Assignment 6
Star UML Diagrams

(may be done by a team of at most two students)

Assigned: Monday, November 26

Due: Thursday, December 6 (11:59 pm)

Part 1: Activity Diagram

Consider the TreeGUI interface of Assignment 5. Assume for this assignment that we are only working with 'normal trees'. The six UI buttons correspond to six use-cases and we want to invoke them repeatedly as appropriate. Draw a `StarUML Activity Diagram` that gives the flow of control for these use-cases. Develop your solution as follows:

- Name the 'actions' in the activity diagram as `insert`, `delete`, `undo`, `min`, `max`, and `clear`.
- Have an 'initial' node for the diagram but there is no need for a 'final' node.
- Use 'decision' and 'merge' nodes to specify the following flow of control:
  - `min`, `max`, `delete` and `clear` should be done only when the tree is nonempty;
  - `undo` should be done only where there is some operation to be undone; and
  - `insert` may be done at any time.
- To help specify the flow of control, assume that there are three integer variables `i`, `d`, and `u`, all initialized to 0. Every insert operation that changes the state of the tree increments `i`; every delete operation that changes the state of the tree increments `d`; and every undo operation increments `u`. Every clear operation sets resets the three variables to 0.
- The variables `i`, `d`, and `u` are updated automatically and this updating should not be shown in the activity diagram. These variables are to be used in guard conditions after a decision node.

Note: A guard condition is not always required after the decision node and more than two branches may emanate from a decision node. By omitting the guard, you are stating that any one of the branches may be taken.

Save the StarUML model in a file called `Activity.mdj` and also export the diagram in a file called `Activity.png`.

Part 2: Class Diagram

For this assignment, we define a *circuit* as either a *primitive circuit* or a *series circuit* or a *parallel circuit*. There are two types of primitive circuits: *battery* and *resistor*. A series circuit must contain a minimum of two sub-circuits, each of which may be a primitive circuit or a parallel circuit. Similarly, a parallel circuit must contain a minimum of two sub-circuits, each of which may be a primitive circuit or a series circuit.

Draw a `StarUML Class Diagram` showing classes named circuit, primitive, resistor, battery, series, and parallel reflecting the entities described above. In drawing a class, it suffices to show just the class name. Two additional classes, sub-series and sub-parallel, may be shown in the class diagram in order to help specify the sub-circuit requirements for series and parallel circuits. Also draw the inheritance

and aggregation relationships (with multiplicity annotations) that exist between the classes according to the above description.

Save your StarUML model in a file called `Circuit.mdj` and also export the diagram in a file called `Circuit.png`.

***What to Submit*.** Prepare a top-level directory named *A6_Part12_UBITId1_UBITId2* if the assignment is done by a team of two students; otherwise, name it as *A6_Part12_UBITId* if the assignment is done solo. (Order the *UBITId*s in alphabetic order, in the former case.) In this directory, place the files `Activity.mdj`, `Activity.png`, `Circuit.mdj` and `Circuit.png`. Compress the directory and submit the compressed file using the `submit_cse522` command. Only one submission per team is required.

Part 3: State Diagram

To Be Assigned.

End of Assignment 6 – Parts 1 and 2