DAY 3 TASKS

**Task 9:** Define an Exportable interface with a method to export transaction data to CSV.

CODE:

package day3tasks9to12  
  
import java.io.BufferedWriter  
import java.io.File  
import java.io.FileWriter  
  
interface Exportable {  
 fun exportToCSV(transactions: List<Transaction>, fileName: String)  
}  
  
  
data class Transaction(val id: Int, var date: String, var amount: Double, val description: String)  
  
class TransactionExporter : Exportable {  
 override fun exportToCSV(transactions: List<Transaction>, fileName: String) {  
 val file = File(fileName)  
 BufferedWriter(FileWriter(file)).*use* **{** writer **->** writer.write("ID,Date,Amount,Description\n")  
  
 for (transaction in transactions) {  
 writer.write("${transaction.id},${transaction.date},${transaction.amount},${transaction.description}\n")  
 }  
 **}** *println*("Exported to $fileName successfully.")  
 }  
}  
  
fun main() {  
 val transactions = *listOf*(  
 Transaction(1, "2025-01-10", 200.0, "Payment received successfully"),  
 Transaction(2, "2025-01-11", 50.0, "Payment failed"),  
 Transaction(3, "2025-01-12", 100.0, "Payment pending")  
 )  
  
 val exporter = TransactionExporter()  
 exporter.exportToCSV(transactions, "transactions.csv")  
}

OUTPUT:

Exported to transactions.csv successfully.

ID,Date,Amount,Description  
1,2025-01-10,200.0,Payment received successfully  
2,2025-01-11,50.0,Payment failed  
3,2025-01-12,100.0,Payment pending

**Task 10:** Apply encapsulation to Transaction properties using getters and setters ensuring sensitive data is protected.

CODE:

package day3tasks9to12  
class TransactionByEncapsulation(id: Int, amount: Double, description: String) {  
  
 private val Id: Int = id  
 private var Amount: Double = amount  
 private var Description: String = description  
  
 val id: Int  
 get() = Id  
  
 var amount: Double  
 get() = Amount  
 set(value) {  
 if (value >= 0) {  
 Amount = value  
 } else {  
 throw IllegalArgumentException("Amount cannot be negative")  
 }  
 }  
  
 var description: String  
 get() = Description  
 set(value) {  
 Description = value.*trim*()  
 }  
}  
fun main() {  
 val transaction = TransactionByEncapsulation(1, 100.0, "Travel")  
  
 *println*("ID: ${transaction.id}")  
 *println*("Amount: ${transaction.amount}")  
 *println*("Description: ${transaction.description}")  
  
 transaction.amount = 300.0  
 transaction.description = "Rent Payment"  
  
 try {  
 transaction.amount =-500.0  
 } catch (e: IllegalArgumentException) {  
 *println*("Error: ${e.message}")  
 }  
 *println*("Updated Amount: ${transaction.amount}")  
 *println*("Updated Description: ${transaction.description}")  
}

OUTPUT:

ID: 1

Amount: 100.0

Description: Travel

Error: Amount cannot be negative

Updated Amount: 300.0

Updated Description: Rent Payment

**Task 11:** Create generic functions to handle different types of collections (List, Set, Map) of transactions.  
CODE:

package day3tasks9to12  
  
data class TransactionCollection(var id: Int, var date: String, var amount: Double, var description: String)  
  
fun <T> TransactionsList(collection: T) where T : Collection<TransactionCollection> {  
 for (transaction in collection) {  
 *println*("ID: ${transaction.id}, Date: ${transaction.date}, Amount: ${transaction.amount}, Description: ${transaction.description}")  
 }  
}  
  
fun <T> TransactionsSet(collection: T) where T : Set<TransactionCollection> {  
 for (transaction in collection) {  
 *println*("ID: ${transaction.id}, Date: ${transaction.date}, Amount: ${transaction.amount}, Description: ${transaction.description}")  
 }  
}  
  
fun <K, V> TransactionsMap(collection: Map<K, V>) where V : TransactionCollection {  
 for ((key, transaction) in collection) {  
 *println*("Key: $key, ID: ${transaction.id}, Date: ${transaction.date}, Amount: ${transaction.amount}, Description: ${transaction.description}")  
 }  
}  
  
fun main() {  
 val transaction1 = TransactionCollection(1, "2025-01-10", 500.0, "Food")  
 val transaction2 = TransactionCollection(2, "2025-01-11", 1000.0, "Rent")  
 val transaction3 = TransactionCollection(3, "2025-01-12", 300.0, "Groceries")  
  
 val transactionList = *listOf*(transaction1, transaction2, transaction3)  
 *println*("List of Transactions:")  
 *TransactionsList*(transactionList)  
  
 val transactionSet = *setOf*(transaction1, transaction2, transaction3)  
 *println*("\nSet of Transactions:")  
 *TransactionsSet*(transactionSet)  
  
 val transactionMap = *mapOf*(  
 1 *to* transaction1,  
 2 *to* transaction2,  
 3 *to* transaction3  
 )  
 *println*("\nMap of Transactions:")  
 *TransactionsMap*(transactionMap)  
}

OUTPUT:

List of Transactions:

ID: 1, Date: 2025-01-10, Amount: 500.0, Description: Food

ID: 2, Date: 2025-01-11, Amount: 1000.0, Description: Rent

ID: 3, Date: 2025-01-12, Amount: 300.0, Description: Groceries

Set of Transactions:

ID: 1, Date: 2025-01-10, Amount: 500.0, Description: Food

ID: 2, Date: 2025-01-11, Amount: 1000.0, Description: Rent

ID: 3, Date: 2025-01-12, Amount: 300.0, Description: Groceries

Map of Transactions:

Key: 1, ID: 1, Date: 2025-01-10, Amount: 500.0, Description: Food

Key: 2, ID: 2, Date: 2025-01-11, Amount: 1000.0, Description: Rent

Key: 3, ID: 3, Date: 2025-01-12, Amount: 300.0, Description: Groceries

**Task 12:** Utilize Kotlin's collection libraries to manage a collection of User objects, enabling the addition and removal of users.  
  
CODE:

package day3tasks9to12  
  
data class User(val id: Int, val name: String, val email: String)  
class CollectonOfUser {  
 private val users: MutableList<User> = *mutableListOf*()  
  
 fun addUser(user: User) {  
 users.add(user)  
 *println*("User added: $user")  
 }  
  
 fun removeUserById(userId: Int) {  
 val user = users.*find* **{ it**.id == userId **}** if (user != null) {  
 users.remove(user)  
 *println*("User removed: $user")  
 } else {  
 *println*("User with ID $userId not found.")  
 }  
 }  
  
 fun displayUsers() {  
 if (users.isEmpty()) {  
 *println*("Users Not Found In Collection")  
 } else {  
 *println*("List of Users:")  
 users.*forEach* **{** *println*(**it**) **}** }  
 }  
}  
  
fun main() {  
 val userManager = CollectonOfUser()  
  
 val user1 = User(1, "Moulali", "moulali@gmail.com")  
 val user2 = User(2, "Abdhul", "abdhul@gmail.com")  
 val user3 = User(3, "Subbu", "subbu@gmail.com")  
  
 userManager.addUser(user1)  
 userManager.addUser(user2)  
 userManager.addUser(user3)  
  
 // Display all users  
 userManager.displayUsers()  
  
 userManager.removeUserById(2)  
  
 userManager.displayUsers()  
  
 userManager.removeUserById(7)  
}

OUTPUT:

User added: User(id=1, name=Moulali, email=moulali@gmail.com)

User added: User(id=2, name=Abdhul, email=abdhul@gmail.com)

User added: User(id=3, name=Subbu, email=subbu@gmail.com)

List of Users:

User(id=1, name=Moulali, email=moulali@gmail.com)

User(id=2, name=Abdhul, email=abdhul@gmail.com)

User(id=3, name=Subbu, email=subbu@gmail.com)

User removed: User(id=2, name=Abdhul, email=abdhul@gmail.com)

List of Users:

User(id=1, name=Moulali, email=moulali@gmail.com)

User(id=3, name=Subbu, email=subbu@gmail.com)

User with ID 7 not found.