**Chun-Wei Chen**

**CSC 142.04 F11**

**Assignment #9**

**Part 1: Drawing Object Diagrams**

**Objective:**

* Practice analyzing the operation of an object-oriented program by drawing object diagrams

Analyze the following short program and draw an object diagram that represents the contents of memory after the statements have been executed. Refer to the rules for drawing object diagrams in the Week 6 Supplemental Reading. In particular, remember that the arrow representing an object reference always points from a reference variable **to an object** -- never to another variable. When an object has instance variables, show those inside the box that represents the object. The Point and LineSegment classes in the code below are from the Week 6 Sample Programs page.

|  |
| --- |
| double y = 1.8; Point a = new Point( 3, 4 ); Point b = new Point( 0.5, -1.2 ); Point c = new Point( -3.5, y ); LineSegment cb = new LineSegment( c, b ); b = a; double answer = cb.getSecondPoint().getX(); |

**Part 2: ArrayList**

**Objective:**

* Practice tracing code that uses an ArrayList

Analyze the code shown below. Show the console output produced by this code. Also carefully show the contents of the ArrayList (including an indication of which element is at the beginning of the list) after all statements have been executed.

|  |
| --- |
| ArrayList<Double> myList = new ArrayList<Double>( ); myList.add( new Double(8.5) ); myList.add( 0, new Double(7.2) ); myList.add( 9.9 ); // this works fine because of autoboxing myList.add( 0.0 );  System.out.println( myList.get( 2 ) ); myList.set( 2, 3.2 ); System.out.println( myList.remove( 1 ) ); System.out.println( myList.remove( 1 ) ); myList.add( 0, new Double(1.1) ); System.out.println( myList.get( 2 ) ); System.out.println( myList.size( ) ); |

**Part 3: HashMap**

**Objective:**

* Practice tracing code that uses a HashMap

Analyze the code shown below. Show the console output produced by this code. Also carefully show the contents of the HashMap after all statements have been executed.

|  |
| --- |
| HashMap<String, Point> myMap  = new HashMap<String, Point>( );  myMap.put( "a", new Point(3,4) );  myMap.put( "b", new Point(5,12) );  myMap.put( "o", new Point() );  System.out.println( myMap.size( ) ); System.out.println( myMap.get( "b" ) );  System.out.println( myMap.get( "a" ).getX() );  System.out.println( myMap.get( "c" ) );  System.out.println( myMap.size( ) ); myMap.put( "b", new Point(2,2) );  myMap.put( "q", new Point(6,0) );  System.out.println( myMap.size( ) );  System.out.println( myMap.remove( "a" ) );  System.out.println( myMap.containsKey( "b" ) );  System.out.println( myMap.containsKey( "a" ) ); |

**Part 4: Sorting**

**Objective:**

* Practice tracing the Selection Sort algorithm
* Practice tracing the Insertion Sort algorithm

**Instructions:**

Fill in 3 numbers to complete the following 'array', as shown:

35 62 \_\_\_ 40 \_\_\_ 18 \_\_\_ 1 55 28

add the month, day, and year you were born in the 3 spaces shown (in that order).

Using the array of numbers above, write 4 different sort trace tables like those produced by the practice program linked above:

1. Sort using the **Selection Sort** algorithm in **ascending** order.
2. Sort using the **Selection Sort** algorithm in **descending** order.
3. Sort using the **Insertion Sort** algorithm in **ascending** order.
4. Sort using the **Insertion Sort** algorithm in **descending** order.
5. **Selection Sort ascending**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 36 | 62 | 5 | 40 | 17 | 18 | 92 | 1 | 55 | 28 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

1. **Selection Sort descending**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 36 | 62 | 5 | 40 | 17 | 18 | 92 | 1 | 55 | 28 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

1. **Insertion Sort ascending**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 36 | 62 | 5 | 40 | 17 | 18 | 92 | 1 | 55 | 28 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

1. **Insertion Sort descending**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 36 | 62 | 5 | 40 | 17 | 18 | 92 | 1 | 55 | 28 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |