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

Студент группы М8О-307 МАИ *Днепров Иван*, №10 по списку
Контакты: vanya.dneprov@gmail.com
Работа выполнена: 05.05.2020

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 №5.9)

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 прямоугольный-р, возвращающую T, если треугольник с такими вершинами является прямоугольным.

4. Оборудование студента

MacBook (13-inch, Mid 2010), процессор 2,4 GHz Intel Core 2 Duo, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение

Mac OS 10.13.6, компилятор clisp, текстовый редактор Sublime Text 3.

6. Идея, метод, алгоритм

Создал класс triangle с тремя параметрами – точками. Точки могут принимать значения класса cart или polar, описанных на сайте. Для проверки прямоугольности тре-

угольника я написал функцию прямоугольный-р, в которой я в начале вычисляю длины всех сторон треугольника при помощи метода calcLen. Этот метод работает с экземплярами класса cart и polar в качестве точек. После того, как я получил длины всех сторон я проверяю теорему пифагора для всех комбинаций сторон, и если хоть одна комбинация соответствует теореме пифагора, то возвращаю T, в противном случае Nil. Теорему пифагора я проверяю с точность 0.0001, так как присутствуют нецелочисленные вычисления.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

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

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

(defclass triangle ()
 ((dot1 :initarg :dot1 :reader dot1)
   (dot2 :initarg :dot2 :reader dot2)
   (dot3 :initarg :dot3 :reader dot3)))



(defmethod print-object ((triang triangle) stream)
  (format stream "[TRIANG ~s ~s ~s]"
    (dot1 triang) (dot2 triang) (dot3 triang)))

(defmethod x ((p polar))
  (* (radius p) (cos (angle p))))
```

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

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

(defun sqare (x)
  (* x x))

(defmethod calcDist ((c1 cart) (c2 cart))
  (sqrt (+ (sqare (abs (- (x c1) (x c2))) ) (sqare (abs (- (y c1)
   (y c2)))))))

(defmethod calcLen ((p1 polar) (p2 polar))
  (setq dot1 (polarToCart p1))
  (setq dot2 (polarToCart p2))
  (calcDist dot1 dot2))

(defmethod calcLen ((c1 cart) (c2 cart))
  (calcDist c1 c2))

(defmethod calcLen ((c1 cart) (p2 polar))
  (setq dot2 (polarToCart p2))
  (calcDist c1 dot2))

(defmethod calcLen ((p1 polar) (c2 cart))
  (setq dot1 (polarToCart p1))
  (calcDist dot1 c2))

(defun прямоугольный-p (triang)
  (setq a (calcLen (dot1 triang) (dot2 triang)))
  (setq b (calcLen (dot2 triang) (dot3 triang)))
  (setq c (calcLen (dot3 triang) (dot1 triang)))
  (cond
    ((< (abs (- (sqare a) (+ (sqare b) (sqare c)))) 0.0001)  T)
    ((< (abs (- (sqare b) (+ (sqare c) (sqare a)))) 0.0001)  T)
    ((< (abs (- (sqare c) (+ (sqare a) (sqare b)))) 0.0001)  T)))
```

```
(setq triang0 (make-instance 'triangle
  :dot1 (make-instance 'cart :x 1 :y 0)
  :dot2 (make-instance 'cart :x 0 :y 1)
  :dot3 (make-instance 'cart :x 0 :y 0)))

(print прямоугольный(-p triang0))


(setq triang1 (make-instance 'triangle
  :dot1 (make-instance 'polar :radius 5 :angle 0)
  :dot2 (make-instance 'cart :x 0 :y 5)
  :dot3 (make-instance 'polar :radius 0 :angle 0)))

(print прямоугольный(-p triang1))

(setq triang2 (make-instance 'triangle
  :dot1 (make-instance 'polar :radius 5 :angle 0)
  :dot2 (make-instance 'polar :radius 4 :angle 1)
  :dot3 (make-instance 'polar :radius 0 :angle 0)))

(print прямоугольный(-p triang2))

(setq triang3 (make-instance 'triangle
  :dot1 (make-instance 'cart :x 100 :y 0)
  :dot2 (make-instance 'cart :x 7 :y 6)
  :dot3 (make-instance 'polar :radius 1 :angle (/ pi 4))))

(print прямоугольный(-p triang3))
```

8.2. Результаты работы

```
   T
T
Nil
NIl
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

До этой лабораторной работы я даже не продполагал, что в Lisp можно создавать классы. Такую возможность считаю замечательной. Но всё-таки по-моему на Lisp сложно присать что-то большое, потому что код помучается трудно читаемым и воспринимаемым. Но это субъективное мненик, конечно. Для того, чтобы было проще разобраться в том, что я написал, я старался делить большие функции на много маленьких, и в итоге разобраться в написанном довольно просто.

11. Выводы

Я научилась работать с простейшими классами, порождать экземпляры классов, производить различные действия над ними, вспомнил геометрию в школе.