1 00:00:04

Функция — это

экземпляр типа FunctionN, где N — количество аргументов

блок из нескольких выражений

альтернативная запись метода с параметрами

базовый тип JVM

Задание 2 00:00:09

Отметьте верные утверждения.

функцию нельзя вызвать дважды

функция должна иметь как минимум один аргумент

функция имеет собственные методы

функция может являться значением переменной

функция может быть передана в качестве параметра в метод

функция может иметь побочные эффекты

функция имеет тип возвращаемого значения

Задание 3 00:04:46

Заполните пропуски в приведенных сценариях подходящими функциональными методами.

Сценарий 1 filter

Seq("one", "two", "three", "four", "five")

.\_\_\_\_\_\_(\_.startsWith("t"))

res0: Seq[String] = List(two, three)

Сценарий 2 map

Seq("one", "two", "three", "four", "five")

.\_\_\_\_\_\_(s => s.substring(0, 1))

res0: Seq[String] = List(o, t, t, f, f)

Сценарий 3 map

Seq("one two three", "four five")

strings

{{ flatMap }}(\_.split(" "))

.\_\_\_\_\_\_(\_.length)

res1: Seq[Int] = List(3, 3, 5, 4, 4)

Задание 4 00:00:00

Выберите тип функции, соответствующий сокращенной записи (String, String) => Double:

Function3[String, String, Double]

Function2[Double, String, String]

Function3[Double, String, String]

Function[Double, String, String]

Function1[Double, String, String]

Function1[String, String, Double]

Function[String, String, Double]

Function0[Double, String, String]

Function0[String, String, Double]

Function2[String, String, Double]

Задание 5 00:00:00

Рассмотрим определение метода map в классе Fraction:

**class** **Fraction**(val numerator: Int,

val denominator: Int) {

**def** map(numF: Int => Int, denF: Int => Int): Fraction =

**new** Fraction(numF(numerator), denF(denominator))

}

Отметьте синтаксически корректные способы вызова метода map (f— это экземпляр класса Fraction).

f.map({n => n + 1}, {d => d - 1})

f.map(\_ + 1, \_ - 1)

f.map(\_ + 1, -1)

f.map(+1, -1)

f.map(n => n + 1, d => d - 1)

f.map(n => 1, d => -1)

Попробуйте проработать нашу геометрическую модель дальше. Вот несколько идей.

### Треугольники

Из курса геометрии Вы знаете о трех способах задания треугольника:

* по двум сторонам и углу между ними;
* по двум углам и стороне;
* по трем сторонам.

Спроектируйте все способы задания треугольника:

* сделайте класс Triangle интерфейсом, сделайте абстрактными все его стороны и углы;
* добавьте конкретные финальные классы TwoSidesAngleTriangle, TwoAnglesSideTriangle и ThreeSidesTriangle с вычислением недостающих сведений.

### Четырехугольники

Введите понятие Quadrilateral (четырехугольник) и соответствующие ему подтипы:

* трапеция Trapezium;
* параллелограмм Parallelogram;
* ромб Rhombus.

Перепишите соответствующим образом классы Rectangle и Square, учитывая, что:

* прямоугольник — это частный случай параллелограмма;
* квадрат — это частный случай прямоугольника и ромба.

abstract class Shape {

def perimeter: Double

def area: Double

}

class Circle(val radius: Double) extends Shape {

def diameter = 2 \* radius

def perimeter = math.Pi \* diameter

def area = math.Pi \* radius \* radius

}

class Rectangle(val ab: Double,

val ac: Double)

extends Shape {

def cd = ab

def bd = ac

def bc = math.sqrt(ab \* ab + ac \* ac)

def ad = bc

def perimeter = 2 \* (ab + ac)

def area = ab \* ac

}

trait Triangle extends Shape {

def a: Double

def b: Double

def c: Double

def ab: Double

def ac: Double

def bc: Double

}

final abstract class TwoSidesAngleTriangle extends Shape

with Triangle {

def bc = math.sqrt(

(ab \* ab) + (ac \* ac) - 2 \* ab \* ac \* math.cos(a))

def b = math.atan(ac / ab)

def c = math.atan(ab / ac)

def perimeter = ab + ac + bc

def area = ab \* ac \* math.sin(a) / 2

}

final abstract class TwoAnglesSideTriangle extends Shape

with Triangle {

def c = 180-a-b

def bc = ab\*(math.sin(a)/math.sin(c))

def ac = ab\*(math.sin(b)/math.sin(c))

def perimeter = ab + ac + bc

def area = ab \* ac \* math.sin(a) / 2

}

final abstract class ThreeSidesTriangle extends Shape

with Triangle {

def a=math.acos((ab \* ab) + (ac \* ac) - (bc \* bc))/(2\*ab\*ac)

def b=math.acos((ab \* ab) + (bc \* bc) - (ac \* ac))/(2\*ab\*bc)

def c=180-(a+b)

def perimeter = ab + ac + bc

def area = ab \* ac \* math.sin(a) / 2

}