Assignment: 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 IntelliJ IDEA

1. Download IntelliJ IDEA:

- Go to the [JetBrains website](https://www.jetbrains.com/idea/download/).

- Select community version

- Download and run the installer.

- Launch IntelliJ IDEA after installation.

Step 2: Install Kotlin Plugin

1. Open IntelliJ IDEA.

2. Install Kotlin Plugin:

- Go to file -> Settings

- Select Plugins from the sidebar.

- Click on the Marketplace tab and search for Kotlin.

- Click Install.

Step 3: Create a New Kotlin Project

1. Create a New Project:

- Open IntelliJ IDEA.

- Click New Project or go to File -> New -> Project.

2. Select Project Type:

- Select Kotlin from the sidebar.

- Choose Kotlin/JVM and click Next.

3. Configure Project Settings:

- Name your project (e.g. HelloWorld).

- Choose a location for your project.

- Click on Finish.

Step 4: Write and Run "Hello World" Program

1. Create a Kotlin File:

- In the Project Explorer, right-click the src folder.

- Select New -> Kotlin File.

- Name the file Main.

2. Write the Code:

- Open Main.kt and add:

```
fun main() {

println("Hello, World!")

}
```

3. Run the Program:

- Click the green play button next to the main function.

- Or right-click Main.kt and select Run Main.main().

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

Kotlin REPL (Read-Eval-Print Loop) is a great way to experiment with Kotlin syntax and basic operations interactively.

Step 1: Open Kotlin REPL in IntelliJ IDEA
- A new window will open where you can enter Kotlin code and see the results.
Step 2: Kotlin Syntax and Basic Operations

1. Basic Arithmetic:

```
val sum = 8 + 1
println(sum)
// Output: 9

val difference = 15 - 3
println(difference)
// Output:12

val product = 8 * 2
println(product)
// Output: 16

val quotient = 80 / 2
println(quotient)
```

```
// Output: 40
```

2. String Interpolation

```
val name = “Rohit"
println("Hello, $name")
// Output: Hello, Rohit
```

3. Variables and Constants

```
var mutableVariable = 100
println(mutableVariable)
// Output: 100

val immutableVariable = 3005
println(immutableVariable)
// Output: 3005
```

4. Control Structures

```
val number = 10
if (number > 5) {
println("$number is greater than 5")
} else {
println("$number is not greater than 5")
}
// Output: 10 is greater than 5
```

5. Loops

Do-While loop:

```
var i = 0
do {
println(i)
i++
} while(i < 5)
```

While loop:

```
var i = 0
while(i < 10) {
if(i == 4) {
i++
continue
}
if(i == 6) {
break
}
println(i)
i++
}
```

For loop:

```
for(num in 34..45){
println(num)
}
```

6. Functions

```
fun greet(name: String) {
println("Hello, $name!")
}
greet("Aryan")
// Output: Hello, Aryan!
```

7. Array

```
val cars = arrayOf("Fortuner", "BMW 330i", "Tourbillion", "RR", "Nexon", "Porsche “)
for(car in cars) {
println(car)
}
// output: Fortuner   BMW 330i   Tourbillion   RR   Nexon   Porsche
```

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

Step 1: create a Kotlin Project

Step 2: Create a Kotlin Class

1. Create a New Kotlin File:
- In the Project Explorer, right-click on the src folder.
- Select New -> Kotlin File/Class.
- Name the file Transaction.

2. Define the Transaction Class:-
- Open the newly created Transaction.kt file and add the following code

```
import java.util.Date
data class Transaction(
val amount: Double,
val date: Date,
val category: String
 )
```

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

```
import java.util.Date

data class Transaction(
val amount: Double,
val date: Date,
val category: String
 )
{
fun displayTransactionDetails() {
println("Transaction Details:")
 println("Amount: $$amount")
println("Date: $date")
println("Category: $category")
}
fun categorizeTransaction(): String
{
 return when (category.toLowerCase())
 { "food" -> "Food"
"utilities" -> "Utilities"
"entertainment" -> "Entertainment"
 else -> "Other"
}}}
```

- Open the newly created `Main.kt` file and add the following code

```
fun main() {
val transactions = listOf(
Transaction(amount = 150.0, date = Date(), category = "Food"),
Transaction(amount = 200.0, date = Date(), category = "Utilities"),
Transaction(amount = 50.0, date = Date(), category = "Entertainment"),
Transaction(amount = 100.0, date = Date(), category = "Misc") )
 // Display transaction details and categories
for (transaction in transactions) {
 transaction.displayTransactionDetails()
 val category = transaction.categorizeTransaction()
 println("Categorized as: $category")
println()
} }
```

3. Run the Program:
- Click the green play button next to the main function in Main.kt.