Introduction to Java for Python and C# Programmers

Author: Yiru Zhang
ESILV

Guiline for reading: Section 1: What is Java, Section 2: Basic Syntax Overview, and Section 3: Quick Java Coding Tips for Python and C# Developers give a brief comparative introduction on Java for a quick-start coding in Java.

Section 4: Brief Introduction to Maven gives basic concepts of Maven, a depencency management tool for Java project, along with an explication of the configuration file.

Section 5: introduces the environment setting procedures for Java and Maven. Normally you have already installed Java. Section 6: guides you build the project with Maven.

If you want to quick starting the lab, you can start from Section 4.

1. What is Java?

- **Java** is a high-level, object-oriented programming language widely used for building enterprise-scale applications, web services, and more.
- **Key Features:**
    - **Platform Independence:** Write once, run anywhere (WORA) via the Java Virtual Machine (JVM).
    - **Object-Oriented:** Emphasizes objects and classes, similar to C#.
    - **Rich Standard Library:** Extensive APIs for various functionalities.

2. Basic Syntax Overview

While Java shares similarities with C#, it has distinct syntax and conventions. Here's a quick comparison:

Hello World Example

Python:

```
print("Hello, World!")
```

C#:

```
using System;

class Program {
    static void Main(string[] args) {
        Console.WriteLine("Hello, World!");
    }
}
```

Java:

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Key Differences:

- **Classes and Methods:**
  - Java requires all code to reside within classes.
  - The `main` method is the entry point, similar to C#.
- **Data Types:**
  - Primitive types (e.g., `int`, `double`, `boolean`) are similar to C#.
  - Object types (e.g., `String`) are used extensively.
- **Access Modifiers:**
  - `public`, `private`, `protected` control visibility, akin to C#.

---

3. Quick Java Coding Tips for Python and C# Developers

To help you get started with Java coding, here are some quick tips highlighting differences and similarities with Python and C#:

a. Variable Declaration and Types

- **Static Typing:** Java requires explicit type declarations.

  Python:

  ```python
  name = "Alice"
  age = 30
  ```

  Java:

  ```java
  String name = "Alice";
  int age = 30;
  ```

- **Primitive vs. Object Types:**
  - Java has primitive types (`int`, `double`, `boolean`) and their corresponding wrapper classes (`Integer`, `Double`, `Boolean`).

b. Control Structures

- **Similar to C#:**

  If-Else Statement:

  ```java
  if (age > 18) {
      System.out.println("Adult");
  } else {
      System.out.println("Minor");
  }
  ```

  For Loop:

  ```java
  for (int i = 0; i < 5; i++) {
      System.out.println(i);
  }
  ```

c. Methods and Functions

- **Method Declaration:**

  Python:

  ```python
  def greet(name):
      print(f"Hello, {name}!")
  ```

  Java:

  ```java
  public void greet(String name) {
      System.out.println("Hello, " + name + "!");
  }
  ```

- **Static Methods:**

  ```java
  public static void greet(String name) {
      System.out.println("Hello, " + name + "!");
  }
  ```

d. Classes and Objects

- **Class Definition:**

  Python:

  ```python
  class Person:
      def __init__(self, name, age):
          self.name = name
          self.age = age
  ```

  Java:

  ```java
  public class Person {
      private String name;
      private int age;

      public Person(String name, int age) {
          this.name = name;
          this.age = age;
      }

      // Getters and setters
      public String getName() {
          return name;
      }

      public void setName(String name) {
          this.name = name;
      }

      public int getAge() {
          return age;
      }

      public void setAge(int age) {
          this.age = age;
      }
  }
  ```

e. Exception Handling

- **Try-Catch Blocks:**

  Python:

  ```python
  try:
      result = 10 / 0
  ```

```
except ZeroDivisionError:
    print("Cannot divide by zero.")
```

Java:

```
try {
    int result = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Cannot divide by zero.");
}
```

f. Access Modifiers

- **Public, Private, Protected:**
  - Similar to C#, Java uses these keywords to control access to classes, methods, and variables.

---

4. Brief introduction to Maven

Apache Maven is a **build automation and dependency management tool** for Java projects. It simplifies project builds, dependency resolution, and lifecycle management. We only introduce the core concepts in Maven for a quick-start of the project.

POM (`pom.xml`) Maven Uses a `pom.xml` file to define dependencies, plugins, and project configuration. In this Jena project, the `pom.xml` file is already configurerd and you do not need to modify it unless necessary.

Let's have a quick look of the `pom.xml` file of this project:

```
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.esilv</groupId>
    <artifactId>websemantics</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>23</maven.compiler.source>
        <maven.compiler.target>23</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.apache.jena</groupId>
            <artifactId>apache-jena-libs</artifactId>
            <type>pom</type>
            <version>5.2.0</version>
        </dependency>
    </dependencies>
</project>
```

The minimum requirement for a POM are the following:

`project` root
`modelVersion` - should be set to 4.0.0
`groupId` - the id of the project's group.
`artifactId` - the id of the artifact (project)
`version` - the version of the artifact under the specified group
`maven.compiler.source`, `maven.compiler.target` - Java version used to compile the project. In this file, Java 23 is specified. If you are using another version, modify this value to your version.
`depencencies` - all dependencies required by the project, with each one described by a `depencency`.
`denpencency` - description of one depencency. In this project, we are using apache jena of version 5.2.0. (You can set to other versions).\

5. Setting Up Your Java Development Environment (with Maven)

To work with Java and Maven, you'll need to set up your development environment.

Check if you have already installed Java and Maven

Run

```
java -version
```

and

```
mvn -version
```

(`mvn` implies *Maven*)

If **Java is installed**, the expected output should be like:

```
openjdk version "23.0.2" 2025-01-21
OpenJDK Runtime Environment Homebrew (build 23.0.2)
OpenJDK 64-Bit Server VM Homebrew (build 23.0.2, mixed mode, sharing)
```

If **Maven is installed**, the expected output should be like:

```
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: /opt/homebrew/Cellar/maven/3.9.9/libexec
Java version: 23.0.2, vendor: Homebrew, runtime:
/opt/homebrew/Cellar/openjdk/23.0.2/libexec/openjdk.jdk/Contents/Home
```

```
Default locale: en_CN, platform encoding: UTF-8
OS name: "mac os x", version: "14.5", arch: "aarch64", family: "mac"
```

if **Java or Maven is not installed**, you'll see:

```
command not found: java
```

and

```
command not found: mvn
```

or some similar instructions implying that the command "java" or "mvn" cannot be found or recogonized.

Below are procedures to install a. **Java** and b. **Maven**, following the correspoding ones for what you don't have.

a. Install Java Development Kit (JDK)

1. **Download JDK:**
   - Visit the Oracle JDK Downloads or use OpenJDK (recommended for open-source projects).
   - Choose the appropriate version (Java 17 LTS is recommended).
2. **Install JDK:**
   - Follow the installation instructions for your operating system.
3. **Set `JAVA_HOME` Environment Variable:**
   - **Windows:**
     - Right-click on `This PC` > `Properties` > `Advanced system settings` > `Environment Variables`.
     - Click `New` under System variables.
     - Variable name: `JAVA_HOME`
     - Variable value: Path to your JDK installation (e.g., `C:\Program Files\Java\jdk-17`)
   - **macOS/Linux:**
     - Add the following to your `.bash_profile` or `.bashrc`:

       ```
       export JAVA_HOME=/path/to/jdk
       export PATH=$JAVA_HOME/bin:$PATH
       ```
4. **Verify Installation:**

   ```
   java -version
   ```

- You should see the installed Java version.

b. Install Maven

Words before starting this step: If you are using Ubuntu, install Maven directly by

```
sudo apt install maven -y
```

If you are using Mac OS, installing Maven with **Homebrew** is quite simple. Otherwise, follow the steps below:

1. **Download Maven:**
   - Visit the Apache Maven Downloads.
   - Download the binary zip archive.
2. **Install Maven:**
   - Extract the archive to a directory of your choice (e.g., `C:\Program Files\Maven`).
3. **Set `MAVEN_HOME` Environment Variable:**
   - **Windows:**
     - Variable name: `MAVEN_HOME`
     - Variable value: Path to your Maven installation (e.g., `C:\Program Files\Maven\apache-maven-3.8.4`)
   - **macOS/Linux:**
     - Add to your `.bash_profile` or `.bashrc`:

       ```
       export MAVEN_HOME=/path/to/maven
       export PATH=$MAVEN_HOME/bin:$PATH
       ```
4. **Verify Installation:**

   ```
   mvn -version
   ```

   - You should see Maven version details.

---

6. Build the project with Maven

Download and unzip `labJena.zip`. You can choose to build this project either by command line or by IDEs for Java (such as Eclipse, IntelliJ, VSCode, etc.) For maven with IDEs, please refer to the corresponding sites:

- **Maven with Eclispe**
- **Maven with VSCode**
- **Maven with IntelliJ**

The following instruction build the project with Terminal (or command Line/Powershell)

Setup Instructions (with Terminal)

1. Ensure you have **Java 17+** and **Maven** installed.
2. Open a terminal and navigate to the project directory:

   ```
   cd [Path_to_your_project]
   ```

3. Run the following command to download dependencies and build:

   ```
   mvn clean compile
   ```

4. Run the application:

   ```
   mvn exec:java -Dexec.mainClass="applications.Main"
   ```

Maven will automatically download dependencies specified in pom.xml. Now you should see the query result corresponding to the sentence in `query.txt`:

```
----------------------------
| user     | age           |
============================
| ns:Marie | "69"^^xsd:int |
| ns:Peter | "70"^^xsd:int |
----------------------------
```

Now, you can continue the Lab Part 4.

7. (Additional) Basic Maven Commands

For the reference, you can use Maven commands to manage the project (or with IDEs).

a. Common Maven Commands

- **Compile the Project:**

  ```
  mvn compile
  ```

- **Run Tests:**

  ```
  mvn test
  ```

- **Package the Project:**

  ```
  mvn package
  ```

  - This creates a JAR or WAR file in the `target` directory.

- **Clean the Project:**

  ```
  mvn clean
  ```

- **Install the Package to Local Repository:**

  ```
  mvn install
  ```