pkg

group of drawable objects that will be used by renderer

<<enum>>
PointColor
+ RED : int
+ BLUE : int

DrawableGroup
- points : List<Point>
- lines : List<Line>
+ getPoints() : List<Point>
+ getLines() : List<Line>

Line
- endPoint1 : Point
- endPoint2 : Point
+ getEndPoint1() : Point
+ getEndPoint2() : Point

Point
- x : int
- y : int
- color : PointColor
+ getX() : int
+ getY() : int
+ getColor() : PointColor

<<enum>>
UpdateOption
+ Line : int
+ Cluster : int

List of options checked by user

---

update():

1. call notifyObservers() with the option list. Observers should update the points and lines in board
2. return group of updated points and lines based on the options
   For example,
   1. (options -> empty list, return group of only points with no color. lines should be empty list)
   2. (options -> [Cluster], return group of only points with colors. lines should be empty list)
   3. (options -> [Lines], return group of only lines. points should be empty list)
   4. (option -> [Cluster, Lines], return group of lines and points with colors)

(remeber that a user can check options, run and then uncheck options, and run again)

addPoint():

1. add new point to List<Point> container.
2. do not call notifyObservers()
3. return gruop of points and lines

notifyObservers():

1. pass update options to observers

---

render():

1. draw lines and points. If a member of DrawableGroup is empty list, skip drawing the list.

For example, if List<Line> is empty list, just draw points

ClassifierView object is created in program constructor

<<interface>>
ClassifierModel
+ *update(options : List<UpdateOption>) : DrawableGroup*
+ *addPoint(newPoint : Point) : DrawableGroup*
+ *getPoints() : List<Point>*
+ *setPoints(points : List<Point>) : void*
+ *setLines(lines : List<Line>) : void*

ClassifierController
- board : ClassifierModel
- renderer : ClassifierView
+ runOptions(options : List<UpdateOption>) : void
+ addNewPoint(newPoint : Point) : void
+ ClassifierController(board : ClassifierModel, renderer : ClassifierView) : void

<<interface>>
ClassifierView
+ *render(group : DrawableGroup) : void*

Board
-   instance : ClassifierModel
- points : List<Point>
- lines : List<Line>
+ getInstance() : ClassifierModel
- ClassifierModel() : void
+ notifyObservers(options : List<UpdateOption>) : void

Observable

Observer

<<interface>>
PointHandler
+ *update(observable : Board, options : List<UpdateOption>) : void*

KMClusterHandler

NearestNeighborHandler

---

runOptions():

1. This method is action listener for clicking run button (you can create inner class instead of calculate method)
2. Call ClassifierModel.update() and ClassifierView.render()
3. When calling this method, pass options checked by a user.

addNewPoint():

1. This method is mouse event listener for adding new points to Board Panel. (you can create inner class instead of addNewPoint method)
2. Call ClassifierModel.addPoint() and ClassifierView.render()

Mouse and button click listener should be added in the constructor

ClassifierController object is created in program constructor

---

JFrame

JPanel

MainFrame

BoardPanel

MouseAdapter

JPanel

OptionPanel

ActionListener

If the user unchecks every option, pass empty list to event listeners

---

remeber that a user can check options, run and then uncheck options, and run again.
In this case, List<UpdateOption> is empty list, so you need to erase color in point object and remove all elements in List<Line>

KMClusterHandler should update List<Point>

NearestNeighborHandler should update List<List>

<<interface>>
ClassifierModel
+ *update(options : List<UpdateOption>) : DrawableGroup*
+ *addPoint(newPoint : Point) : DrawableGroup*
+ *getPoints() : List<Point>*
+ *setPoints(points : List<Point>) : void*
+ *setLines(lines : List<Line>) : void*

Classifier
+ main(args : String[]) : void
+ Classifier() : void

<<interface>>
ClassifierView
+ *render(group : DrawableGroup) : void*

ClassifierController
- board : ClassifierModel
- renderer : ClassifierView
+ runOptions(options : List<UpdateOption>) : void
+ addNewPoint(newPoint : Point) : void
+ ClassifierController(board : ClassifierModel, renderer : ClassifierView) : void

Classifier() constructor should create model, view, and controller object