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 № 4

Название: Внутренние классы. Интерфейсы

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.

1. Создать класс Художественная Выставка с внутренним классом, с помощью объектов которого можно хранить информацию о картинах, авторах и времени проведения выставок.
2. Создать класс Календарь с внутренним классом, с помощью объектов которого можно хранить информацию о выходных и праздничных днях.

Вариант 2.

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

1. interface Здание <- abstract class Общественное Здание <- class Театр.
2. interface Mobile <- abstract class Siemens Mobile <- class Model.

Задача 1

Создан класс Выставка, в котором создан класс Картина. Картина обладает названием, автором, а также датой начала и окончания её выставки. Также определён формат преобразования картины в строку. В выставке содержится массив картин, а также имеется метод для создания и добавления картины к выставке. Код решения приведён в листинге 1.

Листинг 1 – Задача 1

```
import com.github.nscala_time.time.Imports._

class ArtExhibition {
  // Internal class for storing painting information
  class Painting(val title: String, val artist: String, val startExhibition:
DateTime, val endExhibition: DateTime):
    override def toString(): String =
      s"""|Painting '$title', by $artist.
      |Exhibited from ${startExhibition.toString("YYYY-MM-dd")} till
${endExhibition.toString("YYYY-MM-dd")}""".stripMargin
  // Collection to store paintings
  private var paintings = List[Painting]()
  // Method for adding a painting to the collection
  def newPainting(title: String, artist: String, startExhibition: DateTime,
endExhibition: DateTime): Painting = {
    val painting = new Painting(title, artist, startExhibition, endExhibition)
    paintings = painting :: paintings
    painting
  }
  // Method for retrieving all paintings in the collection
  def getPaintings(): List[Painting] = paintings

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

def test() =
  val exhibition = new ArtExhibition()

  def newPaintingInDates(title: String, artist: String, exhibition:
ArtExhibition = exhibition) =
    val start = DateTime.parse("2023-03-21")
    val end = start + 2.months
    exhibition.newPainting(title, artist, start, end)

  newPaintingInDates("Diana Bathing","Jean-Baptiste Camille Corot")
  newPaintingInDates("Blue Dancers", "Edgar Degas")
  newPaintingInDates("White Water Lilies", "Claude Monet")

  val msg =
    s"""|Currently exibited:
        |$exhibition""".stripMargin
  println(msg)
```

Результат работы программы приведён в листинге 2.

Листинг 2 – Решение задачи 1

```
Variant 1
Question 4
Currently exibited:
Painting 'White Water Lilies', by Claude Monet.
Exhibited from 2023-03-21 till 2023-05-21
------
Painting 'Blue Dancers', by Edgar Degas.
Exhibited from 2023-03-21 till 2023-05-21
------
Painting 'Diana Bathing', by Jean-Baptiste Camille Corot.
Exhibited from 2023-03-21 till 2023-05-21
------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Wed Mar 22 18:07:20 MSK 2023
```

Задача 2

Создан класс Календарь, в котором объявлен класс Нерабочие дни. Нерабочие дни содержат два массива: один с названиями дней недели, считающимися выходными; второй содержит даты (для упрощения в виде строк) праздничных дней. В классе также есть методы для добавления выходных и праздников, а также для проверки, является ли день выходным или праздником. Код решения задачи приведён в листинге 3.

Листинг 3 – Задача 2

```
class Calendar:
  class DaysOff:
    private var holidays = List[String]()
    private var weekends = List[String]("Saturday", "Sunday")

    def addHoliday(date: String): Unit =
      holidays = date :: holidays

    def addWeekend(day: String): Unit =
      weekends = day :: weekends

    def isHoliday(date: String): Boolean =
      holidays.contains(date)

    def isWeekend(day: String): Boolean =
      weekends.contains(day)

def test() =
  val calendar = new Calendar()
  val daysOff = new calendar.DaysOff()

  daysOff.addHoliday("2023-04-01")
  daysOff.addWeekend("Friday")

  val msg =
    s"""|Is 2023-04-01 a holiday? ${daysOff.isHoliday("2023-04-01")}
        |Is 2023-04-02 a holiday? ${daysOff.isHoliday("2023-04-02")}
        |Is friday a weekend? ${daysOff.isWeekend("Friday")}
        |Is monday a weekend? ${daysOff.isWeekend("Monday")}""".stripMargin
  println(msg)
```

Результат работы программы приведён в листинге 4.

Листинг 4 – Решение задачи 2

```
Variant 1
Question 5
Is 2023-04-01 a holiday? true
Is 2023-04-02 a holiday? false
Is friday a weekend? true
Is monday a weekend? false
------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Wed Mar 22 18:17:28 MSK 2023
```

Задача 3

В соответствии с заданием реализована иерархия классов. Интерфейс (trait) Здания требует, чтобы у объекта были методы для получения названия здания и количества комнат в нём. Общественное здание имеет поля с названием и количеством комнат, реализует методы интерфейса и добавляет абстрактный метод проверки, открыто ли здание. Театр наследуется от абстрактного класса и определяет, что он всегда открыт. Код задачи приведён в листинге 5.

Листинг 5 – Задача 3

```
trait Building:
  def getName(): String
  def getNumberOfRooms(): Int

abstract class PublicBuilding(name: String, numberOfRooms: Int) extends
Building:
  def getName(): String = name
  def getNumberOfRooms(): Int = numberOfRooms

  def isOpen(): Boolean

class Theatre(name: String, numberOfRooms: Int) extends PublicBuilding(name,
numberOfRooms):
  def isOpen(): Boolean = true

def test() =
  val theatre: Theatre = new Theatre("Palace Theatre", 3)

  println(s"""|Treating theatre as a theatre:
              |Theatre name: ${theatre.getName()}
              |Number of rooms: ${theatre.getNumberOfRooms()}
              |Is open: ${theatre.isOpen()}""".stripMargin)

  val building: Building = theatre

  println(s"""|Treating theatre as a building:
              |Building name: ${building.getName()}
              |Number of rooms: ${building.getNumberOfRooms()}""".stripMargin)
```

Результат работы программы приведён в листинге 6.

Листинг 6 – Решение задачи 3

```
Variant 2
Question 4
Treating theatre as a theatre:
Theatre name: Palace Theatre
Number of rooms: 3
Is open: true
Treating theatre as a building:
Building name: Palace Theatre
Number of rooms: 3
------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Wed Mar 22 18:26:27 MSK 2023
```

Задача 4

В соответствии с заданием реализована иерархия классов. Интерфейс мобильного телефона определяет, что должен быть бренд и метод совершения звонка. Абстрактный класс Siemens определяет бренд телефона и метод звонка, а также добавляет абстрактный метод идентификации устройства. Класс Модель наследуется от Siemens и определяет, какой фразой будет проводиться идентификация. Код задачи приведён в листинге 7.

Листинг 7 – Задача 4

```
trait Mobile:
  def call(): Unit
  def brand: String

abstract class SiemensMobile(owner: String) extends Mobile:
  def call(): Unit = println(s"$owner is making a call with Siemens Mobile")

  def brand: String = "Siemens"

  def identify(): String

class Model(modelName: String, owner: String) extends SiemensMobile(owner):
  def identify(): String = s"This is Siemens $modelName"

def test() =
  val m55 = new Model("M55", "Shrek")

  println(s"""|Testing new mobile phone!
              |This is a mobile phone by ${m55.brand}
              |${m55.identify()}""".stripMargin)

  m55.call()
```

Результат работы программы приведён в листинге 8.

Листинг 8 – Решение задачи 4

```
Variant 2
Question 5
Testing new mobile phone!
This is a mobile phone by Siemens
This is Siemens M55
Shrek is making a call with Siemens Mobile
-------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 3 15:39:00 MSK 2023
Task completed (this run) on: Wed Mar 22 18:34:43 MSK 2023
```

ВЫВОДЫ

Изучены особенности внутренних классов в Scala. Изучены способы их использования.

Изучены особенности интерфейсов (traits) и абстрактных классов в Scala. Освоены принципы их использования.