Building apps using Gradle

Initializing an application

First of all, create a new empty folder named as you want (e.g., demo). In this folder, invoke the gradle init command to start initializing a new Gradle-based project.

This command will show you a dialog form to set up the project you need.

In this form, choose application as the type of the project and **Java** or **Kotlin** as the implementation language.

```
Select type of project to generate:
  1: basic
  2: application
  3: library
  4: Gradle plugin
Enter selection (default: basic) [1..4] 2

Select implementation language:
  1: C++
  2: Groovy
  3: Java
  4: Kotlin
  5: Scala
  6: Swift
Enter selection (default: Java) [1..6] 3
```

Type org.hyperskill.gradleapp as the project name if you would like to precisely follow our example (but it isn't required). You can choose their default options for all other questions since it doesn't matter now.

```
Split functionality across multiple subprojects?:
  1: no - only one application project
  2: yes - application and library projects
Enter selection (default: no - only one application project) [1..2]

Select build script DSL:
  1: Groovy
  2: Kotlin
Enter selection (default: Groovy) [1..2]

Select test framework:
  1: JUnit 4
  2: TestNG
  3: Spock
  4: JUnit Jupiter
```

Enter selection (default: JUnit 4) [1..4]

Project name (default: demo): org.hyperskill.gradleapp
Source package (default: org.hyperskill.gradleapp):

After completing the initialization, the project structure will be the following:

```
.
├── app
│   ├── build.gradle
│   └── src
│       ├── main
│       │   ├── java
│       │   │   └── org
│       │   │       └── hyperskill
│       │   │           └── gradleapp
│       │   │               └── App.java
│       │   └── resources
│       └── test
│           ├── java
│           │   └── org
│           │       └── hyperskill
│           │           └── gradleapp
│           │               └── AppTest.java
│           └── resources
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── gradlew
├── gradlew.bat
└── settings.gradle
```

There is a folder app that exists because you've chosen application as the type of project, and the folder represents our application.

There is also the src directory inside app. It contains two sub-directories main and test. This is quite a standard project structure when using Gradle. In our case, the package org.hyperskill.gradleapp has some Java source code (App.java).

If you chose Kotlin as the implementation language, the project structure would be the same except for Kotlin source code files (.kt instead of .java) and kotlin folders instead of java ones.

Please note it is a good practice for Java and Kotlin projects to include the

name of your organization in the path to your source code files as a package name like org.hyperskill. We follow this recommendation, too.

Running the application

If you look at the list of available tasks for managing the project using the command gradle tasks --all, you will see that the list is fairly long. Here is a shortened version of it:

```
Application tasks
-----------------
run - Runs this project as a JVM application

Build tasks
-----------
assemble - Assembles the outputs of this project.
build - Assembles and tests this project.
...
```

To start the application, you can use the run command of Gradle. To do it, invoke the gradle run command, or you can use a Gradle wrapper script for your OS. This command will build and run the application. Here is an output example:

```
> Task :app:run
Hello World!

BUILD SUCCESSFUL in 623ms
2 actionable tasks: 1 executed, 1 up-to-date
```

As you can see, the autogenerated application can already display a welcome string. If you get a similar result, it means that everything is OK: your application works, and Gradle can manage it!

If you look at the project structure again, you will see that it has some new files, including files with bytecode (App.class, AppTest.class). Actually, Gradle built and started the App.class file when we invoked the run command.

Building the application

If you would like to generate a bundle of the application with all its dependencies and a script to start the application, use the gradle build command.

```
BUILD SUCCESSFUL in 1s
7 actionable tasks: 7 executed
```

If everything is OK, Gradle will have produced the archive in two formats for you: app/build/distributions/app.tar and app/build/distributions/app.zip. Now, you can distribute your application!