

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

```
1  /*
2   * Descripción: Calcula el nivel de rendimiento y el dinero extra que
3   * recibir un empleado según su puntuación.
4   * Autor: [Tu Nombre]
5   * Fecha creación: 25/08/2025
6   * Última modificación: 25/08/2025
7   */
8 fun calcularNivel(puntuacion: Double): String {
```

```

9     return when {
10         puntuacion in 0.0..3.0 -> "Inaceptable"
11         puntuacion in 4.0..6.0 -> "Aceptable"
12         puntuacion in 7.0..10.0 -> "Meritorio"
13         else -> "Puntuaci n inv lida"
14     }
15 }
16
17 fun calcularDineroExtra(salario: Double, puntuacion: Double): Double {
18     return salario * (puntuacion / 10)
19 }
20
21 fun main() {
22     print("Ingrese su salario mensual: ")
23     val salario = readLine()?.toDoubleOrNull() ?: 0.0
24
25     print("Ingrese su puntuaci n (0 a 10): ")
26     val puntuacion = readLine()?.toDoubleOrNull() ?: 0.0
27
28     val nivel = calcularNivel(puntuacion)
29     if (nivel == "Puntuaci n inv lida") {
30         println("Error: la puntuaci n debe estar entre 0 y 10.")
31     } else {
32         val dineroExtra = calcularDineroExtra(salario, puntuacion)
33         println("Nivel de Rendimiento: $nivel")
34         println("Cantidad de dinero recibido: \$${"%.2f".format(
35             dineroExtra)})")
36     }
36 }

```

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

```

1 /*
2  Descripc i n: Juego de piedra, papel o tijera contra la computadora.
3  Autor: [Tu Nombre]
4  Fecha creaci n: 25/08/2025
5  ltima modificaci n: 25/08/2025
6 */
7
8 import kotlin.random.Random
9
10 fun obtenerOpcionComputadora(): String {
11     val opciones = listOf("piedra", "papel", "tijera")
12     return opciones[Random.nextInt(opciones.size)]
13 }
14

```

```

15 fun determinarGanador(usuario: String, computadora: String): String {
16     return if (usuario == computadora) {
17         "Empate"
18     } else if (
19         (usuario == "piedra" && computadora == "tijera") ||
20         (usuario == "papel" && computadora == "piedra") ||
21         (usuario == "tijera" && computadora == "papel")
22     ) {
23         "Ganaste"
24     } else {
25         "Perdiste"
26     }
27 }
28
29 fun main() {
30     println("Elige piedra, papel o tijera: ")
31     val usuario = readLine()?.lowercase() ?: ""
32     val computadora = obtenerOpcionComputadora()
33
34     if (usuario !in listOf("piedra", "papel", "tijera")) {
35         println("Opción invalida.")
36     } else {
37         println("Computadora eligió: $computadora")
38         println(determinarGanador(usuario, computadora))
39     }
40 }
```

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

```

1 /*
2      Descripción: Calculadora básica que realiza suma, resta,
3      multiplicación y división.
4      Autor: [Tu Nombre]
5      Fecha creación: 25/08/2025
6      Última modificación: 25/08/2025
7 */
8
8 fun mostrarMenu() {
9     println("==== Menú ====")
10    println("1. Suma")
11    println("2. Resta")
12    println("3. Multiplicación")
13    println("4. División")
14    println("5. Salir")
15 }
16
```

```

17 fun suma(a: Double, b: Double) = a + b
18 fun resta(a: Double, b: Double) = a - b
19 fun multiplicacion(a: Double, b: Double) = a * b
20 fun division(a: Double, b: Double): Double? = if (b != 0.0) a / b else
21     null
22
23 fun main() {
24     var opcion: Int
25
26     do {
27         mostrarMenu()
28         print("Seleccione una opción: ")
29         opcion = readLine()?.toIntOrNull() ?: 0
30
31         if (opcion in 1..4) {
32             print("Ingrese el primer número: ")
33             val num1 = readLine()?.toDoubleOrNull() ?: 0.0
34
35             print("Ingrese el segundo número: ")
36             val num2 = readLine()?.toDoubleOrNull() ?: 0.0
37
38             val resultado = when (opcion) {
39                 1 -> suma(num1, num2)
40                 2 -> resta(num1, num2)
41                 3 -> multiplicacion(num1, num2)
42                 4 -> division(num1, num2) ?: run {
43                     println("Error: División entre cero")
44                     continue
45                 }
46                 else -> 0.0
47             }
48
49             println("Resultado: $resultado")
50         } else if (opcion != 5) {
51             println("Opción invalida, intenta de nuevo.")
52         }
53     } while (opcion != 5)
54
55     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

```

1 /*
2      Descripción: Juego para adivinar un número aleatorio entre 1 y 30
3      con máximo 5 intentos.

```

```
3     Autor: [Tu Nombre]
4     Fecha creaci n: 25/08/2025
5         ltima modificaci n: 25/08/2025
6 */
7
8 import kotlin.random.Random
9
10 fun main() {
11     val numeroSecreto = Random.nextInt(1, 31)
12     var intentos = 5
13     var adivinaste = false
14
15     println("Adivina el n mero entre 1 y 30. Tienes 5 intentos.")
16
17     while (intentos > 0) {
18         print("Ingresa tu n mero: ")
19         val intento = readLine()?.toIntOrNull() ?: 0
20
21         if (intento == numeroSecreto) {
22             println(" Felicidades ! Adivinaste el n mero .")
23             adivinaste = true
24             break
25         } else if (intento < numeroSecreto) {
26             println("El n mero secreto es mayor .")
27         } else {
28             println("El n mero secreto es menor .")
29         }
30
31         intentos --
32         println("Te quedan $intentos intentos .")
33     }
34
35     if (!adivinaste) {
36         println("Game Over. El n mero era $numeroSecreto .")
37     }
38 }
```

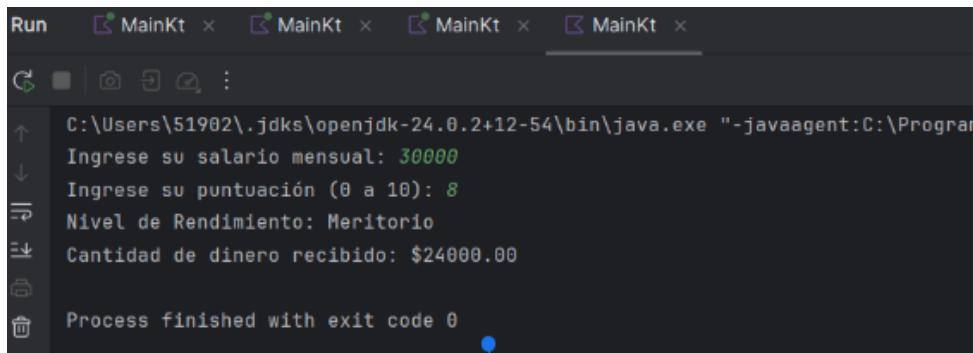
Listing 4: AdivinaNumero.kt

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

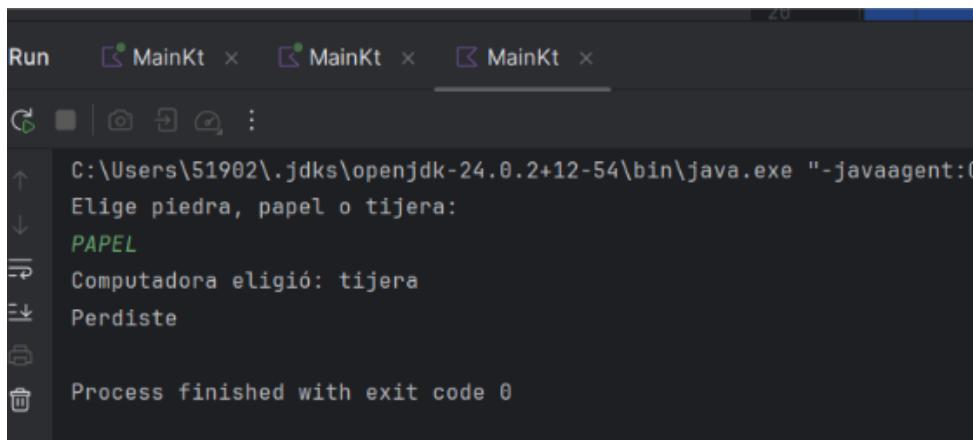


The screenshot shows a terminal window within an IDE. The title bar has four tabs labeled "Run" and three "MainKt" tabs. The terminal output is as follows:

```
C:\Users\51902\.jdks\openjdk-24.0.2+12-54\bin\java.exe "-javaagent:C:\Program  
Ingrese su salario mensual: 30000  
Ingrese su puntuación (0 a 10): 8  
Nivel de Rendimiento: Meritorio  
Cantidad de dinero recibido: $24000.00  
Process finished with exit code 0
```

Figura 1: Ejecución del programa EvaluacionEmpleados.kt

4.2. Piedra, Papel o Tijera

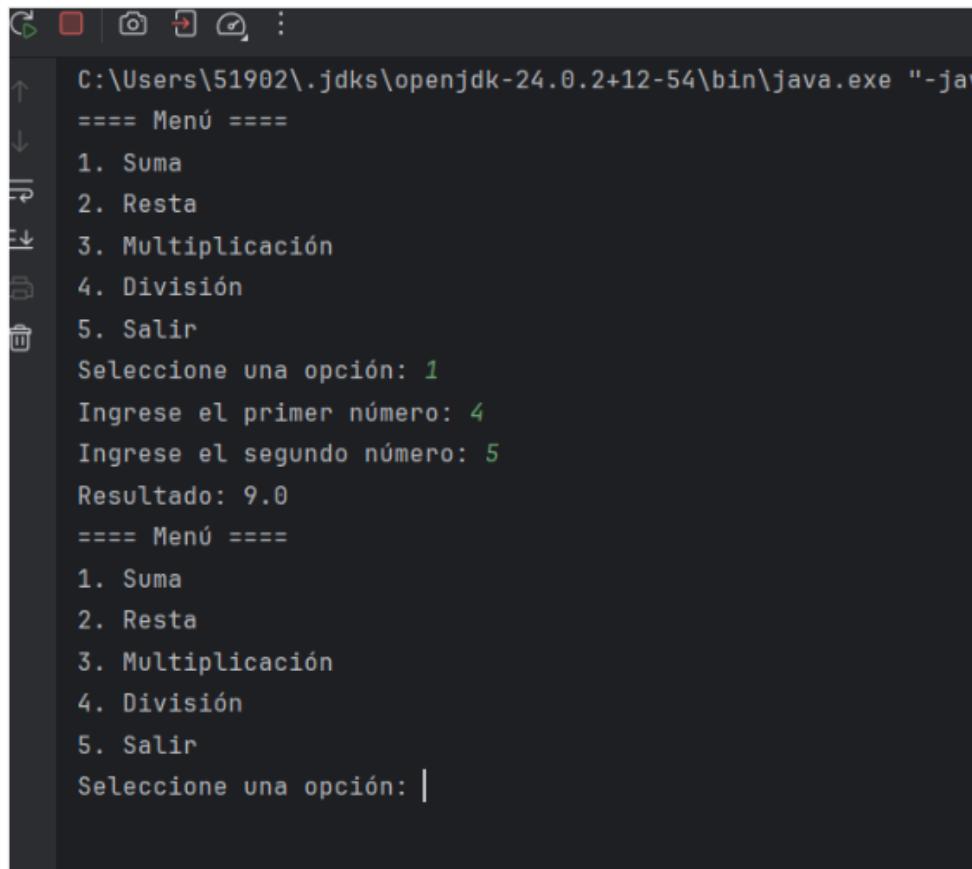


The screenshot shows a terminal window within an IDE. The title bar has four tabs labeled "Run" and three "MainKt" tabs. The terminal output is as follows:

```
C:\Users\51902\.jdks\openjdk-24.0.2+12-54\bin\java.exe "-javaagent:C:\Program  
Elige piedra, papel o tijera:  
PAPEL  
Computadora eligió: tijera  
Perdiste  
Process finished with exit code 0
```

Figura 2: Ejecución del programa PiedraPapelTijera.kt

4.3. Calculadora Elemental



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are standard window control icons (minimize, maximize, close) and a title bar. Below the title bar, the path is displayed: C:\Users\51902\.jdks\openjdk-24.0.2+12-54\bin\java.exe "-javaw". The main content of the window is a menu-based application. It starts with a menu header "===== Menú =====". Below it is a numbered list of five options: 1. Suma, 2. Resta, 3. Multiplicación, 4. División, and 5. Salir. The user is prompted to "Seleccione una opción: 1". They then enter "4" when asked to "Ingresar el primer número: 4". When asked to "Ingresar el segundo número: 5", they enter "5". The application then displays the result "Resultado: 9.0". After the result, the menu is shown again with the same five options. The user is then prompted to "Seleccione una opción: |", where the cursor is visible.

Figura 3: Ejecución del programa Calculadora.kt

4.4. Adivina el Número

```
C:\Users\51902\.jdks\openjdk-24.0.2+12-54\bin\java.exe "-javaagent:C:\Users\51902\IdeaProjects\AdivinaNumero\lib\idea_rt.jar" -Dfile.encoding=UTF-8 MainKt
Adivina el número entre 1 y 30. Tienes 5 intentos.
Ingresá tu número: 4
El número secreto es mayor.
Te quedan 4 intentos.
Ingresá tu número: 15
El número secreto es mayor.
Te quedan 3 intentos.
Ingresá tu número: 25
El número secreto es menor.
Te quedan 2 intentos.
Ingresá tu número: 22
El número secreto es mayor.
Te quedan 1 intentos.
Ingresá tu número: 24
El número secreto es menor.
Te quedan 0 intentos.
Game Over. El número era 23.

Process finished with exit code 0
```

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.