

1

```

1 fun calculator() { new *
2 while (true) {
3     try {
4         println("Введите первое число:")
5         val num1 = readLine()!!.toDouble()
6
7         println("Введите операцию (+, -, *, /):")
8         val operation = readLine()!!
9
10        println("Введите второе число:")
11        val num2 = readLine()!!.toDouble()
12
13        val result = when (operation) {
14            "+" -> num1 + num2
15            "-" -> num1 - num2
16            "*" -> num1 * num2
17            "/" -> {
18                if (num2 != 0.0) num1 / num2
19                else {
20                    println("Ошибка: деление на ноль!")
21                    continue
22                }
23            }
24            else -> {
25                println("Неверная операция!")
26                continue
27            }
28        }
29
30        println("Результат: $result")
31
32        println("Хотите продолжить? (да/нет):")
33        if (readLine()!!.toLowerCase() != "да") break
34
35    } catch (e: NumberFormatException) {
36        println("Ошибка: введите корректное число!")
37    }
38 }
39 }
40
41 fun main() { new *

```

Run \_4K1 x  
Practic7 > src > Main.kt  
43.2 LF UTF-8 4 spaces

2

```

1 fun isPalindrome(word: String): Boolean {
2     val cleanWord = word.toLowerCase().replace("[^a-zA-Z0-9_]+", "")
3     return cleanWord == cleanWord.reversed()
4 }
5
6 fun main() {
7     println("Введите слово:")
8     val word = readLine()!!
9     if (isPalindrome(word)) {
10        println("$word - это палиндром")
11    } else {
12        println("$word - не палиндром")
13    }
14 }

```

Run \_4K1 x  
Practic7 > src > 2.kt  
14.2 CRLF UTF-8 4 spaces

3a

```

1 fun calculatePoints(wins: Int, draws: Int, losses: Int): Int {
2     return wins * 3 + draws * 1 + losses * 0
3 }
4
5 fun main() {
6     println("Введите количество побед:")
7     val wins = readLine()!!.toInt()
8
9     println("Введите количество ничьих:")
10    val draws = readLine()!!.toInt()
11
12    println("Введите количество поражений:")
13    val losses = readLine()!!.toInt()
14
15    val points = calculatePoints(wins, draws, losses)
16    println("Команда набрала $points очков")
17 }
  
```

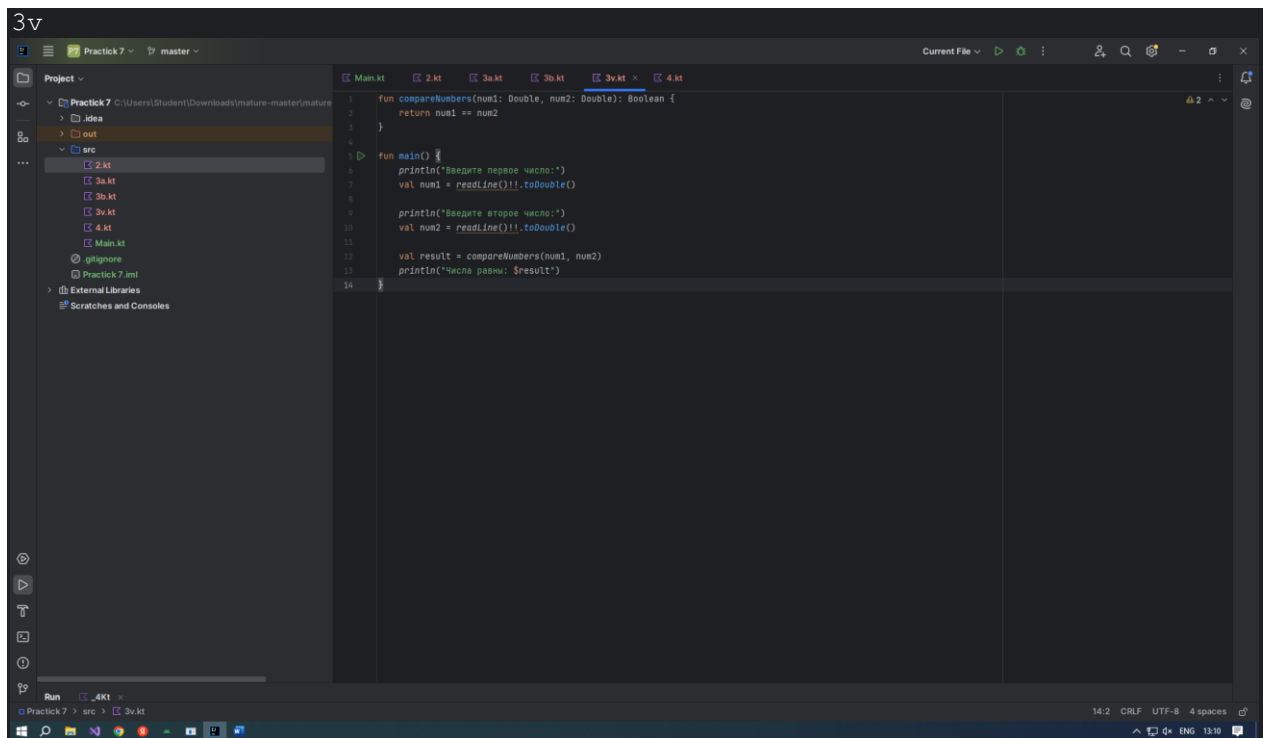
The screenshot shows an IDE with a project named 'Practic7'. The file explorer on the left shows a directory structure with files like '2.kt', '3a.kt', '3b.kt', '3v.kt', '4.kt', 'Main.kt', 'gIgnore', and 'Practic7.iml'. The main editor displays the code for '3a.kt'. The code defines a function 'calculatePoints' that takes three integers (wins, draws, losses) and returns a total score based on the formula: wins \* 3 + draws \* 1 + losses \* 0. The 'main' function prompts the user to enter the number of wins, draws, and losses, reads the input, and then prints the total points.

3b

```

1 fun findMinimum(numbers: List<Int>): Int {
2     if (numbers.isEmpty()) {
3         throw IllegalArgumentException("Чисел не может быть пустым")
4     }
5     var minimum = numbers[0]
6     for (number in numbers) {
7         if (number < minimum) {
8             minimum = number
9         }
10    }
11    return minimum
12 }
13
14 fun main() {
15    try {
16        println("Введите числа через пробел:")
17        val input = readLine()!!
18        val numbers = input.split("\\s+").map { it.toInt() }
19
20        if (numbers.isEmpty()) {
21            println("Вы не ввели числа")
22            return
23        }
24
25        val minimum = findMinimum(numbers)
26        println("Минимальное число: $minimum")
27    } catch (e: NumberFormatException) {
28        println("Ошибка: введите корректные числа")
29    } catch (e: IllegalArgumentException) {
30        println(e.message)
31    }
32 }
  
```

The screenshot shows the same IDE with the project 'Practic7'. The file explorer shows the same directory structure. The main editor displays the code for '3b.kt'. The code defines a function 'findMinimum' that takes a list of integers and returns the minimum value. It includes a check for an empty list and throws an exception. The 'main' function prompts the user to enter numbers separated by spaces, splits the input into a list of integers, and then prints the minimum value. It also includes error handling for 'NumberFormatException' and 'IllegalArgumentException'.



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", "B",
            "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: ${playerHand.joinToString()}")
    println("${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()  
}
```