Tecnológico Nacional de México.

Instituto Tecnológico de Ciudad Juárez.

Práctica Final de Kotlin online

Diseño y Desarrollo de Aplicaciones Móviles Reactivas

Docente: Ramos Silva Jaime Alonso

Alumno: Ángel Damián Reyes García

Fecha de Entrega: 07-04-2025

Código

```
class Estudiante(val nombre: String, val edad: Int, val calificaciones: List<Double>) {

    init {
        require(nombre.isNotBlank()) { "El nombre no puede estar vacío." }
        require(edad in 1..120) { "La edad debe estar entre 1 y 120 años." }
        require(calificaciones.isNotEmpty() && calificaciones.all { it in 0.0..100.0 }) {
            "Las calificaciones deben estar entre 0 y 100."
        }
    }

    fun promedio(): Double {
        return calificaciones.average()
    }

    fun estaAprobado(): Boolean {
        return promedio() >= 60
    }

    override fun toString(): String {
        return "Nombre: ".plus(nombre) +
                ", Edad: ".plus(edad.toString()) +
                ", Calificaciones: ".plus(calificaciones.toString()) +
                ", Promedio: ".plus(String.format("%.2f", promedio())) +
                ", " + if (estaAprobado()) "Aprobado" else "Reprobado"
    }
}

class GestorEstudiantes {
    private val estudiantes = mutableListOf<Estudiante>()

    fun agregarEstudiante(nombre: String, edad: Int, calificaciones: List<Double>) {
        try {
            val estudiante = Estudiante(nombre, edad, calificaciones)
            estudiantes.add(estudiante)
            println("Estudiante agregado exitosamente.\n")
        } catch (e: IllegalArgumentException) {
            println("Error al agregar estudiante: " + e.message + "\n")
        }
    }

    fun mostrarEstudiantes() {
        if (estudiantes.isEmpty()) {
            println("No hay estudiantes registrados.\n")
        } else {
            for (est in estudiantes) {
                println(est.toString())
            }
            println()
        }
    }

    fun buscarEstudiante(nombre: String) {
        val encontrados = estudiantes.filter { it.nombre.equals(nombre, ignoreCase = true) }
        if (encontrados.isEmpty()) {
            println("No se encontraron estudiantes con ese nombre.\n")
        } else {
            for (e in encontrados) {
                println(e.toString())
            }
            println()
        }
    }
```

```
 fun ordenarPorPromedio() {
    estudiantes.sortByDescending { it.promedio() }
    println("Estudiantes ordenados por promedio (descendente):")
    for (est in estudiantes) {
        println(est.toString())
    }
    println()
}
    fun eliminarEstudiante(nombre: String) {
        val eliminado = estudiantes.removeIf { it.nombre.equals(nombre, ignoreCase = true) }
        if (eliminado) {
            println("Estudiante(s) eliminado(s) exitosamente.\n")
        } else {
            println("No se encontró ningún estudiante con ese nombre.\n")
        }
    }
}
```

```kotlin
fun main() {
    val gestor = GestorEstudiantes()

    while (true) {
        println("------ MENÚ ------")
        println("1. Agregar estudiante")
        println("2. Mostrar estudiantes")
        println("3. Buscar estudiante por nombre")
        println("4. Ordenar estudiantes por promedio")
        println("5. Eliminar estudiante")
        println("6. Salir")
        print("Seleccione una opción: ")

        when (readLine()?.trim()) {
            "1" -> {
                try {
                    print("Nombre: ")
                    val nombre = readLine() ?: ""
                    print("Edad: ")
                    val edad = readLine()?.toIntOrNull() ?: throw Exception("Edad inválida.")

                    print("Ingrese las calificaciones separadas por comas: ")
                    val calificacionesStr = readLine() ?: ""
                    val calificaciones = calificacionesStr.split(",").map {
                        it.trim().toDouble()
                    }

                    gestor.agregarEstudiante(nombre, edad, calificaciones)
                } catch (e: Exception) {
                    println("Error al ingresar los datos: " + e.message + "\n")
                }
            }

            "2" -> gestor.mostrarEstudiantes()

            "3" -> {
                print("Nombre a buscar: ")
                val nombre = readLine() ?: ""
                gestor.buscarEstudiante(nombre)
            }

            "4" -> gestor.ordenarPorPromedio()

            "5" -> {
                print("Nombre del estudiante a eliminar: ")
                val nombre = readLine() ?: ""
                gestor.eliminarEstudiante(nombre)
            }

            "6" -> {
                println("Saliendo del programa...")
                break
            }

            else -> println("Opción inválida.\n")
        }
    }
}
```

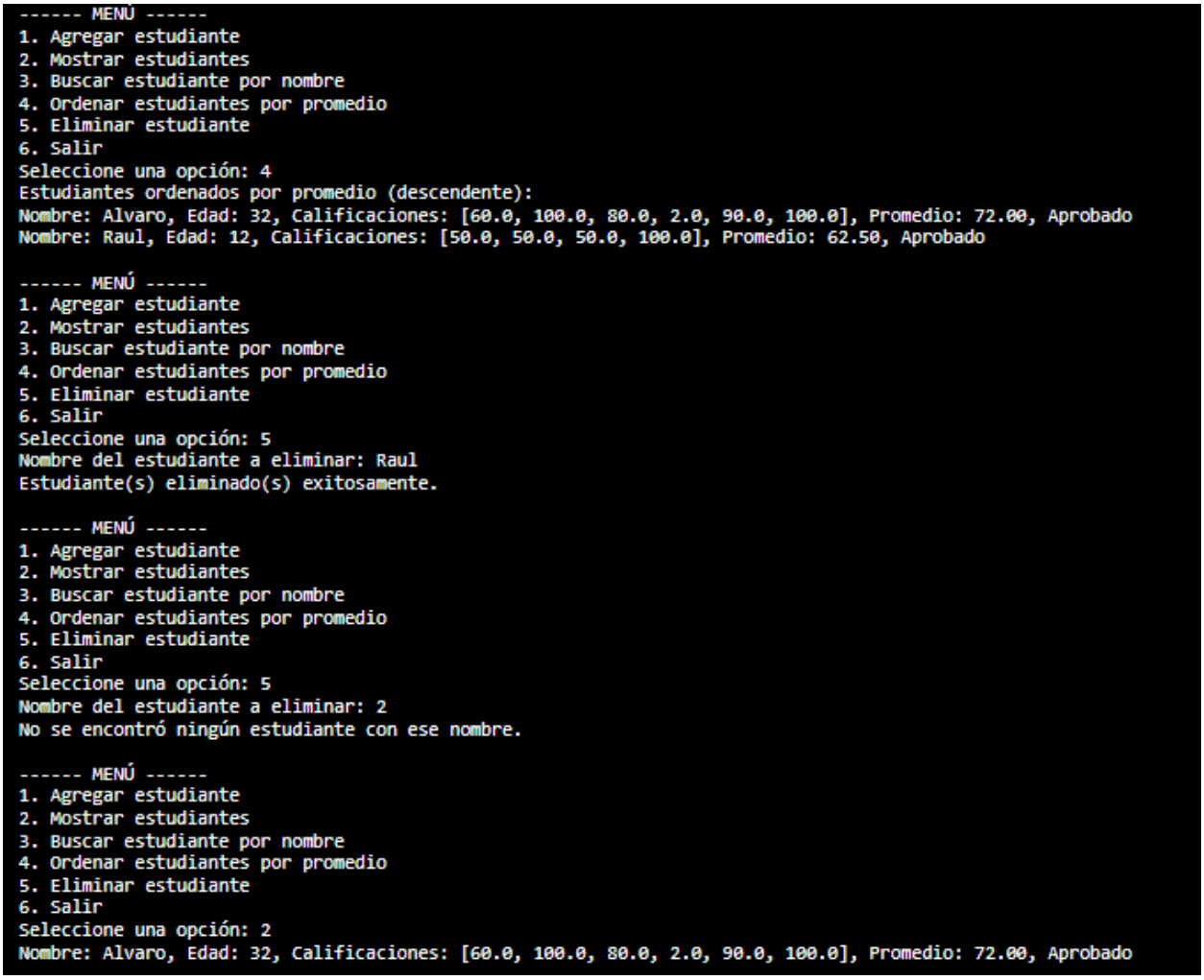
Ejecución

```
------ MENÚ ------
1. Agregar estudiante
2. Mostrar estudiantes
3. Buscar estudiante por nombre
4. Ordenar estudiantes por promedio
5. Eliminar estudiante
6. Salir
Seleccione una opción: 1
Nombre: Raul
Edad: 12
Ingrese las calificaciones separadas por comas: 50,50,50,100
Estudiante agregado exitosamente.

------ MENÚ ------
1. Agregar estudiante
2. Mostrar estudiantes
3. Buscar estudiante por nombre
4. Ordenar estudiantes por promedio
5. Eliminar estudiante
6. Salir
Seleccione una opción: 1
Nombre: Alvaro
Edad: 32
Ingrese las calificaciones separadas por comas: 60,100,80,2,90,100
Estudiante agregado exitosamente.

------ MENÚ ------
1. Agregar estudiante
2. Mostrar estudiantes
3. Buscar estudiante por nombre
4. Ordenar estudiantes por promedio
5. Eliminar estudiante
6. Salir
Seleccione una opción: 2
Nombre: Raul, Edad: 12, Calificaciones: [50.0, 50.0, 50.0, 100.0], Promedio: 62.50, Aprobado
Nombre: Alvaro, Edad: 32, Calificaciones: [60.0, 100.0, 80.0, 2.0, 90.0, 100.0], Promedio: 72.00, Aprobado

------ MENÚ ------
1. Agregar estudiante
2. Mostrar estudiantes
3. Buscar estudiante por nombre
4. Ordenar estudiantes por promedio
5. Eliminar estudiante
6. Salir
Seleccione una opción: 3
Nombre a buscar: Raul
Nombre: Raul, Edad: 12, Calificaciones: [50.0, 50.0, 50.0, 100.0], Promedio: 62.50, Aprobado
```



Conclusiones:

Con esta actividad aprendí a aplicar los conceptos fundamentales de la programación orientada a objetos en Kotlin, como la creación de clases, objetos y métodos. Pude practicar el uso de listas para almacenar datos dinámicos. También aprendí la importancia de validar correctamente los datos que recibe el programa, para evitar errores en tiempo de ejecución, y cómo implementar control de errores usando bloques try-catch.