COMP20220 Programming II (Conversion)

Practical 7

Q1 Download the `Student`, `BirthDate`, and `TestStudent` classes from Moodle. Assume the goal is to make the `Student` class immutable.

Through the use of **packages** and **visibility modifiers** (i.e. do not delete any data fields or methods in any class), how can the `Student` class be made immutable?

Q2 Implement a class named `StopWatch` that contains:

- Data fields `startTime` and `endTime` with getter methods.
- A no-arg constructor that initialises `startTime` with the current time.
- A method named `start()` that resets the `startTime` to the current time.
- A method named `stop()` that sets the `endTime` to the current time.
- A method named `getElapsedTime()` that returns the elapsed time for the stopwatch in milliseconds.

Write a test program that measures the execution time of sorting 100,000 randomly generated integers (in the range 0 to 999, inclusive) using (a) the qucksort algorithm (use the `sort` method in the `Arrays` class) and (b) the parallel sort algorithm (use the `parallelSort` method in the `Arrays` class).

Hint: You can invoke `System.currentTimeMillis()` to return the current time. Refer to the Java API for details.

Q3 In an n-sided regular polygon, all sides have the same length and all angles have the same degree. Implement a class named `RegularPolygon` that contains:

- An `int` data field named `n` that defines the number of sides in the polygon (default value 3).
- A `double` data field named `length` that stores the length of each side (default value 1).
- A `double` data field named `x` that defines the x-coordinate of the polygon's center (default value 0).
- A `double` data field named `y` that defines the y-coordinate of the polygon's center (default value 0).
- A no-arg constructor that creates a regular polygon with default values.
- A constructor that creates a regular polygon with the specified number of sides and length of side, centered at (0, 0).
- A constructor that creates a regular polygon with the specified number of sides, length of side, and x- and y-coordinates.
- The accessor and mutator methods for all data fields.
- A method `getPerimeter()` that returns the perimeter of the polygon.
- A method `getArea()` that returns the area of the polygon. The formula for computing the area of a regular polygon is (where s denotes the length of the side) :

$$Area = \frac{n \times s^2}{4 \times \tan\left(\frac{\pi}{n}\right)}.$$

- A method `toString()` that returns a string representation of a polygon (i.e. returns the values of the data fields as a string).

Write a test program that creates an array of five `RegularPolygon` objects as follows:

- Polygon 1: number of sides = 3, side length = 10.0, center point = (2.5, 8.0)
- Polygon 2: number of sides = 6, side length = 4.0, center point = (0.0, 0.0)
- Polygon 3: number of sides = 3, side length = 1.0, center point = (0.0, 0.0)
- Polygon 4: number of sides = 10, side length = 5.0, center point = (0.0, 0.0)
- Polygon 5: number of sides = 4, side length = 6.0, center point = (3.0, 6.8)

For each object, display its string representation, perimeter and area. Also, find and display the string representation of the polygon in the array with (a) the smallest perimeter and (b) the largest area.

Q4 Using the `Circle` class (from Practical 6 solutions), write a test program that creates an array named `circles` of 10 `Circle` objects. Each circle should have a radius of 3 and the x- and y-coordinates of the centre point should be set to randomly generated numbers in the range [0, 5). Display the circles using the `toString()` method in class `Circle`.

Compute the pairwise distances (using the `getDistance(Circle another)` method in class `Circle`) between the center points of all `Circle` objects. Store these pairwise distances in 2-D array named `pwd` such that `pwd[i][j]` stores the distance between `circles[i]` and `circles[j]`.

Find the two circles which are furthest in terms of distance and display the string representation of these circles (using the `toString()` method in class `Circle`).

A sample run of the program is shown below.

```
Circles:
   0 | radius = 3.0, center point = (0.25, 2.48)
   1 | radius = 3.0, center point = (3.06, 0.66)
   2 | radius = 3.0, center point = (4.57, 1.85)
   3 | radius = 3.0, center point = (0.13, 3.06)
   4 | radius = 3.0, center point = (0.53, 1.18)
   5 | radius = 3.0, center point = (3.55, 1.09)
   6 | radius = 3.0, center point = (1.33, 3.93)
   7 | radius = 3.0, center point = (2.05, 0.63)
   8 | radius = 3.0, center point = (4.75, 1.66)
   9 | radius = 3.0, center point = (3.92, 4.68)

Pairwise distances between circles:
   0 |  0.00  3.35  4.37  0.59  1.33  3.58  1.81  2.58  4.57  4.28
   1 |  3.35  0.00  1.92  3.79  2.58  0.65  3.70  1.01  1.96  4.11
   2 |  4.37  1.92  0.00  4.60  4.10  1.27  3.85  2.80  0.26  2.90
   3 |  0.59  3.79  4.60  0.00  1.92  3.95  1.48  3.10  4.83  4.12
   4 |  1.33  2.58  4.10  1.92  0.00  3.02  2.86  1.62  4.25  4.87
   5 |  3.58  0.65  1.27  3.95  3.02  0.00  3.60  1.57  1.33  3.61
   6 |  1.81  3.70  3.85  1.48  2.86  3.60  0.00  3.38  4.10  2.70
   7 |  2.58  1.01  2.80  3.10  1.62  1.57  3.38  0.00  2.89  4.46
   8 |  4.57  1.96  0.26  4.83  4.25  1.33  4.10  2.89  0.00  3.13
   9 |  4.28  4.11  2.90  4.12  4.87  3.61  2.70  4.46  3.13  0.00

Furthest circles:
   4 | radius = 3.0, center point = (0.53, 1.18)
   9 | radius = 3.0, center point = (3.92, 4.68)
   (distance = 4.87)
```