Name: Siddhartha Chhabra
ID: 111461745

**Objective:**

Implementing a randomized incremental algorithm for the construction of minimum enclosing ball of uniformly distributed points.

**Solution/Algorithm Details:**

1) Just to make the problem statement more concise I need to find the circle of smallest radius that contains a given set of points in its interior or on its boundary.
2) I used one of the most simple and elegant algorithm for this purpose i.e a randomized incremental algorithm.
3) The algorithm is quite simple and straightforward and is as follows:

---

***MinDisc0({p1 ,p2 ,…,pn })***

*(i) Randomly permute {p1 ,p2 ,…,pn }*
*(ii) Let D2 be the smallest disc containing p1 and p2*
*(iii) For i = 3 to n do*
*if pi in Di-1 then Di := Di-1*
*else Di := MinDisc1({p1 ,p2 ,…,pi-1 }, pi )*
*(iv) Return Dn*

---

4) The analysis of the complexity of the algorithm can be broken in two cases:

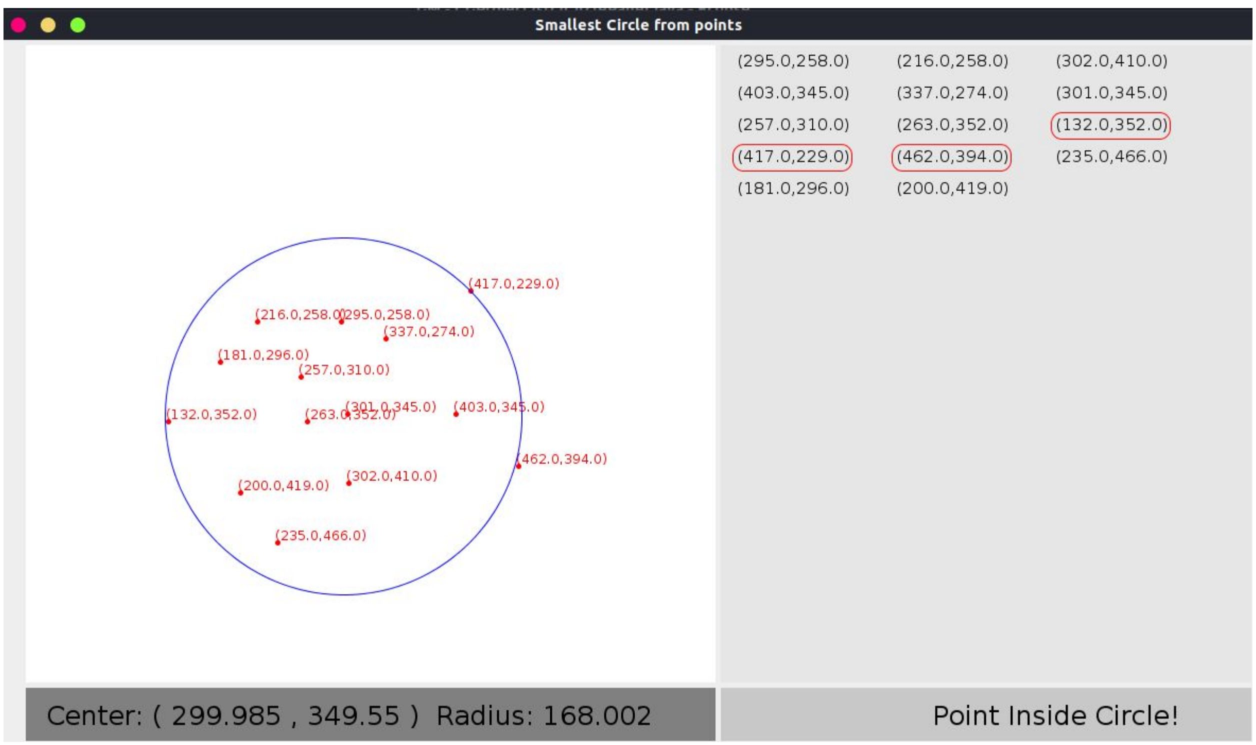
   (i) **Case-1**: When the point lies inside the circle, we can check in constant time and this point will not be a part of our new circle.Complexity: O(1)
   (ii) **Case-2**: When point is outside,
   P(case-2) <= 3/i (backwards analysis)
   E( work to insert point ) = (3/i )*O(i) = O(1)
   Total expected work for n points = O(n).
   **Overall Complexity: O(n)**

5) For this to work it is important that the points are randomly distributed otherwise we can easily achieve a complexity of $O(n^2)$, for example by placing every point outside the circle.

**Implementation Details:**

1) I used Java as my primary language for implementation.
2) I used Java Swing framework for UI and Animations.
3) I used java.math.Random for uniformly random points generation.

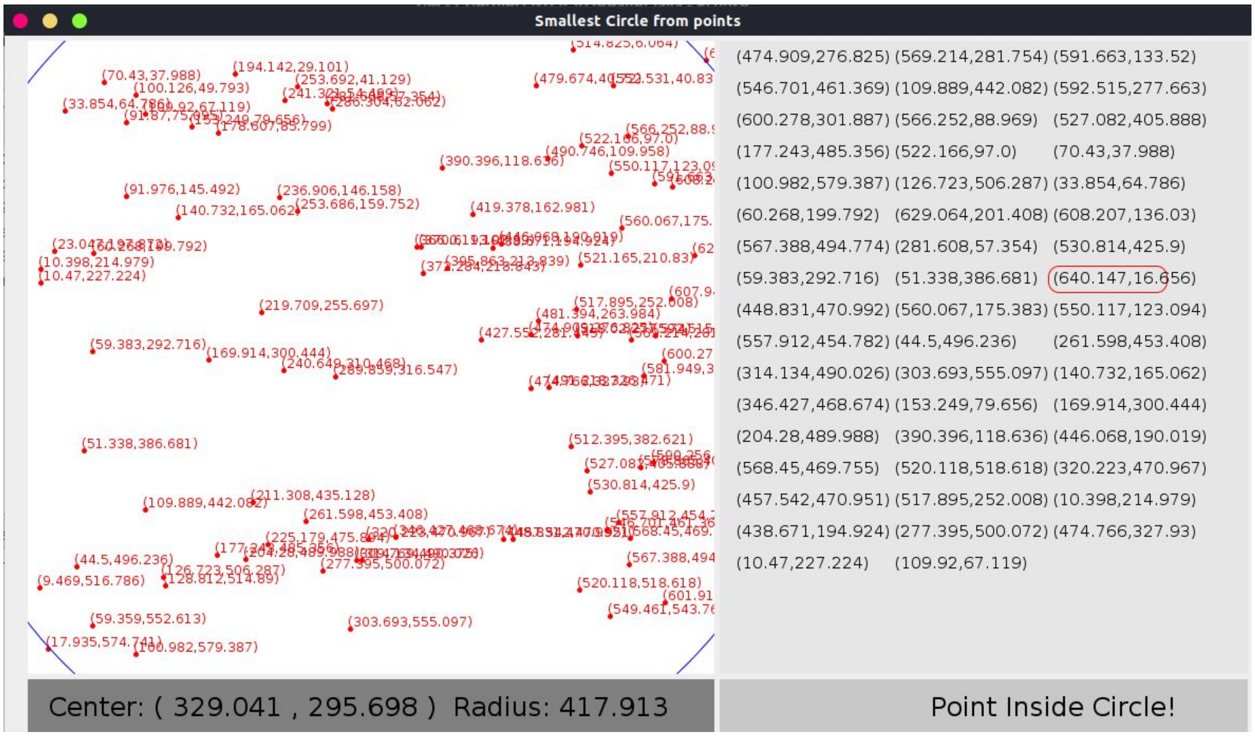
**Results:**




This is one of the images that my project generates. It consist of 4 panels:

1) **Top-Right part** which shows the current state of points and the smallest enclosing ball.
2) **Top-Left part** which shows the three extreme points that determine the current circle.
3) **Bottom-Left part** which tells the current minimum enclosing ball's center and radius.
4) **Bottom-Right part** which shows if the latest point is inside the circle or updates the circle.

Here is another example for a bigger randomly generated circle for more points:




**Some results for different number of randomly generated points(without UI/Animation/Logging):**

| No of Points | Time Taken(ms) | Approximate Complexity |
|---|---|---|
| 1000 | 0.25 | 25n |
| 1000000 | 113.4 | 12n |
| 100000000 | 9971.35 | 9n |

The complexity is written assuming every instruction takes about $10^{-8}$ sec to execute.

**References:**

1) https://en.wikipedia.org/wiki/Smallest-circle_problem.
2) https://www.nayuki.io/page/smallest-enclosing-circle.
3) AMS 545/ CSE 555 Course Material, Spring 2018.