**Lab 13- Coroutines in Kotlin**

In this exercise, we'll set up a Kotlin project with Maven and use coroutines to simulate asynchronous tasks.

**Setup:**

* Create a new directory for your project.
* Inside the project directory, create a file named pom.xml with the following Maven configuration:

<!-- pom.xml -->

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>coroutine-lab</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

<kotlin.version>1.5.30</kotlin.version>

</properties>

<dependencies>

<dependency>

<groupId>org.jetbrains.kotlin</groupId>

<artifactId>kotlin-stdlib-jdk8</artifactId>

<version>${kotlin.version}</version>

</dependency>

<dependency>

<groupId>org.jetbrains.kotlinx</groupId>

<artifactId>kotlinx-coroutines-core</artifactId>

<version>1.5.0</version>

</dependency>

</dependencies>

</project>

**Task:**

Inside the project directory, create a Kotlin file named CoroutineLab.kt.

**Exercise:**

Define a simple function that simulates an asynchronous task using a coroutine.

// CoroutineLab.kt

import kotlinx.coroutines.GlobalScope

import kotlinx.coroutines.delay

import kotlinx.coroutines.launch



suspend fun fetchData(): String {

delay(2000) // Simulate a delay (e.g., network request)



return "Data from server"



}

Usage:

In the main function or another function, call the fetchData function using a coroutine.

// CoroutineLab.kt

import kotlinx.coroutines.runBlocking

fun main() = runBlocking {

println("Fetching data...")

// Launch a coroutine to fetch data asynchronously

val result = fetchData()



// This line will be printed after the delay in fetchData

println("Received data: $result")

}

**Build and Run:**

**Observation:**

Observe the output in the console. The "Fetching data..." message should be printed, followed by the "Received data: Data from server" message after a simulated delay.

This simple lab exercise introduces the basic concepts of coroutines in Kotlin using Maven for project management. Feel free to experiment further or extend the exercise to gain a deeper understanding of coroutine concepts.

import kotlinx.coroutines.\*

fun main() = runBlocking {

println("Program starts on: ${Thread.currentThread().name}")

// Launch a coroutine

launch {

println("Coroutine starts on: ${Thread.currentThread().name}")

delay(1000L) // Simulate a delay

println("Coroutine ends on: ${Thread.currentThread().name}")

}

println("Main program continues on: ${Thread.currentThread().name}")

delay(2000L) // Keep the main thread alive for a while

println("Program ends on: ${Thread.currentThread().name}")

}