Федеральное агентство связи

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической Кибернетики и Информационных Технологий

Отчет по лабораторной работе

по предмету «Функциональное программирование»

на тему:

«Язык программирования «Scala» ЛР 1»

Выполнил: студент группы

БВТ1802

Дворянинов Павел Владимирович

Руководитель:

Мосева Марина Сергеевна

Москва 2020

Выполнение

File Classes.scala

```
package exercise1
/*
 * a) Создать класс Animal, который имеет следующие поля:
 *      - name: String (название)
 *      - species: String (вид)
 *      - food: String
 *
 * Синтаксис: class MyClass(val publicField: Int, privateField: String) {
 * // остальные поля и методы
 * }
 *
 * b) Создайте объект-
 * компаньон для класса Animal и добавьте следующие сущности как поля:
 *      - cat, mammal, meat
 *      - parrot, bird, vegetables
 *      - goldfish, fish, plants
 *
 *    Синтаксис: object MyClass {
 *               // статические поля и методы
 *             }
 *
 * c) Добавьте следующие метод в Animals:
 *      def eats(food: String): Boolean
 *
 *     который проверяет ест ли животное определенную пищу
 *
 * d) Переопределите ваш класс Animal как трейт и создайте объекты класса-
 * образца для Mammals, Birds и Fishs.
 *    Вам все еще нужно поле `species`?
 *
 * e) Добавьте следующие функции в объект-компаньон Animal:
 *      def knownAnimal(name: String): Boolean  // true если это имя одного из
 * трех животных из (b)
 *      def apply(name: String): Option[Animal] // возвращает одно из трех
 * животных в соответствии с именем (Some) или ничего (None), см. ниже
 *
 * f) Создайте трейт Food со следующими классами-образцами:
 *      - Meat
 *      - Vegetables
 *      - Plants
 *   и добавьте это в определение Animal. Так же добавьте объект-
 * компаньон с методом apply():
 *      def apply(food: String): Option[Food]
 */
```

```scala
sealed trait Animal {
  val name: String
  val food: String
  def eats(food: String): Boolean = return this.food.equals(food)
}

case class Mammals(name: String, food: String) extends Animal
case class Birds(name: String, food: String) extends Animal
case class Fishs(name: String, food: String) extends Animal

object Animal {

  sealed trait Food

  case object Meat extends Food
  case object Vegetables extends Food
  case object Plants extends Food

  val cat = Mammals("cat", "meat")
  val parrot = Birds("parrot", "vegetables")
  val goldfish = Fishs("goldfish", "seaweed")

  def knownAnimal(name: String): Boolean =
    name.equals(cat.name) || name.equals(parrot.name) ||
    name.equals(goldfish.name)

  def apply(name: String): Option[Animal] = {
    name match {
      case cat.name => Some(cat)
      case parrot.name => Some(parrot)
      case goldfish.name => Some(goldfish)
      case other => None
    }
  }
}

object program extends App {
  println(Animal("cat").get.eats("meat"))
  println(Animal("parrot").get.eats("seaweed"))
}
```

File Functions.scala

```scala
/*
 * Напишите отдельные функции, решающие поставленную задачу.
 *
 * Синтаксис:
 * // метод
 * def myFunction(param0: Int, param1: String): Double = // мело
 *
 * // значение
 * val myFunction: (Int, String) => Double (param0, param1) => // мело
 */
object Functions extends App {
  /*
   * Напишите функцию, которая рассчитывает площадь окружности r^2 * Math.PI
   */
  def CircleArea(r: Double): Double = r * r * Math.PI
  /*
   * Примените вашу функцию из пункта здесь, не изменяя сигнатуру.
   */
  def testCircle(r: Double): Double = CircleArea(r)

  /*
   * Напишите карированную функцию которая рассчитывает площадь прямоугольника
   * a * b.
   */
  def RectangeAreaCur(a: Double)(b:Double) = a * b
  /*
   * Примените вашу функцию из пукта здесь, не изменяя сигнатуру.
   */
  def testRectangleCur(a: Double, b: Double): Double = RectangeAreaCur(a)(b

  /*
   * Напишите не карированную функцию для расчета площади прямоугольника.
   */
  def RectangleArea(a: Double, b:Double): Double = a * b
  /*
   * Примените вашу функцию из пункта здесь, не изменяя сигнатуру.
   */
  def testRectangleUc(a: Double, b: Double): Double = RectangleArea(a, b)

  println("testCircle:\t\t" + testCircle(31))
  println("testRectangleCur:\t" + testRectangleCur(10, 10))
  println("testRectangleUc:\t" + testRectangleUc(10, 10))
}
```

File HiOrder.scala

```scala
/*
 * Напишите ваши решения в виде функций.
 */
object HigherOrder extends App {
  val plus: (Int, Int) => Int = _ + _
  val multiply: (Int, Int) => Int = _ * _
  /*
   * Напишите функцию, которая принимает `f: (Int, Int) => Int`, параменты
   *`a` и `b`
   * и коэффициент умножения `n` и возвращает n * f(a, b). Назовите `nTimes`.
   */
  def nTimes(f: (Int, Int) => Int, a: Int, b: Int, n: Int): Int = n * f(a, b)
  /*
   * Примените вашу функцию (a) здесь, не изменяйте сигнатуру.
   */
  def testNTimes(f: (Int, Int) => Int, a: Int, b: Int, n: Int): Int = nTimes(f,
  a, b, n)
  /*
   * Напишите анонимную функцию, функцию без идентификатора ((a, b) => ???)
   * для `nTimes` которая
   * выполняет следующее: if (a > b) a else b
   */
  def testAnonymousNTimes(a: Int, b: Int, n: Int): Int = nTimes((a: Int,b: Int) =
  > { if (a > b) a else b },  a: Int, b: Int, n: Int)

  println("testNTimes:\t" + testNTimes(plus,  10, 20, 30))
  println("testAnonymousNTimes:\t" + testAnonymousNTimes(20, 30, 40))
}
```

File Patterns.scala

```scala
/*
 * Напишите решение в виде функции.
 *
 * Синтаксис:
 *   val a: Int = ???
 *
 *   a match {
 *     case 0 => true
 *     case _ => false
 *   }
 */
object PatternMatching extends App {
    sealed trait Hand
    case object Rock    extends Hand
    case object Paper   extends Hand
    case object Scissor extends Hand
```

```scala
  sealed trait Result
  case object Win  extends Result
  case object Lose extends Result
  case object Draw extends Result

  sealed trait Food
  case object Meat       extends Food
  case object Vegetables extends Food
  case object Plants     extends Food

  sealed trait Animal {
  val name: String
  var food: Food
}
  case class Mammal(name: String, var food: Food, weight: Int) extends Animal
  case class Fish(name: String, var food: Food)                extends Animal
  case class Bird(name: String, var food: Food)                extends Animal
/*
 * Напишите функцию, которая ставит в соответствие числу строку слудеющим
 * образом:
 * Если:
 *     1 => "it is one"
 *     2 => "it is two"
 *     3 => "it is three"
 *     иначе => "what's that"
 */
def intToString(value: Int): String =
  value match {
    case 1 => "it is one"
    case 2 => "it is two"
    case 3 => "it is three"
    case other => "what's that"
  }
/*
 * Примените вашу функцию из пункта (a) здесь, не изменяя сигнатуру.
 */
def testIntToString(value: Int): String = intToString(value)
/*
 * Напишите функцию которая возвращает true если переменная `value` принимает
 * значение:
 * "max" или "Max
 * "moritz" или "Moritz"
 */
def isMaxAndMoritz(value: String): Boolean =
  value match {
    case "max" | "Max" | "moritz" | "Moritz"=> true
    case other => false
  }
```

```scala
/*
 *примените функции из пункта здесь, не изменяя сигнатуру
 */
def testIsMaxAndMoritz(value: String): Boolean = isMaxAndMoritz(value)
/*
 * Напишите функцию проверки является ли `value` четным
 */
def isEven(value: Int): Boolean =
  value % 2 match {
  case 0 => true
  case 1 => false
  }
/*
 * Примените функции из пункта здесь, не изменяя сигнатуру
 */
def testIsEven(value: Int): Boolean = isEven(value)
/*
 * Напишите функцию, моделирующую игру в Камень ножницы бумага
 * 1. камень побеждает ножницы
 * 2. ножницы побеждают бумагу
 * 3. бумага побеждает камень
 * Выиграет ли игрок `a`?
 */
def winsA(a: Hand, b: Hand): Result =
  a match {
    case Rock => b match {
      case Rock => Draw
      case Paper => Lose
      case Scissor => Win
    }
    case Paper => b match {
      case Rock => Win
      case Paper => Draw
      case Scissor => Lose
    }
    case Scissor => b match {
      case Rock => Lose
      case Paper => Win
      case Scissor => Draw
    }
  }
/*
 * Примените вашу функцию из пункта здесь, не изменяя сигнатуру.
 */
def testWinsA(a: Hand, b: Hand): Result = winsA(a, b)
```

```
  /*
   * Примечание: используйте определение Animals
   * Верните вес (weight: Int) объекта Mammal, иначе верните -1.
   */
  def extractMammalWeight(animal: Animal): Int =
    animal match {
      case mammal: Mammal => mammal.weight
      case other => -1
    }
  /*
   * Примените вашу функцию из пункта здесь, не изменяя сигнатуру.
   */
  def testExtractMammalWeight(animal: Animal): Int = extractMammalWeight(animal)
  /*
   * Измените поле еда объектов классов Fishes и Birds на Plants, класс Mammels
   * оставьте неизмененным.
   */
  def updateFood(animal: Animal): Animal =
    animal match {
      case fish: Fish => fish.food = Plants; fish
      case bird: Bird => bird.food = Plants; bird
      case other => animal
    }
  /*
   * Примените вашу функцию из пункта здесь, не изменяя сигнатуру.
   */
  def testUpdateFood(animal: Animal): Animal = updateFood(animal)

  println("testIntToString:\t\t" + testIntToString(1))
  println("testIsMaxAndMoritz:\t\t" + testIsMaxAndMoritz("max"))
  println("testIsEven:\t\t\t" + testIsEven(5))
  println("testWinsA:\t\t\t" + testWinsA(Paper, Rock))
  println("testExtractMammalWeight:\t" + testExtractMammalWeight(Mammal("cat", Me
  at, 5)))
  println("testUpdeteFood:\t\t\t" +testUpdateFood(Bird("parrot", Vegetables)))
}
```

Результат работы программы

File Classes.scala

```
D:\4 семестр\(экзамен) ФП>scala LAB_1.scala
true
false
```

File Functions.scala

```
D:\4 семестр\(экзамен) ФП>scala LAB_1.scala
testCircle:             3019.0705400997913
testRectangleCur:       100.0
testRectangleUc:        100.0
```

File HiOrder.scala

```
D:\4 семестр\(экзамен) ФП>scala LAB_1.scala
testNTimes:     900
testAnonymousNTimes:    1200
```

File Patterns.scala

```
D:\4 семестр\(экзамен) ФП>scala LAB_1.scala
testIntToString:                it is one
testIsMaxAndMoritz:             true
testIsEven:                     false
testWinsA:                      Win
testExtractMammalWeight:        5
testUpdeteFood:                 Bird(parrot,Plants)
```