Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Санкт-Петербургский Государственный Электротехнический
Университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»

Факультет компьютерных технологий и информатики
Кафедра автоматики и процессов управления

ОТЧЕТ

по лабораторной работе № 6.1
по дисциплине «СМиСПИС»

« Разработка простого MDA-приложения »

Студент гр. 5371 _____ Мартынов М.

Студентка гр. 5371 _____ Козлова С.

Студент гр. 5371 _____ Аверкиев В.

Преподаватель _____ Кораблев Ю.А.

Санкт-Петербург
2020

1. Цель работы

Цель работы: ознакомиться с архитектурой, управляемой моделью MDA; научиться создавать простое приложение по технологии MDA.

.

2. Задание на лабораторную работу №6.1. Вариант №3.

Необходимо реализовать приложение, приведенное в п.2, используя технологию MDA. Приложение рассчитывает площадь и периметр фигуры (прямоугольника, ромба и треугольника).

Затем добавить возможность находить и выводить радиус вписанной и описанной окружности.

a. На диаграмме необходимо добавить два атрибута «radiusExternal» и «radiusInternal» и добавить два метода «getRadiusExternal» и «getRadiusInternal» для расчета значений радиусов.
b. Для Прямоугольника найти радиус описанной окружности и найти (если есть) радиус вписанной окружности.
c. Для Ромба найти радиус вписанной окружности и найти (если есть) радиус описанной окружности.
d. Для треугольника найти радиус вписанной и описанной окружности, если треугольник с заданными длинами сторон существует.

3. Выполнение лабораторной работы

3.1 Выбор технологий

Для выполнения данной лабораторной работы использован язык программирования Kotlin с компиляцией под JVM, а также обертка над фремйворком Java FX для языка Kotlin - Tornado FX.

3.2 UML диаграмма разработанных классов

На рис.1 изображена диаграмма разработанных классов

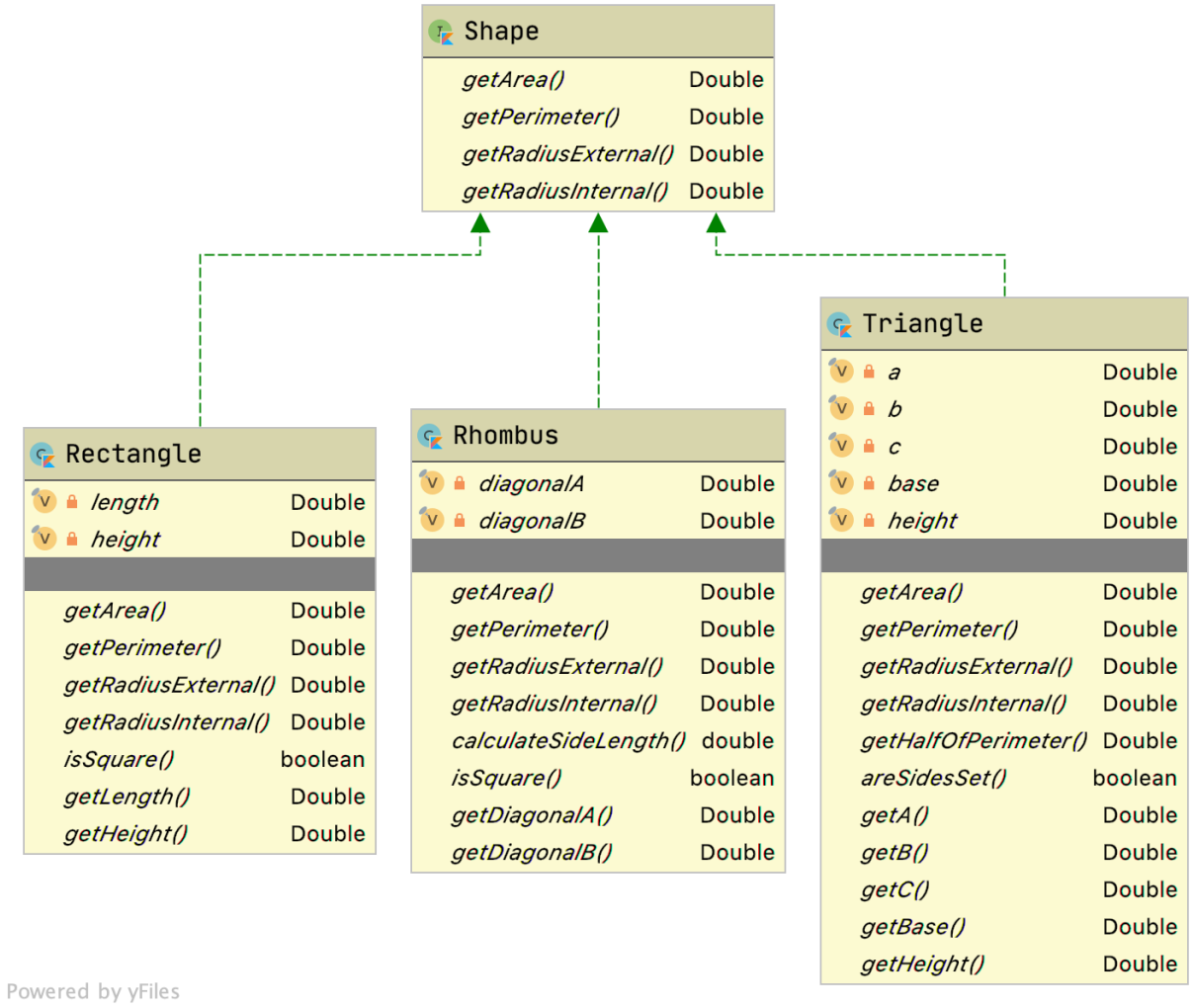

Рис. 1 – UML диаграмма классов Model

Помимо стандартных методов getArea() и getPerimeter() интерфейс Shape содержит два метода, указанных в задании: getRadiusExternal() и getRadiusInternal().

3.3 Создание интерфейса

На рис.2 изображен вид программы при запуске с фигурой по умолчанию «Треугольник».

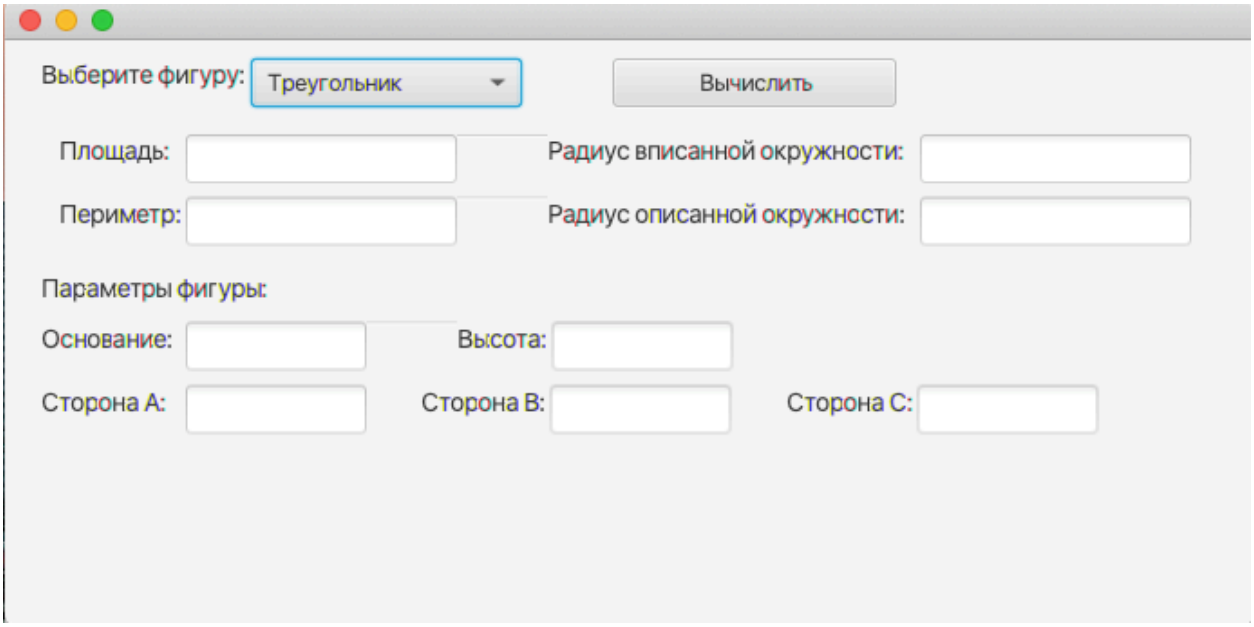

Рисунок 2 – Вид формы с фигурой «Треугольник»

На рис. 3 показан пример работы программы, когда треугольник существует.

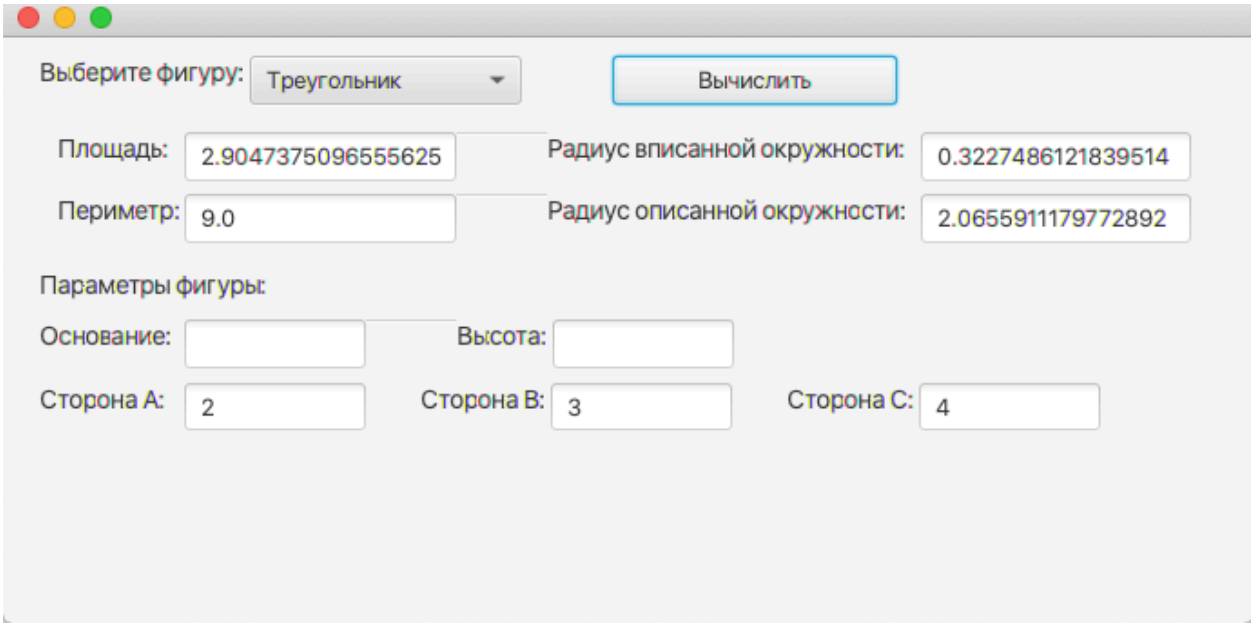

Рисунок 3 – Пример работы программы с существующей фигурой «Треугольник»

На рис. 4 показан пример работы программы, когда треугольник не существует.

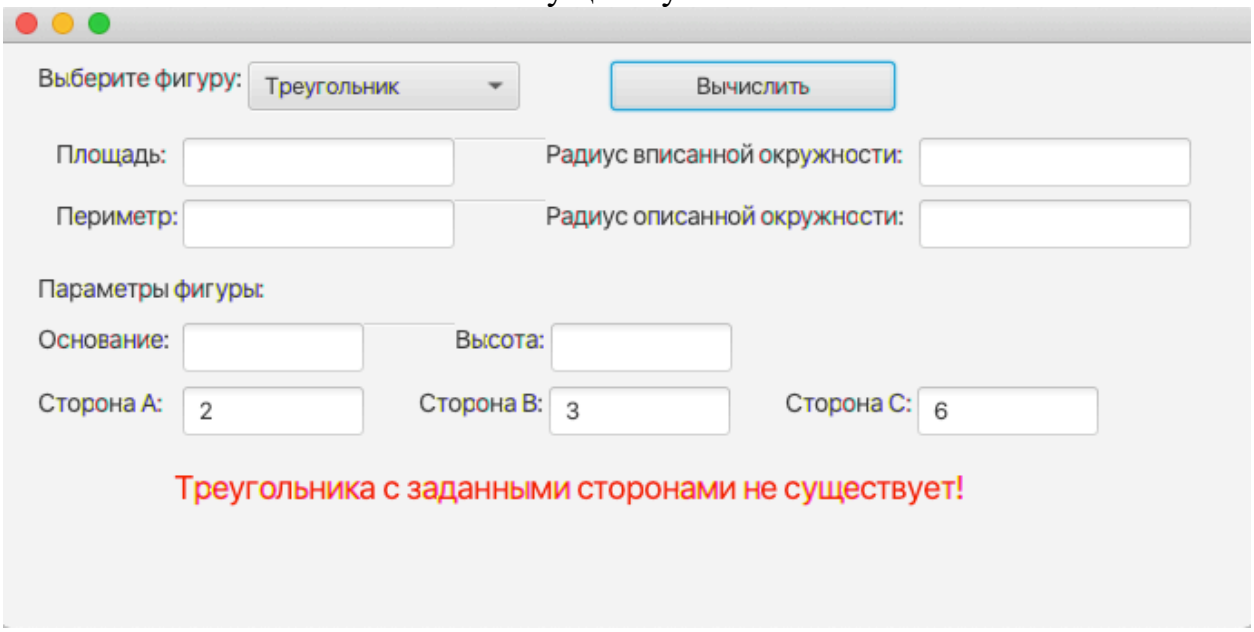

Рисунок 4 – Пример работы программы с не существующей фигурой «Треугольник»

На рис. 5 показан пример работы программы с фигурой «Ромб»

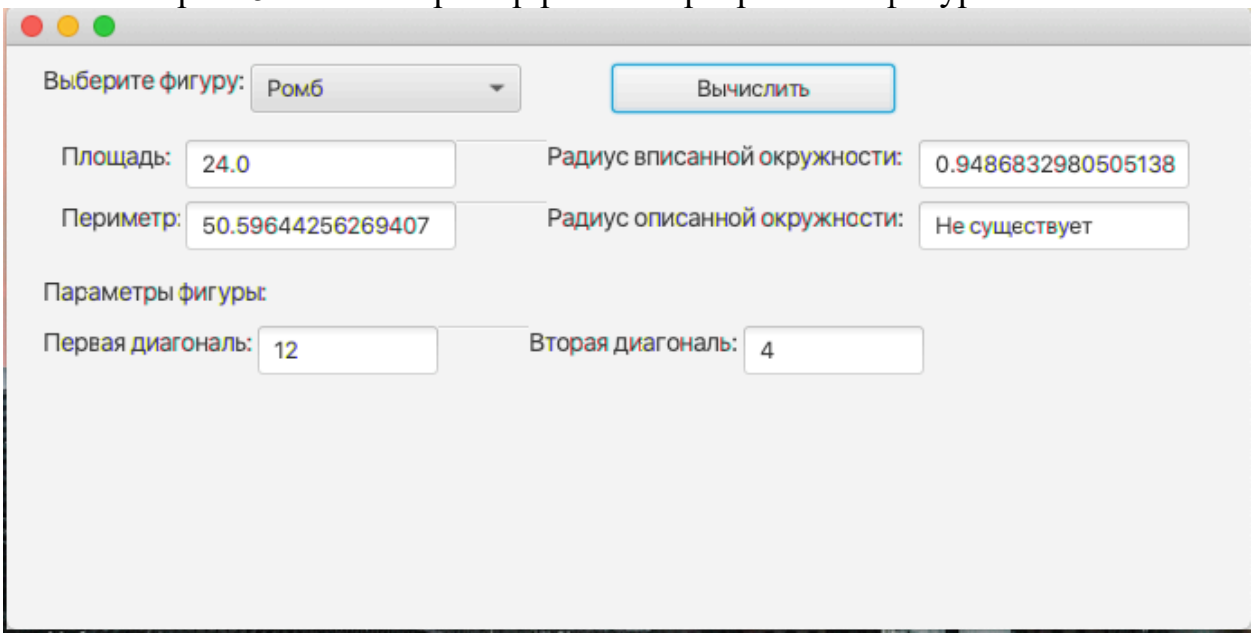

Рисунок 5 – Пример работы программы с фигурой «Ромб»

На рис. 6 показан пример работы программы с фигурой «Прямоугольник»

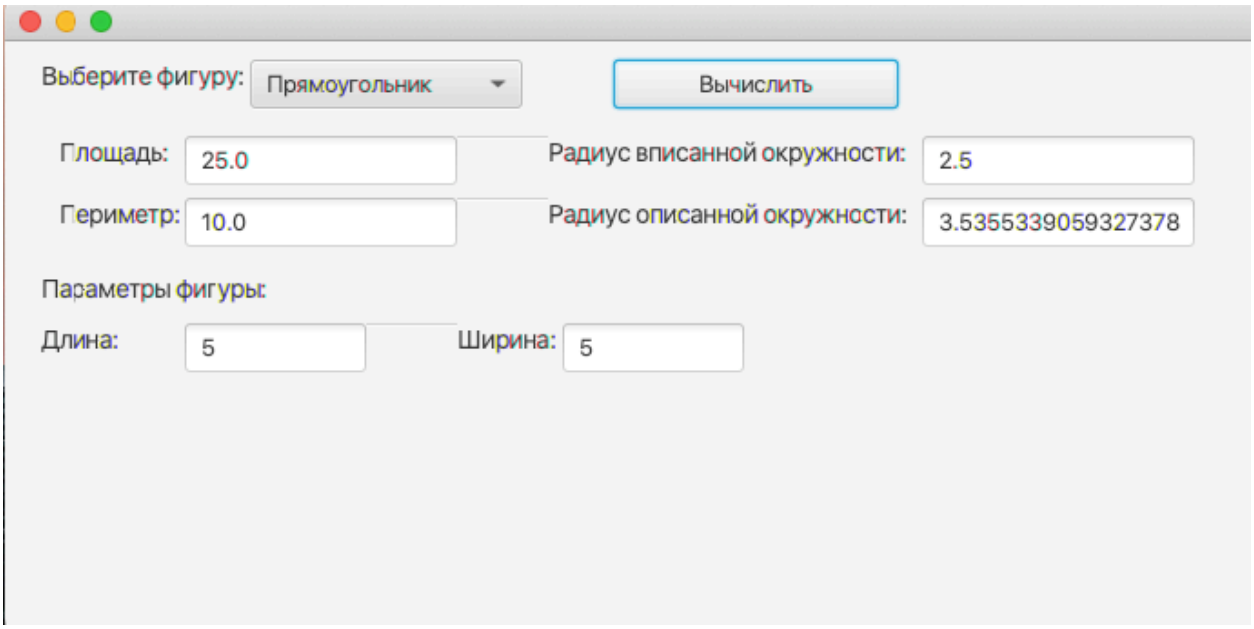

Рисунок 6 – Пример работы программы с фигурой «Прямоугольник»

3.4 Исходный код

Исходный код MasterView

```
package com.github.sindicat.lab6_1.views

import com.github.sindicat.lab6_1.controller.MasterViewController
import com.github.sindicat.lab6_1.dto.InputViewData
import com.github.sindicat.lab6_1.dto.OutputViewData
import javafx.collections.FXCollections
import javafx.scene.control.ChoiceBox
import javafx.scene.control.Label
import javafx.scene.control.TextField
import javafx.scene.layout.BorderPane
import tornadofx.View


class MasterView : View() {
   override val root: BorderPane by fxml(location = "MasterView.fxml")

   val area: TextField by fxid()
   val perimeter: TextField by fxid()
   val inRadius: TextField by fxid()
   val outRadius: TextField by fxid()
   val length: TextField by fxid()
   val height: TextField by fxid()
   private val aSide: TextField by fxid()
   private val bSide: TextField by fxid()
```

```
    private val cSide: TextField by fxid()
    private val aSideLabel: Label by fxid()
    private val bSideLabel: Label by fxid()
    private val cSideLabel: Label by fxid()
    private val lengthLabel: Label by fxid()
    private val heightLabel: Label by fxid()
    private val infoLabel: Label by fxid()

    val shapeSelector: ChoiceBox<ShapeChoice> by fxid()

    val masterViewController: MasterViewController = MasterViewController()

    init {
        val choices = FXCollections.observableArrayList<ShapeChoice>()
        choices.add(ShapeChoice.RECTANGLE)
        choices.add(ShapeChoice.RHOMBUS)
        choices.add(ShapeChoice.TRIANGLE)
        shapeSelector.items = choices
        shapeSelector.value = ShapeChoice.TRIANGLE
        setNameForLabels("Основание: ", "Высота: ")

        shapeSelector.selectionModel.selectedItemProperty().addListener { _, _, newValue:
ShapeChoice ->
            updatedFormWithCalculatedMeasures(OutputViewData())
            when (newValue) {
                ShapeChoice.TRIANGLE -> {
                    setVisibilityTriangleSpecificInputs(isVisible = true)
                    setNameForLabels("Основание: ", "Высота: ")
                }
                ShapeChoice.RECTANGLE -> {
                    setVisibilityTriangleSpecificInputs(isVisible = false)
                    setNameForLabels("Длина: ", "Ширина: ")
                }
                ShapeChoice.RHOMBUS -> {
                    setVisibilityTriangleSpecificInputs(isVisible = false)
                    setNameForLabels("Первая диагональ: ", "Вторая диагональ: ")
                }
            }
        }
    }

    private fun setVisibilityTriangleSpecificInputs(isVisible: Boolean) {
        aSide.isVisible = isVisible
        bSide.isVisible = isVisible
        cSide.isVisible = isVisible
        aSideLabel.isVisible = isVisible
        bSideLabel.isVisible = isVisible
        cSideLabel.isVisible = isVisible
    }

    private fun setNameForLabels(lengthLabelName: String, heightLabelName: String) {
        lengthLabel.text = lengthLabelName
        heightLabel.text = heightLabelName
    }
```

```
    fun onCalculateButtonPressed() {
        val outputViewData: OutputViewData = masterViewController.getUpdatedViewData(
            InputViewData(
                length = length.text,
                height = height.text,
                aSide = aSide.text,
                bSide = bSide.text,
                cSide = cSide.text,
                shapeChoice = shapeSelector.value
            )
        )
        updatedFormWithCalculatedMeasures(outputViewData)
    }

    private fun updatedFormWithCalculatedMeasures(outputViewData: OutputViewData) {
        area.text = outputViewData.area
        perimeter.text = outputViewData.perimeter
        inRadius.text = outputViewData.internalRadius
        outRadius.text = outputViewData.externalRadius
        infoLabel.text = outputViewData.infoMsg
    }
}

enum class ShapeChoice(private var shapeName: String) {
    RHOMBUS("Ромб"),
    TRIANGLE("Треугольник"),
    RECTANGLE("Прямоугольник");

    override fun toString(): String = shapeName
}
```

Исходный код MasterViewController

```
package com.github.sindicat.lab6_1.controller

import com.github.sindicat.lab6_1.dto.InputViewData
import com.github.sindicat.lab6_1.dto.OutputViewData
import com.github.sindicat.lab6_1.model.Rectangle
import com.github.sindicat.lab6_1.model.Rhombus
import com.github.sindicat.lab6_1.model.Shape
import com.github.sindicat.lab6_1.model.Triangle
import com.github.sindicat.lab6_1.views.ShapeChoice

const val NOT_EXIST_MSG = "Не существует"

class MasterViewController {

    fun getUpdatedViewData(inputViewData: InputViewData): OutputViewData {
        if (inputViewData.shapeChoice == ShapeChoice.TRIANGLE
            && isTriangleNotExist(
                a = inputViewData.aSide.toDoubleOrNull(),
                b = inputViewData.bSide.toDoubleOrNull(),
```

```
            c = inputViewData.cSide.toDoubleOrNull()
        )
    ) {
        return OutputViewData(infoMsg = "Треугольника с заданными сторонами не существует!")
    }

    val shape: Shape = createShape(inputViewData)
    return createOutputViewData(shape)
  }

  private fun createShape(inputViewData: InputViewData): Shape {
    return when (inputViewData.shapeChoice) {
      ShapeChoice.RHOMBUS -> Rhombus(
        diagonalA = inputViewData.length.toDoubleOrNull(),
        diagonalB = inputViewData.height.toDoubleOrNull()
      )
      ShapeChoice.RECTANGLE -> Rectangle(
        length = inputViewData.length.toDoubleOrNull(),
        height = inputViewData.height.toDoubleOrNull()
      )
      ShapeChoice.TRIANGLE -> Triangle(
        a = inputViewData.aSide.toDoubleOrNull(),
        b = inputViewData.bSide.toDoubleOrNull(),
        c = inputViewData.cSide.toDoubleOrNull(),
        base = inputViewData.length.toDoubleOrNull(),
        height = inputViewData.height.toDoubleOrNull()
      )
    }
  }

  private fun createOutputViewData(shape: Shape): OutputViewData = OutputViewData().apply {
    area = shape.getArea()?.toString() ?: ""
    perimeter = shape.getPerimeter()?.toString() ?: ""
    internalRadius = shape.getRadiusInternal()?.let { getDecodedValue(it) } ?: ""
    externalRadius = shape.getRadiusExternal()?.let { getDecodedValue(it) } ?: ""
  }

  private fun getDecodedValue(value: Double): String = if (value == -1.0) NOT_EXIST_MSG else value.toString()

  private fun isTriangleNotExist(a: Double?, b: Double?, c: Double?): Boolean =
    if (a == null || b == null || c == null) false else !(a < b + c && b < a + c && c < a + b)
}
```

Исходный исходный код Shape

```
package com.github.sindicat.lab6_1.model

interface Shape {

    fun getArea(): Double?

    fun getPerimeter(): Double?

    fun getRadiusExternal(): Double?

    fun getRadiusInternal(): Double?

}
```

Исходный исходный код Triangle

```
package com.github.sindicat.lab6_1.model

import kotlin.math.pow

class Triangle(
    val a: Double?,
    val b: Double?,
    val c: Double?,
    val base: Double?,
    val height: Double?
) : Shape {

    override fun getArea(): Double? {
        return if (areSidesSet()) {
            (1.0 / 4) * ((a!! + b!! + c!!) * (b + c - a) * (a + c - b) * (a + b - c)).pow(0.5)

        } else if (base != null && height != null) {
            0.5 * base * height
        } else null
    }

    override fun getPerimeter(): Double? = if (areSidesSet()) a!! + b!! + c!! else null

    override fun getRadiusExternal(): Double? {
        return if (areSidesSet() && getArea() != null) {
            (a!! * b!! * c!!) / (4 * getArea()!!)
        } else null
    }

    override fun getRadiusInternal(): Double? {
        return if (getHalfOfPerimeter() != null) {
            (((getHalfOfPerimeter()!! - a!!) * (getHalfOfPerimeter()!! - b!!) * (getHalfOfPerimeter()!! -
c!!)) / (4 * getHalfOfPerimeter()!!)).pow(0.5)
        } else null
    }

    private fun getHalfOfPerimeter(): Double? = if (getPerimeter() != null) getPerimeter()!! / 2
else null

    private fun areSidesSet(): Boolean = a != null && b != null && c != null
}
```

Исходный исходный код Rhombus

```
package com.github.sindicat.lab6_1.model

import kotlin.math.pow

class Rhombus(val diagonalA: Double?, val diagonalB: Double?) : Shape {

  override fun getArea(): Double? {
    return if (diagonalA != null && diagonalB != null) (diagonalA * diagonalB) / 2
    else null
  }

  override fun getPerimeter(): Double? {
    return if (diagonalA != null && diagonalB != null) 4 * calculateSideLength()
    else null
  }

  override fun getRadiusExternal(): Double? {
    return if (diagonalA != null && diagonalB != null) {
      if (isSquare()) diagonalA / 2 else -1.0
    } else null
  }

  override fun getRadiusInternal(): Double? {
    return if (diagonalA != null && diagonalB != null) (diagonalA * diagonalB) / (4 *
calculateSideLength())
    else null
  }

  private fun calculateSideLength(): Double = (diagonalA!!.pow(2) +
diagonalB!!.pow(2)).pow(0.5)

  private fun isSquare(): Boolean {
    return if (diagonalA == null || diagonalB == null) false
    else diagonalA.compareTo(diagonalB) == 0
  }

}
```

Исходный исходный код Rectangle

```
package com.github.sindicat.lab6_1.model

import kotlin.math.pow

class Rectangle(val length: Double?, val height: Double?) : Shape {

   override fun getArea(): Double? = if (length == null || height == null) null else length *
height

   override fun getPerimeter(): Double? = if (length == null || height == null) null else length
+ height

   override fun getRadiusExternal(): Double? = if (length == null || height == null) null
   else ((length.pow(2) + height.pow(2)).pow(0.5)) / 2

   override fun getRadiusInternal(): Double? =  if (length == null || height == null) null
   else if (isSquare()) length / 2 else -1.0

   private fun isSquare(): Boolean =  if (length == null || height == null) false  else
length.compareTo(height) == 0

}
```

Вывод

В ходе выполнения лабораторной работы были получены знания об архитектуре приложения, управляемой моделью MDA.

Было создано простое MDA-приложение для расчета площади, периметра, радиусов фигур. В качестве модели была выбрана UML-диаграмма классов. Было разработано приложение в программной среде IntelliJ IDEA на языке Kotlin.

Вывод

В процессе выполнения лабораторной работы были изучены основы разработки клиент-серверных Web-приложений с использованием технологий Spring Boot и Vue JS, а также разработан Web-сервер и Web-клиент для него.