Java Version Compatibility with Spring Boot

Understanding the Error

When you run `mvn spring-boot:run` with Spring Boot 3.4.3 and encounter the error “class file has wrong version 61.0, should be 52.0,”it indicates a Java version compatibility issue. In Java, class files have a version number corresponding to the Java version they were compiled with:

- **Version 52.0** corresponds to Java 8.
- **Version 61.0** corresponds to Java 17.

The error suggests that your Spring Boot application’s class files were compiled with Java 17 (version 61.0), but the Java Virtual Machine (JVM) attempting to run them expects class files compatible with Java 8 (version 52.0). Since Java 8’s JVM cannot execute class files compiled with Java 17 (because Java is backward compatible but not forward compatible in this context), you get this error.

Why This Happens

Spring Boot 3.4.3 requires **Java 17 or later** as its minimum supported version. This means your project should be both compiled and run using at least Java 17. The mismatch likely occurs because:

1. **Compilation**: Your project is configured to compile with Java 17, producing class files with version 61.0. This is typically set in your `pom.xml` with the `maven-compiler-plugin` or the `<java.version>` property.
2. **Runtime**: When you execute `mvn spring-boot:run`, the Spring Boot Maven plugin launches a new Java process to run the application. However, the `java` executable it uses is from a Java 8 installation (version 52.0), which cannot handle the Java 17-compiled class files.

Steps to Fix the Issue

To resolve this, you need to ensure that both the compilation and runtime environments use Java 17. Here’ s how to fix it:

1. Verify Your Project’s Java Version First, confirm that your project is set to use Java 17. In your `pom.xml`, check for the following:

```
<properties>
    <java.version>17</java.version>
</properties>
```

This property tells the `maven-compiler-plugin` to compile your code with Java 17. Spring Boot 3.4.3 sets this by default, but it’s good to verify. If it’s missing or set to a different version (e.g., 8), update it to 17.

2. Install Java 17 Ensure that Java 17 is installed on your system. You can download it from:

- Adoptium (Eclipse Temurin)
- Oracle JDK (if you accept the license terms)

To check if Java 17 is installed, open a terminal and run:

```
java -version
```

If it doesn't show Java 17 (e.g., `openjdk 17.x.x` or similar), install it and proceed.

3. Update Your Environment to Use Java 17 The Spring Boot Maven plugin uses the `java` executable from your environment, typically determined by the `JAVA_HOME` environment variable or the `java` command in your system's PATH. If your current `java` command points to Java 8, you need to update it.

Option A: Set JAVA_HOME and PATH Set your `JAVA_HOME` environment variable to point to the Java 17 installation and ensure the `java` command in your PATH uses it.

- **On Linux/Mac**:
  1. Find your Java 17 installation directory (e.g., `/usr/lib/jvm/java-17-openjdk` or wherever you installed it).
  2. Set `JAVA_HOME` and update `PATH` in your terminal:
     ```
     export JAVA_HOME=/path/to/java-17
     export PATH=$JAVA_HOME/bin:$PATH
     ```
  3. Verify:
     ```
     java -version
     ```
     It should now show Java 17.

  To make this permanent, add the `export` lines to your shell configuration file (e.g., `~/.bashrc`, `~/.zshrc`).

- **On Windows**:
  1. Find the Java 17 installation directory (e.g., `C:\Program Files\Java\jdk-17`).
  2. Set `JAVA_HOME`:
     - Open the System Properties window (search for "Environment Variables"in the Start menu).
     - Under "System Variables,"add or update `JAVA_HOME` to `C:\Program Files\Java\jdk-17`.
  3. Update `Path`:
     - In "System Variables,"edit `Path` and add `%JAVA_HOME%\bin` (ensure it's at the top or remove older Java paths like Java 8).
  4. Open a new Command Prompt and verify:
     ```
     java -version
     ```
     It should show Java 17.

Option B: Use a Specific Java Version for Maven If you have multiple Java versions installed and don't want to change your system-wide `JAVA_HOME`, you can run Maven with a specific Java version:

- Before running `mvn spring-boot:run`, set `JAVA_HOME` just for that command:

  ```
  JAVA_HOME=/path/to/java-17 mvn spring-boot:run
  ```

  This ensures the Spring Boot Maven plugin uses Java 17 without altering your system settings.

4. Run the Application Again With Java 17 properly set, run:

```
mvn spring-boot:run
```

The plugin will now use Java 17 to execute your application, matching the compiled class file version (61.0), and the error should disappear.

Additional Checks

- **Maven's Java Version**: Run `mvn -version` to see which Java version Maven uses. It should match your `JAVA_HOME`. If not, ensure your PATH and `JAVA_HOME` are correctly configured.
- **Dependencies**: If the error persists, a dependency might be compiled with Java 17 while your runtime is still misconfigured. Since Spring Boot 3.4.3 enforces Java 17, ensuring your runtime is Java 17 should resolve this.

Why Not Java 8?

Spring Boot 3.4.3 does not support Java 8. If you need to use Java 8, you'd have to downgrade to an older Spring Boot version (e.g., 2.x), but that's not recommended unless you have specific constraints. Since your class files are version 61.0, sticking with Java 17 aligns with Spring Boot 3.4.3's requirements.

Final Solution Summary

To fix "class file has wrong version 61.0, should be 52.0"when running `mvn spring-boot:run` with Spring Boot 3.4.3:

1. Ensure `<java.version>17</java.version>` is in your `pom.xml`.
2. Install Java 17 if not already present.
3. Set `JAVA_HOME` to your Java 17 installation directory.
4. Update your PATH so `java -version` shows Java 17.
5. Run `mvn spring-boot:run` again.

This ensures your runtime matches your compiled Java 17 class files, resolving the compatibility issue.