Lab 4: Multi-module Maven build with parallel test execution

Step 1 — Project Structure

We'll create a **multi-module finance app** with:

- `core` → forex conversion logic
- `tax` → simple tax calculator
- `app` → integrates both modules

GITHUB Reference: GitHub - cloud-dev-user/finance-multi-module

```
finance-multi-module/
 ├── pom.xml (parent)
 ├── core/
 │   ├── pom.xml
 │   └── src/main/java/com/example/core/ForexConverter.java
 │   └── src/test/java/com/example/core/ForexConverterTest.java
 ├── tax/
 │   ├── pom.xml
 │   └── src/main/java/com/example/tax/TaxCalculator.java
 │   └── src/test/java/com/example/tax/TaxCalculatorTest.java
 └── app/
     ├── pom.xml
     └── src/main/java/com/example/app/MainApp.java

```

Step 2 — Parent `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>finance-parent</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>pom</packaging>

    <modules>
        <module>core</module>
        <module>tax</module>
        <module>app</module>
    </modules>

    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
        <junit.version>5.9.3</junit.version>
    </properties>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.junit.jupiter</groupId>
                <artifactId>junit-jupiter</artifactId>
                <version>${junit.version}</version>
                <scope>test</scope>
            </dependency>
        </dependencies>
```

```
    </dependencyManagement>
</project>

```

Step 3 — Core Module

core/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">
    <parent>
        <groupId>com.example</groupId>
        <artifactId>finance-parent</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>core</artifactId>
</project>

```

ForexConverter.java

```
package com.example.core;

public class ForexConverter {
    public double inrToUsd(double inr) { return inr * 0.012; }
    public double usdToInr(double usd) { return usd / 0.012; }
}

```

ForexConverterTest.java

```
package com.example.core;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class ForexConverterTest {
    @Test
    void testInrToUsd() {
        ForexConverter fx = new ForexConverter();
        assertEquals(12, fx.inrToUsd(1000), 0.5);
    }
}

```

Step 4 — Tax Module

tax/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">
    <parent>
        <groupId>com.example</groupId>
        <artifactId>finance-parent</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>tax</artifactId>
</project>
```

```
13
```

TaxCalculator.java

```
package com.example.tax;

public class TaxCalculator {
    public double calculateGST(double amount) {
        return amount * 0.18;
    }
}

```

TaxCalculatorTest.java

```
package com.example.tax;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class TaxCalculatorTest {
    @Test
    void testGST() {
        TaxCalculator t = new TaxCalculator();
        assertEquals(18, t.calculateGST(100), 0.1);
    }
}

```

Step 5 — App Module

app/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">
    <parent>
        <groupId>com.example</groupId>
        <artifactId>finance-parent</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>app</artifactId>

    <dependencies>
        <dependency>
            <groupId>com.example</groupId>
            <artifactId>core</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
        <dependency>
            <groupId>com.example</groupId>
            <artifactId>tax</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
    </dependencies>
</project>

```

MainApp.java

```
package com.example.app;

import com.example.core.ForexConverter;
import com.example.tax.TaxCalculator;

```

```
public class MainApp {
    public static void main(String[] args) {
        ForexConverter fx = new ForexConverter();
        TaxCalculator tx = new TaxCalculator();

        System.out.println("1000 INR in USD: " + fx.inrToUsd(1000));
        System.out.println("GST on 1000 INR: " +
tx.calculateGST(1000));
    }
}

```

Step 6 — Jenkins Scripted Pipeline with Parallel Tests

`Jenkinsfile`

```
node {
    stage('Checkout') {
        checkout scm
    }

    stage('Build') {
        sh 'mvn clean install -DskipTests'
    }

    stage('Parallel Tests') {
        parallel(
            "Core Tests": {
                dir('core') {
                    sh 'mvn test'
                    junit 'target/surefire-reports/*.xml'
                }
            },
            "Tax Tests": {
                dir('tax') {
                    sh 'mvn test'
                    junit 'target/surefire-reports/*.xml'
                }
            },
            "App Tests": {
                dir('app') {
                    sh 'mvn test'
                    junit 'target/surefire-reports/*.xml'
                }
            }
        )
    }

    stage('Package') {
        sh 'mvn package -DskipTests'
        archiveArtifacts artifacts: '**/target/*.jar', fingerprint:
true
    }
}

```

Step 7 — Validation

1. Run pipeline → ensure **tests execute in parallel** (faster build).
2. Verify JUnit results in Jenkins.
3. Check `target/` folders for built `.jar`s.

Optional Challenges

🔹 **Challenge 1: Add Code Coverage with JaCoCo**

- Update `pom.xml` in parent:

```
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.8</version>
    <executions>
        <execution>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
        </execution>
        <execution>
            <id>report</id>
            <phase>verify</phase>
            <goals>
                <goal>report</goal>
            </goals>
        </execution>
    </executions>
</plugin>

```

- Add Jenkins stage:

```
stage('Code Coverage') {
    sh 'mvn verify'
    publishHTML([allowMissing: false,
                 alwaysLinkToLastBuild: true,
                 keepAll: true,
                 reportDir: 'target/site/jacoco',
                 reportFiles: 'index.html',
                 reportName: 'JaCoCo Coverage'])
}

```

🔹 **Challenge 2: Integrate SonarQube**

- Configure SonarQube server in Jenkins (`Manage Jenkins → Configure System`).
- Add Jenkins stage:

```
stage('SonarQube Analysis') {
    withSonarQubeEnv('MySonarQube') {
        sh 'mvn sonar:sonar'
    }
}

```

🔹 **Challenge 3: Integration Test Across Modules**

- Write an **integration test** in `app/src/test/java` that:
  - Calls `ForexConverter` + `TaxCalculator` together.
  - Example:

```
@Test
void testFullWorkflow() {
    ForexConverter fx = new ForexConverter();
    TaxCalculator tx = new TaxCalculator();
```

```
    double usd = fx.inrToUsd(1000);
    double gst = tx.calculateGST(1000);
    assertTrue(usd > 0 && gst > 0);
}

```