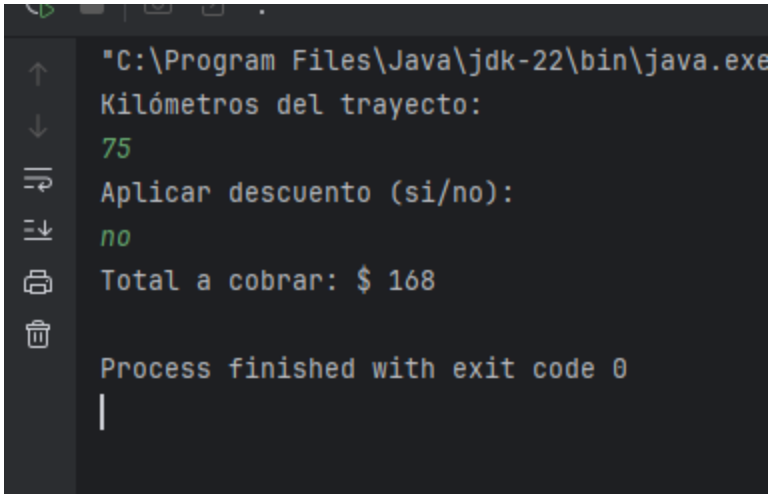
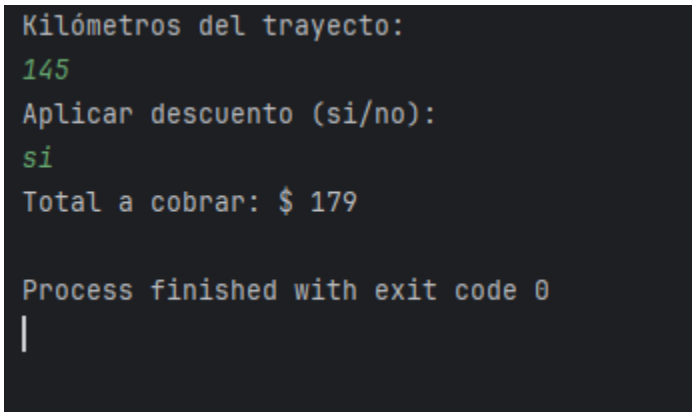
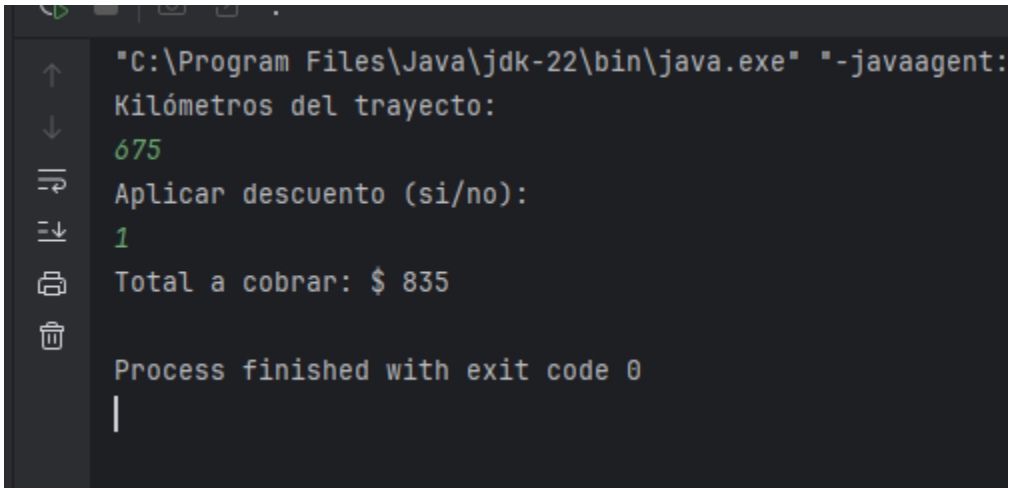
ESTRUCTURAS, ARREGLOS Y CICLOS

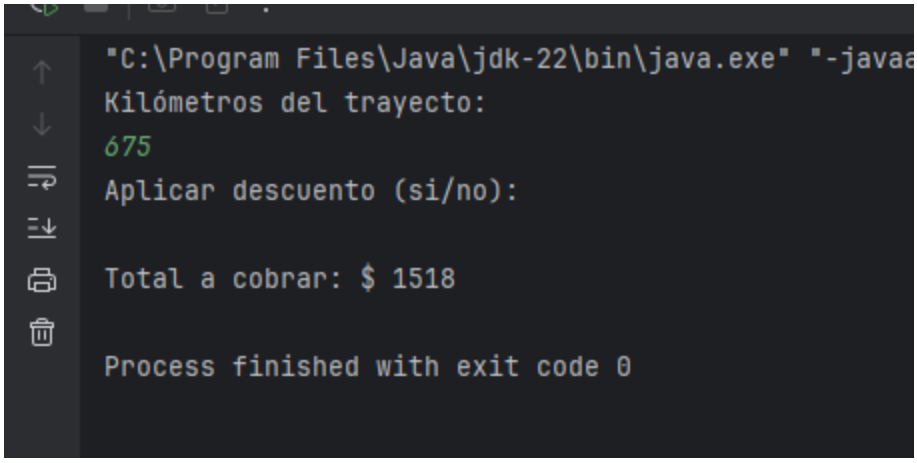
2CV22 HERNANDEZ GONZALEZ JOSEPH FABRIZZIO

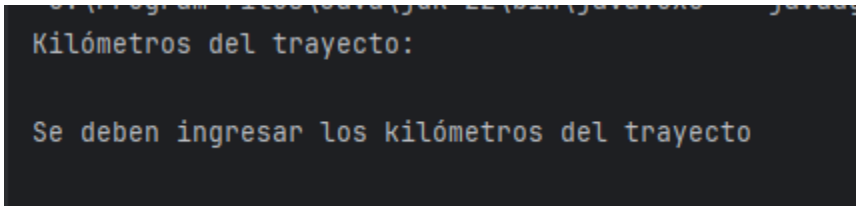
Kotlin

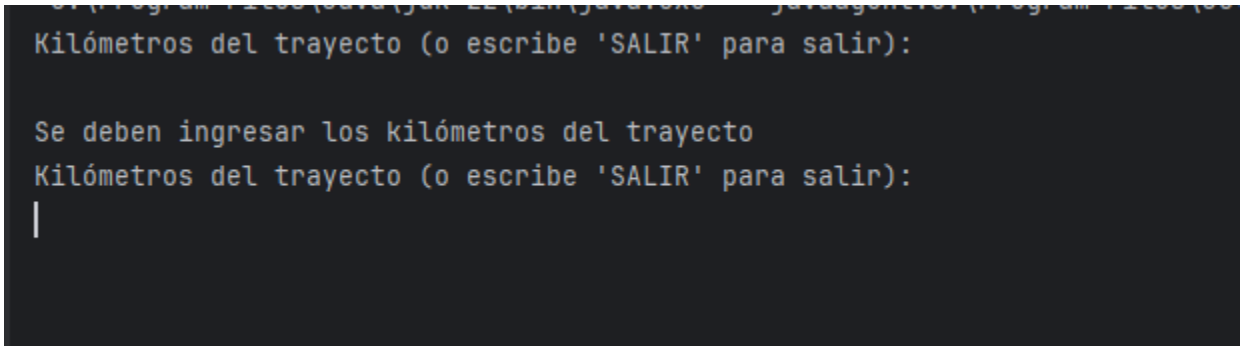

"C:\Program Files\Java\jdk-22\bin\java.exe
Kilómetros del trayecto:
75
Aplicar descuento (si/no):
no
Total a cobrar: $ 168
Process finished with exit code 0


Kilómetros del trayecto:
145
Aplicar descuento (si/no):
si
Total a cobrar: $ 179
Process finished with exit code 0


"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:
Kilómetros del trayecto:
675
Aplicar descuento (si/no):
1
Total a cobrar: $ 835
Process finished with exit code 0


"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaa
Kilómetros del trayecto:
675
Aplicar descuento (si/no):
Total a cobrar: $ 1518
Process finished with exit code 0


Kilómetros del trayecto:
Se deben ingresar los kilómetros del trayecto


Kilómetros del trayecto (o escribe 'SALIR' para salir):
Se deben ingresar los kilómetros del trayecto
Kilómetros del trayecto (o escribe 'SALIR' para salir):

CODIGO

```
// Created by: 2CV22 HERNANDEZ GONZALEZ JOSEPH FABRIZZIO
fun calculateFare(kilometers: Double, applyDiscount: String?): Int {
    val baseFare = 2.25
    var total = baseFare * kilometers
    if (applyDiscount?.lowercase() == "si" || applyDiscount == "1") {
        total *= 0.55 // Aplicar descuento del 45%
    }
    return total.toInt() // Redondear a números enteros
}

fun main() {
    while (true) {
        println("Kilómetros del trayecto (o escribe 'SALIR' para salir):")
        val input = readLine()

        // Comprobar si el usuario quiere salir
        if (input.equals("SALIR", ignoreCase = true)) {
            println("Saliendo del programa...")
            break
        }

        val kilometers = input?.toDoubleOrNull()
        if (kilometers == null || kilometers <= 0) {
            println("Se deben ingresar los kilómetros del trayecto")
            continue // Volver al inicio del bucle
        }

        println("Aplicar descuento (si/no):")
        val applyDiscount = readLine()

        val total = calculateFare(kilometers, applyDiscount)
        println("Total a cobrar: $ $total")
    }
}
```

PROGRAMA 2

```
Promedio de calificación: 70.0
Calificación más alta: 98.5
Calificación más baja: 31.0
Reprobados:
José, 31.0
Erika, 46.5
```

Codigo

```
// Created by: 2CV22 HERNANDEZ GONZALEZ JOSEPH FABRIZZIO
fun calculateAverage(grades: Array<Double>): Double {
    return grades.average()
}

fun highestGrade(grades: Array<Double>): Double {
    return grades.maxOrNull() ?: 0.0
}

fun lowestGrade(grades: Array<Double>): Double {
    return grades.minOrNull() ?: 0.0
}

fun failingStudents(students: Array<String>, grades: Array<Double>):
List<Pair<String, Double>> {
    return students.zip(grades).filter { it.second < 60.0 }
}

fun main() {
    val students = arrayOf("José", "Alberto", "Laura", "Noel", "Erika",
"Daniel")
    val grades = arrayOf(31.0, 94.0, 98.5, 75.0, 46.5, 75.0)

    val average = calculateAverage(grades)
    val maxGrade = highestGrade(grades)
    val minGrade = lowestGrade(grades)
    val failing = failingStudents(students, grades)

    println("Promedio de calificación: $average")
    println("Calificación más alta: $maxGrade")
    println("Calificación más baja: $minGrade")
```

```
    println("Reprobados:")
    for ((student, grade) in failing) {
        println("$student, $grade")
    }
}
```