АННОТАЦИЯ

Документ содержит реверс инжениринг примеров статьи "Разнорукое программирование"А.И.Легалова в терминах формальной модели объектно - ориентированного программирования.

Содержание

1 ВВЕДЕНИЕ

Предложенная в работе [2] формальная модель объектно ориентированного программирования будет использоваться для записи примеров с геометрическими фигурами из статьи [1] . Своего рода реверс-инжениринг. Для записи схем (кода спецификации) будет использоваться система записи [3] .

2 Пример oop_examp1.zip

Пример написан на языке C++. Находится в файле oop_examp1.zip на странице http://www.softcraft.ru/paradigm/dhp/dhp06.shtml. Количество строчек в формальной модели oop

container.rsl:27
rectangle.rsl:18
shape.rsl:19
triangle.rsl:23

Количество строчек на языке C++

container_Area.cpp:19
container_Constr.cpp:22
container_In.cpp:38
container_Out.cpp:20
main.cpp:20
rectangle_Area.cpp:14
rectangle_Constr.cpp:12
rectangle_In.cpp:17
rectangle_Out.cpp:16
shape_In.cpp:41
triangle_Area.cpp:16
triangle_Constr.cpp:12
triangle_In.cpp:17
triangle_Out.cpp:18

2.1 Shape

Задан класс shape и вспомагательные функции. Например, ввод - вывод целого числа.

---- File:shape.rsl

```
SHAPE =
 class
  type
    shape = (,{In},{Out, Area}),     -- класс родитель для будущих фигур

  value
     null : shape                -- несозданный объект

    ,Area : shape -> Real      -- функция для вычисления площади фигуры
     , In : shape -> shape
     , Out: shape  -> Unit
                                        -- вспомагательные функции
     , In : Int -> Int          -- ввод целого
     , Out: Text -> Unit         -- вывод текста
     , Out: Int -> Unit         -- вывод целого
     , Real: Int -> Real         -- преобразование целое в вещественное

     , sqrt : Real  -~-> Real    -- извлечение корня
  axiom
      all x: Real => exist s: Real :-
       sqrt (x) = s /\ s*s = x /\ s >= 0.

 end
```

---- End Of File:shape.rsl

2.2 Container

Описание списка фигур.

---- File:container.rsl

```
CONTAINER extend SHAPE with =
 class
  type
    container = (shape-list,                    --
          {In, container, Clear}, {len, Out, Area})

  value
     null : shape
   ,container = -\ x: shape-list :- <..>  -- -\ лямбда выражение. вернуть пустой список <..>
     ,Clear = -\ x : shape-list :-  container (x)
     ,in : shape-list -> shape-list
      in (sl) is
        let s = In(null) in
          if s != null then
            s ^ in(sl)                      --  если удалось ввести фигуру
       else                    -- то положить ее в голову списка,сделанного позже
            <..>                            --  иначе пустой список
          end
        end

     ,len = -\ x: shape-list :- len x        -- длина списка
     ,out : shape-list -> Unit
    out (sl) is all s in sl :- Out(s)    -- для всех s в списке sl выполнить  Out(s)

     ,Area = -\ x: shape-list :-
       if len x > 0                         --
      then Area (tl x) +  Area (hd x)    -- сложить сумму площадей хвоста (tl x) с площадью фигуры
        else 0. end                         --  в голове списка (hd x)


 end

---- End Of File:container.rsl
```

2.3 Triangle

Наследование треугольника из фигуры.

```
---- File:triangle.rsl
```

```
TRIANGLE extend SHAPE with =
 class
  type
   triangle = shape >< ( (a: Int , b:Int , c:Int ),{triangle},),    -- класс родитель для фигуры

  value

       triangle = -\ t: triangle, (a,b,c): Int  >< Int >< Int :-
         if a > 0 /\ b > 0 /\ c > 0 then (a,b,c)
           else null
           end
     , Area = -\ t: triabgle :-
         let p = a(t) + b(t) + c(t) / 2. in
           sqrt ( p * (p- a(t)) * (p-b(t) * (p-c(t)))
         end

     , In = -\ t : triangle :-  triangle (In(0), In(0), In(0))
     , Out = -\ t: triangle :-
          Out("It is Triangle: a = ");
          Out(a(t));
          Out(", b = ");
          Out(b(t));
          Out(", c = ");
          Out(c(t));
 end

---- End Of File:triangle.rsl
```

2.4 Rectangle

Наследование четырезугольника из фигуры.

```
---- File:rectangle.rsl


RECTANGLE extend SHAPE with =
 class
  type
   rectangle = shape >< ( (x: Int, y:Int), {rectangle},),    -- класс родитель для фигуры
```

```
value

    rectangle = -\ t: rectangle, (_x,_y):  Int >< Int :-
      if _x > 0 /\ _y > 0  then (_x,_y)
        else null
        end
  , Area = -\ t: rectangle :-  Real(x(t)  * y(t))

  , In = -\ t : rectangle :-  rectangle (In(0), In(0))
  , Out = -\ t: rectangle :-
      Out("It is rectangle: x = ");
      Out(x(t));
      Out(", y = ");
      Out(y(t));


end

---- End Of File:rectangle.rsl
```

Предметный указатель

ССЫЛКИ

[1] А.И. Легалов. Разнорукое программирование, 2001. http://www.softcraft.ru/paradigm/dhp/index.shtml. 3

[2] А.Г. Пискунов. Формализация парадигмы обьектно-ориентированного программирования: критика определения Гради Буча, 2007. http://i.com.ua/~agp1/ru/oopFormalizm.html. 3

[3] Chris George. Introduction to RAISE. The RAISE Development Method, 2001. http://users.iptelecom.net.ua/~agp1/arts/report249.pdf. 3