Отчет по лабораторной работе № 5
по курсу «Функциональное программирование»

Студент группы М8О-307-19 МАИ *Тимофеев Алексей Владимирович*, №21 по списку
Контакты: TimofeevAV8f@yandex.ru
Работа выполнена: 29.05.2022

Преподаватель: Иванов Дмитрий Анатольевич, доц. каф. 806
Отчет сдан:
Итоговая оценка:
Подпись преподавателя:

1. Тема работы

Обобщённые функции, методы и классы объектов.

2. Цель работы

Цель работы: научиться определять простейшие классы, порождать экземпляры классов, считывать и изменять значения слотов, научиться определять обобщённые функции и методы.

3. Задание (Вариант 5.25)

Дан экземпляр класса triangle, причем все вершины треугольника могут быть заданы как декартовыми координатами (экземплярами класса cart), так и полярными (экземплярами класса polar).

Задание: Определить обычную функцию медиана, возвращающую объект-отрезок (экземпляр класса line), являющийся медианой первого угла vertex1. Концы результирующего отрезка могут быть получены либо в декартовых, либо в полярных координатах.

```
(setq tri (make-instance 'triangle
            :1 (make-instance 'cart-или-polar ...)
            :2 (make-instance 'cart-или-polar ...)
            :3 (make-instance 'cart-или-polar ...)))
(медиана tri) => [ОТРЕЗОК ...]
```

4. Оборудование студента

Процессор Intel Core i5-10600K @ 4.10GHz, память: 16 Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Ubuntu 20.04.4 LTS, комилятор GNU CLISP 2.49.92, текстовый редактор VS Code

6. Идея, метод, алгоритм

Я взял с сайта нашего курса объявление классов cart, polar, line, triangle, а также метод для печати каждого класса print-object. Далее написал функцию median работающую только с cart. Так как треугольник в cart обрабатывать я уже умею, было решено сделать функцию, которая преобразует декартовы в полярные координаты.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defclass cart ()
  ((x :initarg :x :reader cart-x)
   (y :initarg :y :reader cart-y)))

(defmethod print-object ((c cart) stream)
  (format stream "[CART x ~d y ~d]"
    (cart-x c) (cart-y c)))

(defclass polar ()
  ((radius :initarg :radius :accessor radius)
    (angle  :initarg :angle  :accessor angle)))

(defmethod print-object ((p polar) stream)
  (format stream "[POLAR radius ~d angle ~d]"
    (radius p) (angle p)))

(defmethod radius ((c cart))
  (sqrt (+ (* (cart-x c) (cart-x c))
           (* (cart-y c) (cart-y c)))))

(defmethod angle ((c cart))
  (atan (cart-y c) (cart-x c)))

(defmethod cart-x ((p polar))
  (* (cos (angle p)) (radius p)))

(defmethod cart-y ((p polar))
  (* (sin (angle p)) (radius p)))
```

```
(defgeneric to-cart (arg)
  (:method ((c cart)) c)
  (:method ((p polar))
    (make-instance 'cart :x (cart-x p) :y (cart-y p))))

(defmethod add ((c1 cart) (c2 cart))
    (make-instance 'cart
    :x (+ (cart-x c1) (cart-x c2))
        :y (+ (cart-y c1) (cart-y c2))))

(defmethod add ((p1 polar) (p2 polar))
    (make-instance 'cart
    :x (+ (cart-x p1) (cart-x p2))
        :y (+ (cart-y p1) (cart-y p2))))

(defmethod make-half ((c1 cart))
  (make-instance 'cart
    :x (/ (cart-x c1) 2)
    :y (/ (cart-y c1) 2)))

(defclass line ()
  ((start :initarg :start :accessor line-start)
   (end   :initarg :end   :accessor line-end)))

(defmethod print-object ((lin line) stream)
  (format stream "[SEGMENT ~s ~s]"
    (line-start lin) (line-end lin)))

(defclass triangle ()
  ((vertex1 :initarg :1 :reader vertex1)
   (vertex2 :initarg :2 :reader vertex2)
   (vertex3 :initarg :3 :reader vertex3)))

(defmethod print-object ((tri triangle) stream)
  (format stream "[TRIANGLE ~s ~s ~s]"
    (vertex1 tri) (vertex2 tri) (vertex3 tri)))

(defun median (tri)
  (make-instance 'line
           :start (to-cart (vertex1 tri))
           :end (make-half (add (vertex2 tri) (vertex3 tri)))))
```

8.2. Результаты работы

```
[1]> (load "lab5.lisp")
;; Loading file lab5.lisp ...
;; Loaded file lab5.lisp
#P"/mnt/d/education/education/FP/Mylab/lab5/lab5.lisp"
[2]> (setq triangleCartesian1 (make-instance 'triangle :1
   (make-instance 'cart :x 2 :y 8)
:2 (make-instance 'cart :x 2 :y 3)
:3 (make-instance 'cart :x 6 :y 6)))
[TRIANGLE [CART x 2 y 8] [CART x 2 y 3] [CART x 6 y 6]]
[3]> (setq trianglePolar2 (make-instance 'triangle
           :1 (make-instance 'polar :radius (radius (vertex1
   triangleCartesian1)) :angle (angle (vertex1
   triangleCartesian1)))
           :2 (make-instance 'polar :radius (radius (vertex2
   triangleCartesian1)) :angle (angle (vertex2
   triangleCartesian1)))
           :3 (make-instance 'polar :radius (radius (vertex3
   triangleCartesian1)) :angle (angle (vertex3
   triangleCartesian1)))))
[TRIANGLE [POLAR radius 8.246211 angle 1.3258177] [POLAR radius
   3.6055512 angle 0.9827938] [POLAR radius 8.485281 angle
   0.7853981]]
[4]> (median triangleCartesian1)
[SEGMENT [CART x 2 y 8] [CART x 4 y 9/2]]
[5]> (median trianglePolar2)
[SEGMENT [CART x 1.9999996 y 7.9999995] [CART x 4.0 y 4.5]]
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Программа работает с небольшой погрешностью в методе расчета полярных координат.
Возможно это из-за внутренней реализации тригонометрических функций.

11. Выводы

При выполнении лабораторной работы № 5 я научился определять простейшие классы, порождать экземпляры классов, считывать и изменять значения слотов,а также научился определять обобщённые функции и методы.