Analysis of Change Request

**Task 5.** Implement an "owner" for each figure: An owner is a user who put that figure onto the canvas, and only the owner is allowed to move and modify it. At the beginning, each session declares a session owner, and this session owner will own all new figures created in that session. No other user will be allowed to manipulate them. At the beginning of a session, user inputs an ID and a password. Any function that attempts to modify a figure must check that the figure owner and the current session owner are the same.

According to the change request, one canvas should have several sessions; each session corresponds to one owner; each session has several figures; each owner (or session) can and only can move or modify owned figures.

Based on this analysis, a log-in and user-exchange component should be added (to test); each figure should has an attribute ownerID when created; the validation happens only in move operation or modify operations (delete / resize / other specific operation to certain shape); the change should consider all existing shapes in source code as well as future shapes.

### Process Record of JRipples

1. This task wants to modify the whole framework rather than SimpleApplet of Drawlets. Because this framework has 9 classes that contain main() and 4 different entries of Applets, it is hard to know which one is the topmost, in this task, pick class SimpleApplet to begin with.

2. In first phase, focus on "session", "figure" and "owner". Besides, “only the owner is allowed to move and modify” means Copy/Paste operations needn't validation. In other words, only adding/deleting/moving/reshaping/changing operations are concerned.

3. ownerID and password is completely new feature, because the canvas is loaded in SimpleApplet, so there should be a dependency from this class to login component which means this class may be marked as Impacted in second phase.

4. DrawingCanvas filed is initialized by SimpleDrawingCanvas in class SimpleApplet, so go to class SimpleDrawingCanvas after checking DrawingCanvas.

5. figureHandles in SimpleDrawingCanvas controls the figure associations. figureChange() relates to the validation. All figure modifying operations on canvas are here.

6. Goto Figure which has dependencies from SimpleDrawingCanvas. In interface Figure, there are lots of relative methods. Mark it as Propagating to get all its subclasses.

7. Goto AbstractFigure which is the first Next after marking Figure. This class implements Figure and provides most basic operations. Mark it as Propagating to get all its subclasses.

8. Goto AbstractPaintable which is the first Next, there is no need to change it, mark it as Unchanged.

9. Then go to AbstractShape which is the 1st Next. This class inherits AbstractFigure and provides reshaping operations.

10. Next is AdornedLine, it draws adorned lines on canvas.

11. Next is AdornedLineTool, no need to change.

12. Go to AnySidedPolygonTool which is 1st Next now. Several methods may be impacted.

14. Next is ConnectingLine. Constructors and several methods are impacted.

15. ConnectingLineTool creates new connecting lines, but the change can be implemented in class ConnectingLine. So mark it as Unchanged.

16. ConstructionTool, no need to change.

17. DrawLineTool draws new lines, if change constructors of Line, no need to change it.

18. Ellipse is related to drawing ellipses.

19. FilledShape inherits AbstractShape, mark it as Propagating to get all direct subtypes.

20. LabelEditHandle is for editing content of labels.

21. There is no need to change LabelTool.

22. Class Line is for drawing lines. Located.

23. Mark LineFigure as Propagating to get all its direct subclasses which implement it.

24. Goto LineFigureRelativePoint. Unchanged.

25. LinePointHandle has move() that may be changed.

26. LineTool. No need to change.

27. Several methods in LinearShape should be changed.

28. PolygonFigure, mark it as Propagating.

29. PolygonPointHandle, irrelevant.

30. PolygonShape is for drawing polygons. PolygonTool creates several kinds of polygons. Only need to change one of them.

31. Go to PrototypeConstructionTool. This class creates new Figures based on a prototypical Figure, but no need to change.

32. RectangleTool needn’t to change, but RectangleShape used in should be changed.

33. RectangularCreationTool is for adding rectangular shapes on canvas.

34. RoundedRectangleShape is for drawing rounded rectangles.

35. Mark SequenceOfFigures to get all its direct subclasses which implement it.

36. Go to TLabel. It is used to draw labels.
37. Decide to add the log-in component in SimpleApplet and add two new methods in Figure, change their marks to Located. Now phase 1 has been done.

38. Go to phase 2, all recommended types are irrelevant or needn’t to change, mark them as Unchanged.

Process Record of ArchSummary

0. Open a java file to initialize ArchSummary, this time is Drawing.java.

1. In MIC, the first one PropertyChangeEvent is a library type, so go to the second one class SimpleDrawingCanvas. See MCBI of type Drawing and Component. Impacted.

2. Figure is frequently used in SimpleDrawingCanvas and relates to the change request, but cannot be seen in MIC/MIRC/MCBI, so open it directly from source code and check. Impacted.

3. In Figure’s MIRC, open the 3rd rank class Line because 1st and 2nd have been checked. In this class, see MCBI of type Locator. Impacted.

4. In Line’s MIRC, open the 3rd rank class LinePointHandle with same reason above. Impacted.

5. In LinePointHandle, see MCBI of type LineFigure, open class ConnectingLine. Impacted.

6. In ConnectingLine’s MIRC, go to 5th rank TLabel. Impacted.

7. In TLabel’s MIRC, go to 6th rank PolygonShape. Impacted.

8. In PolygonShape’s MIRC, go to 7th class Ellipse. Impacted.

9. In Ellipse’s MIRC, go to 8th class RectangleShape. Impacted.

10. In RectangleShape’s MIRC, go to 9th class ValueAdapter. Unchanged.

11. In ValueAdapter’s MIRC, go to 8th class SimplePanel, it’s the class that is suitable for log-in part. Impacted. See MCBI of type DrawingCanvas.

12. In SimplePanel’s MIRC, go to 5th class SimpleModelPanel. Unchanged.


15. In StylePalette’s MIRC, go to 14th class ToolBar, no need to change. Unchanged.

16. Return to StylePalette’s MIRC, go to 15th class PaintableViewer. Unchanged.

17. In PaintableViewer’s MIRC, go to 13th class SimpleDrawing which seems to be related. Change some methods and test. Methods in SimpleDrawingCanvas are valid before. So, Unchanged.

18. In SimpleDrawing’s MIRC, go to 8th class SingleDrawingModel. Unchanged.

19. In SingleDrawingModel’s MIRC, no more class seems relevant. So return to class SimpleDrawing. Repeat similar steps until in SimpleModelPanel’s MIRC. Go to 18th rank class Main. Unchanged.

20. Return to ValueAdapter’s MIRC, go to 12th rank class LocatorConnectionHandle. Unchanged. This time we find a field of type Figure. Go to MCBI view and all classes in it have already been checked.

21. In MIRC view, go to 11th rank class ConnectingLinePointHandle. Unchanged.

22. In MIRC view, go to 20th rank class SelectionTool. Unchanged.
23. In MIRC view, go to 21st rank class LabelEditHandle. Impacted.
24. In MIRC view, go to 22nd rank PolygonPointHandle. See MCBI of type PolygonFigure. Unchanged.
25. In MIRC view, go to 23rd rank BoxSelectionHandle. Unchanged.
27. Return to BoxSelectionHandle’s MIRC, go to 25th class RectangularCreationTool. Impacted.
28. Return to BoxSelectionHandle’s MIRC, go to 26th class EllipseTool. Unchanged.
29. Return to BoxSelectionHandle’s MIRC, go to 27th rank RectangleTool. Unchanged.
30. Return to BoxSelectionHandle’s MIRC, go to 29th rank AnySidedPolygonTool. Impacted.
31. Open interface PolygonFigure directly from the source code after checking MCBI view. Unchanged.
32. Return to BoxSelectionHandle’s MIRC, go to 30th rank LabelTool. Unchanged.
33. Return to TextLabel’s MIRC, go to 44th rank BasicStringRenderer. See MCBI of type FontMetrics. Unchanged.
34. Return to Line’s MIRC, go to 12th rank FigureRelativePoint in case ignoring important classes. Unchanged.
35. Return to SimpleDrawingCanvas’ MIRC and find no class need to check. So go to MIC view and get a similar conclusion, too. After this finish Impact Analysis and make the change based on results.