Lab 1: Basic Java

Yuxuan Zheng

Summer 2023

Table of contents

Java as a programming language

Java is a compiled programming language that is

- High level
- Class based
- Object oriented

The syntax of Java is similar to that of C and C++, but has fewer low-level facilities.

Compiled Java applications are run on *Java virtual machine* (JVM) regardless of the underlying computer architecture (cross platform).

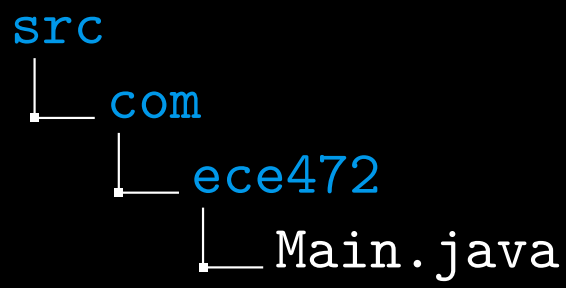
In particular, Hadoop is written in Java, thus knowledge of Java basics is essential to learn Hadoop.

Java development kit

The *Java development kit* (JDK) contains tools we need to use to build a Java application, such as

- Compiler
- Java runtime environment (JRE)
- Archiver (jar)
- Other useful tools

A typical structure of a Java project may look like



Here, `com.ece472` is the *base package*. In Java, a package is used to group a bunch of related classes.

By convention, the base package is the domain name of your company in reverse.

$$\texttt{company.com} \longrightarrow \texttt{com.company}$$

Note: You don't need to have an actual domain on the internet. It is just a way to create a namespace for classes.

Let's take a look at what's inside `Main.java`.

```
package com.ve472;

public class Main {
    public static void main(String[] args) {
        // write your code here
    }
}
```

The class name should be the same as the file name (`Main` inside `Main.java`), and the `main` method should be announced as `public static void`.

Naming conventions

- **Classes:** PascalNamingConvention
- **Methods:** camelNamingConvention
- **Packages:** lower.case.letters

Variable types

Primitive types

- int
- float
- boolean
- ...

Primitive types store values.

Reference types are objects. They store references to objects rather than values.

Note: Although `String` is a reference type, Java allows us to declare a string variable "as if" it is primitive. Declare using

```
String str = "abcd";
```

rather than

```
String str = new String("abcd");
```

Compile a java source code using `javac`.

```
$ javac Main.java
```

It yields a binary file `Main.class`. Change to the `src` directory and run the program with the full package name plus the class name.

```
$ java com.ece472.Main
```

Exercise I

Build a “Hello world” project in Java. Compile and execute the program with command line.

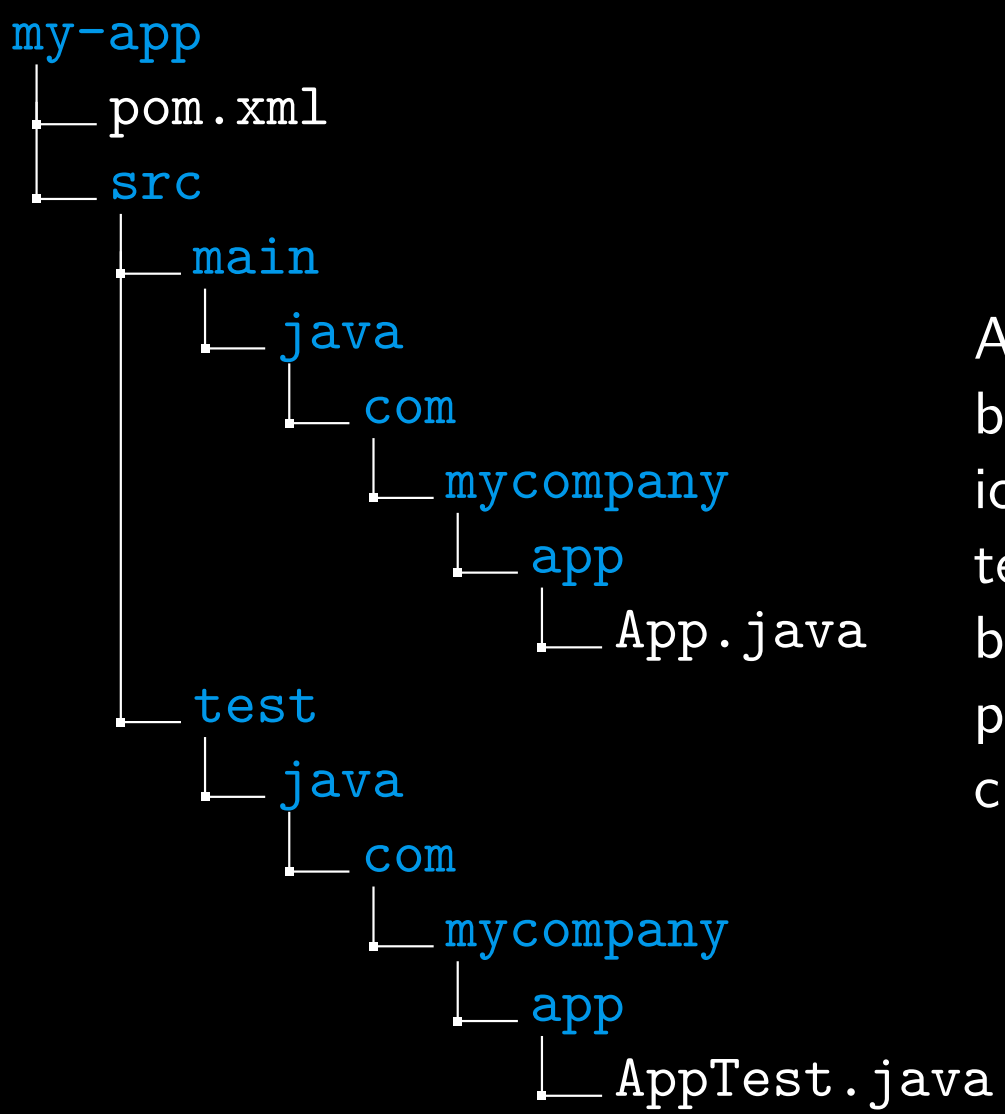
Table of contents

What is Maven?

Maven is a tool that is used to build and manage any Java-based project. Areas of concerns Maven deals with include

- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Encouraging better development practices

Standard project structure



A `target` folder will be generated automatically under `app` after the Maven project is built. It contains compiled classes and test-classes.

The *Project Object Model* (POM) is the core of a Maven project.

- Held in a XML file `pom.xml`
- Contains all necessary information about a project and the build process

```
<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>ece472</groupId>
  <artifactId>ece47211</artifactId>
  <version>1.0</version>
</project>
```

- **groupId:** Unique amongst an organisation or a project
- **artifactId:** Name of the project
- **version:** Version of the project

```
<project ...>
  ...
  <dependencies>
    <dependency>
      <groupId>commons-cli</groupId>
      <artifactId>commons-cli</artifactId>
      <version>1.4</version>
    </dependency>
    ...
  </dependencies>
</project>
```

A dependency list is the cornerstone of POM, since most projects depend on others to build and run correctly.

```
<project ...>
  ...
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.6</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Build handles things like declaring your project's directory structure and managing plugins.

By default, the packages are installed on local under the path `${HOME}/.m2/repository/`.

You can modify `settings.xml` of Maven to customise configurations.

Compile a Maven project using

```
$ mvn compile
```

The output (class binary executables), by default, will be dumped into `${basedir}/target`.

After compilation, run a program using

```
$ mvn exec:java -D exec.mainClass=<your main class>
-D exec.args="arg1 arg2"
```

Use the `pom.xml` given in the lab manual to build a Maven project. The main method takes a command line argument followed by `-i`, and prints the argument to stdout directly.

E.g.

```
$ mvn exec:java -D exec.mainClass=com.ve472.l1.Main
-D exec.args="-i hello"
```

Output:

```
hello
```

An IDE (e.g, IDEA) simplifies the build process, and it could be very helpful.

Thank you!
Any questions?