

Session 10 : AADK Task 10 Report

Title: Kotlin Classes, Objects, and Lambdas (Student Management System)

Objective:

To strengthen Kotlin fundamentals by creating a Student Management System using custom classes, objects, and higher-order functions with lambda expressions. The task focuses on writing clean, modular, and reusable logic.

Concepts Covered:

- Custom Data Classes (Student)
- Object Creation and List Management
- Higher-Order Functions
- Lambda Expressions
- List Operations (map, filter, forEach)
- Immutability using copy()

Complete Implementation Code:

```
data class Student(  
    val name: String,  
    val department: String,  
    val marks: Int  
)  
  
fun processStudents(  
    students: List<Student>,  
    action: (Student) -> Unit  
) {  
    students.forEach { action(it) }  
}  
  
fun applyGraceMarks(  
    students: List<Student>,  
    graceRule: (Int) -> Int  
) : List<Student> {  
    return students.map { student ->  
        student.copy(marks = graceRule(student.marks))  
    }  
}  
  
fun main() {  
    val students = listOf(  
        Student("John", "Computer Science", 85),  
        Student("Alice", "Mathematics", 92),  
        Student("Bob", "Physics", 78),  
        Student("Charlie", "Chemistry", 88),  
        Student("David", "Biology", 75),  
        Student("Eve", "History", 82),  
        Student("Frank", "Literature", 70),  
        Student("Grace", "Art", 65),  
        Student("Heidi", "Music", 72),  
        Student("Ivan", "Sports", 68),  
        Student("Julia", "Dance", 75),  
        Student("Leo", "Gardening", 60),  
        Student("Mia", "Reading", 80),  
        Student("Noah", "Writing", 78),  
        Student("Olivia", "Drawing", 62),  
        Student("Peter", "Coding", 85),  
        Student("Quinn", "Golfing", 55),  
        Student("Rachel", "Fishing", 60),  
        Student("Samuel", "Hiking", 70),  
        Student("Tina", "Swimming", 65),  
        Student("Uma", "Skiing", 72),  
        Student("Victor", "Climbing", 68),  
        Student("Wendy", "Judo", 75),  
        Student("Xavier", "Taekwondo", 60),  
        Student("Yara", "Karate", 70),  
        Student("Zoe", "Martial Arts", 65)
```

```

        Student("Arjun", "ISE", 78),
        Student("Meera", "CSE", 85),
        Student("Ravi", "ECE", 67),
        Student("Ananya", "ISE", 92),
        Student("Kiran", "ME", 55)
    )

    println("--- Student Details ---")
    processStudents(students) {
        println("Name: ${it.name} | Dept: ${it.department} | Marks: ${it.marks}")
    }

    val updatedMarks = applyGraceMarks(students) { marks ->
        marks + 5
    }

    println("\n--- After Adding 5 Grace Marks ---")
    updatedMarks.forEach {
        println("Name: ${it.name} | Updated Marks: ${it.marks}")
    }

    val topStudents = students.filter { it.marks > 80 }

    println("\n--- Top Students (Marks > 80) ---")
    topStudents.forEach {
        println("${it.name} (${it.marks})")
    }
}

```

Explanation of the Code:

1. A Student data class is created to store structured information such as name, department, and marks.
2. The function processStudents() is a higher-order function because it accepts another function as a parameter. This allows flexible processing of student data.
3. The function applyGraceMarks() uses map() to return a new list with updated marks based on a lambda rule. The original list remains unchanged.
4. Lambda expressions are passed in main() to dynamically add grace marks and filter top-performing students.

Learning Outcome:

By completing this task, students learned how Kotlin combines object-oriented and functional programming. They practiced writing modular, reusable, and scalable code using classes, objects, and lambda expressions in a real-world student scenario.