Project 2 Test Report

Daniel Tyebkhan, Sai Lyon Ho, Anhad Gande, Lawson Wheately

GUI Tests

The following tests were performed to check that our GUI performs properly. Each one represents a different possible action. Inputs that use empty strings or cancel are used to test edge cases.

Project 3

1. Undo/Redo (ctrl+z does undo and ctrl+y does redo on the diagram in focus)
   a. Adding/Removing an object can be undone and redone
      i. Passed
   b. Adding/Removing a method to an object can be undone and redone
      i. Passed
   c. Adding/Removing a parameter to a method can be undone and redone
      i. Passed
   d. Adding/Removing a stereotype to an object can be undone and redone
      i. Passed
   e. Adding/Removing a variable to an object can be undone and redone
      i. Passed
   f. Adding/Removing an arrow can be undone and redone
      i. Passed
   g. Adding a parameter can be undone and redone
      i. Passed
   h. Undo can go all the way back to beginning of diagram
      i. Passed
   i. Redo can go back to first undo
      i. Passed
   j. Redo/Undo do nothing when there is nothing to redo/undo
      i. Passed
2. Parameters
   a. Right clicking on a method brings up menu with “Add Parameter”
      i. Passed
   b. Clicking “Add Parameter” brings up an input prompt
      i. Passed
   c. Entering an empty string to the parameter box does nothing
      i. Passed
   d. Cancelling in the parameter box does nothing
      i. Passed
   e. Entering a name in the parameter box adds it
      i. Passed
3. Tabs
   a. Clicking “Add Tab” brings up an input box
      i. Passed
   b. Entering empty or cancelling the add tab input box does nothing
      i. Passed

   c. Entering a name creates a new tab with a blank diagram
      i. Passed
   d. Undo/Redo affects only the current tab
      i. Passed
4. Themes
   a. Changing the background color changes all diagrams’ background colors
      i. Passed
   b. Changing the arrow color changes all arrow colors
      i. Passed
   c. Changing the class color changes all class colors
      i. Passed
   d. Changing the border color changes all border colors
      i. Passed
   e. Changing to light uses light theme
      i. Passed
   f. Changing to dark uses dark theme
      i. Passed
5. Serialization
   a. Serializing classes, methods, variables and inheritance to python works
      i. Passed

Project 2

1. Add Object
   a. Right clicking on the background of the diagram and clicking add object brings up an input box.
      i. Passed
   b. Entering an empty string does not change the diagram
      i. Passed
   c. Canceling does not change the diagram
      i. Passed
   d. Inputting a name adds a class with the name to the diagram
      i. Passed
2. Dragging on any part of object moves the entire object
   a. Passed
3. Dragging Object outside of panel boundary stops it moving
   a. Failing (We did not successfully implement this)
4. Object Menu
   a. Right clicking on an object brings up a menu
      i. Passed
   b. Selecting add method adds a method to the object if the input is not empty or cancelled
      i. Passed
   c. Selecting add variable adds a method to the object if the input is not empty or cancelled
      i. Passed

    d. Selecting add stereotype adds a method to the object if the input is not empty or cancelled
        i. Passed
    e. Selecting delete removes the object from the diagram
        i. Passed
    f. Selecting add arrow adds an arrow between two items which persists over dragging and repainting
        i. Passed
5. Selecting an arrow type paints the correct type
    a. Passed
6. Notable Menu
    a. Right clicking a notable brings up a menu
        i. Passed
    b. Selecting remove removes the item
        i. Passed
    c. Selecting add arrow adds an arrow to any other component which persists with dragging and repainting
        i. Passed
7. Double clicking export saves the diagram as an image
    a. Passes when given a valid path to save at including the correct file extension in the filename
    b. Fails when given an invalid path
8. Double clicking save as saves the file
    a. Passes when give a valid path to save at including correct file extension
    b. Fails when given an invalid path or filename
9. Double clicking open opens a file and loads a diagram
    a. Passes

<u>Unit Tests</u>

The following tests were performed to check that our Document objects perform properly. All our tests were running.

1. *ArrowTests* creates an Arrow() object that takes a *type, from, and to*
    a. getType()
        i. Passes when assertEquals the object Arrow *type* SUBTYPE to a ArrowType.SUBTYPE
    b. getFrom()
        i. Passes when assertTrue has Arrow containing a *from* that equals a new Arrow object has a *from* containing the same *from*
    c. getTo()
        i. Passes when assertTrue has Arrow containing a *to* that equals a new Arrow object has a *to* containing the same *to*
    d. equalsTrue()

        i. Passes when assertTrue has Arrow object containing *type, to, from* that are equal to a new Arrow object with the same *type, to, from*
    e. equalsFalse()
        i. Passes when assertTrue has Arrow object containing *type, to, from* that are not equal to a new Arrow object with different *type, to, from*

2. *NotableTests* creates a Notable() object that takes *name* and *note* as Strings
    a. getName()
        i. Passes when assertEquals the object Notable gets the *name* to a new Notable object with the same *name*
    b. getNote()
        i. Passes when assertEquals the object Notable gets the *note* to a new Notable object with the same *note*
    c. setName()
        i. Passes when assertEquals the object Notable sets the *name* to a new Notable object with the same *name*
    d. setNote()
        i. Passes when assertEquals the object Notable sets the *note* to a new Notable object with the same *note*
    e. equals()
        i. Passes when assertTrue has Notable object containing *name, note* that are equal to a new Notable object with the same *name, note*
3. ObjectClass creates an ObjectClass() that takes a *name* and *position*
    a. getName()
        i. Passes when assertEqual has ObjectClass containing a string *name* that is equal to the string “test”
    b. addChild()
        i. Passes when assertTrue has ObjectClass that contains the a new child
    c. setName()
        i. Passes when assertEqual has ObjectClass with new *name* the same as the *name*
    d. addInstanceVariable()
        i. Passes when assertEqual has ObjectClass with a new variable *name* that is equal to the new ObjectClass containing the same variable *name*
    e. removeInstanceVariable()
        i. Passes when assertTrue has ObjectClass with no or empty array of variable *name*
    f. addStereotype()
        i. Passes when assertEqual has ObjectClass with a new stereotype *name* that is equal to the new ObjectClass containing the same stereotype *name*
    g. removeStereotype()
        i. Passes when assertTrue has ObjectClass with no or empty array of stereotype *name*
    h. addMethod()

            i. Passes when assertEqual has ObjectClass with a new method *name* that is equal to the new ObjectClass containing the same method *name*
    i. removeMethod()
        i. Passes when assertTrue has ObjectClass with no or empty array of method *name*
    j. equals()
        i. Passes when assertEqual has ObjectClass with a *name* and *position* that is equal to the new ObjectClass containing the same *name* and *position*
    k. setPosition()
        i. Passes when assertEqual has ObjectClass with a new set *position* that is equal to the new ObjectClass containing the same *position*

4. StorageTest adds *ObjectClass* and *Arrow* objects to the Storage class
    a. addArrow()
        i. Passes assertTrue when Storage contains an *Arrow*
    b. removeArrow()
        i. Passes assertFalse when Storage does not contain an *Arrow*
    c. addObject()
        i. Passes assertTrue when Storage contains an *ObjectClass*
    d. removeObject()
        i. Passes assertFalse when Storage does not contain an *ObjectClass* and *Arrow*
    e. testUpdate()
        i. Passes verification whether notifiyObservers() gets called when update() is called.
    f. testObserverPatter()
        i. Passes verification that adding an object to Storage notifies observers.
5. *DiagramPanelTests* adds *ObjectClass* objects to *Storage* and verifies if *DiagramPanel* repaints
    a. testDiagramUpdate()
        i. Passes verification that update() method calls repaint()
    b. testDiagramUpdate()
        i. Passes verification that repaint() is not called when update() is not called
    c. testDiagramAddObject()
        i. Passes verification that adding an ObjectClass to Storage repaints DiagramPanel.
    d. testUpdateWithObjectCompnent()
        i. Passes verification that an instantiation of ObjectComponent does not repaint DiagramPanel.