Java Boot Camp

QUIZ 1 - PRIMER

Develop Java Application

If I want to develop a Java application, what do I need to have installed on my development computer?

- ○ Nothing
- ○ The Java Development Kit (JDK)
- ○ The Java Runtime Environment (JRE)
- ○ The Just In Time (JIT) Compiler

Java Bytecode

Which of the following are true regarding Java Bytecode?

- ❑ Machine/OS dependent code
- ❑ Produced by the compiler when compiling Java source code
- ❑ A form of instruction set designed for efficient execution by the Java JIT compiler
- ❑ Each Bytecode is composed of one byte that represents the opcode, along with zero or more bytes for operands

Java Application

Which of the following are true regarding Java Applications?

- ☐ Java applications developed on a Mac OS can only run on a Mac OS or a Linux based OS
- ☐ Java applications can run anywhere where the JRE is installed
- ☐ Java applications can run anywhere natively
- ☐ Java applications can be found on smartcards, laptops, and datacentres

Advantages of Gradle

What are some advantages of Gradle (or similar build tools such as Maven)?

- ❑ Gradle manages the project dependencies
- ❑ Gradle configuration is part of the source repository, thus anyone can easily clone the repository and build the project without worrying about its dependencies
- ❑ Gradle standardise the way of working with a given project, such as `./gradlew clean build` will build the project
- ❑ Gradle, through its tasks, helps the developer packaging the application into a fat, executable JAR

The Gradle Wrapper

What is the Gradle wrapper?

The Gradle Wrapper is…

- ○ a shortcut to the Gradle installed on the OS
- ○ a script that invokes a declared version of Gradle, downloading it beforehand if necessary
- ○ a script that works with Maven and other build tools
- ○ a Java compiler

Gradle Plugins and Dependencies

What is the difference between the plugins and the dependencies declared in Gradle?

- Plugins defined the project type, while Dependencies are required by the plugins
- Plugins add tasks to Gradle, while Dependencies add libraries to the project
- Plugins add libraries to the project, while Dependencies add tasks to Gradle
- Plugins and Dependencies can be used interchangeably and there is no difference

Gradle Plugins

Are the Gradle plugins packaged as of the application?

- ❑ No, the Gradle plugins add tasks to Gradle
- ❑ No, the Gradle plugins are defined in the Gradle configuration file, which is part of the application source
- ❑ Yes, the Gradle plugins are compiled and packaged with the application as a fat JAR
- ❑ No, a task defined by a Gradle plugin may be used to build or create a fat JAR, but the plugin itself is not part of the fat JAR

Configure Gradle

Given that I am in the project directory, which file do I need to modify to add new Gradle plugins or dependencies?

- ○ `gradle`
- ○ `gradlew`
- ○ `build.gradle`
- ○ `gradle/wrapper/gradle-wrapper.properties`

Gradle Configuration Language

What languages can be used to configure Gradle?

- ❑ Groovy
- ❑ Kotlin DSL for Gradle
- ❑ C++
- ❑ Python

JAR File

What is a JAR file?

- ❑ A JAR is a package file format typically used to aggregate many Java class files and associated metadata and resources into one file for distribution
- ❑ JAR files are archive files that include a Java-specific manifest file (`MANIFEST.MF`)
- ❑ They are built on the ZIP format and typically have a `.jar` file extension
- ❑ JAR file is a typical deployment unit for a Java application

Fat JAR

Why should we create a fat JAR?

- ❑ Fat JARs are not recommended and it is best to packed the application without its dependencies
- ❑ Fat JARs simplify delivery of application as the whole application is packaged in one file
- ❑ Fat JARs simplify execution as only the `-jar` option needs to be provided
- ❑ No need to create a fat JARs as the `-cp` option can <u>easily</u> achieve a similar results as the `-jar`

Running a Fat JAR

Given we have a fat JAR, named `application.jar`, how can we run it?

- `$ javac -jar application.jar`
- `$ java -jar application.jar`
- `$ java application.jar`
- `$ application.jar`

Docker

What are some of the advantages of Docker?

- ❑ Simplifies the application distribution
- ❑ Creates a standard way of how application can be managed by dev-ops
- ❑ Docker is independent from the programming language used to develop the application
- ❑ Only Java applications can be dockerized

Docker Images and Containers

Which of the following are true regarding Docker Images and Containers

- ☐ A Docker Image is a definition of a Docker Container
- ☐ A Docker Container is an instance of a Docker Image
- ☐ Docker Images and Containers are used interchangeably and mean the same thing
- ☐ Only one Docker Container can exists for every Docker Image

Build Docker Image

Given that we have the required docker configuration, how can we build a docker image?

- `$ docker compile . -t my-image:local`
- `$ docker build . -t my-image:local`
- `$ docker execute . -t my-image:local`
- `$ docker it . -t my-image:local`

Dockerize Java Application

How can a Java application be Dockerized?

- Create a fat JAR and then run `java -toDocker application.jar` command
- Create a `Dockerfile` and add the necessary configurations
- A Java application cannot be dockerized
- Compile the Java application using the docker compiler

Dockerize Java Application

What do I need to have to Dockerize a Java application?

- Java applications cannot be dockerized
- Convert the Java application to native code
- Create a docker image that has the corrected version of Java installed
- Run a docker container and install everything you need on it