Informe de Programas Kotlin

Esther Mariana Chunga Pacheco

25 de Agosto de 2025

1. Introducción

Este documento presenta el análisis y desarrollo de cuatro programas implementados en Kotlin, enfocándose en el uso de lectura de datos por teclado y generación de valores aleatorios.

2. Conceptos Clave Utilizados

2.1. Lectura de Datos por Teclado

Se utilizó `readLine()` y `readln()` para la lectura de datos ingresados por el usuario:

- `readLine()?:` Lee una línea de texto y retorna `null` si ocurre un error
- `toDoubleOrNull()`, `toIntOrNull()`: Conversión segura a tipos numéricos
- Operador Elvis (`?:`): Proporciona valores por defecto en caso de valores nulos

2.2. Generación de Valores Aleatorios

Se empleó `Random` de `kotlin.random` para generar valores aleatorios:

- `Random.nextInt(from, until)`: Números enteros en rango
- `Random.nextInt(size)`: Índices aleatorios para listas

3. Programas Desarrollados

3.1. 1. Evaluación de Empleados

```
/*
    Descripci n: Calcula el nivel de rendimiento y el dinero extra que
   recibir  un empleado seg n su puntuaci n.
    Autor: [Tu Nombre]
    Fecha creaci n: 25/08/2025
     ltima  modificaci n: 25/08/2025
*/

fun calcularNivel(puntuacion: Double): String {
```

```
    return when {
        puntuacion in 0.0..3.0 -> "Inaceptable"
        puntuacion in 4.0..6.0 -> "Aceptable"
        puntuacion in 7.0..10.0 -> "Meritorio"
        else -> "Puntuaci n inv lida"
    }
}

fun calcularDineroExtra(salario: Double, puntuacion: Double): Double {
    return salario * (puntuacion / 10)
}

fun main() {
    print("Ingrese su salario mensual: ")
    val salario = readLine()?.toDoubleOrNull() ?: 0.0

    print("Ingrese su puntuaci n (0 a 10): ")
    val puntuacion = readLine()?.toDoubleOrNull() ?: 0.0

    val nivel = calcularNivel(puntuacion)
    if (nivel == "Puntuaci n inv lida") {
        println("Error: la puntuaci n debe estar entre 0 y 10.")
    } else {
        val dineroExtra = calcularDineroExtra(salario, puntuacion)
        println("Nivel de Rendimiento: $nivel")
        println("Cantidad de dinero recibido: \$${"%.2f".format(
   dineroExtra)}")
    }
}
```

Listing 1: EvaluacionEmpleados.kt

Características:

- Uso de `readLine()` para capturar salario y puntuación
- Función `when` para rangos de evaluación
- Validación de entrada con valores por defecto
- Formateo de salida monetaria

3.2. 2. Piedra, Papel o Tijera

```
/*
    Descripci n: Juego de piedra, papel o tijera contra la computadora.
    Autor: [Tu Nombre]
    Fecha creaci n: 25/08/2025
     ltima  modificaci n: 25/08/2025
*/

import kotlin.random.Random

fun obtenerOpcionComputadora(): String {
    val opciones = listOf("piedra", "papel", "tijera")
    return opciones[Random.nextInt(opciones.size)]
}

```

```
fun determinarGanador(usuario: String, computadora: String): String {
    return if (usuario == computadora) {
        "Empate"
    } else if (
        (usuario == "piedra" && computadora == "tijera") ||
        (usuario == "papel" && computadora == "piedra") ||
        (usuario == "tijera" && computadora == "papel")
    ) {
        "Ganaste"
    } else {
        "Perdiste"
    }
}

fun main() {
    println("Elige piedra, papel o tijera: ")
    val usuario = readLine()?.lowercase() ?: ""
    val computadora = obtenerOpcionComputadora()

    if (usuario !in listOf("piedra", "papel", "tijera")) {
        println("Opci n inv lida.")
    } else {
        println("Computadora eligi : $computadora")
        println(determinarGanador(usuario, computadora))
    }
}
```

Listing 2: PiedraPapelTijera.kt

Características:

- `Random.nextInt()` para elección de la computadora
- Validación de opciones del usuario
- Lógica de comparación para determinar ganador
- Manejo de case insensitive con `lowercase()`

3.3. 3. Calculadora Elemental

```
/*
    Descripci n: Calculadora b sica que realiza suma, resta,
   multiplicaci n y divisi n.
    Autor: [Tu Nombre]
    Fecha creaci n: 25/08/2025
     ltima  modificaci n: 25/08/2025
*/

fun mostrarMenu() {
    println("==== Men  ====")
    println("1. Suma")
    println("2. Resta")
    println("3. Multiplicaci n")
    println("4. Divisi n")
    println("5. Salir")
}

```

```
fun suma(a: Double, b: Double) = a + b
fun resta(a: Double, b: Double) = a - b
fun multiplicacion(a: Double, b: Double) = a * b
fun division(a: Double, b: Double): Double? = if (b != 0.0) a / b else
    null

fun main() {
    var opcion: Int

    do {
        mostrarMenu()
        print("Seleccione una opci n: ")
        opcion = readLine()?.toIntOrNull() ?: 0

        if (opcion in 1..4) {
            print("Ingrese el primer n mero: ")
            val num1 = readLine()?.toDoubleOrNull() ?: 0.0

            print("Ingrese el segundo n mero: ")
            val num2 = readLine()?.toDoubleOrNull() ?: 0.0

            val resultado = when (opcion) {
                1 -> suma(num1, num2)
                2 -> resta(num1, num2)
                3 -> multiplicacion(num1, num2)
                4 -> division(num1, num2) ?: run {
                    println("Error: Divisi n entre cero")
                    continue
                }
                else -> 0.0
            }

            println("Resultado: $resultado")
        } else if (opcion != 5) {
            println("Opci n inv lida, intenta de nuevo.")
        }
    } while (opcion != 5)

    println("Saliendo de la calculadora...")
}
```

Listing 3: Calculadora.kt

Características:

- Menú interactivo con `do-while`
- Funciones lambda para operaciones matemáticas
- Validación de división por cero
- Lectura múltiple de datos numéricos

3.4. 4. Adivina el Número

```
/*
    Descripci n: Juego para adivinar un n mero aleatorio entre 1 y 30
    con m ximo 5 intentos.
```

```
    Autor: [Tu Nombre]
    Fecha creaci n: 25/08/2025
     ltima  modificaci n: 25/08/2025
*/

import kotlin.random.Random

fun main() {
    val numeroSecreto = Random.nextInt(1, 31)
    var intentos = 5
    var adivinaste = false

    println("Adivina el n mero entre 1 y 30. Tienes 5 intentos.")

    while (intentos > 0) {
        print("Ingresa tu n mero: ")
        val intento = readLine()?.toIntOrNull() ?: 0

        if (intento == numeroSecreto) {
            println("  Felicidades  ! Adivinaste el n mero.")
            adivinaste = true
            break
        } else if (intento < numeroSecreto) {
            println("El n mero secreto es mayor.")
        } else {
            println("El n mero secreto es menor.")
        }

        intentos--
        println("Te quedan $intentos intentos.")
    }

    if (!adivinaste) {
        println("Game Over. El n mero era $numeroSecreto.")
    }
}
```

Listing 4: AdivinaNumero.kt

Características:

- `Random.nextInt(1, 31)` para número secreto
- Contador de intentos con bucle `while`
- Retroalimentación de mayor/menor
- Manejo de estado del juego

4. Ejecuciones y Resultados

4.1. Evaluación de Empleados

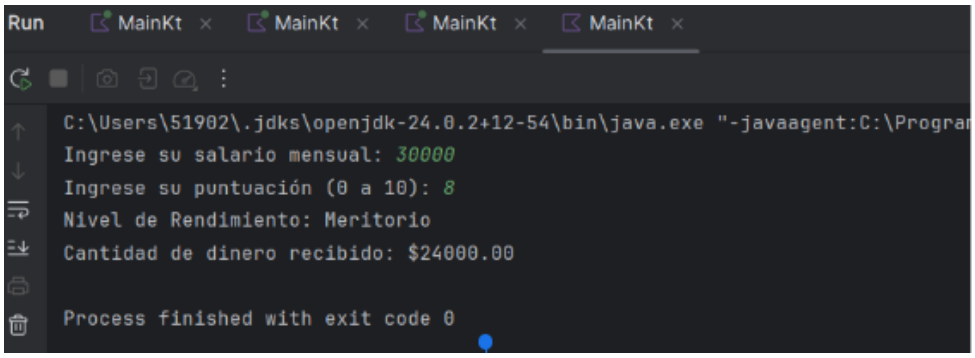

Figura 1: Ejecución del programa EvaluacionEmpleados.kt

4.2. Piedra, Papel o Tijera

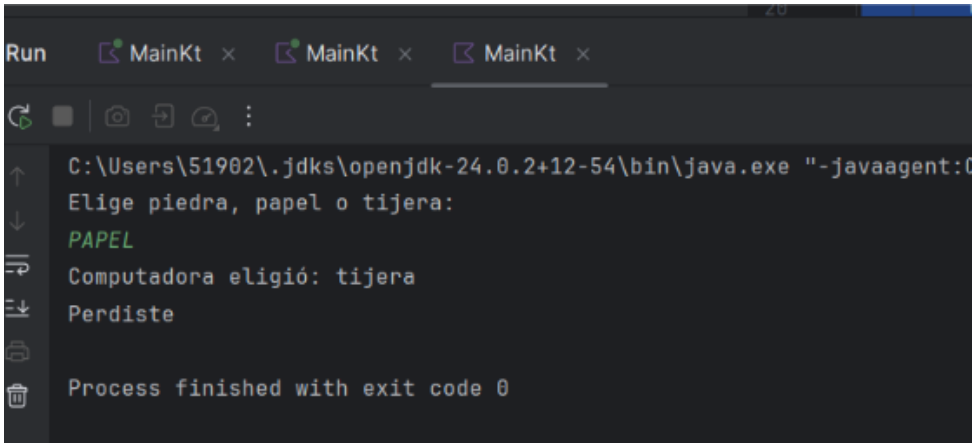

Figura 2: Ejecución del programa PiedraPapelTijera.kt

4.3. Calculadora Elemental

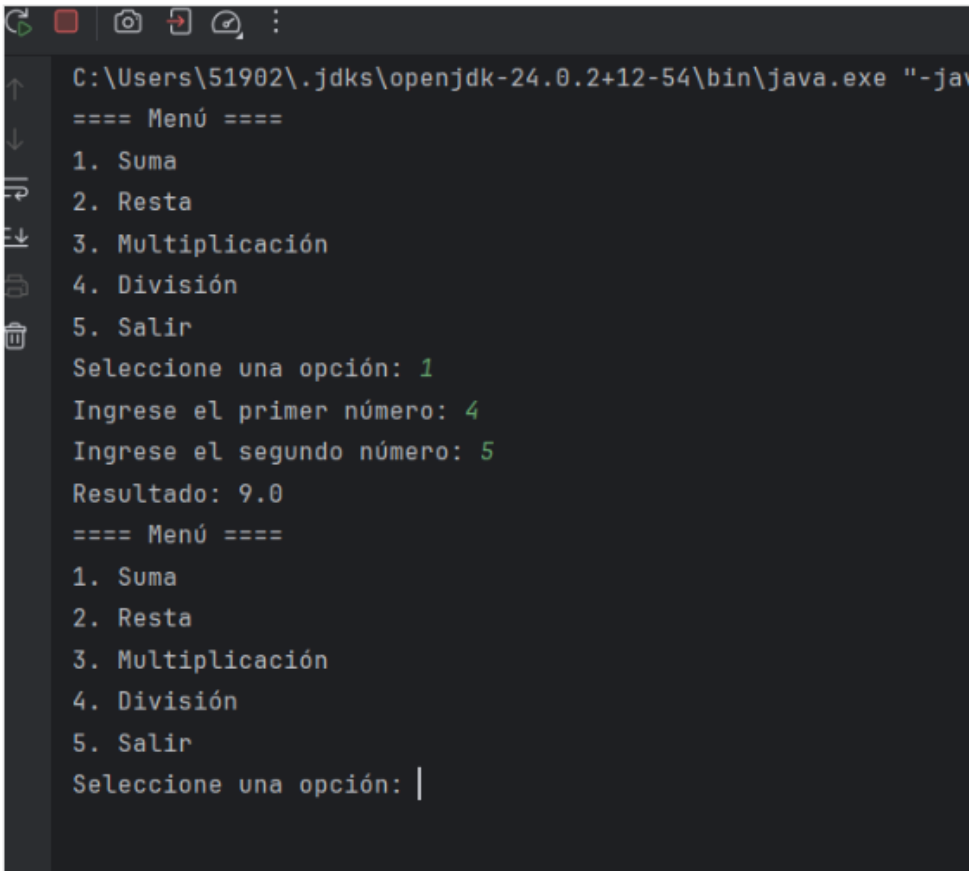

Figura 3: Ejecución del programa Calculadora.kt

4.4. Adivina el Número

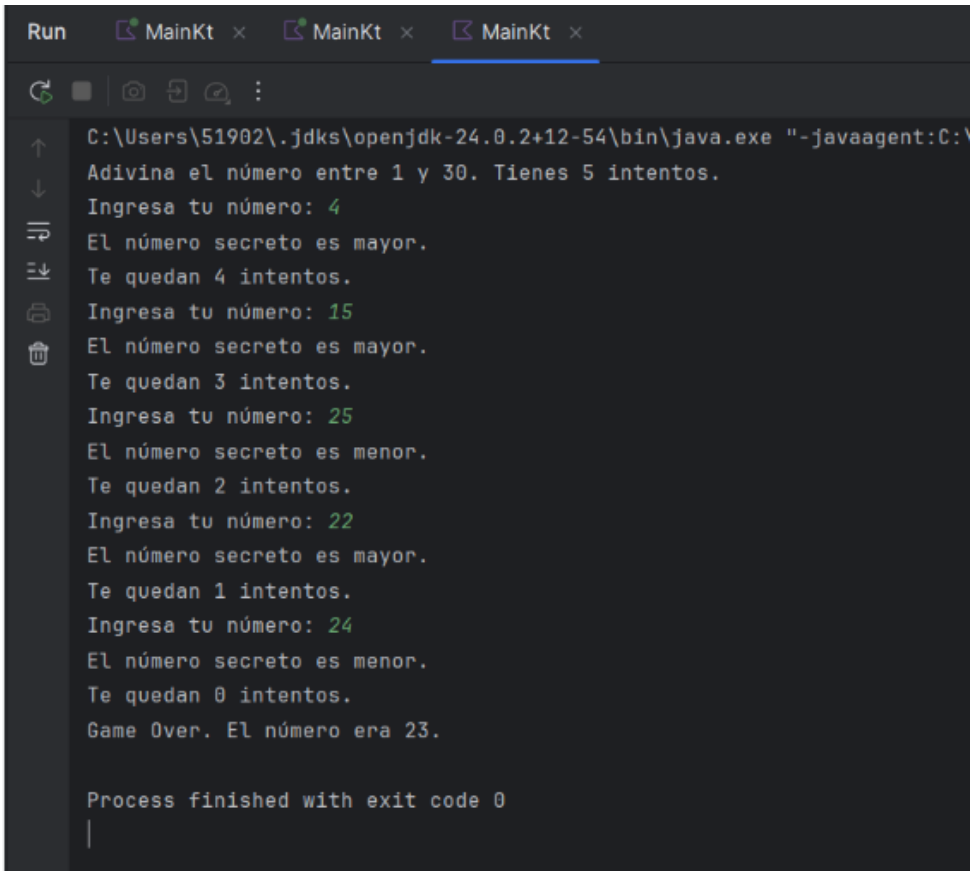

Figura 4: Ejecución del programa AdivinaNumero.kt

5. Conclusiones

Los programas desarrollados demuestran el dominio de:

- **readLine() y readln()**: Para lectura segura de datos del usuario con manejo de valores nulos
- **Random**: Para generación de valores aleatorios en diferentes contextos
- **Manejo de errores**: A través del operador Elvis y null safety
- **Estructuras de control**: Bucles, condicionales y funciones
- **Validación de entrada**: Verificación de opciones válidas

La implementación de estos conceptos en Kotlin muestra la eficiencia y seguridad que ofrece el lenguaje para el desarrollo de aplicaciones interactivas. El uso de `readLine()` junto con los métodos de conversión segura (`toDoubleOrNull()`, `toIntOrNull()`) garantiza que los programas no fallen con entradas inesperadas, mientras que la clase `Random` proporciona una forma sencilla de generar valores aleatorios para diferentes propósitos.