MA117 Project 1: Circles on the Plane

Administrative Details

- This project is the first of the three assignments required for the assessment in this course. It is to be submitted by **5pm, 26th February 2016**. Details of the method of the submission via the Tabula system have been described in the lecture notes and are also available on the course web page.
- This assignment will count for **20%** of your total grade in the course.
- The automated submission system requires that you closely follow instructions about the format of certain files; failure to do so will result in the severe loss of points in this assessment.
- You may work on the assignment during the lab session, provided you have completed the other tasks that have been set. You can use the work areas at all times when they are not booked for teaching, 7 days per week. If you are working on the assignment on your home system you are advised to make regular back-up copies (for example by transferring the files to the University systems). You should note that **no** allowance will be made for domestic disasters involving your own computer system. You should make sure well ahead of the deadline that you are able to transfer all necessary files to the University system and that it works there as well.
- The Tabula system will be open for the submission of this assignment starting from **9am, 8th February 2015**. You will not be able to test your code for correctness using Tabula but you can resubmit your work several times, until the deadline, if you find a mistake after your submission. A later submission always replaces the older one, but you have to re-submit **all** files.
- Remember that all work you submit should be **your own work**. Do not be tempted to copy work; this assignment is not meant to be a team exercise. There are both human and automated techniques to detect pieces of the code which have been copied from others. If you are stuck then ask for assistance in the lab sessions. TAs will not complete the exercise for you but they will help if you do not understand the problem, are confused by an error message, need advice on how to debug the code, require further explanation of a feature of Java or similar matters.
- If you have more general or administrative problems e-mail me immediately. Always include the course number (MA117) in the subject of your e-mail.

1 Formulation of the Problem

This assignment focuses neither on mathematical/modelling issues nor on any advanced techniques in scientific computing. It will test your ability to write a simple Java code consisting of a few classes and manipulate simple data structures. Remember that marks are available for properly documenting and structuring your code.

You are given a set of data stored in a text (ASCII) file. Each line in this file stores 3 real numbers that represent:

- the coordinates (x, y) of a point in the plane and
- a radius r of the circle with the centre at that point.

Hence each line in the file represents a circle in the plane. Your task is to write a code which will analyse this set of circles. The statistical analysis required in this step is very simple and does not require any sophisticated techniques. The information you want to extract from this set of data is:

1. The number of circles in the set. When reading the description of circles from the file you will need to ignore those circles which are "singular", i.e. those for which the radius is equal to zero, within a certain tolerance (described below). However, you will **not** need to check whether two circles might be identical. Also you may assume that the data in the file are in a proper format so you **do not** need to perform any additional checks to, for example, check for an incorrect number of points per line.
2. The largest ($S_{\max}$) and the smallest ($S_{\min}$) area of a circle in the file.
3. The average area of a circle, $\mu = \frac{1}{N}\sum_{i=1}^{N} S_i$ (where S_i is the area of the ith non-singular circle in the data file).
4. The standard deviation of area; i.e., compute
$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(S_i-\mu)^2} = \sqrt{\left(\frac{1}{N}\sum_{i=1}^{N} S_i^2\right) - \mu^2}$$
5. The median of the areas; i.e. the value M such that, if the areas are ordered in ascending order, half of them are larger than M, and the other half are smaller. If there is even number of values in the set then the median is usually defined as the average of the two elements in the middle. For example, the median of the set $\{1, 7, 3, 5, 3\}$ is 3, and the median of $\{1, 2, 6, 5\}$ is 3.5.
6. Whether the first (non-singular) and last (non-singular) circle in the set overlap, touch or are disjoint.

2 Programming Instructions, classes in your code and hints

The code will be structured in an object-oriented manner. Hence one class, `Circle`, will describe the circle. To break the description down further, it will use another class, `Point`, and a real number that defines the radius.

Just as in Project 0, the files provided on the course webpage have some predefined methods which are either complete, or have names and parameters. You **must** keep the names of all `public` objects and methods as they are in the templates. The files define the three basic classes for your project.

- `Point.java` which represents points $(x, y) \in \mathbb{R}^2$;
- `Circle.java` to represent a circle using a `Point` and a radius r;
- `Project1.java` in which you will use `Circle` and `Point` to process the data file.
- `Project1.data` contains the data set you are supposed to analyse. This file is rather large so for testing you may want to experiment with your own smaller data set, which may also allow you to test different aspects of your code.

  You can create such a file easily in any text editor (for example `nedit`). Each circle is defined on a single line in this file, and it contains three numbers: coordinates x_i, y_i, and the radius r_i.

Finally, there is the file `ReadFile.java`, which is **not** explicitly part of Project 1. It is a piece of example code which documents using the `MaInput` class to read in the data from a file, and find the maximum and minimum of that data. You **should not** submit it.

2.1 The `Point` class

The `Point` class will contain:

1. `private` variables `double X, Y`, that store the co-ordinates of the point represented by an instance of this class.
2. All of the standard methods that we identified in the previous exercises as necessary for usual classes of the similar type. In particular: constructors, setters, getters, converters, and the method `equals` which allows us to compare two `Point`s.

   In particular, the comparison between two points is important, and different from the usual mathematical notion of equality. Due to the way that floating point numbers are stored in computer memory, we never usually test whether two floating point numbers are equal; just whether they are 'close' to within an arbitrarily chosen tolerance. You should have seen examples of this in the week 13 lab notes.

   Given two points (x_1, y_1) and (x_2, y_2) we say that they are equal in our model if and only if $|x_1 - x_2| \leq \Delta$ and $|y_1 - y_2| \leq \Delta$. The quantity Δ is named `GEOMTOL` in the `Point` class and it is defined as `public`, `final` and `static`. In your computations the value of this variable should be 10^{-6}.

   Moreover, whenever you test for equality that involves two of the geometric objects in this project, you will allow for the tolerance Δ. For example to test that the distance between two points $d(A, B)$ is equal to a certain value a you would evaluate $|d(A, B) - a| \leq \Delta$.
3. Implement the method `distance` that computes the Euclidean distance between two points $x = (x_1, y_1)$ and $y = (x_2, y_2)$, which is defined as

$$d(x, y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

4. You can, and in fact should, also include a tester (as the method `main`) for the methods as usual. See week 14 lecture notes for more information on this.

2.2 The `Circle` class

While the methods in the class `Point` are rather standard, for the class which defines instances for circles you need to do more work.

1. You should start by defining two private variables: `Point A` and `double r` which store the centre and the radius of the circle.
2. You then have to define all standard method as above in the class `Point`, i.e., constructors, setters, getters and converters. Read carefully instructions in the file `Circle.java` as they explain the names of each of these methods, along with their expected behaviour.
3. The method `equals` that compares two circles. Obviously, two circles are equal if they have identical (within the tolerance Δ) centers and radii.
4. The core of this class are methods that actually implement functions needed for the circles. In particular, in this exercise you will need two methods: `area` and `overlap`.

   `area` simply computes and returns the area of the circle. `overlap` is slightly more complex. It accepts as a parameter another `Circle`, and checks to see whether it overlaps this one. It returns:

- 0 if the circles are disjoint – i.e. they do not touch at all;
- 1 if the circles overlap;
- 2 if the circles overlap at only a single point – i.e. they touch;
- 3 if the circles are identical.

Hint: If the centre points of the circles are C_1 and C_2 with radii r_1 and r_2, then they overlap if $d(C_1, C_2) < r_1 + r_2$. Two circles have a contact point if $d(C_1, C_2) = r_1 + r_2$. Remember that when you test the equality you will again allow for the tolerance Δ.

2.3 The `Project1` class

This class will perform the statistical analysis of the data and should produce the results you are asked to perform in the formulation. You have to design and implement the core method `results` which will calculate all of the quantities outlined in the formulation above. It takes the argument `fileName`, which contains the name of the file from which the data will be entered. An example how to read the data from the given file is provided in a file `ReadData.java`, which suggests how you can read data from a file until the end of the file using methods from the `MaInput` class.

Within `results`, you should not print anything to the screen with `System.out.println`. Instead, you should store the results in the predefined public variables at the top of the class:

- `int circleCounter`: The number of (non-singular) circles in the file.
- `int posFirstLast`: Whether the first and last circles in the file overlap, using the numbering scheme outlined in the `Circle` class above.
- `double maxArea`: The maximum area $S_{\max}$.
- `double minArea`: The minimum area $S_{\min}$.
- `double averageArea`: The average area μ.
- `double stdArea`: The standard deviation of the areas.
- `double medArea`: The median of the areas.

Remember not to change the names of any of these variables.

Hint: While most of the implementation is straightforward, one task requires a bit more algorithmic thinking. Let A_i with $0 \leq i < N$ be an increasing sequence ($A_{i+1} \geq A_i \; \forall \, i$) which represents a data set which is sorted. Then the median $M = A_{\frac{N-1}{2}}$ if N is odd or $M = \frac{1}{2}(A_{\frac{N}{2}-1} + A_{\frac{N}{2}})$ if N is even. Hence you will need to store areas of entered circles into an array and then sort the array. We will cover this topic in week 16, but you should try and figure this out for yourself first!

3 Submission

You should submit the files `Project1.java`, `Circle.java` and `Point.java` using Tabula. Before you submit, test that all your methods work properly (use the `main` method within each class) and that the `results` method computes correct results when tested on a smaller file of your own input data. Make sure to include `main` in the class `Project1` which should call the `results` method and print the results stored in the public variables of an instance of this method. You should **not** submit the data file `Project1.data` or the example code `ReadData.java`.