Session 9 Lab Report

Inventory Management System using Kotlin Collections and Generics

1. Objective

To design and implement a Kotlin-based inventory management program demonstrating the use of data classes, lists, maps, generics, and higher-order functions.

2. Concepts Used

- Data Classes
- Lists and Maps
- Generic Functions
- Lambda Expressions
- groupBy() for data grouping

3. Implementation with Code Explanation

```
// Data Class
data class Item(
    val name: String,
    val category: String,
    val quantity: Int
)

// Function to Print Inventory
fun printInventory(items: List<Item>) {
    for (item in items) {
        println("Name: ${item.name} | Category: ${item.category} | Qty: ${item.quantity}")
    }
}

// Generic Filter Function
fun <T> filterList(list: List<T>, condition: (T) -> Boolean): List<T> {
    val result = mutableListOf<T>()
    for (element in list) {
        if (condition(element)) {
            result.add(element)
        }
    }
    return result
}

fun main() {
    val inventory = listOf(
        Item("Laptop", "Electronics", 12),
        Item("Keyboard", "Electronics", 5),
        Item("Chair", "Furniture", 20),
        Item("Table", "Furniture", 8),
        Item("Notebook", "Stationery", 50)
    )
    println("--- Full Inventory ---")
```

```
    printInventory(inventory)

    val lowStock = filterList(inventory) { it.quantity < 10 }
    println("--- Low Stock (<10) ---")
    printInventory(lowStock)

    val electronicsItems = filterList(inventory) { it.category == "Electronics" }
    println("--- Electronics Category ---")
    printInventory(electronicsItems)

    println("--- Total Quantity per Category ---")
    val groupedItems = inventory.groupBy { it.category }

    for ((category, items) in groupedItems) {
        var totalQuantity = 0
        for (item in items) {
            totalQuantity += item.quantity
        }
        println("$category: $totalQuantity")
    }
}
```

4. Explanation

The Item data class represents each product in the inventory. The printInventory() function prints items in a structured format. The generic function filterList() allows filtering of any list type using a condition lambda, demonstrating reusability through generics. The groupBy() function groups items by category and calculates total quantities for each category.

5. Conclusion

This lab successfully demonstrates intermediate Kotlin concepts including generics, collections, higher-order functions, and data grouping techniques. It reinforces clean code structure and reusable function design.