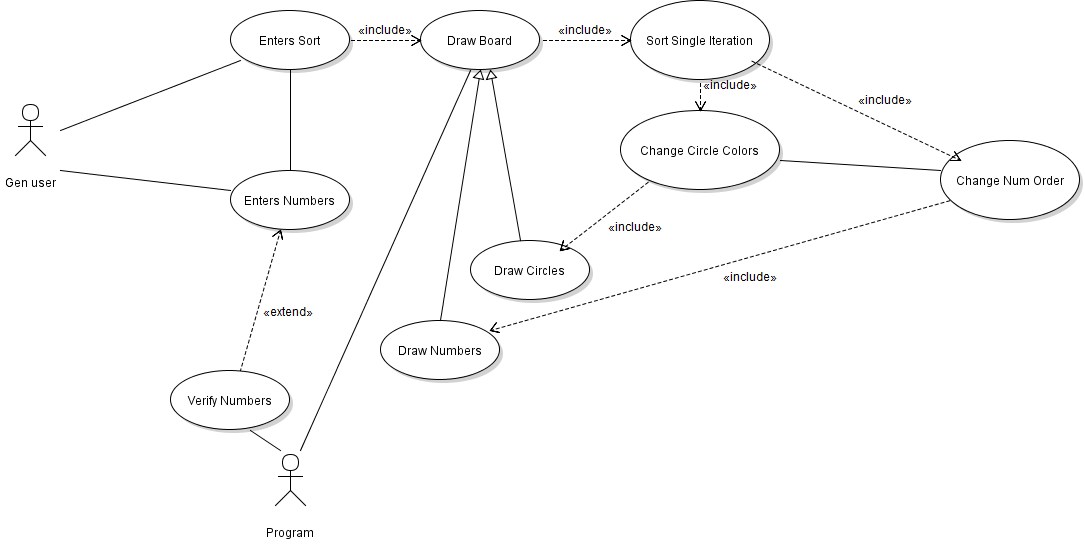
Group I

Patricia Sipes, Garion Armoogam, Germain Matali, Darius Rodriguez

Design Specifications, Use Case, and Class Descriptions

1. Requirements
   1. “Each member must study all five sorting algorithms, design and implement a graphical user interface based animations. Users should be able to enter a list of numbers (10), select a sorting algorithm, and see the sorted list. The animation MUST be done in programming.”
2. Flow Design
   1. User Inputs numbers in textfield
   2. Program registers enter and stores in string, validating numbers (if not, textual prompt to try again)
   3. User selects dropdown for sort type
   4. Program registers change and, if numbers are valid, calls the sort type.
   5. The circles get drawn, the numbers drawn on top.
   6. At each loop/recursion/iteration of the sort, the screen redraws the circles and text.
   7. Circles that have been “moved” change colors, text represents the switching of the numbers.
   8. When finished, all circles change to green to show finished sort.
3. Use Case Diagram



1. Basic Logic
   1. An arrayList stores the numbers at the end of each pass of the algorithm.
   2. A second arrayList stores a Boolean as to whether the item at that index has changed
   3. After algorithm finished, cycle through the arrayLists drawing the circles (red for not changed ever, orange for change, green for finished) and the numbers on top.
2. Class Structure
   1. SortDriver
      1. Drives the program
   2. SortFrame
      1. Draws the Frame
      2. Our Frame is not resizeable
   3. SortPanels
      1. All the visual components and their logic
      2. Internal Classes:
         1. KeyListen, KeyListener for number entry
         2. AnimationListener, ActionListener for the animation timer
         3. ButtonListener, ActionListener for the sort button
         4. TypeListener, ItemListener for the dropdown menu/sort type selector
   4. MasterSort:
      1. ABSTRACT
      2. Contains the basic setup for all the sort objects
      3. Defines the method for storing the array for animation at the end of each step of algorithm
      4. Abstract method for sort types to define as their own sort algorithm
   5. BubbleSort
      1. Extends MasterSort
      2. Recursive BubbleSort Method
   6. SelectionSort
      1. Extends MasterSort
      2. Recursive SelectionSort Method
   7. InsertionSort
      1. Extends MasterSort
      2. Recursive InsertionSort Method
   8. MergeSort
      1. Extends MasterSort
      2. Multiple methods for Merge algorithm
   9. QuickSort
      1. Extends MasterSort
      2. Multiple methods for Quick algorithm