Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**
МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших данных в системах поддержки принятия решений**

О Т Ч Е Т
по лабораторной работе № 3

Название: Классы, наследование, полиморфизм

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-23М _____ М.А. Гейне
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ П.В. Степанов
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

ЗАДАНИЕ

Вариант 1

4. Определить класс Матрица размерности (n x n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения матриц. Объявить массив объектов. Создать методы, вычисляющие первую и вторую нормы матрицы

$$||a||_1 = \max_{1 \le i \le n} \sum_{j=1}^{n} (a_{ij}), \|a\|_2 = \max_{1 \le j \le n} \sum_{i=1}^{n} (a_{ij})$$

Определить, какая из матриц имеет наименьшую первую и вторую нормы.

5. Определить класс Матрица размерности (m x n). Класс должен содержать несколько конструкторов. Объявить массив объектов. Передать объекты в метод, меняющий местами строки с максимальным и минимальным элементами k-го столбца. Создать метод, который изменяет i-ю матрицу путем возведения ее в квадрат.

Вариант 2

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

4. Abiturient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Оценки. Создать массив объектов. Вывести: a) список абитуриентов, имеющих неудовлетворительные оценки; b) список абитуриентов, средний балл у которых выше заданного; c) выбрать заданное число n абитуриентов, имеющих самый высокий средний балл (вывести также полный список абитуриентов, имеющих полупроходной балл).

5. Book: id, Название, Автор(ы), Издательство, Год издания, Количество страниц, Цена, Переплет. Создать массив объектов. Вывести: a) список книг заданного автора; b) список книг, выпущенных заданным издательством; c) список книг, выпущенных после заданного года.

Вариант 3

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

4. Создать объект класса Простая дробь, используя класс Число. Методы: вывод на экран, сложение, вычитание, умножение, деление.

5. Создать объект класса Дом, используя классы Окно, Дверь. Методы: закрыть на ключ, вывести на консоль количество окон, дверей.

Вариант 4

Построить модель программной системы.

4. Система Вступительные экзамены. Абитуриент регистрируется на Факультет, сдает Экзамены. Преподаватель выставляет Оценку. Система подсчитывает средний балл и определяет Абитуриентов, зачисленных в учебное заведение.

5. Система Библиотека. Читатель оформляет Заказ на Книгу. Система осуществляет поиск в Каталоге. Библиотекарь выдает Читателю Книгу на абонемент или в читальный зал. При невозвращении Книги Читателем он может быть занесен Администратором в «черный список».

Задание 1

Создан класс матрицы в соответствии с заданием. Имеется проверка размера матрицы, возможность получения элементов матрицы по индексам, а также поддержка операторов. Код класса матрицы приведён в листинге 1.

Листинг 1 -- Задача 1

```
class Matrix(private val data: Array[Array[Int]]):
  require(data.length == data.head.length, "Matrix must be square!")

  def this(n: Int) (fill_with: => Int) =
    this(data = Array.fill(n,n)(fill_with))

  def size: Int = data.length

  def apply(i: Int, j: Int): Int= data(i)(j)

  def update(i: Int, j: Int, value: Int): Unit =
    data(i)(j) = value

  def sum (other: Matrix): Matrix =
    require(this.size == other.size, "Dimensions of the matrixes should be
equal!")
    val newData = Array.fill(size, size)(0)
    for (i <- 0 until size; j <- 0 until size) {
      newData(i)(j) = data(i)(j) + other(i, j)
    }
    new Matrix(newData)
  def +(other: Matrix): Matrix = sum(other)

  def sub (other: Matrix): Matrix =
    require(this.size == other.size, "Dimensions of the matrixes should be
equal!")
    val newData = Array.fill(size, size)(0)
    for (i <- 0 until size; j <- 0 until size) {
      newData(i)(j) = data(i)(j) - other(i, j)
    }
    new Matrix(newData)

  def -(other: Matrix): Matrix = sub(other)

  def mul(other: Matrix) =
    require(this.size == other.size, "Dimensions of the matrixes should be
equal!")
    val newData = Array.fill(size, size)(0)
    for (i <- 0 until size; j <- 0 until size) {
      for (k <- 0 until size) {
        newData(i)(j) += data(i)(k) * other(k, j)
      }
    }
    new Matrix(newData)
  def *(other: Matrix): Matrix = mul(other)

  override def toString: String = "\n" + data.map(_.mkString("
")).mkString("\n")

  def norm1:Int =
    data.map(_.sum).max

  def norm2:Int =
    (0 until size).map(j => data.map( row => row(j)).sum).max
```

Результат работы тестовой программы приведён в листинге 2.

Листинг 2 -- Решение задачи 1

```
Variant 1
Question 4
Matrix op test.
Matrix A:
1 0 4 1
4 2 3 3
3 4 0 3
1 3 0 1
Matrix B:
0 3 4 2
0 3 4 0
1 4 2 0
1 4 0 0
A+B=
1 3 8 3
4 5 7 3
4 8 2 3
2 7 0 1
A-B=
1 -3 0 -1
4 -1 -1 3
2 0 -2 3
0 -1 0 1
A*B=
5 23 12 2
6 42 30 8
3 33 28 6
1 16 16 2
Matrix array & norms test.
Array:
List(
2 3 0 1
1 3 1 2
3 3 1 3
3 2 0 3,
2 1 0 0
3 1 3 4
0 1 0 1
2 4 1 4,
3 3 0 3
4 2 1 2
3 3 1 1
2 0 0 3,
2 0 3 1
1 0 1 0
4 1 4 2
0 4 1 2)
Norm1: List(10, 11, 9, 11)
Norm2: List(11, 9, 12, 9)
Max norm 1 in matrix: 1
Max norm 2 in matrix: 2
Test complete!
------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Thu Mar 16 16:22:57 MSK 2023
```

Задача 2

Создан класс матрицы в соответствии с заданием. Дополнительный конструктор позволяет создавать матрицу определённого размера и заполнять её значениями переданного выражения. Код класса матрицы приведён в листинге 3.

Листинг 3 -- Задача 2

```
class Matrix(val data: Array[Array[Int]]) {
  // Constructor that takes in the dimensions and data array
  // and initializes the rows, cols and data instance variables
  def this(rows: Int, cols: Int) (fill_with: => Int) = this(data =
Array.fill[Int](rows, cols)(fill_with))

  // Method to access a specific element in the matrix
  def apply(row: Int, col: Int): Double = data(row)(col)

  // Method to update a specific element in the matrix
  def update(row: Int, col: Int, value: Int): Unit = data(row)(col) = value

  // Method to print the matrix
  override def toString: String = "\n" + data.map(_.mkString("
")).mkString("\n")
}

def swap(m:Matrix, k:Int): Matrix =
  val min = m.data.minBy(row => row(k))
  val minIndex = m.data.indexOf(min)
  val maxIndex = m.data.indexOf(m.data.maxBy(row => row(k)))
  m.data(minIndex) = m.data.maxBy(row => row(k))
  m.data(maxIndex) = min
  m

def square(arr: List[Matrix], i: Int): List[Matrix] =
  require(arr(i).data.length == arr(i).data(0).length, "To square a matrix it
should be n*n dimensions!")
  val old = arr(i).data
  val size = old.length
  val newData = Array.fill(size, size)(0)
  for (i <- 0 until size; j <- 0 until size) {
    for (k <- 0 until size) {
      newData(i)(j) += old(i)(k) * old(k)(j)
    }
  }
  arr.updated(i, Matrix(newData))
```

Результат исполнения программы приведён в листинге 4.

Листинг 4 -- Решение задачи 2

```
Variant 1
Question 5
List of matrices: List(
4 1 0 0
2 4 4 2
3 1 1 0
1 1 2 4,
2 3 0 0
2 2 1 3
0 2 1 4
0 0 2 4,
4 2 2 0
4 1 4 3
2 0 2 4
1 2 3 1)
Swap by col 1: List(
2 4 4 2
4 1 0 0
3 1 1 0
1 1 2 4,
0 0 2 4
2 2 1 3
0 2 1 4
2 3 0 0,
2 0 2 4
4 1 4 3
4 2 2 0
1 2 3 1)
Square matrix 2: List(
2 4 4 2
4 1 0 0
3 1 1 0
1 1 2 4,
0 0 2 4
2 2 1 3
0 2 1 4
2 3 0 0,
16 12 20 12
31 15 29 22
24 6 20 22
23 10 19 11)
------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Thu Mar 16 16:35:45 MSK 2023
```

Задача 3

Реализованы классы Abiturient и AbitList (обёртка над массивом Abiturient) в соответствии с заданием. Сеттеры и геттеры записаны в соответствии с

принятой в Scala нотацией. Имеется ряд вспомогательных методов для работы с массивом. Код классов приведён в листинге 5.

Листинг 5 -- Задача 3

```
class Abiturient(
  private var _id: Int,
  private var _surname: String,
  private var _name: String,
  private var _patronymic: String,
  private var _address: String,
  private var _telephone: String,
  private var _marks: List[Int]
):
  require(!(_marks.exists(mark => (mark>100) || (mark<0))), "Marks should be
in range 0..100")
  def id = _id
  def id_=(id:Int) = _id = id
  def surname = _surname
  def surname_=(surname:String) = _surname = surname
  def name = _name
  def name_=(name:String) = _name = name
  def patronymic = _patronymic
  def patronymic_=(patronymic:String) = _patronymic = patronymic
  def address = _address
  def address_=(address:String) = _address = address
  def telephone = _telephone
  def telephone_=(telephone:String) = _telephone = telephone
  def marks = _marks
  def marks_=(marks:List[Int]) = _marks = marks
  override def toString: String = s"""|Abiturient №$id: $surname $name
$patronymic
                                      |Address: $address
                                      |Telephone: $telephone
                                      |Marks: ${marks.mkString(",
")}""".stripMargin


class AbitList(private val _lst: List[Abiturient]):
  def this(abts: Abiturient* ) =
    this(_lst = List.from(abts))
  override def toString: String = lst.mkString("\n------\n")
  def lst = _lst
  def apply(i:Int) = lst(i)
  def unsatisfactory =
    AbitList(_lst.filter(abt => abt.marks.exists(_<60)))

  def avgOver(threshold: Int) =
    AbitList(_lst.filter(abt => (abt.marks.sum / abt.marks.length.toDouble) >
threshold))

  def top(n: Int = _lst.length) =
    AbitList(_lst.sortBy(abt => (abt.marks.sum /
abt.marks.length.toDouble))(Ordering[Double].reverse).take(n))
```

Результат работы программы приведён в листинге 6.

Листинг 6 -- Решение задачи 3

```
Variant 2
Question 4
Abiturients:
Abiturient №0: itpi3sd lwcN6aGyATP vxK
Address: gz
Telephone: 5348185155
Marks: 87, 80, 55
------
Abiturient №1: 36AWbaWYEjrzXP wJU U8Of0efLFq4co
Address: SYoG0b
Telephone: 8685761322
Marks: 70, 55, 61
------
Abiturient №2: Ils l7X1goRe1 kJUmhvAHHh
Address: Ox6GZQa
Telephone: 1502378174
Marks: 50, 71, 89
------
Abiturient №3: 520DHoMm8zmeEI edgmT1C5ii0 2zgq9SlBo
Address: 3uyK8pTXxRltJ4gP
Telephone: 3789710775
Marks: 85, 94, 67
------
Abiturient №4: FYlIsqsfF 3uXW0rC PW9HhSj5K
Address: 6s1niz4qmhEwa
Telephone: 6204270098
Marks: 86, 77, 88
------
Abiturient №5: LiJGhPd L KjuIkQ1D
Address: qE8oewtpO
Telephone: 1500438614
Marks: 72, 74, 98
------
Abiturient №6:  KH BNaAvA5
Address: utTPN7EVN
Telephone: 5786981683
Marks: 89, 74, 55
~~~~~~~
Unsatisfactory marks:
Abiturient №0: itpi3sd lwcN6aGyATP vxK
Address: gz
Telephone: 5348185155
Marks: 87, 80, 55
------
Abiturient №1: 36AWbaWYEjrzXP wJU U8Of0efLFq4co
Address: SYoG0b
Telephone: 8685761322
Marks: 70, 55, 61
------
Abiturient №2: Ils l7X1goRe1 kJUmhvAHHh
Address: Ox6GZQa
Telephone: 1502378174
Marks: 50, 71, 89
------
Abiturient №6:  KH BNaAvA5
Address: utTPN7EVN
Telephone: 5786981683
Marks: 89, 74, 55
~~~~~~~
Average over 80:
Abiturient №3: 520DHoMm8zmeEI edgmT1C5ii0 2zgq9SlBo
Address: 3uyK8pTXxRltJ4gP
Telephone: 3789710775
Marks: 85, 94, 67
------
Abiturient №4: FYlIsqsfF 3uXW0rC PW9HhSj5K
Address: 6s1niz4qmhEwa
Telephone: 6204270098
Marks: 86, 77, 88
------
Abiturient №5: LiJGhPd L KjuIkQ1D
Address: qE8oewtpO
Telephone: 1500438614
Marks: 72, 74, 98
~~~~~~~
Top 2:
Abiturient №4: FYlIsqsfF 3uXW0rC PW9HhSj5K
Address: 6s1niz4qmhEwa
Telephone: 6204270098
Marks: 86, 77, 88
------
Abiturient №3: 520DHoMm8zmeEI edgmT1C5ii0 2zgq9SlBo
Address: 3uyK8pTXxRltJ4gP
Telephone: 3789710775
Marks: 85, 94, 67
-------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Fri Mar 17 14:18:57 MSK 2023
```

Задача 4

Аналогично задаче 3 были реализованы классы Book и BookList с рядом вспомогательных функций. Код приведён в листинге 7.

Листинг 7 -- Задача 4

```
class Book(
  private var _id: Int,
  private var _title: String,
  private var _authors: List[String],
  private var _publisher: String,
  private var _year: Int,
  private var _pages: Int,
  private var _price: Int,
  private var _cover: String
):
  require(_authors.length>=1, "There must be at least 1 author of a book!")
  def id = _id
  def id_=(id:Int) = _id = id
  def title = _title
  def title_=(title:String) = _title = title
  def authors = _authors
  def authors_=(authors:List[String]) = _authors = authors
  def publisher = _publisher
  def publisher_=(publisher:String) = _publisher = publisher
  def year = _year
  def year_=(year:Int) = _year = year
  def pages = _pages
  def pages_=(pages:Int) = _pages = pages
  def price = _price
  def price_=(price:Int) = _price = price
  def cover = _cover
  def cover_=(id:String) = _cover = cover

  override def toString(): String = s"""|Book №$id: \'$title\'
                                        |By ${authors.mkString(", ")}
                                        |Pub: $publisher, $year
                                        |Pages: $pages; Cover: $cover
                                        |Price: $price""".stripMargin

class BookList(val lst: List[Book]):
  def byAuthor(name: String) =
    BookList(lst.filter(book => book.authors.contains(name)))

  def byPub(pub: String) =
    BookList(lst.filter(book => book.publisher.equals(pub)))

  def publishedAfter(year: Int) =
    BookList(lst.filter(_.year > year))

  override def toString: String = lst.mkString("\n------\n")

  def apply(i: Int) = lst.apply(i)
```

Результат работы программы приведён в листинге 8.

Листинг 8 -- Решение задачи 4

```
Variant 2
Question 5
Books:
Book %0: 'hrhkISMkVJSfN'
By 16hX7St, m, ri
Pub: 3Lx, 1702
Pages: 890; Cover: Soft
Price: 34606
------
Book %1: 'Kihfvp9'
By m, 16hX7St, ri
Pub: 3Lx, 1596
Pages: 358; Cover: Hard
Price: 7170
------
Book %2: 'ZsyjoQnUiKcGmnp'
By w9cBBcPWfAJ
Pub: , 1776
Pages: 1135; Cover: Hard
Price: 16843
------
Book %3: 'LXDJEzJn0Vm5gnlbU3V'
By sU8N9j7
Pub: , 2000
Pages: 456; Cover: Soft
Price: 19439
------
Book %4: 'so3nsp'
By ri, w9cBBcPWfAJ, ri
Pub: , 1861
Pages: 333; Cover: Soft
Price: 11079
------
Book %5: 'pk'
By sU8N9j7, w9cBBcPWfAJ, w9cBBcPWfAJ
Pub: 3Lx, 1923
Pages: 1194; Cover: Soft
Price: 19004
------
Book %6: 'Tt9b78'
By sU8N9j7, w9cBBcPWfAJ
Pub: 3Lx, 1948
Pages: 918; Cover: Soft
Price: 30002
~~~~~~~
Books by m:
Book %0: 'hrhkISMkVJSfN'
By 16hX7St, m, ri
Pub: 3Lx, 1702
Pages: 890; Cover: Soft
Price: 34606
------
Book %1: 'Kihfvp9'
By m, 16hX7St, ri
Pub: 3Lx, 1596
Pages: 358; Cover: Hard
Price: 7170
~~~~~~~
Published by :
Book %2: 'ZsyjoQnUiKcGmnp'
By w9cBBcPWfAJ
Pub: , 1776
Pages: 1135; Cover: Hard
Price: 16843
------
Book %3: 'LXDJEzJn0Vm5gnlbU3V'
By sU8N9j7
Pub: , 2000
Pages: 456; Cover: Soft
Price: 19439
------
Book %4: 'so3nsp'
By ri, w9cBBcPWfAJ, ri
Pub: , 1861
Pages: 333; Cover: Soft
Price: 11079
~~~~~~~
Published after 1800:
Book %3: 'LXDJEzJn0Vm5gnlbU3V'
By sU8N9j7
Pub: , 2000
Pages: 456; Cover: Soft
Price: 19439
------
Book %4: 'so3nsp'
By ri, w9cBBcPWfAJ, ri
Pub: , 1861
Pages: 333; Cover: Soft
Price: 11079
------
Book %5: 'pk'
By sU8N9j7, w9cBBcPWfAJ, w9cBBcPWfAJ
Pub: 3Lx, 1923
Pages: 1194; Cover: Soft
Price: 19004
------
Book %6: 'Tt9b78'
By sU8N9j7, w9cBBcPWfAJ
Pub: 3Lx, 1948
Pages: 918; Cover: Soft
Price: 30002
-------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Fri Mar 17 14:25:57 MSK 2023
```

Задача 5

Реализован класс простой дроби Fraction в соответствии с заданием. В классе имеется наибольший общий делитель и сокращённые значения числителя и знаменателя. Хэш подсчитывается исходя из сокращённых значений. Код класса приведён в листинге 9.

Листинг 9 -- Задача 5

```
class Fraction(val numerator: Int, val denominator: Int):
  require(denominator != 0, "denominator must be non-zero")
  def output = println(s"$numerator/$denominator")
  def plus(x: Fraction): Fraction =
    val new_num =
      if this.denominator == x.denominator then
        this.numerator + x.numerator
      else (this.numerator * x.denominator) + (x.numerator * this.denominator)
    val new_denom =
      if this.denominator == x.denominator then
        this.denominator
      else
        this.denominator * x.denominator
    Fraction(new_num, new_denom)
  def +(x:Fraction) = plus(x)
  def minus(x: Fraction): Fraction =
    val new_num =
      if this.denominator == x.denominator then
        this.numerator - x.numerator
      else (this.numerator * x.denominator) - (x.numerator * this.denominator)
    val new_denom =
      if this.denominator == x.denominator then
        this.denominator
      else
        this.denominator * x.denominator
    Fraction(new_num, new_denom)
  def -(x:Fraction) = minus(x)
  def mul(x: Fraction): Fraction =
    val new_num = this.numerator * x.numerator
    val new_denom = this.denominator * x.denominator
    Fraction(new_num, new_denom)
  def *(x: Fraction) = mul(x)
  def div(x: Fraction): Fraction =
    val new_num = this.numerator * x.denominator
    val new_denom = this.denominator * x.numerator
    Fraction(new_num, new_denom)
  def /(x: Fraction) = div(x)
  // reduce fraction to its lowest terms
  private val gcd = BigInt(numerator).gcd(BigInt(denominator))
  val reducedNumerator = numerator / gcd.toInt
  val reducedDenominator = denominator / gcd.toInt

  override def equals(x: Any): Boolean = x match
    case that: Fraction => reducedDenominator == that.reducedDenominator &&
reducedNumerator == that.reducedNumerator
    case _ => false

  override def toString(): String = s"$numerator/$denominator"

  override def hashCode(): Int = 31 * (31 + reducedNumerator) +
reducedDenominator
```

Результат работы программы приведён в листинге 10.

Листинг 10 -- Решение задачи 5

```
Variant 3
Question 4
Fraction A: 1/2
Fraction B: 3/4
A+B=10/8
A-B=-2/8
A*B=3/8
A/B=4/6
A==B: false
A==2/4: true
A hashCode: 994
A output test:
1/2
------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Fri Mar 17 14:35:37 MSK 2023
```

Задача 6

В соответствии с заданием заданы классы Door, Window и House. Их код приведён в листинге 11.

Листинг 11 -- Задача 6

```
class Door(val name: String, var locked: Boolean = false):
  def lock() = locked = true
  def unlock() = locked = false
  def toggle() = locked = !locked
  override def toString(): String = s"Door $name, locked: $locked"
  override def hashCode(): Int = name.hashCode()
  override def equals(x: Any): Boolean = x match
    case that: Door => name.equals(that.name)
    case _ => false

class Window(val name: String):
  override def toString(): String = s"Window $name"
  override def hashCode(): Int = name.hashCode()
  override def equals(x: Any): Boolean = x match
    case that: Window => name.equals(that.name)
    case _ => false

class House(val name: String, val doors: List[Door], val windows:
List[Window]):
  require(doors.length > 0, "There should be at least 1 door in a house!")
  def lock() = doors.foreach(_.lock())
  def unlock() = doors.foreach(_.unlock())
  def toggle() = doors.foreach(_.toggle())
  def countDoors = doors.length
  def countWindows = windows.length

  override def toString(): String =
    s"""|House $name:
        |Doors:
        |${doors.mkString("\n")}
        |Windows:
        |${windows.mkString("\n")}
        |""".stripMargin

  override def equals(x: Any): Boolean = x match
    case that: House => name.equals(that.name) && doors.equals(that.doors) &&
windows.equals(that.windows)
    case _ => false

  override def hashCode(): Int = name.hashCode() + doors.map(_.hashCode()).sum
+ windows.map(_.hashCode()).sum
```

Результат работы программы приведён в листинге 12.

Листинг 12 -- Решение задачи 6

```
Variant 3
Question 5
House1:
House adin:
Doors:
Door door1, locked: false
Door door2, locked: false
Windows:
Window Shindows

House2:
House Dva:
Doors:
Door door2, locked: false
Door backdoor, locked: false
Door door1, locked: false
Windows:
Window window1

Lock all doors in house1
House adin:
Doors:
Door door1, locked: true
Door door2, locked: true
Windows:
Window Shindows

Doors in house2: 3
Windows in house1: 1
house1==house2: false
house1==eq_house1: true
house1 hash: -259262704
------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Fri Mar 17 14:42:45 MSK 2023
```

Задача 7

На основе решения задачи 3 была выполнена задача 7. Классы Abiturient и AbitList были переписаны с использованием Map для хранения оценок с названиями экзаменов. Добавлены классы Faculty, Exam, Instructor и Enrollment. Код классов приведён в листинге 13.

Листинг 13 -- Задача 7

```
class Abiturient(
  private var _id: Int,
  private var _surname: String,
  private var _name: String,
  private var _patronymic: String,
  private var _address: String,
  private var _telephone: String,
  private var _marks: Map[String, Int] = Map.empty
):
  require(!(_marks.exists(mark => (mark._2 > 100) || (mark._2 < 0))), "Marks
should be in range 0..100")
``````
  def marks = _marks
  def marks_=(marks: Map[String, Int]) = _marks = marks

  override def toString: String = s"""|Abiturient №$id: $surname $name
$patronymic
                                      |Address: $address
                                      |Telephone: $telephone
                                      |Marks: ${marks.mkString(",
")}""".stripMargin

class AbitList(private val _lst: List[Abiturient]):
``````
  def unsatisfactory =
    AbitList(_lst.filter(abt => abt.marks.values.exists(_<60)))
  def avgOver(threshold: Int, exams: List[Exam] = Nil) =
    if exams.isEmpty then
      AbitList(_lst.filter(abt => (abt.marks.values.sum /
abt.marks.values.size.toDouble) > threshold))
    else
      val subjects = exams.map(_.subject)
      val qualifiedApplicants = _lst.filter(abt => subjects.forall(subject =>
abt.marks.contains(subject)))
      AbitList(qualifiedApplicants.filter(
        abt => (abt.marks.filterKeys(subjects.contains).values.sum /
subjects.length.toDouble) > threshold ))
  def top(n: Int = _lst.length, exams: List[Exam] = Nil) =
    if exams.isEmpty then
      AbitList(_lst.sortBy(abt => (abt.marks.values.sum /
abt.marks.values.size.toDouble))(Ordering[Double].reverse).take(n))
    else
      val subjects = exams.map(_.subject)
      val qualifiedApplicants = _lst.filter(abt => subjects.forall(subject =>
abt.marks.contains(subject)))
      AbitList(qualifiedApplicants.sortBy(
        abt => (abt.marks.filterKeys(subjects.contains).values.sum /
subjects.length.toDouble))
        (Ordering[Double].reverse).take(n))
  def passed(exams: List[Exam]) =
    AbitList(_lst.filter(abt => exams.forall(exam =>
abt.marks.contains(exam.subject))))
```

```
class Faculty(private val _name: String, private var _applicants: AbitList):
  def name = _name
  def applicants = _applicants
  def addApplicant(abt: Abiturient) = _applicants = new
AbitList(_applicants.lst :+ abt)

  def removeApplicant(abt: Abiturient) = _applicants = new
AbitList(_applicants.lst.filterNot(_ == abt))

class Exam(private val _subject: String, private val _maxScore: Int):
  def subject = _subject
  def maxScore = _maxScore

class Instructor(private val _name: String):
  def name = _name
  def grade(abt: Abiturient, exam: Exam, score: Int) =
    require(score <= exam.maxScore, s"Score for the '${exam.subject}' exam
cannot be over ${exam.maxScore}! (given $score)")
    abt.marks = abt.marks + ((exam.subject, score))

class Enrollment(private val _faculty: Faculty, private val _exams:
List[Exam]):
  def faculty = _faculty
  def exams = _exams

  def enrollApplicants(threshold: Int, n: Int =
faculty.applicants.lst.length): AbitList =
    val qualifiedApplicants = _faculty.applicants.avgOver(threshold, _exams)
    qualifiedApplicants.top(n, _exams )
```

Результат работы программы приведён в листинге 14.

Листинг 14 -- Решение задачи 7

```
def test() =
  val faculty = new Faculty("Computer Science", new AbitList())

  val mathExam = new Exam("Math", 100)
  val csExam = new Exam("Computer Science", 100)

  val alice = new Abiturient(1, "Smith", "Alice", "A.", "123 Main St.", "555-
1234")
  val bob = new Abiturient(2, "Jones", "Bob", "B.", "456 Elm St.", "555-5678")

  val instructor = new Instructor("John Doe")

  instructor.grade(alice, mathExam, 90)
  instructor.grade(alice, csExam, 80)
  instructor.grade(bob, mathExam, 70)
  instructor.grade(bob, csExam, 85)

  faculty.addApplicant(alice)
  faculty.addApplicant(bob)

  val enrollment = new Enrollment(faculty, List(mathExam, csExam))

  val enrolled = enrollment.enrollApplicants(75, 1)

  println(enrolled)
Output:
Variant 4
Question 4
Abiturient №1: Smith Alice A.
Address: 123 Main St.
Telephone: 555-1234
Marks: Math -> 90, Computer Science -> 80
------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Fri Mar 17 14:51:57 MSK 2023
```

**Задача 8**

Задача решена с использованием классов из задачи 4 без их изменения. Добавлены классы Reader, Order и Library. Код этих классов приведён в листинге 15.

Листинг 15 -- Задача 8

```
import var2.task2.*
import scala.collection.mutable

class Reader(val name: String) {
  private var _blacklisted: Boolean = false
  def blacklisted: Boolean = _blacklisted
  def blacklist(): Unit = _blacklisted = true
}

class Order(val reader: Reader, val book: Book)

class Library(val catalog: BookList)
  private val _orders: mutable.Set[Order] = mutable.Set.empty
  private val _lentBooks: mutable.Map[Book, Reader] = mutable.Map.empty
  def orderBook(reader: Reader, book: Book): Unit =
    if (catalog.lst.contains(book)) then
      _orders.addOne(new Order(reader, book))
      println(s"${reader.name} has ordered ${book.title}")
    else
      println(s"Sorry, ${book.title} is not in our catalog")

  def lendBook(book: Book, reader: Reader): Unit =
    if (_lentBooks.contains(book)) then
      println(s"Sorry, ${book.title} is already lent to
${_lentBooks(book).name}")
    else
      val order = _orders.find(_.book == book)
      order match
        case Some(o) =>
          if (o.reader == reader) {
            _lentBooks.put(book, reader)
            _orders.remove(o)
            println(s"${book.title} has been lent to ${reader.name}")
          } else {
            println(s"Sorry, ${book.title} has been ordered by
${o.reader.name}")
          }
        case None => println(s"Sorry, ${book.title} has not been ordered")

  def returnBook(book: Book, reader: Reader): Unit = {
    if (_lentBooks.contains(book) && _lentBooks(book) == reader) {
      _lentBooks.remove(book)
      println(s"${book.title} has been returned by ${reader.name}")
    } else {
      println(s"${reader.name} did not borrow ${book.title}")
    }
  }

  def blacklistReader(reader: Reader): Unit = {
    reader.blacklist()
    println(s"${reader.name} has been blacklisted")
  }
}
```

Результат работы программы приведён в листинге 16.

Листинг 16 -- Решение задачи 8

```
Variant 4
Question 5
John Doe has ordered vYb8aLNvU
Jane Smith has ordered vYb8aLNvU
Sorry, SZpMfeiIhY3XYmn is not in our catalog
vYb8aLNvU has been lent to John Doe
Sorry, vYb8aLNvU is already lent to John Doe
Sorry, 2988E7fdF has not been ordered
vYb8aLNvU has been returned by John Doe
Jane Smith did not borrow vYb8aLNvU
Jane Smith has been blacklisted
------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Fri Mar 17 15:00:25 MSK 2023
```

# ВЫВОДЫ

Изучены принципы работы с классами в Scala. Освоены приватные поля, изменяемые и неизменяемые поля.

Освоены способы задания сеттеров и геттеров в нотации Scala.

Освоены переопределяемые методы toString, equals и hashCode.

Изучены способы использования значений по умолчанию.

Изучены методы работы с коллекциями объектов.

Изучен способ объявления имплицитной конверсии классов объектов.