A red sign with white text

Description automatically generated

**P5. Fundamentos de programación en Kotlin**

Alumna: Daniela Mendez Ramirez

Número de Cuenta: 258331-9

**Aplicaciones de Aplicaciones**

Profesor: Omar Vázquez González

Fecha de Entrega: 16 de febrero de 2025

|  |  |
| --- | --- |
| Código | Resultado |
| /\*\*  \* You can edit, run, and share this code.  \* play.kotlinlang.org  \*/  fun main() {  println("Hello, world!!!")  } |  |

2. Variables

|  |  |
| --- | --- |
| Código | Resultado |
| fun main() {  val nombre: String = "Daniela"  var edad: Int = 20  println("Mi nombre es $nombre y tengo $edad años.")  } |  |

3. Condiciones con IF

|  |  |
| --- | --- |
| Código | Resultado |
| fun main() {  val numero = -5    if (numero > 0) {  println("El número es positivo.")  } else if (numero < 0) {  println("El número es negativo.")  } else {  println("El número es cero.")  }  } |  |

4. Funciones

|  |  |
| --- | --- |
| Código | Resultado |
| fun saludar(nombre: String) {  println("Hola, $nombre!")  }  fun main() {  saludar("Daniela")  } |  |

5. Ciclo For

|  |  |
| --- | --- |
| Código | Resultado |
| fun main() {  for (i in 1..10) {  println(i)  }  } |  |

6. Ciclo While

|  |  |
| --- | --- |
| Código | Resultado |
| fun main() {  for (i in 1..10) {  println(i)  }  } |  |

7. Tablas de multiplicar

|  |  |
| --- | --- |
| Código | Resultado |
| fun imprimirTablas() {  for (i in 1..10) {  println("Tabla del $i:")  for (j in 1..10) {  println("$i x $j = ${i \* j}")  }  println() // Línea en blanco para separar las tablas  }  }  fun main() {  imprimirTablas()  } |  |

8. Área de un Círculo Por Valor

|  |  |
| --- | --- |
| Código | Resultado |
| fun calcularAreaPorValor(radio: Double): Double {  return Math.PI \* radio \* radio  }  fun main() {  val radio = 5.0  val area = calcularAreaPorValor(radio)  println("El área del círculo con radio $radio es: $area")  } |  |

9. Área de un Círculo Por Referencia

|  |  |
| --- | --- |
| Código | Resultado |
| class Circulo(var radio: Double, var area: Double = 0.0)  fun calcularAreaPorReferencia(circulo: Circulo) {  circulo.area = Math.PI \* circulo.radio \* circulo.radio  }  fun main() {  val miCirculo = Circulo(5.0)  calcularAreaPorReferencia(miCirculo)  println("El área del círculo con radio ${miCirculo.radio} es: ${miCirculo.area}")  } | A number on a black background  AI-generated content may be incorrect. |

10. Area, Celcius

|  |  |
| --- | --- |
| Código | Resultado |
| fun volumenCubo(lado: Double): Double {  return lado \* lado \* lado  }  fun areaTriangulo(base: Double, altura: Double): Double {  return (base \* altura) / 2  }  fun celsiusAFahrenheit(celsius: Double): Double {  return (celsius \* 9 / 5) + 32  }  fun main() {  val ladoCubo = 3.0  println("El volumen del cubo con lado $ladoCubo es: ${volumenCubo(ladoCubo)}")  val baseTriangulo = 5.0  val alturaTriangulo = 4.0  println("El área del triángulo con base $baseTriangulo y altura $alturaTriangulo es: ${areaTriangulo(baseTriangulo, alturaTriangulo)}")  val temperaturaCelsius = 25.0  println("$temperaturaCelsius°C en Fahrenheit es: ${celsiusAFahrenheit(temperaturaCelsius)}°F")  } |  |

11. Una función que compute la sumatoria de los primeros n números, dado n.

|  |  |
| --- | --- |
| Código | Resultado |
| fun sumatoria(n: Int): Int {  return (n \* (n + 1)) / 2  }  fun main() {  print("Ingrese un número n: ")  val n = readLine()!!.toInt()  println("La sumatoria de los primeros $n números es: ${sumatoria(n)}")  } |  |

12. Fibonacci

|  |  |
| --- | --- |
| Código | Resultado |
| fun fibonacci(n: Int): Int {  if (n == 0) return 0  if (n == 1) return 1  return fibonacci(n - 1) + fibonacci(n - 2)  }  fun main() {  print("Ingrese un número n: ")  val n = readLine()!!.toInt()  println("El término $n de la serie Fibonacci es: ${fibonacci(n)}")  } |  |

13. Auto

|  |  |
| --- | --- |
| Código | Resultado |
| // Definimos la clase Auto con dos propiedades: marca y velocidad  class Auto(val marca: String, val velocidad: Int) {  // Método para calcular cuántos kilómetros avanza en un tiempo dado  fun avanzar(tiempo: Int) {  val distancia = velocidad \* tiempo  println("El auto de marca $marca recorrió $distancia km en $tiempo horas.")  }  }  fun main() {  // Creamos autos con diferentes marcas y velocidades  val auto1 = Auto("Toyota", 100)  val auto2 = Auto("Ferrari", 200)  // Mostramos cuántos kilómetros avanzan en 2 horas  auto1.avanzar(2)  auto2.avanzar(2)  } |  |

14. Alumnos

|  |  |
| --- | --- |
| Código | Resultado |
| // Clase Materia que representa el nombre de la materia  class Materia(val nombre: String)  // Clase Calificación que asocia una materia con una calificación  class Calificacion(val materia: Materia, val nota: Double)  // Clase Alumno con nombre y lista de calificaciones  class Alumno(val nombre: String) {  private val calificaciones = mutableListOf<Calificacion>()  // Método para agregar una calificación  fun agregarCalificacion(materia: Materia, nota: Double) {  calificaciones.add(Calificacion(materia, nota))  }  // Método para calcular el promedio del alumno  fun calcularPromedio(): Double {  if (calificaciones.isEmpty()) return 0.0  return calificaciones.map { it.nota }.average()  }  // Método para mostrar las calificaciones del alumno  fun mostrarCalificaciones() {  println("Calificaciones de $nombre:")  calificaciones.forEach {  println("Materia: ${it.materia.nombre} - Nota: ${it.nota}")  }  println("Promedio: ${calcularPromedio()}\n")  }  }  fun main() {  // Creamos materias  val matematicas = Materia("Matemáticas")  val espanol = Materia("Español")  val ciencias = Materia("Ciencias")  // Creamos un alumno  val alumno1 = Alumno("Lorenzo Grebe")  // Agregamos calificaciones  alumno1.agregarCalificacion(matematicas, 8.5)  alumno1.agregarCalificacion(espanol, 9.0)  alumno1.agregarCalificacion(ciencias, 7.5)  // Mostramos las calificaciones y el promedio del alumno  alumno1.mostrarCalificaciones()  } |  |

Conclusión

A través de estos ejercicios en Kotlin, pude reforzar conceptos clave como estructuras de control, funciones y programación orientada a objetos. Implementé ciclos for y while para resolver problemas matemáticos, además de utilizar funciones con paso por valor y referencia para cálculos como el área de un círculo y la conversión de grados. También trabajé con clases y objetos, lo que me permitió organizar mejor la información en programas como el control de calificaciones y la simulación de autos. Además, usé listas mutables para almacenar y manipular datos de manera dinámica. En general, estos ejercicios me ayudaron a entender la flexibilidad y legibilidad de Kotlin, destacando su eficiencia para desarrollar código limpio y estructurado.