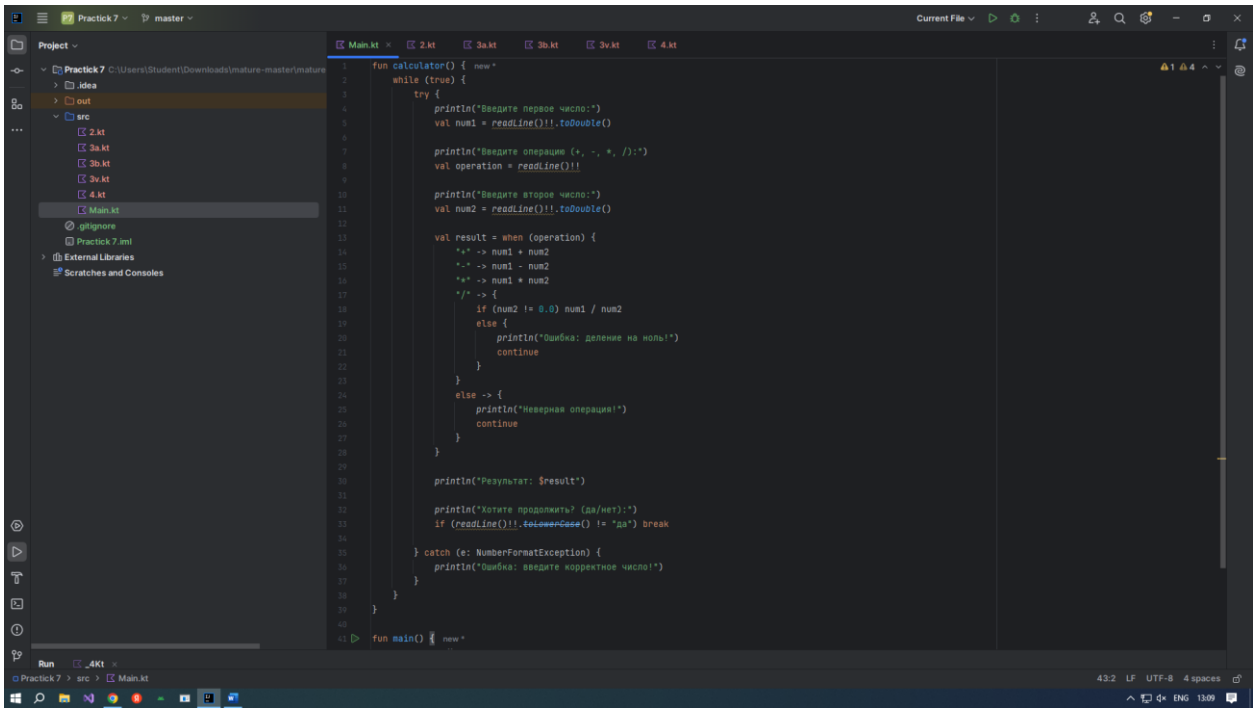
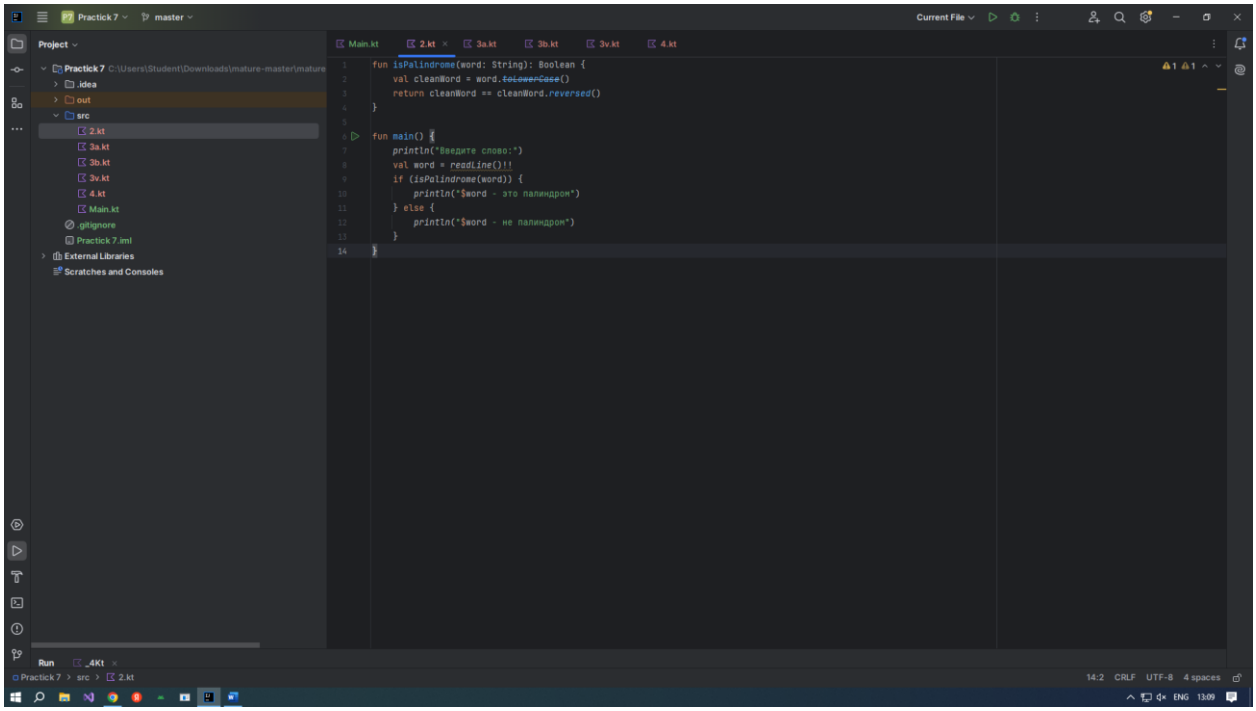
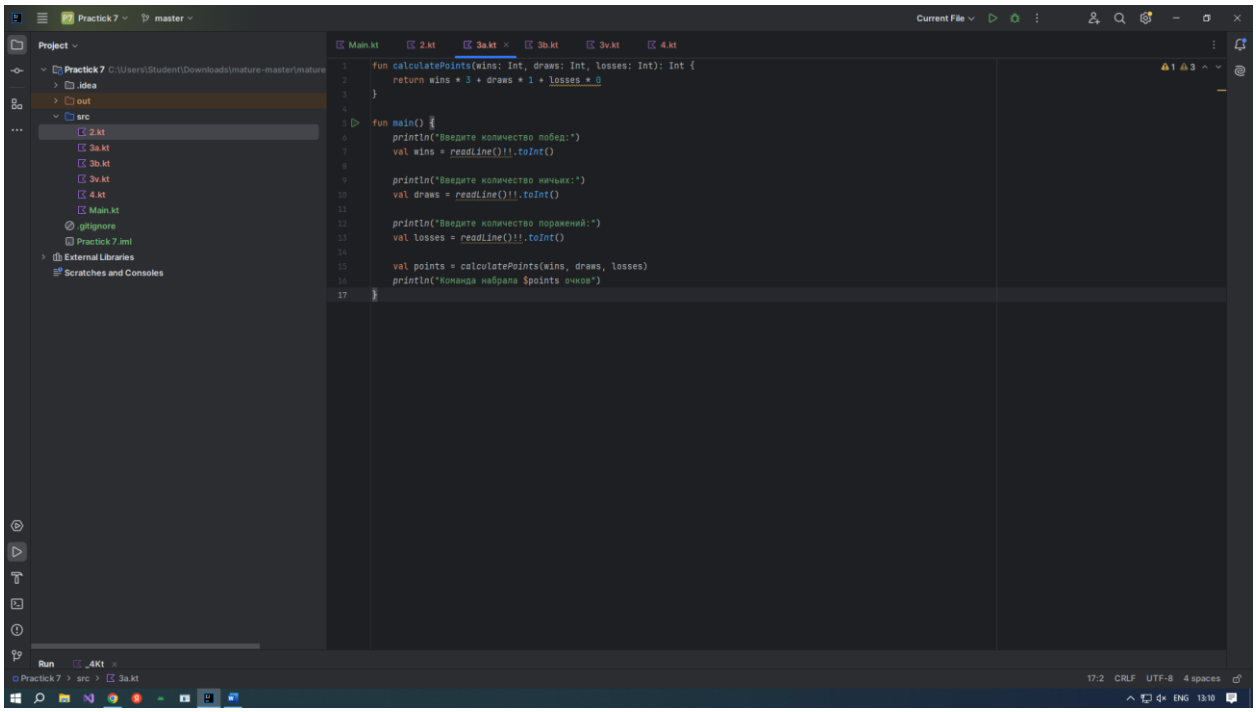
1


fun calculator() {
while (true) {
try {
println("Введите первое число:")
val num1 = readLine()!!.toDouble()
println("Введите операцию (+, -, *, /):")
val operation = readLine()!!
println("Введите второе число:")
val num2 = readLine()!!.toDouble()
val result = when (operation) {
"+" -> num1 + num2
"-" -> num1 - num2
"*" -> num1 * num2
"/" -> {
if (num2 != 0.0) num1 / num2
else {
println("Ошибка: деление на ноль!")
continue
}
}
else -> {
println("Неверная операция!")
continue
}
}
println("Результат: $result")
println("Хотите продолжить? (да/нет):")
if (readLine()!!.toLowerCase() != "да") break
} catch (e: NumberFormatException) {
println("Ошибка: введите корректное число!")
}
}
}
fun main() {

2

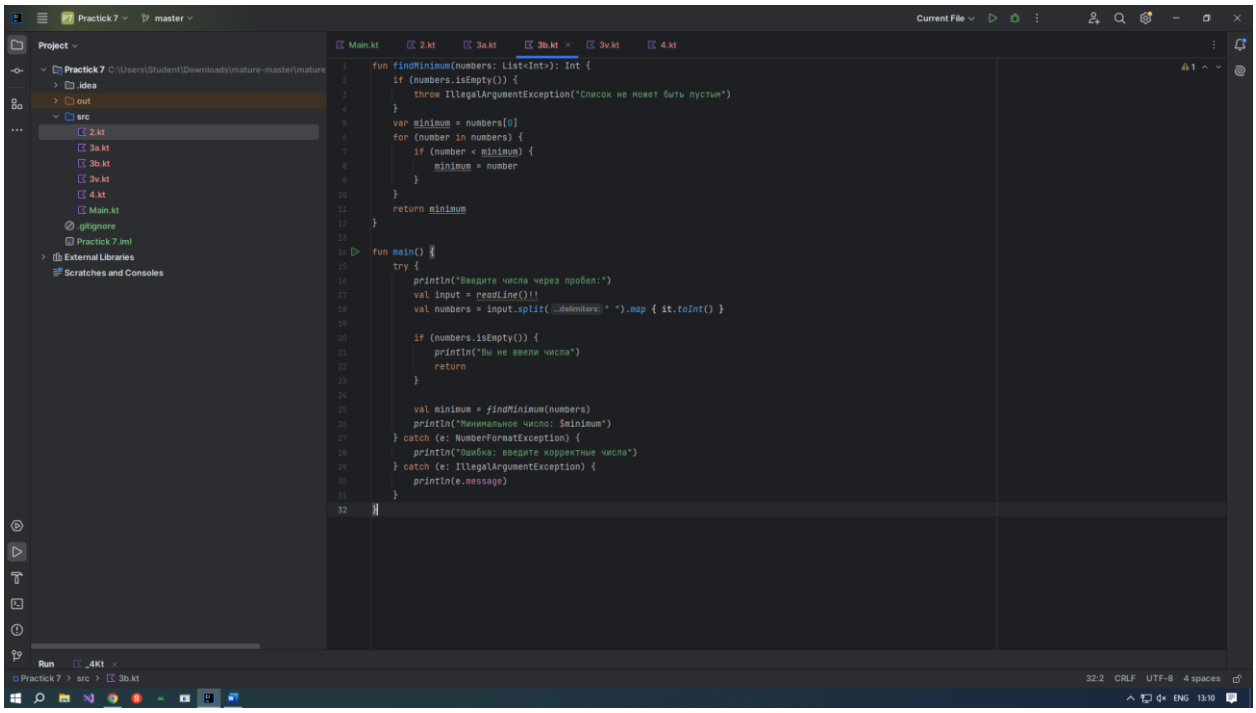

fun isPalindrome(word: String): Boolean {
val cleanWord = word.toLowerCase()
return cleanWord == cleanWord.reversed()
}
fun main() {
println("Введите слово:")
val word = readLine()!!
if (isPalindrome(word)) {
println("$word - это палиндром")
} else {
println("$word - не палиндром")
}
}

3a



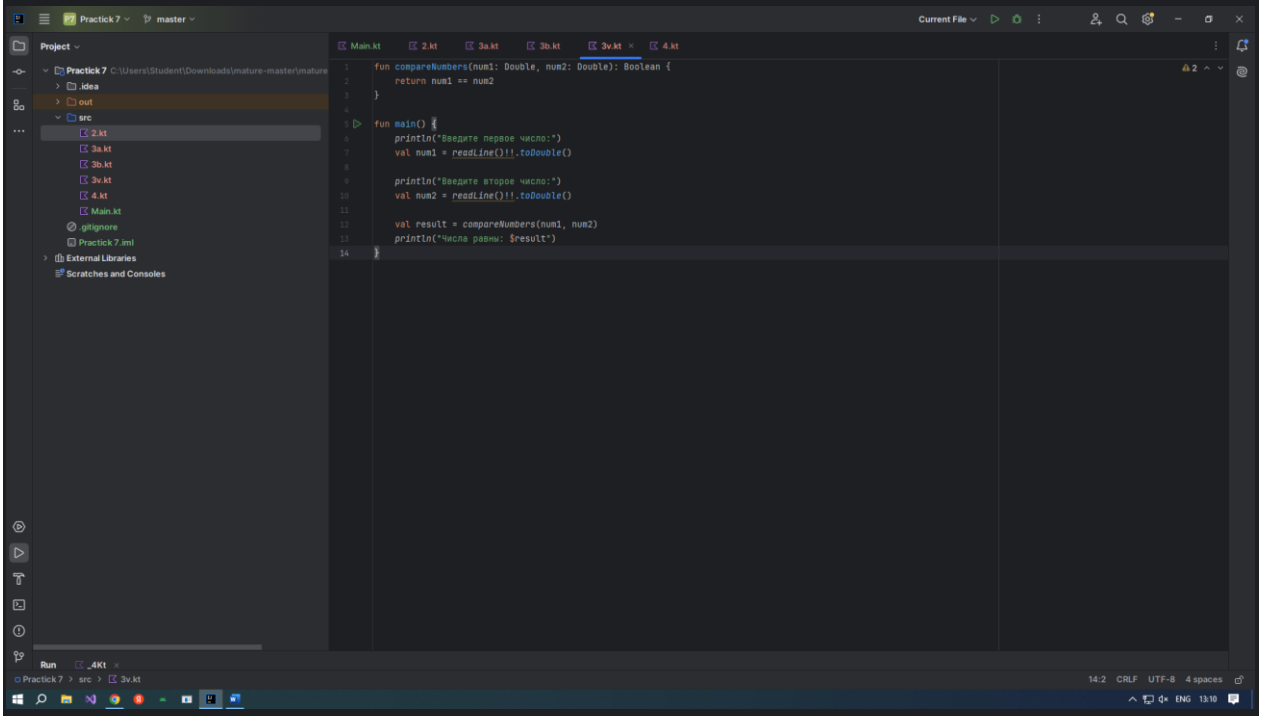
```kotlin
fun calculatePoints(wins: Int, draws: Int, losses: Int): Int {
    return wins * 3 + draws * 1 + losses * 0
}

fun main() {
    println("Введите количество побед:")
    val wins = readLine()!!.toInt()

    println("Введите количество ничьих:")
    val draws = readLine()!!.toInt()

    println("Введите количество поражений:")
    val losses = readLine()!!.toInt()

    val points = calculatePoints(wins, draws, losses)
    println("Команда набрала $points очков")
}
```

3b



```kotlin
fun findMinimum(numbers: List<Int>): Int {
    if (numbers.isEmpty()) {
        throw IllegalArgumentException("Список не может быть пустым")
    }
    var minimum = numbers[0]
    for (number in numbers) {
        if (number < minimum) {
            minimum = number
        }
    }
    return minimum
}

fun main() {
    try {
        println("Введите числа через пробел:")
        val input = readLine()!!
        val numbers = input.split( ...delimiters: " ").map { it.toInt() }

        if (numbers.isEmpty()) {
            println("Вы не ввели числа")
            return
        }

        val minimum = findMinimum(numbers)
        println("Минимальное число: $minimum")
    } catch (e: NumberFormatException) {
        println("Ошибка: введите корректные числа")
    } catch (e: IllegalArgumentException) {
        println(e.message)
    }
}
```

3v



4

```
class Game21 {
    private val deck = mutableListOf<Card>()
    private var playerHand = mutableListOf<Card>()
    private var dealerHand = mutableListOf<Card>()

    data class Card(val rank: String, val value: Int) {
        override fun toString() = rank
    }

    init {
        // колода
        val ranks = listOf("2", "3", "4", "5", "6", "7", "8", "9", "10", "В",
"Д", "К", "Т")
        val values = listOf(2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 11)

        for (i in ranks.indices) {
            repeat(4) { deck.add(Card(ranks[i], values[i])) }
        }
        deck.shuffle()
    }

    private fun dealCard(): Card = deck.removeFirst()

    private fun calculateScore(hand: List<Card>): Int {
        var score = 0
        var aces = 0

        for (card in hand) {
            if (card.value == 11) aces++
            score += card.value
        }

        while (score > 21 && aces > 0) {
```

```
            score -= 10
            aces--
        }

        return score
    }

    fun play() {
        println("Добро пожаловать в игру 21!")

        // раздача
        playerHand.add(dealCard())
        dealerHand.add(dealCard())
        playerHand.add(dealCard())

        // карты
        println("\nВаши карты: ${playerHand.joinToString()}
(${calculateScore(playerHand)})")
        println("Карта дилера: ${dealerHand[0]}")

        // ход игрока
        while (true) {
            println("\nВзять карту? (yes/no)")
            val answer = readLine()?.toLowerCase()

            if (answer == "yes") {
                playerHand.add(dealCard())
                val score = calculateScore(playerHand)
                println("Ваши карты: ${playerHand.joinToString()} ($score)")

                if (score > 21) {
                    println("Перебор! Вы проиграли!")
                    return
                }
            } else if (answer == "no") {
                break
            }
        }

        // ход дилера
        println("\nХод дилера:")
        while (calculateScore(dealerHand) < 17) {
            dealerHand.add(dealCard())
            println("Дилер берет карту: ${dealerHand.last()}")
        }

        // подсчет
        val playerScore = calculateScore(playerHand)
        val dealerScore = calculateScore(dealerHand)

        println("\nВаши карты: ${playerHand.joinToString()} ($playerScore)")
        println("Карты дилера: ${dealerHand.joinToString()} ($dealerScore)")

        when {
            dealerScore > 21 -> println("Дилер перебрал! Вы выиграли!")
            playerScore > dealerScore -> println("Вы выиграли!")
            playerScore < dealerScore -> println("Дилер выиграл!")
            else -> println("Ничья!")
        }
    }
}

fun main() {
    val game = Game21()
```

```
    game.play()
}
```