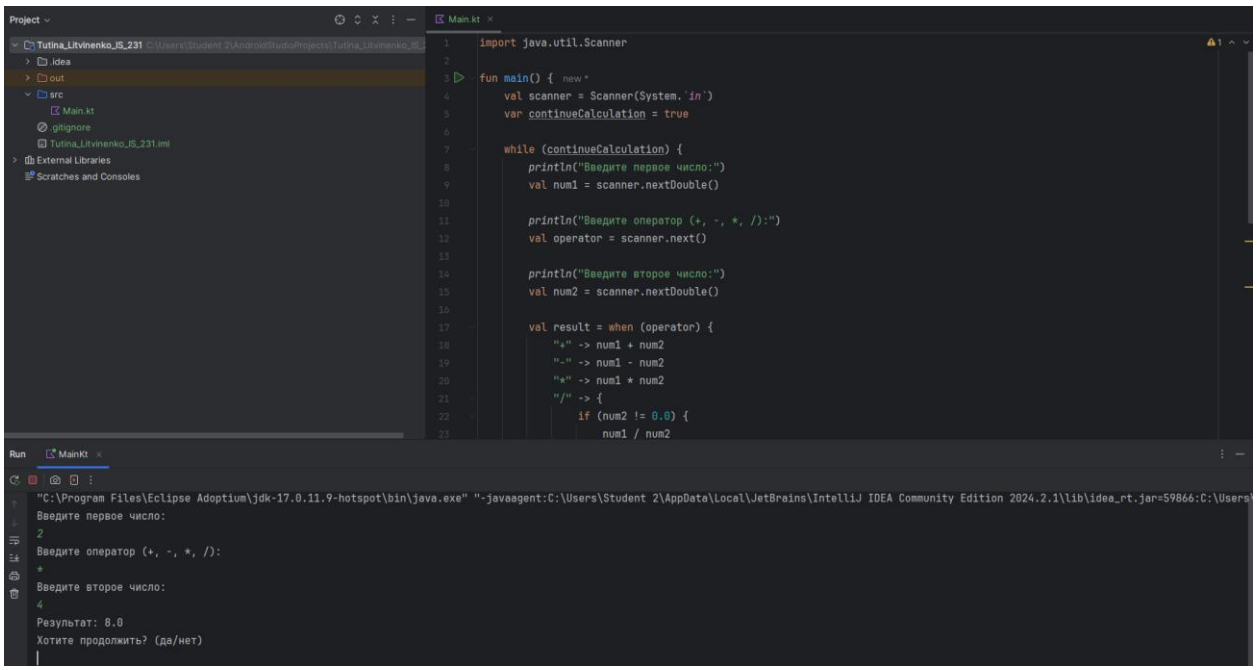
Практическая работа № 7.2

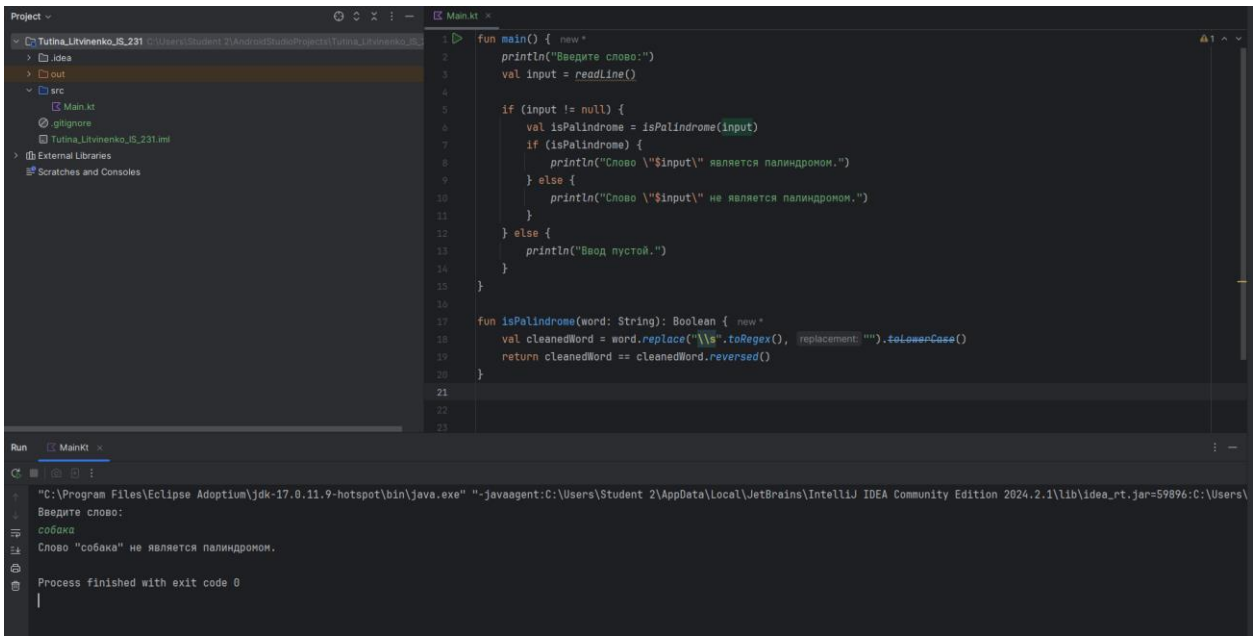
1.

```
import java.util.Scanner

fun main() {
    val scanner = Scanner(System.`in`)
    var continueCalculation = true

    while (continueCalculation) {
        println("Введите первое число:")
        val num1 = scanner.nextDouble()

        println("Введите оператор (+, -, *, /):")
        val operator = scanner.next()

        println("Введите второе число:")
        val num2 = scanner.nextDouble()

        val result = when (operator) {
            "+" -> num1 + num2
            "-" -> num1 - num2
            "*" -> num1 * num2
            "/" -> {
                if (num2 != 0.0) {
                    num1 / num2
                } else {
                    println("Ошибка: деление на ноль!")
                    continueCalculation = false
                    null
                }
            }
            else -> {
                println("Недопустимый оператор!")
                continueCalculation = false
                null
            }
        }

        result?.let { println("Результат: $it") }

        println("Хотите продолжить? (да/нет)")
        val answer = scanner.next()
        continueCalculation = answer.equals("да", ignoreCase = true)
    }
}
```

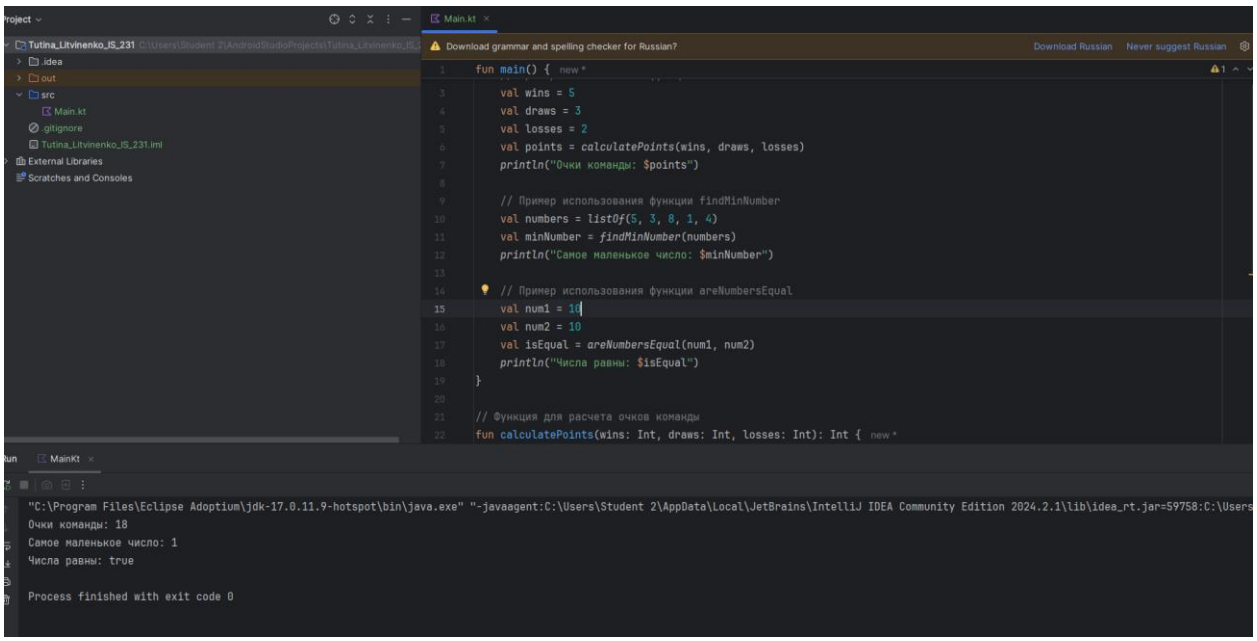


2.

```
fun main() {
    println("Введите слово:")
    val input = readLine()

    if (input != null) {
        val isPalindrome = isPalindrome(input)
        if (isPalindrome) {
            println("Слово \"$input\" является палиндромом.")
        } else {
            println("Слово \"$input\" не является палиндромом.")
        }
    } else {
        println("Ввод пустой.")
    }
}

fun isPalindrome(word: String): Boolean {
    val cleanedWord = word.replace("\\s".toRegex(), "").toLowerCase()
    return cleanedWord == cleanedWord.reversed()
}
```



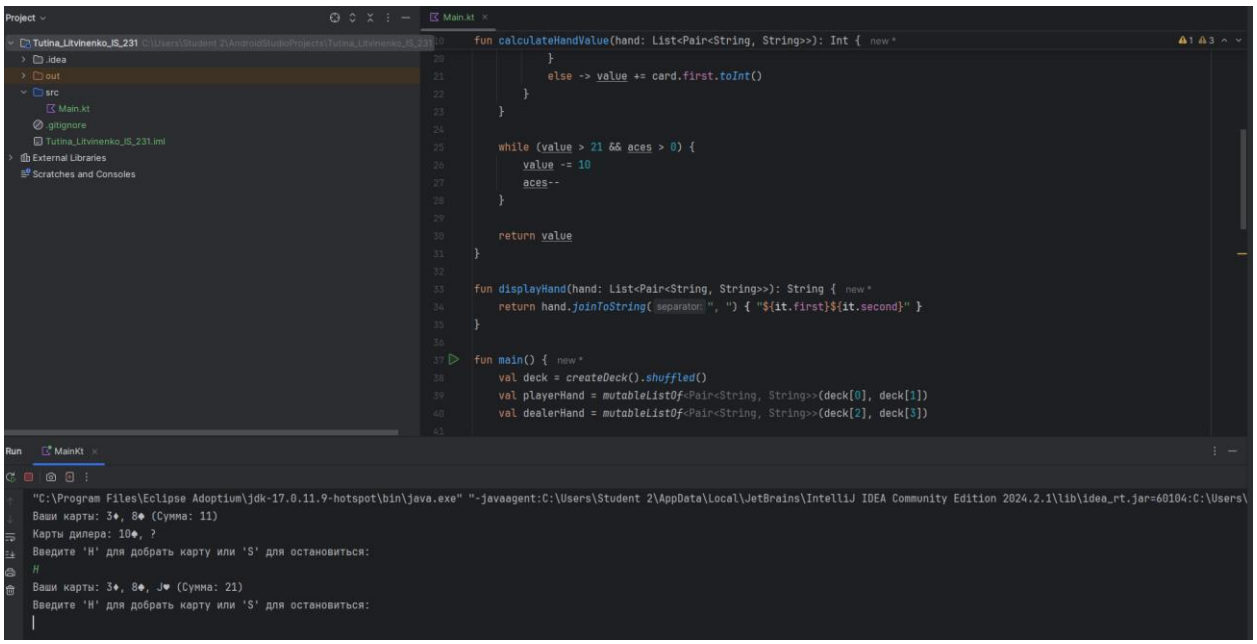
3. fun main()
```
    // Пример использования функции calculatePoints
    val wins = 5
    val draws = 3
    val losses = 2
    val points = calculatePoints(wins, draws, losses)
    println("Очки команды: $points")

    // Пример использования функции findMinNumber
    val numbers = listOf(5, 3, 8, 1, 4)
    val minNumber = findMinNumber(numbers)
    println("Самое маленькое число: $minNumber")

    // Пример использования функции areNumbersEqual
    val num1 = 10
    val num2 = 10
    val isEqual = areNumbersEqual(num1, num2)
    println("Числа равны: $isEqual")
}

// Функция для расчета очков команды
fun calculatePoints(wins: Int, draws: Int, losses: Int): Int {
    return wins * 3 + draws * 1 + losses * 0 // Поражения не дают очков
}

// Функция для нахождения самого маленького числа в списке
fun findMinNumber(numbers: List<Int>): Int? {
    return numbers.minOrNull() // Возвращает минимальное число или null, если список пуст
}

// Функция для сравнения двух чисел
fun areNumbersEqual(num1: Int, num2: Int): Boolean {
    return num1 == num2 // Возвращает true, если числа равны, иначе false
}
```



4.

```
import kotlin.random.Random

fun createDeck(): List<Pair<String, String>> {
    val suits = listOf("♠", "♥", "♦", "♣")
    val values = listOf("2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A")

    return values.flatMap { value -> suits.map { suit -> Pair(value, suit) } }
}

fun calculateHandValue(hand: List<Pair<String, String>>): Int {
    var value = 0
    var aces = 0

    for (card in hand) {
        when (card.first) {
            in listOf("J", "Q", "K") -> value += 10
            "A" -> {
                aces++
                value += 11
            }
            else -> value += card.first.toInt()
        }
    }

    while (value > 21 && aces > 0) {
        value -= 10
        aces--
    }

    return value
}

fun displayHand(hand: List<Pair<String, String>>): String {
    return hand.joinToString(", ") { "${it.first}${it.second}" }
}

fun main() {
```

```
    val deck = createDeck().shuffled()
    val playerHand = mutableListOf<Pair<String, String>>(deck[0], deck[1])
    val dealerHand = mutableListOf<Pair<String, String>>(deck[2], deck[3])

    println("Ваши карты: ${displayHand(playerHand)} (Сумма: ${calculateHandValue(playerHand)})")
    println("Карты дилера: ${dealerHand[0].first}${dealerHand[0].second}, ?")

    while (true) {
        println("Введите 'H' для добрать карту или 'S' для остановиться:")
        val action = readLine()?.trim()?.toUpperCase()

        when (action) {
            "H" -> {
                playerHand.add(deck[playerHand.size + dealerHand.size])
                val playerValue = calculateHandValue(playerHand)
                println("Ваши карты: ${displayHand(playerHand)} (Сумма: $playerValue)")
                if (playerValue > 21) {
                    println("Вы проиграли! (Превышение 21)")
                    return
                }
            }
            "S" -> break
            else -> println("Неверный ввод. Попробуйте снова.")
        }
    }

    var dealerValue = calculateHandValue(dealerHand)
    while (dealerValue < 17) {
        dealerHand.add(deck[playerHand.size + dealerHand.size])
        dealerValue = calculateHandValue(dealerHand)
    }

    println("Карты дилера: ${displayHand(dealerHand)} (Сумма: $dealerValue)")

    when {
        dealerValue > 21 -> println("Вы выиграли! (Дилер превышает 21)")
        calculateHandValue(playerHand) > dealerValue -> println("Вы выиграли!")
        calculateHandValue(playerHand) < dealerValue -> println("Вы проиграли!")
        else -> println("Ничья!")
    }
}
```


Project
Tutina_Litvinenko_IS_231 C:\Users\Student 2\AndroidStudioProjects\Tutina_Litvinenko_IS_231
.idea
out
src
Main.kt
.gitignore
Tutina_Litvinenko_IS_231.iml
External Libraries
Scratches and Consoles
Main.kt
fun calculateHandValue(hand: List<Pair<String, String>>): Int { new *
}
else -> value += card.first.toInt()
}
}
while (value > 21 && aces > 0) {
value -= 10
aces--
}
return value
}
fun displayHand(hand: List<Pair<String, String>>): String { new *
return hand.joinToString(separator: ", ") { "${it.first}${it.second}" }
}
fun main() { new *
val deck = createDeck().shuffled()
val playerHand = mutableListOf<Pair<String, String>>(deck[0], deck[1])
val dealerHand = mutableListOf<Pair<String, String>>(deck[2], deck[3])
Run
MainKt
"C:\Program Files\Eclipse Adoptium\jdk-17.0.11.9-hotspot\bin\java.exe" "-javaagent:C:\Users\Student 2\AppData\Local\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=60104:C:\Users\
Ваши карты: 3♦, 8♣ (Сумма: 11)
Карты дилера: 10♣, ?
Введите 'H' для добрать карту или 'S' для остановиться:
H
Ваши карты: 3♦, 8♣, J♥ (Сумма: 21)
Введите 'H' для добрать карту или 'S' для остановиться: