingArea, Graphics g):void – sets the path’s color when drawing in pencil to the chosen point color by the user.

**Module:**+setEraserColor(DrawingArea drawingArea, Graphics g):void – sets the path’s color when erasing to white.

**Module:**+setStroke(DrawingArea drawingArea, Graphics g):void – sets the path’s stroke and its line thickness

**Module:**+drawPath(DrawingArea drawingArea, Graphics g):void - draws a path and saves the path’s drawing number (when drawn) relative to all the other drawings.

**Module:**+mousePressed(MouseEvent me, DrawingArea drawingArea):void – creates a new

path, stores the new path and saves the new path’s color and thickness. The new path is moved to the mouse’s x and y coordinates.

**Module:**+mouseDragged(MouseEvent me, DrawingArea drawingArea):void – a line is drawn from the path’s starting point to the mouse’s x and y coordinates.

**Module:**+mouseReleased(MouseEvent me, DrawingArea drawingArea):void – the path, its color, and thickness are saved.

**Module:**+deleteAll():void - deletes all previously drawn paths. Each path and its color, thickness, and

**Module:** drawing number (when drawn) relative to all the other drawings are deleted from the path lists storing each paths’ attributes.

**Module:**+getters - return the field variables and their current value.

**Module:**+setters - set the field variables to new values.

**Class: TextDrawings -** used to store each text and their attributes. Also has methods for drawing text, the actions that occur when text is pressed, and deleting all texts and their attributes.

**Module:**+setTextColor(DrawingArea drawingArea, Graphics g):void – sets the text color for the text.

**Module:**+drawText(DrawingArea drawingArea, Graphics g) - draws text and saves the text’s

**Module:**color, string, name (font), style, size, starting x and y coordinates, and drawing number (when drawn) relative to all the other drawings.

**Module:**+mousePressed(MouseEvent me, DrawingArea drawingArea):void – sets the text’s starting x and y coordinates.

**Module:**+deleteAll():void - deletes all previously drawn text. Each text’s color, string, name (font), style, size, starting x and y coordinates, and drawing number (when drawn) relative to all the other drawings are deleted from the text lists storing each texts’ attributes.

**Module:**+getters - return the field variables and their current value.

**Module:**+setters - set the field variables to new values.

**Class: ImageDrawings -** used to store each image and their attributes. Also has methods for drawing an image, and deleting all images and their attributes.

**Module:**+drawImage(DrawingArea drawingArea, Graphics g):void - draws an image and saves the images’s image, starting and ending x and y coordinates, draggable status, and drawing number (when drawn) relative to all the other drawings.

**Module:**+deleteAll():void - deletes all previously drawn image. Each image’s image, starting and ending x and y coordinates, draggable status, and drawing number (when drawn) relative to all the other drawings are deleted from the image lists storing each images’ attributes.

**Module:**+getters - return the field variables and their current value.

**Module:**+setters - set the field variables to new values.

**Class: SelectAction -** used to select a shape or image. Handles the location of the mouse and determines which shape/image is to be selected to allow it to be moved around the screen by the user. Has methods for the actions that occur when the mouse is pressed, dragged, and released and the user is attempting to select and move an image

**Module:**+mousePressed(MouseEvent me, DrawingArea drawingArea):void – determines the shape or image that is to be selected and able to be moved around the screen. First, all shape or image draw numbers that contain the x and y coordinates of the mouse click are stored in a linked list. Second, the highest draw number is determined using a loop. Third and lastly, the shape or image containing the x and y coordinates of the mouse click that has the highest draw number (meaning it is the most recent drawing with the mouse click) is determined to be the selected shape or image and able to be moved around the screen. The select start and end x and y coordinates are set to the selected shape or image’s starting and ending x any y coordinates.

**Module:**+mouseDragged(MouseEvent me, DrawingArea drawingArea):void – calls the mouseDraggedReleased() method.

**Module:**+mouseReleased(MouseEvent me, DrawingArea drawingArea):void - calls the mouseDraggedReleased() method.

**Module:**+mouseDraggedReleased(MouseEvent me, DrawingArea drawingArea):void – the selected shape or image’s starting and ending x and y coordinates are changed as the shape or image is being moved around the screen.

**Module:**+getters - return the field variables and their current value.

**Module:**+setters - set the field variables to new values.

**Class: FillAction -** contains methods to set the fill color for a clicked shape and for the action that occurs when the mouse is pressed and the user is attempting to fill a shape.

**Module:** +setFillColor(DrawingArea drawingArea, Graphics g):void – sets the fill color for a shape when clicked

**Module:** +mousePressed(MouseEvent me, DrawingArea drawingArea):void - – determines the shape to be filled with the chosen fill color. First, all shape draw numbers that contain the x and y coordinates of the mouse click are stored in a linked list. Second, the highest draw number is determined using a loop. Third and lastly, the shape containing the x and y coordinates of the mouse click that has the highest draw number (meaning it is the most recent drawing with the mouse click) is determined to be the shape to be filled and has its fill color changed to the current color and its filled status changed to true.

**Class: RotateAction** - not implemented

**Module:**+mousePressed(MouseEvent me, DrawingArea drawingArea):void – not implemented

**Module:**+mouseDragged(MouseEvent me, DrawingArea drawingArea):void - not implemented

**Module:** +mouseReleased(MouseEvent me, DrawingArea drawingArea):void - not implemented

**Module:** +mouseDraggedReleased(MouseEvent me, DrawingArea drawingArea):void - not implemented

**Module:** +getters - return the field variables and their current value.

**Module:** +setters - set the field variables to new values.

**Class: IOActions -**  Handles all IO actions for the drawing area including save, save as, open, open recent, new project, and import image.

**Module:** +saveAs(DrawingArea drawingArea):void – allows user to save their painting (drawing file) as a JPEG file to a chosen destination. The painting’s save path and save files are set.

**Module:** +save(DrawingArea drawingArea):void – allows user to save their painting as a JPEG to the current (already determined) save path and save file. If no current save path and save file exist, the saveAs() method is called.

**Module:** +open(DrawingArea drawingArea):void – allows user to load an image of a previous painting from a chosen file. The programs’s save path and save file are updated to match the loaded image’s path and file. The image of the previous painting is drawn on the drawing area.

**Module:** +openRecent(DrawingArea drawingArea,Path loadFilePath):void – allows user to load an image of a previously opened painting. The programs’s save path and save file are updated to match the loaded image’s path and file. The image of the previous painting is drawn on the drawing area.

**Module:** +newProject(DrawingArea drawingArea):void+importImage(DrawingArea drawingArea):void – The current save path and save file are set to null. All drawings are deleted from the drawing area, and the drawing area is initialized as if the program was just started.

**Module:** +importImage(DrawingArea drawingArea):void – allows user to import a draggable image into their painting from a chosen file. The imported image is drawn on the drawing area.

**Module:** +getters - return the field variables and their current value.

**Module:** +setters - set the field variables to new values.