Development Scenario 1: Personal Finance Tracker

Day 1: Introduction and Setup and Variables and Control Structures

Task 1: Install Kotlin and configure IntelliJ IDEA. Verify the setup by running a "Hello, World!" program.

Step 1: Install Kotlin

1. Download Kotlin.
2. Set up Environment Variables and Update PATH.
3. Verify Installation in command prompt with command ‘kotlinc -version’.

Step 2: Install IntelliJ IDEA

1. Download IntelliJ IDEA.
2. Launch and open IntelliJ IDEA.

Step 3: Configure IntelliJ IDEA for Kotlin

1. **Create a New Project**
   - Create new project or ‘File -> New ->project’.
   - Select ‘Kotlin’ from list and click next.
2. **Configure Project**
   - Give name to project and give location to save it.
   - Click ‘Finish’.
3. **Write and Run First Program**
   - In the Project view (usually on the left side), right-click on the ‘src’ folder.
   - Choose New -> Kotlin File/Class.
   - Enter a name for your file (e.g., HelloWorld) and click OK.
   - IntelliJ IDEA will create a Kotlin file with a main function.
   - Write Code Inside function
   - Code: fun main(){

     Println(“Hello World”)

     }

   - Save The file
4. **Run The Program**
   - Right-click inside the editor window where your main function is defined.
   - Select Run ‘your file name’ (ex. Run 'HelloWorld’)
   - You should see Hello, World printed in the Run window at the bottom of IntelliJ IDEA.

Task 2: Explore Kotlin REPL (Read-Eval-Print Loop) to familiarize with Kotlin syntax and basic operations.

Kotlin provides an interactive shell known as the Kotlin REPL (Read-Eval-Print Loop) that allows you to quickly evaluate Kotlin code snippets and expressions interactively. Using the Kotlin REPL is straightforward and provides a convenient way to test Kotlin code without needing to compile a full Kotlin program.

1. **Start Kotlin REPL**
    - Open Command prompt and type 'Kotlinc-jvm' and press enter to start Kotlin REPL
2. **Let Explore Some Operations**
    - Once the REPL starts,it will show('>>>') in command prompt by default.
    - Some arithmetic operations:
        i. Addition
            - >>> 5 + 3
            - Output: 8
        ii. Subtraction
            - >>> 5 – 3
            - Output: 2
        iii. Multiplication
            - >>> 5 * 3
            - Output: 15
        iv. Division
            - >>> 5 – 3
            - Output: 1.6
3. **Declare and Use Variables**:
    - Addition operation using Variables
        - >>> val a = 10
          >>> val b = 5
          >>> val sum = a + b
          >>> println("Sum: $sum")
        - Output: Sum: 15
4. **Define Functions**:
    - >>> fun greet(name: String) {
      >>> println("Hello, $name!")
      >>> }
      >>> greet("Chand")
    - Output: Hello, Chand

Task 3: Create a Transaction class with properties such as amount, date, and category.

```
class Transaction(
        val amount: Double,
        val date: LocalDate,
        val category: String
){
        override fun toString() : String{
                return “Transaction(Amount=$amount, Date=$date, Category=$category’)”
        }
}
fun main(){
        val transact = Transaction(20.0, LocalDate.now(), “Groceries”)
        println(transact)
}
```

Output: Amount: 100.0

Date: 2024-07-18

Category: Groceries

Task 4: Implement control structures to categorize transactions (e.g., Food, Utilities, Entertainment) using when statements.

```
class Transaction(
        val amount: Double,
        val date: LocalDate,
        val category: String
){
fun categorize(): String {
        return when (category.toLowerCase()) {
                "Lays", "sweet bun","yippee" -> "Food"
                 "gas", "water", "telephone" -> "Utilities"
                 "movies", "music", "circus" -> "Entertainment"
                 else -> "Other"
        }
```

```
}
override fun toString(): String {
        return "Transaction(amount=$amount, date=$date, category='$category')"
        }
}
fun main() {
   val transaction1 = Transaction(150.0, LocalDate.now(), "Movies")
   val transaction2 = Transaction(60.0, LocalDate.now(), "yippee")
   val transaction3 = Transaction(10.0, LocalDate.now(), "water")
   val transaction4 = Transaction(30.0, LocalDate.now(), "Transportation")

   println("${transaction1.category} is categorized as ${transaction1.categorize()}")
   println("${transaction2.category} is categorized as ${transaction2.categorize()}")
   println("${transaction3.category} is categorized as ${transaction3.categorize()}")
   println("${transaction4.category} is categorized as ${transaction4.categorize()}")
}
```

Output: Movies is categorized as Entertainment

yippee is categorized as Food

water is categorized as Utilities

Transportation is categorized as Other