CS2613: Programming Languages Laboratory (FR02A)
Lab #19 – Winter 2024

Language: Racket (#4)
# of Tasks: 2
Topics:

- Class Definition
- Object Instantiation
- Accessors and Mutators
- Methods

*All tasks are to be completed individually in line with the academic offense guidelines detailed on the syllabus and are **due before the end of the lab period** unless stated otherwise.*

Task #1

Task Style: Peer-Evaluation
Submission Method: Hand-in Grading Form

Description:

You have been given a fake submission for a programming question. The student was asked to complete the following:

> Create a class for Point objects that has an x and y value as instance variables. The class should have accessors and mutators to get and set the x and y values. The class should also have a method that returns the distance between the two points using the Euclidean distance formula.

This student struggled to complete the programming question. Complete the Peer-Evaluation form for (handed-out) the student and try to give the student good feedback on how to improve their code. You may need to correct the code yourself to give appropriate feedback. Just assigning the student a grade will not be enough to get full points on this task.

Resources:

- The Racket **Guide**
  - 6.6
- The Racket **Reference**
  - 6.1 – 6.3 (Basic OOP only – think about the content of CS1073 without inheritance)

Task #2

Task Style: Programming w/ Self-Assessment
Submission Method: Complete self-assessment steps below and let the instructor or teaching assistant know before you leave.

Description:

Create a 3D Point class for points that have x, y, and z values. Mutators and accessors should be provided for all instance variables. A method must be provided to calculate and return the Euclidean distance between the two points.

An additional method should be added to check if two points are within a certain distance apart. For example, the following should evaluate to #t because the distance is ~10.2 which is less than or equal to 11

```
(define p1 (make-object point% 9 3 6))
(define p2 (make-object point% -1 2 8))

(send p1 distance-range p2 11)
```

However, the following should evaluate to #f because ~10.2 which is NOT less than or equal to 10

```
(define p1 (make-object point% 9 3 6))
(define p2 (make-object point% -1 2 8))

(send p1 distance-range p2 10)
```

Finally, implement a method called triangle-perimeter. This method should take two other points that would make up a triangle. This method should calculate the perimeter of the triangle and return it.

To assess, check all methods to make sure they work as intended. Use the calculator below to make sure the output distance is correct.
https://www.omnicalculator.com/math/distance

Resources:

- Resources from Task #1