**Exercise 4: Creating and Configuring a Maven Project**

**Scenario:**

You need to set up a new Maven project for the library management application and add Spring dependencies.

**1. Create a New Maven Project**

In this exercise, we create a new Maven project for the Library Management application. The project is named LibraryManagement. This project will manage a library's operations, including adding/removing books, managing members, and borrowing/returning books.

**2. Add Spring Dependencies in pom.xml**

The pom.xml file is the core of a Maven project. It manages project dependencies, plugins, and configurations. Update the file.

**Updates:**

* **Spring Context Dependency**: This dependency is essential for managing the application context in Spring.
* **Spring AOP Dependency**: This is used for Aspect-Oriented Programming, allowing separation of cross-cutting concerns like logging and transaction management.
* **Spring WebMVC Dependency**: This dependency is crucial for creating web applications, especially if you're using Spring MVC architecture.
* **AspectJ Weaver Dependency**: Used for weaving aspects into your Spring beans at runtime.

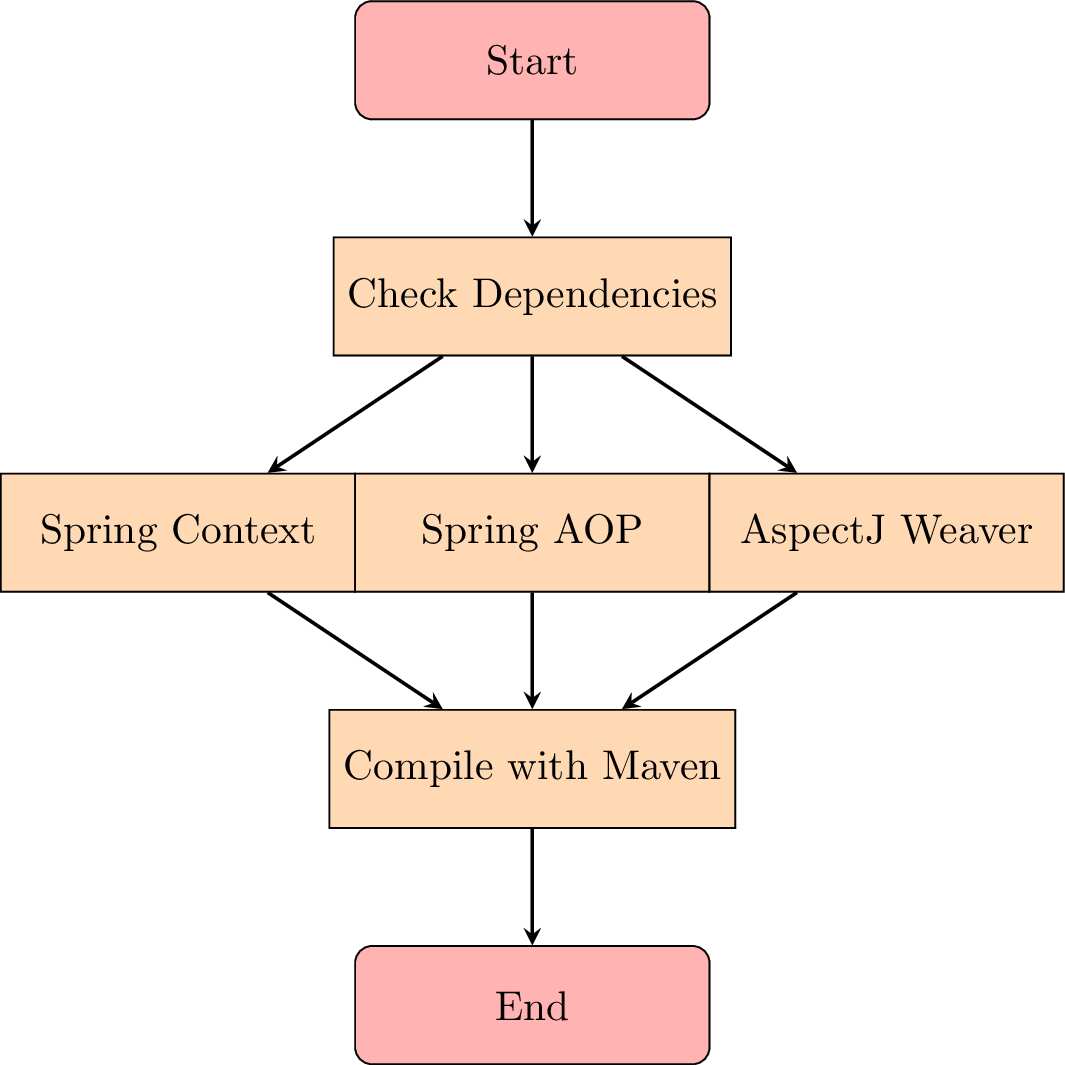
**3. Configure Maven Plugins**

In the pom.xml, we configured the Maven Compiler Plugin to ensure that the project is compiled with Java 1.8. This is critical for maintaining compatibility and ensuring that all features used in the project are supported.

**Explanation:**

* **Maven Compiler Plugin**: This plugin is used to compile Java source files. The configuration ensures that both the source code and the bytecode are compatible with Java version 1.8.

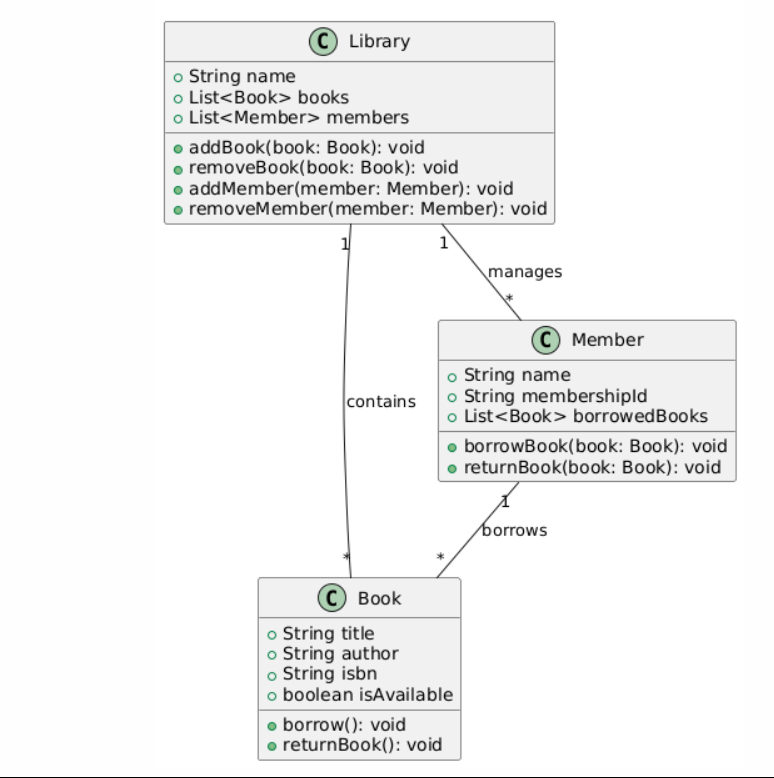
**FLOWCHART of the program :**



**Explanation:**

* The process begins with checking and resolving the necessary dependencies (Spring Context, Spring AOP, and AspectJ Weaver).
* Each dependency is processed individually to ensure that it is correctly integrated into the project.
* Finally, Maven compiles the project using the specified Java version (1.8).

**CLASS DIAGRAM :**



The class diagram represents the core components of the Library Management System, showing the relationships between the Library, Book, and Member classes.

**Explanation:**

* **Library**: Manages the collection of Book and Member objects. It contains methods for adding/removing books and members.
* **Book**: Represents the books in the library. Each book has attributes like title, author, ISBN, and availability status.
* **Member**: Represents the members of the library. Members can borrow and return books.

The relationships between the classes are shown through associations:

* **Library** contains multiple Book and Member objects.
* **Member** can borrow multiple Book objects.

Output:

