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

Студент группы 8О-308 Петросян Виктор, № по списку 15.

Контакты: viko20000@mail.ru
Работа выполнена: 13.05.2020
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

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 ...)))
```

(median tri) => [ОТРЕЗОК ...]

4. Оборудование студента

Ноутбук Samsung NP300E5c-S0URU 15.6, процессор Intel® Core™ i5-2410M CPU 2.30GHz, память 6ГБ, 64-разрядная система.

5. Программное обеспечение

Linux 4.15.0-96-generic #97-Ubuntu SMP, использовал sbcl(для запуска .lisp файлов) + VSCode(для редактирования кода)

6. Идея, метод, алгоритм

Взял с сайта преподавателя объявление классов cart, polar, line, triangle, а также метод для печати каждого класса print-object. Решал задачу по мере поступления проблем. Сначала написал функцию median работающую только с cart. Это оказалось легко. После чего пришлось подумать что делать, когда треугольник задаётся в полярных координатах. Решил эффективно использовать то, что у меня уже было реализовано, поэтому написал функцию, которая преобразует полярные координаты в декартовы.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

Программа

```
(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 "[ОТРЕЗОК ~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 "[ТРЕУГ ~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)))))

(defvar triangleCartesian1 (make-instance 'triangle
    :1 (make-instance 'cart :x 0 :y 5)
    :2 (make-instance 'cart :x 0 :y 0)
    :3 (make-instance 'cart :x 4 :y 0)))
(print (median triangleCartesian1))

(defvar trianglePolar1 (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)))))
(print (median trianglePolar1))
(princ #\Newline)
```

Результаты

```
[ОТРЕЗОК [CART x 0 y 5] [CART x 2 y 0]]
[ОТРЕЗОК [CART x -2.1855695e-7 y 5.0] [CART x 2.0 y 0.0]]
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Считаю, что программа работает верно и ответы 0 и -0.00000021855695 отличаются только из-за точности вычислений. Возможно, дело именно в тригонометрических функциях.

11. Выводы

В данной лабораторной работе, я изучил, как строятся классы в common lisp, как происходит обращение к их внутренним переменным. Так же я узнал, что такое обобщенные методы и научился их создавать и ими пользоваться. Программа работает правильно и прошла все тесты.