BUILD ENVIRONMENT

What is a build environment?

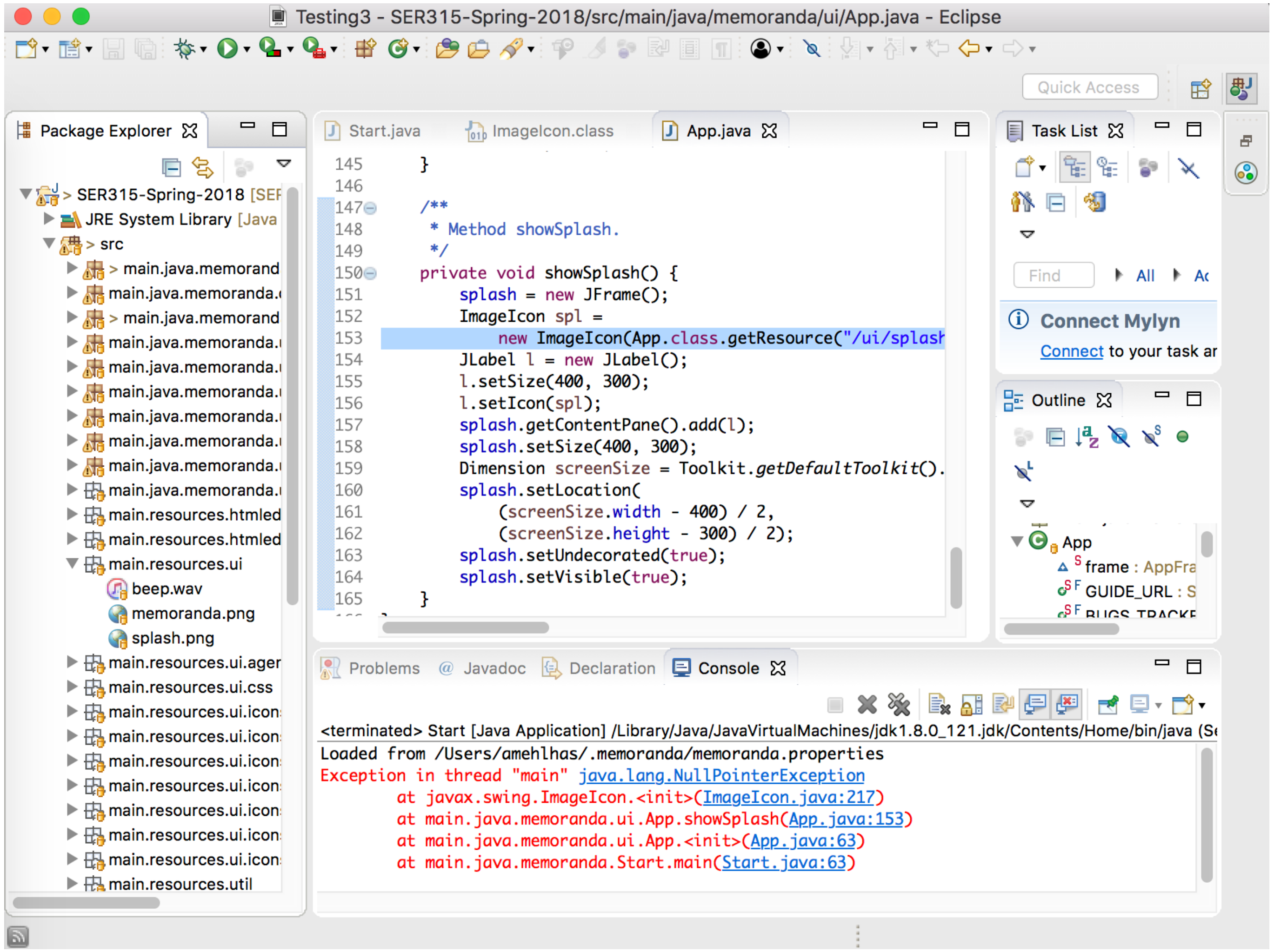
Why do we need it?

Examples?

Gradle

WHAT IS A BUILD ENVIRONMENT

PROBLEM

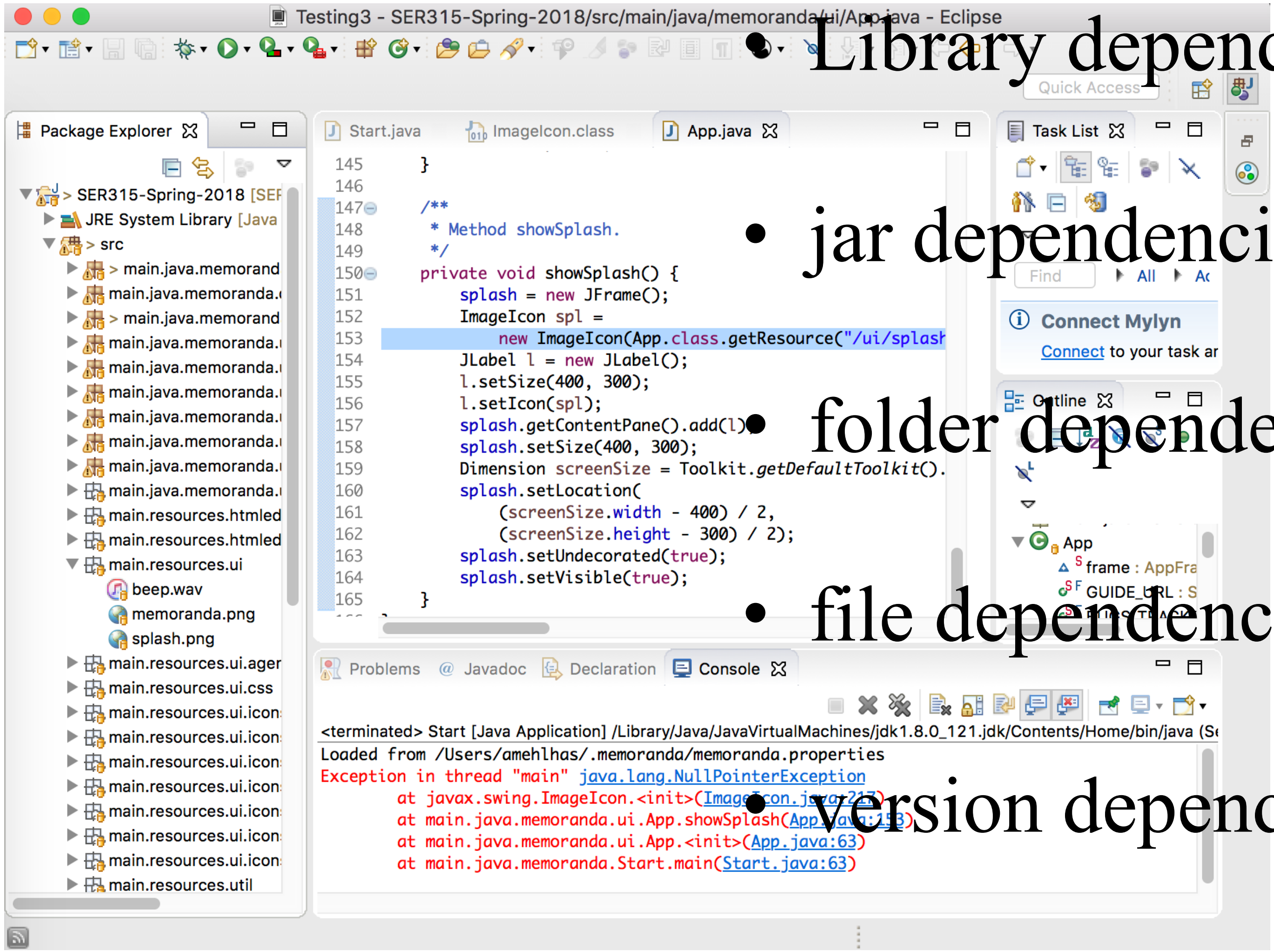


Testing3 - SER315-Spring-2018/src/main/java/memoranda/ui/App.java - Eclipse

```
    }

    /**
     * Method showSplash.
     */
    private void showSplash() {
        splash = new JFrame();
        ImageIcon spl =
            new ImageIcon(App.class.getResource("/ui/splash
        JLabel l = new JLabel();
        l.setSize(400, 300);
        l.setIcon(spl);
        splash.getContentPane().add(l);
        splash.setSize(400, 300);
        Dimension screenSize = Toolkit.getDefaultToolkit().
        splash.setLocation(
            (screenSize.width - 400) / 2,
            (screenSize.height - 300) / 2);
        splash.setUndecorated(true);
        splash.setVisible(true);
    }
```

```
<terminated> Start [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java (S
Loaded from /Users/amehlhas/.memoranda/memoranda.properties
Exception in thread "main" java.lang.NullPointerException
	at javax.swing.ImageIcon.<init>(ImageIcon.java:217)
	at main.java.memoranda.ui.App.showSplash(App.java:153)
	at main.java.memoranda.ui.App.<init>(App.java:63)
	at main.java.memoranda.Start.main(Start.java:63)
```

PROBLEM



```java
    }

    /**
     * Method showSplash.
     */
    private void showSplash() {
        splash = new JFrame();
        ImageIcon spl =
            new ImageIcon(App.class.getResource("/ui/splash
        JLabel l = new JLabel();
        l.setSize(400, 300);
        l.setIcon(spl);
        splash.getContentPane().add(l)
        splash.setSize(400, 300);
        Dimension screenSize = Toolkit.getDefaultToolkit().
        splash.setLocation(
            (screenSize.width - 400) / 2,
            (screenSize.height - 300) / 2);
        splash.setUndecorated(true);
        splash.setVisible(true);
    }
```

```
<terminated> Start [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java (S
Loaded from /Users/amehlhas/.memoranda/memoranda.properties
Exception in thread "main" java.lang.NullPointerException
	at javax.swing.ImageIcon.<init>(ImageIcon.java:217)
	at main.java.memoranda.ui.App.showSplash(App.java:153)
	at main.java.memoranda.ui.App.<init>(App.java:63)
	at main.java.memoranda.Start.main(Start.java:63)
```

- Library dependencies
- jar dependencies
- folder dependencies
- file dependencies
- version dependencies
- …

BUILD TOOLS

Allow to

- define dependencies for a build
- define different builds
- define what to test
- define specific versions

EXAMPLES

BUILD TOOLS

<APACHE ANT>

Maven™

Gradle

ANT

ANT (Another Neat Tool)

- **Java library** to build Java applications
- Similar to Make
- **XML** based
- Build file is **build.xml**
- Phases in build are called **targets**

➔ **Very flexible, very powerful, often hard to read**

```xml
<project>
    <target name="clean">
        <delete dir="classes" />
    </target>

    <target name="compile" depends="clean">
        <mkdir dir="classes" />
        <javac srcdir="src" destdir="classes" />
    </target>

    <target name="jar" depends="compile">
        <mkdir dir="jar" />
        <jar destfile="jar/HelloWorld.jar" basedir="classes">
            <manifest>
                <attribute name="Main-Class"
                  value="antExample.HelloWorld" />
            </manifest>
        </jar>
    </target>

    <target name="run" depends="jar">
        <java jar="jar/HelloWorld.jar" fork="true" />
    </target>
</project>
```

```
amehlhas ~ $ ant clean
```

```
amehlhas ~ $ ant compile
```

```
amehlhas ~ $ ant jar
```

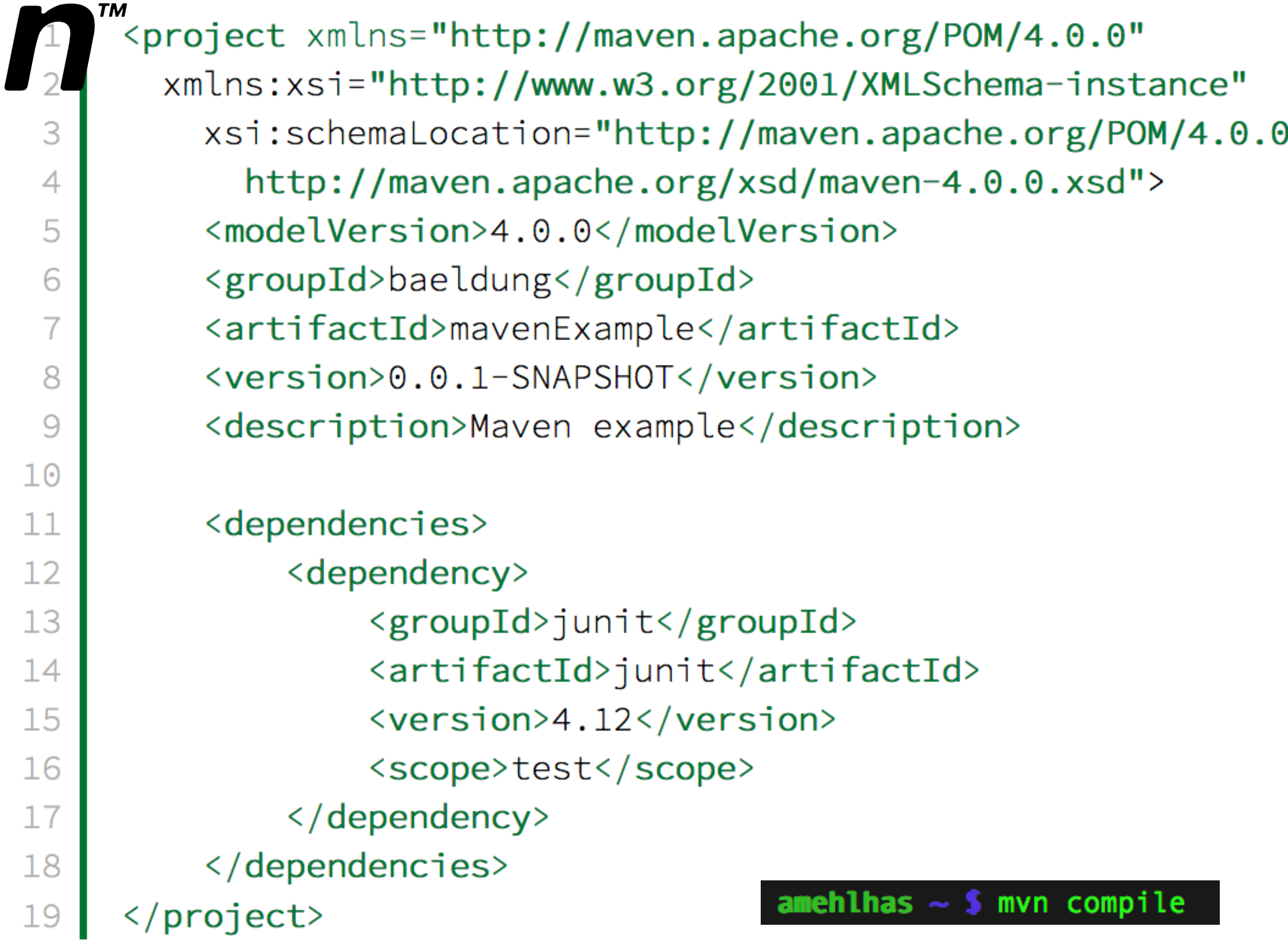
```
amehlhas ~ $ ant run
```

Code from: https://www.baeldung.com/ant-maven-gradle

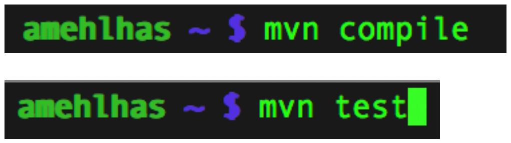
MAVEN

- Primarily for **Java applications**
- **XML** based
- Build file is **pom.xml**
- Relies on **conventions** and predefines commands – less flexible than Ant
- Extensible through Plug-ins

➔ **Not as flexible, not always easy to read, easy to setup**



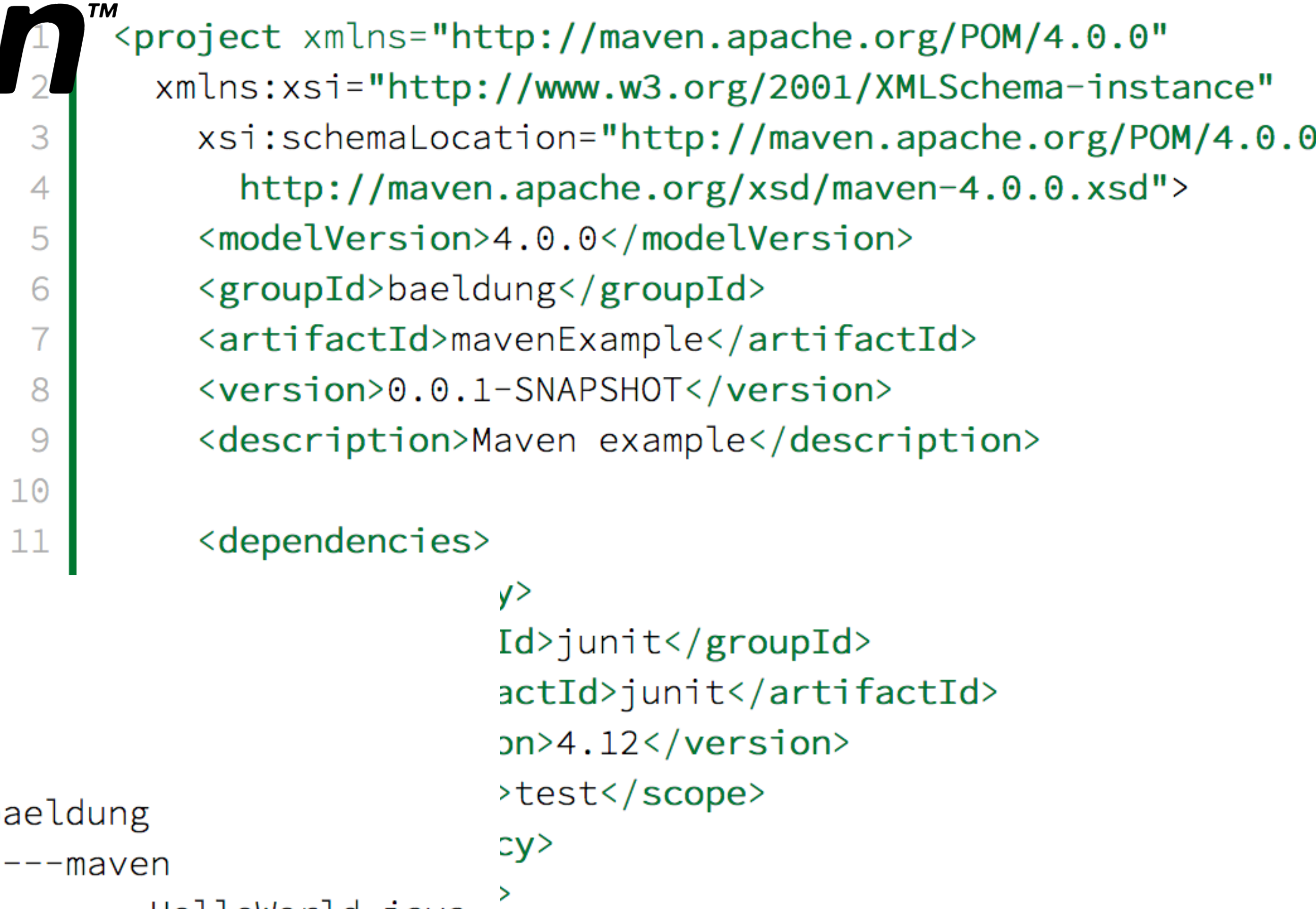
```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
      http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>baeldung</groupId>
    <artifactId>mavenExample</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <description>Maven example</description>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```



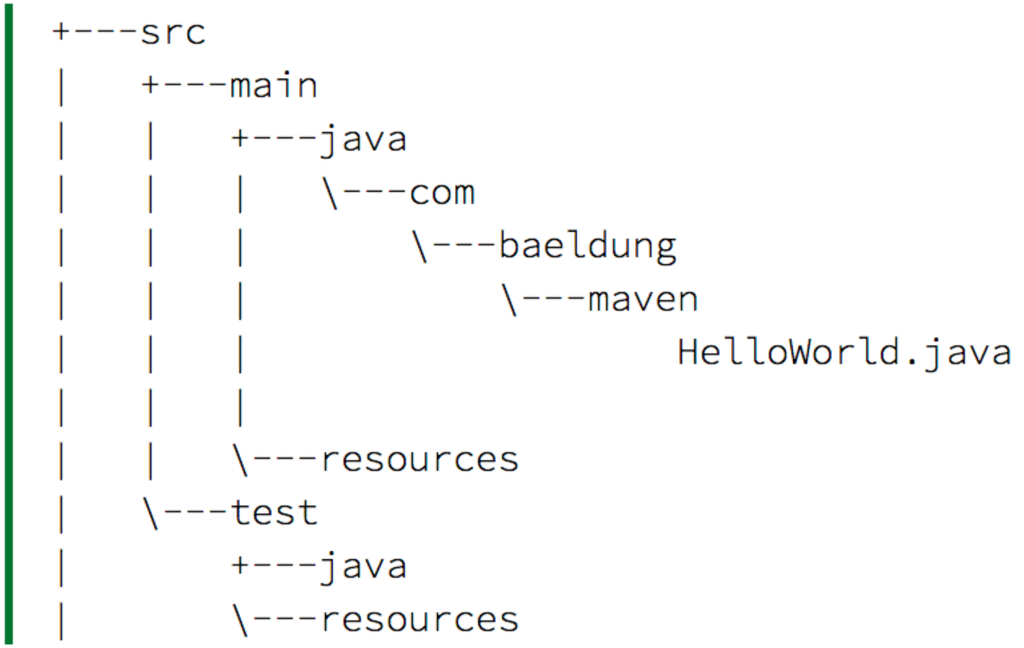
```
amehlhas ~ $ mvn compile
```

```
amehlhas ~ $ mvn test
```

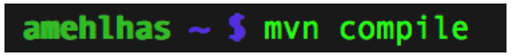
Code from:
https://www.baeldung.com/ant-maven-gradle



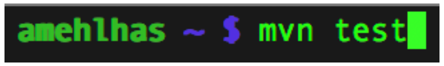
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
      http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>baeldung</groupId>
    <artifactId>mavenExample</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <description>Maven example</description>

    <dependencies>
                 y>
                 Id>junit</groupId>
                 actId>junit</artifactId>
                 on>4.12</version>
                 >test</scope>
                 cy>
                 >
```



```
+---src
|   +---main
|   |   +---java
|   |   |   \---com
|   |   |       \---baeldung
|   |   |           \---maven
|   |   |                   HelloWorld.java
|   |   |
|   |   \---resources
|   \---test
|       +---java
|       \---resources
```



```
amehlhas ~ $ mvn compile
```



```
amehlhas ~ $ mvn test
```

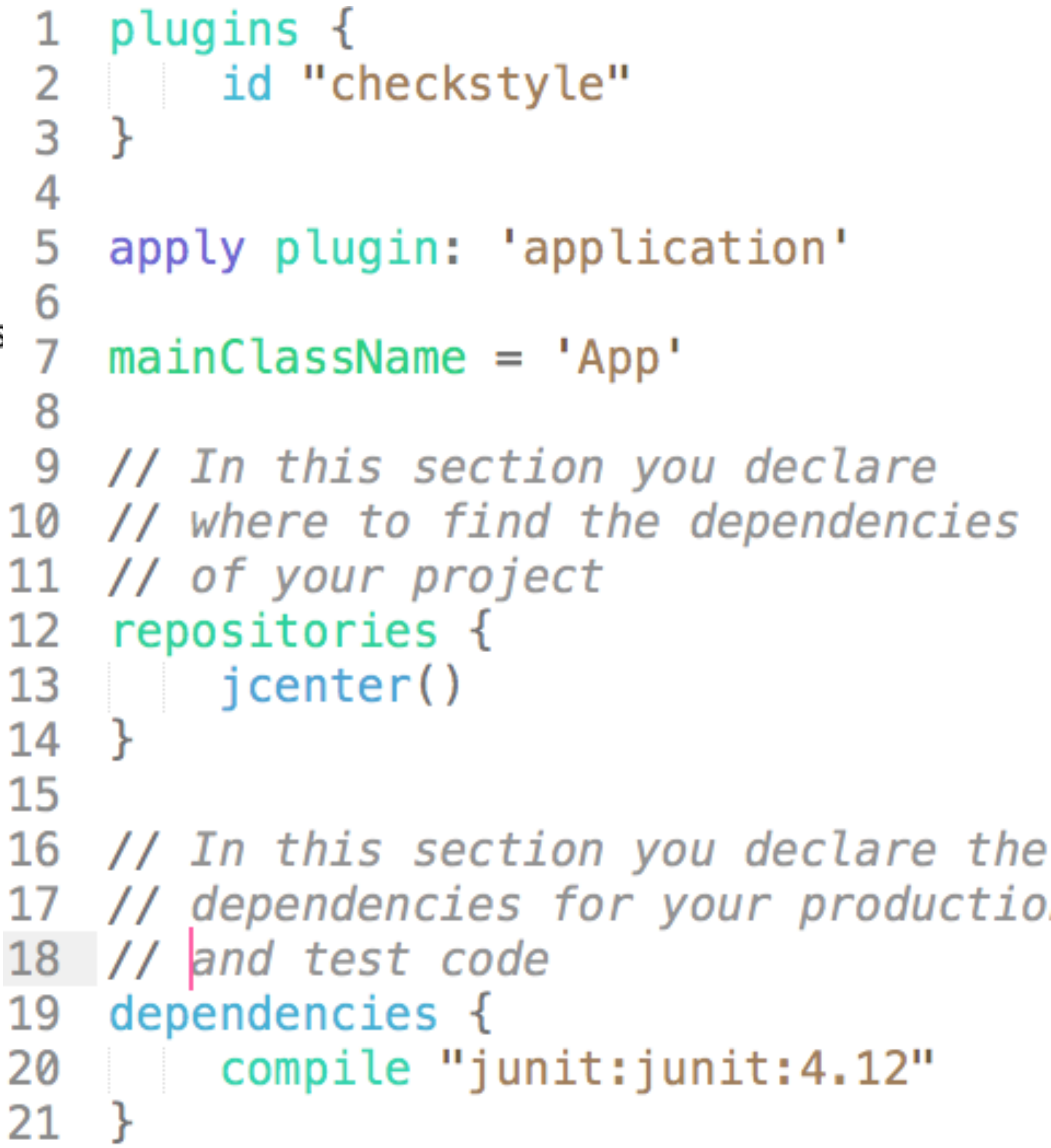
Code from:
https://www.baeldung.com/ant-maven-gradle

GRADLE

Gradle

- Build on Ant and Maven concept
- DSL based on **Groovy** or Kotlin
- **Small configuration** files
- Build file is **build.gradle**
- Uses **tasks**
- Extensible though Plug-ins

```
├── build . gradle
├── gradle
│     └── wrapper
│           ├── gradle-wrapper . jar
│           └── gradle-wrapper . propertie
├── gradlew
├── gradlew . bat
├── settings . gradle
└── src
     ├── main
     │    ├── java
     │    │    └── App.java
     │    └── resources
     └── test
          ├── java
          │    └── demo
          │         └── AppTest.java
          └── resources
```



```
plugins {
    id "checkstyle"
}

apply plugin: 'application'

mainClassName = 'App'

// In this section you declare
// where to find the dependencies
// of your project
repositories {
    jcenter()
}

// In this section you declare the
// dependencies for your productio
// and test code
dependencies {
    compile "junit:junit:4.12"
}
```

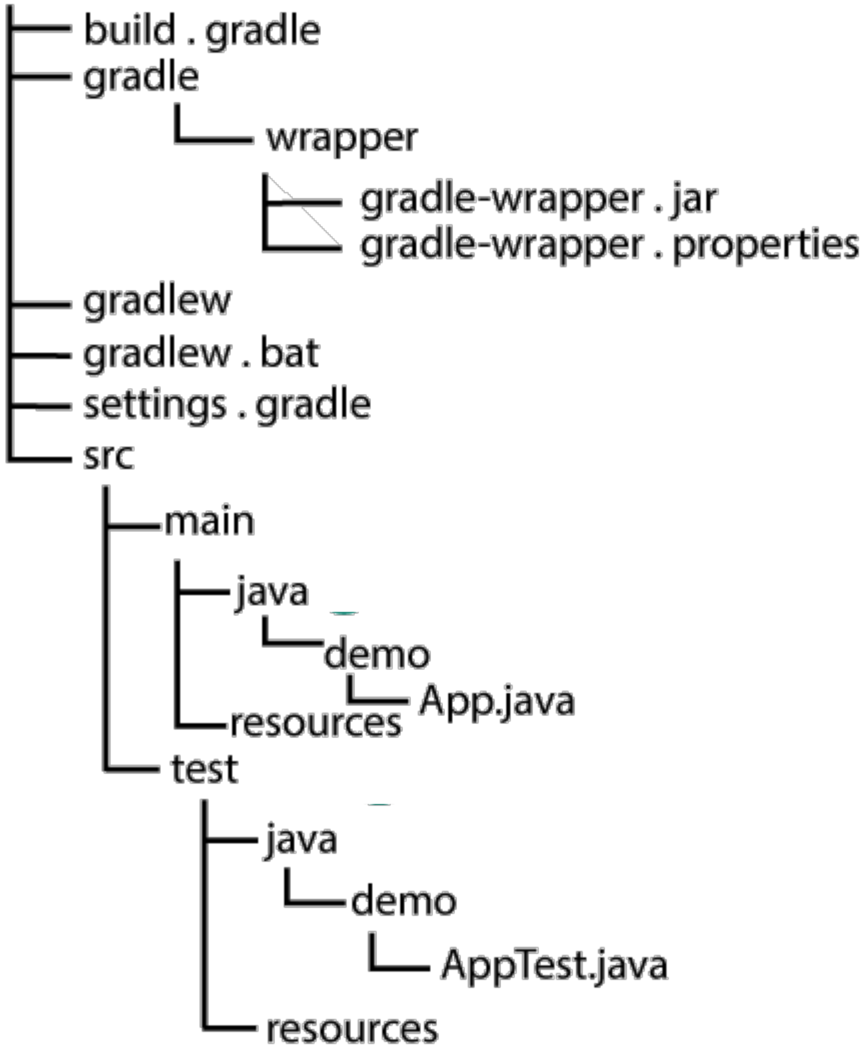
SUMMARY

- Build environments are to compile/build your application
- Ant is most flexible but also ‘most complicated’ (might be subjective)
- Gradle is ‘easiest’ due to DSL (might be subjective)

Gradle

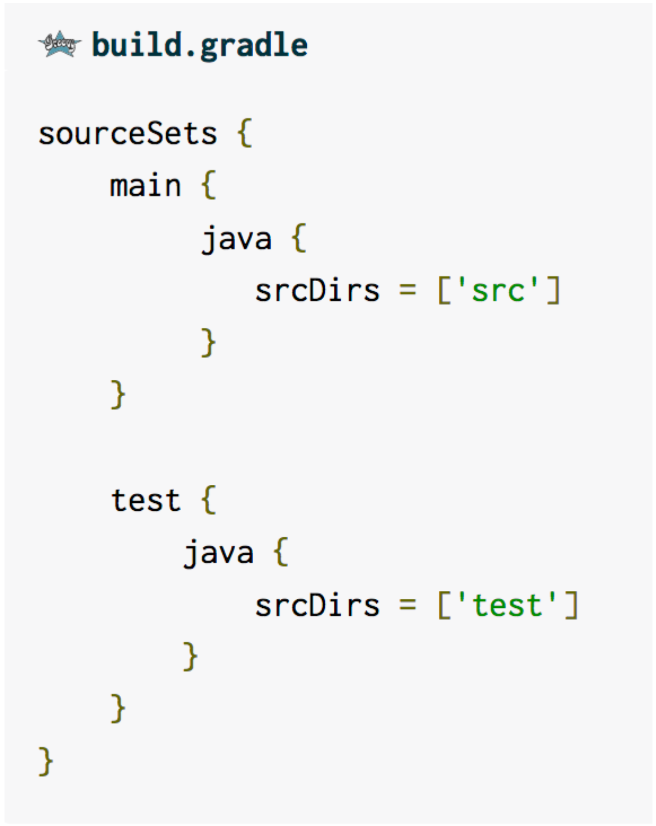
Examples used from:
https://docs.gradle.org/current/userguide/building_java_projects.html

PROJECT STRUCTURE



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

Change the structure:

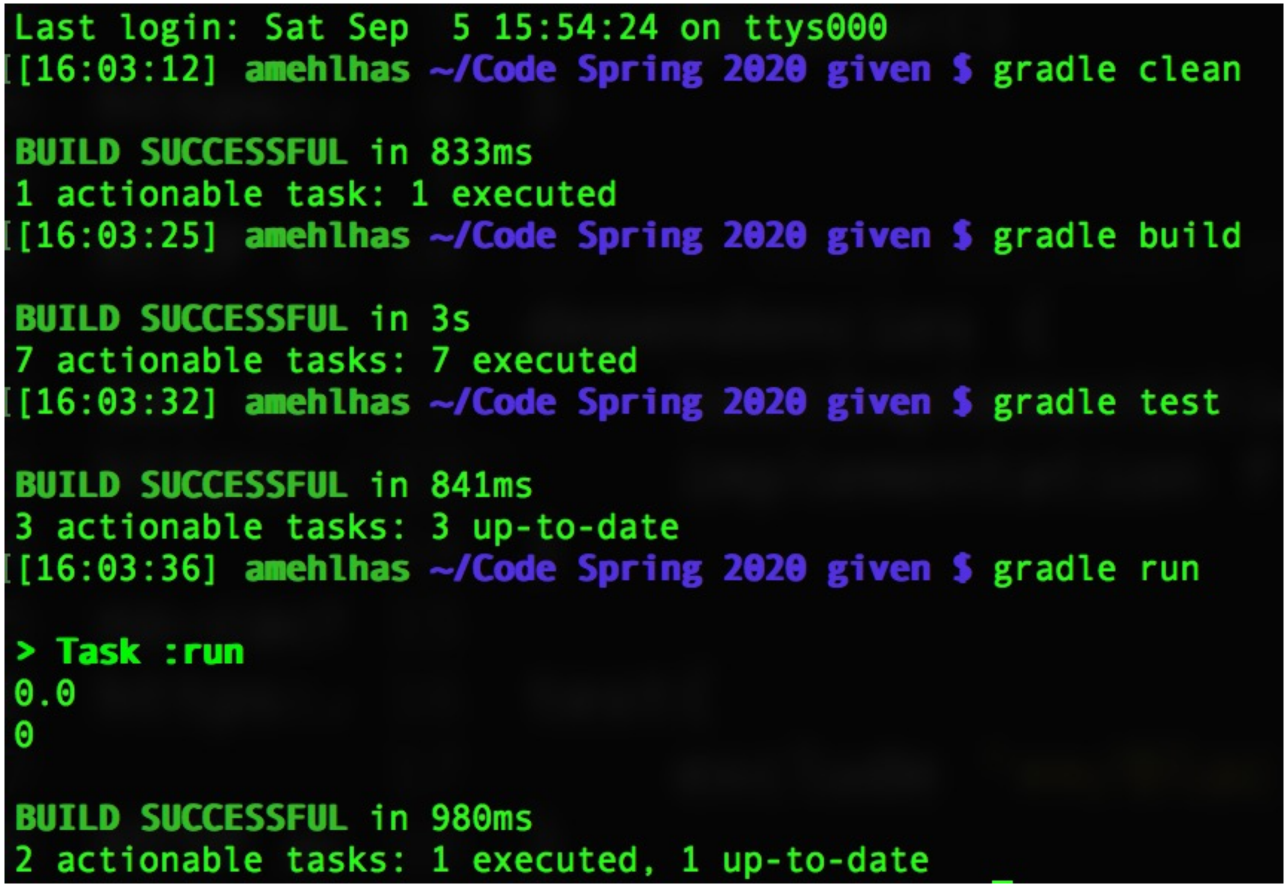


build.gradle

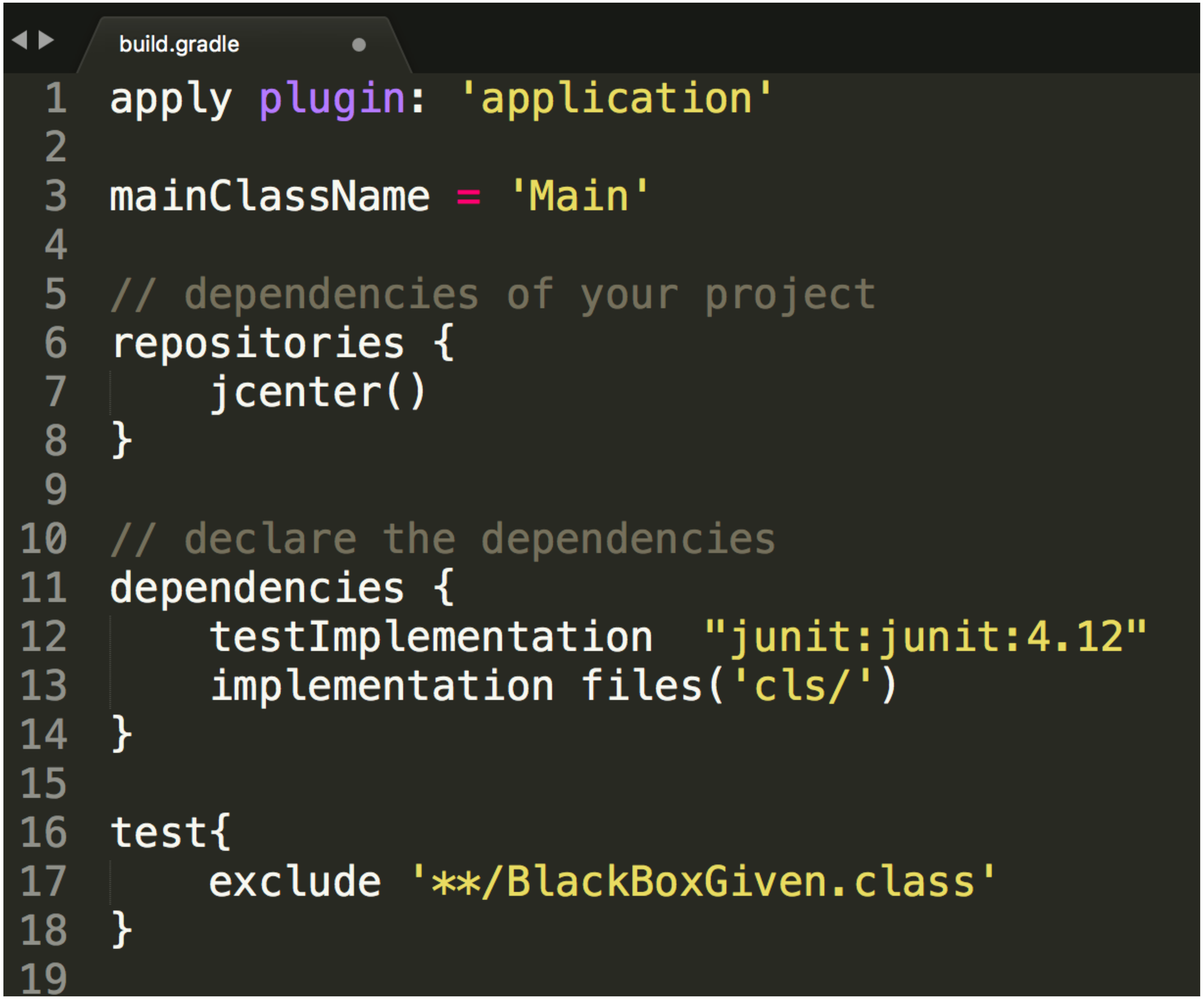
```
sourceSets {
    main {
        java {
            srcDirs = ['src']
        }
    }

    test {
        java {
            srcDirs = ['test']
        }
    }
}
```

MAIN TASKS

- clean
- build
- test
- run

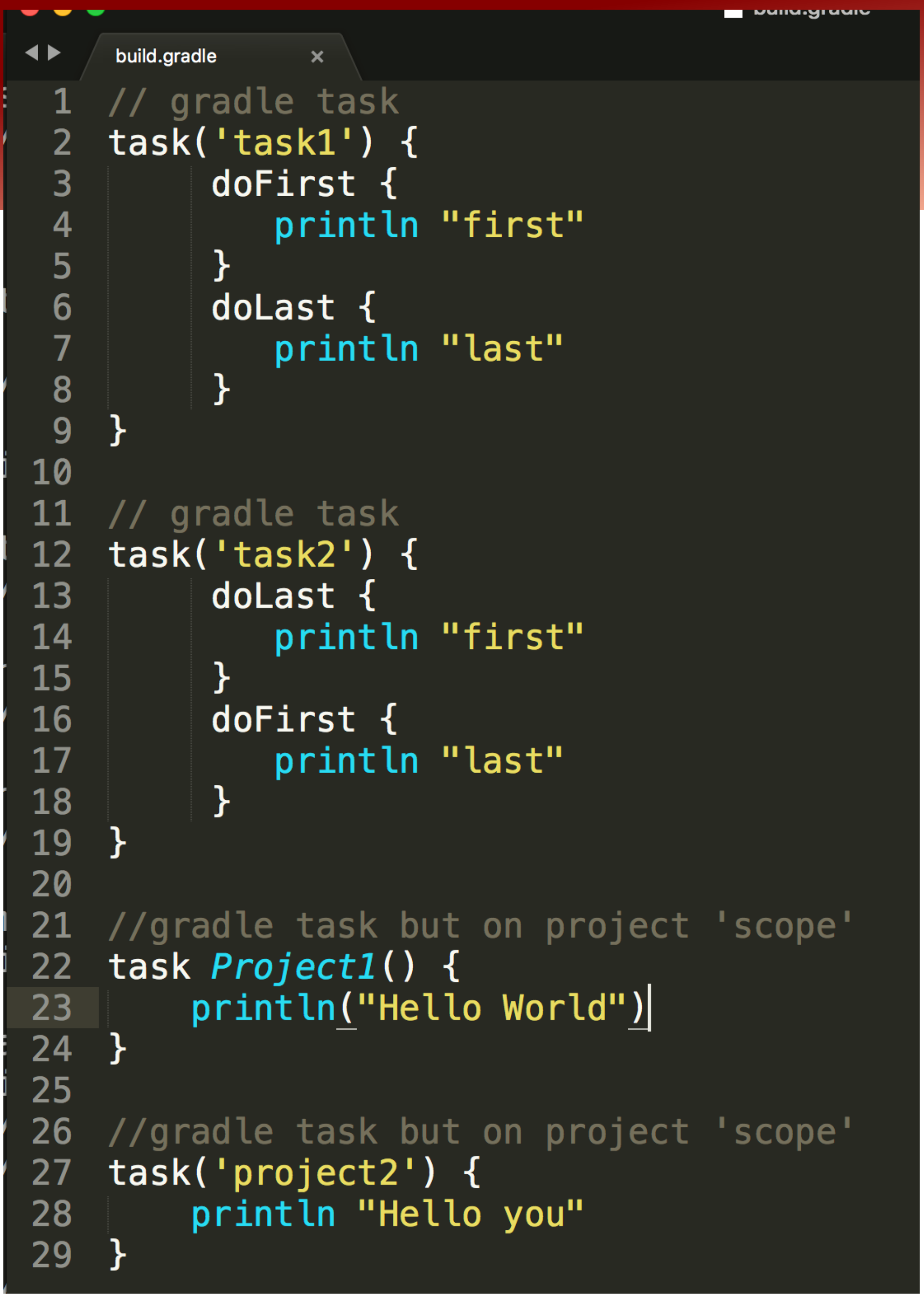


```
Last login: Sat Sep  5 15:54:24 on ttys000
[16:03:12] amehlhas ~/Code Spring 2020 given $ gradle clean

BUILD SUCCESSFUL in 833ms
1 actionable task: 1 executed
[16:03:25] amehlhas ~/Code Spring 2020 given $ gradle build

BUILD SUCCESSFUL in 3s
7 actionable tasks: 7 executed
[16:03:32] amehlhas ~/Code Spring 2020 given $ gradle test

BUILD SUCCESSFUL in 841ms
3 actionable tasks: 3 up-to-date
[16:03:36] amehlhas ~/Code Spring 2020 given $ gradle run

> Task :run
0.0
0

BUILD SUCCESSFUL in 980ms
2 actionable tasks: 1 executed, 1 up-to-date
```



```
apply plugin: 'application'

mainClassName = 'Main'

// dependencies of your project
repositories {
    jcenter()
}

// declare the dependencies
dependencies {
    testImplementation  "junit:junit:4.12"
    implementation files('cls/')
}

test{
    exclude '**/BlackBoxGiven.class'
}

```

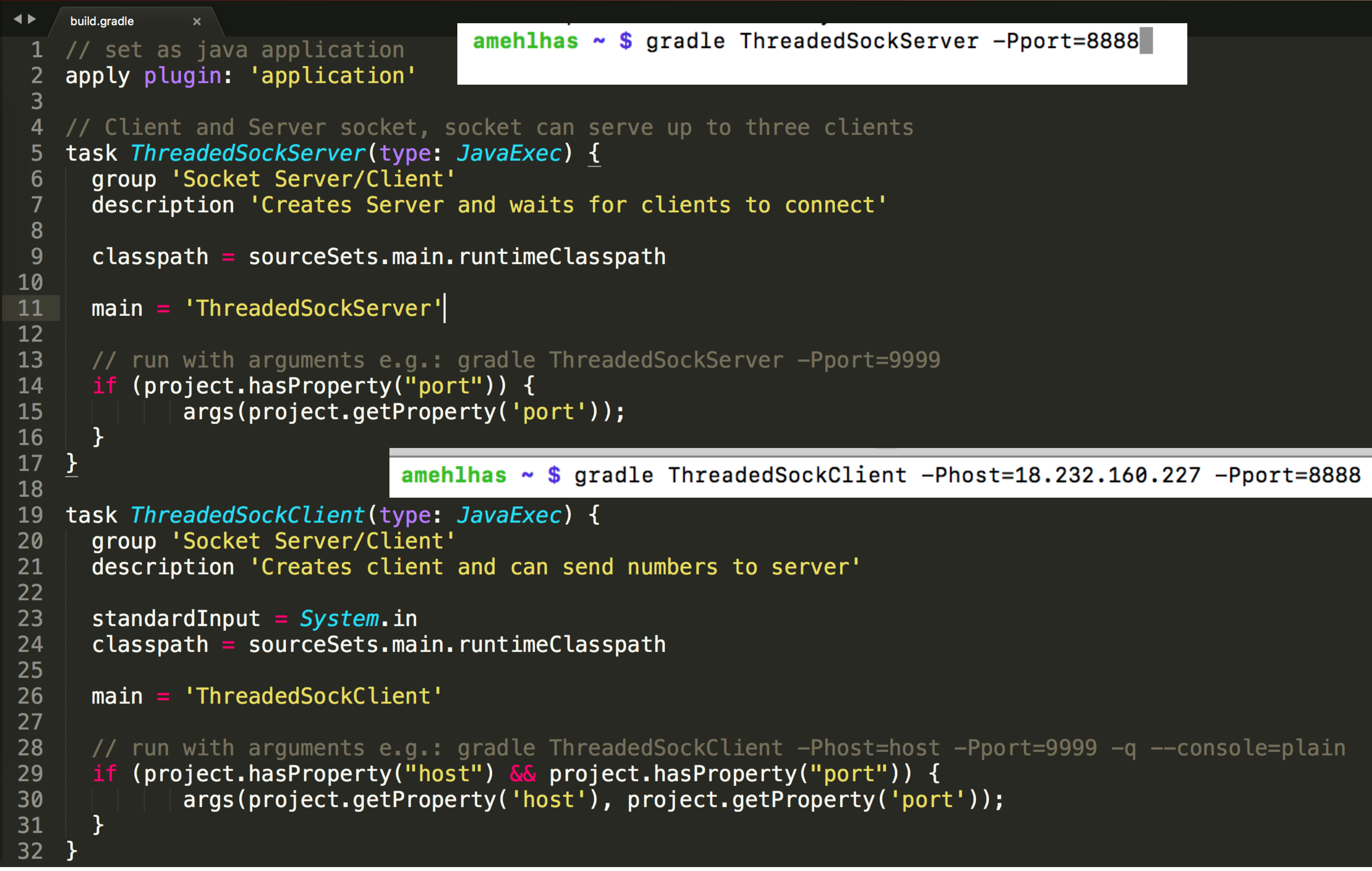
CUSTOM TASKS


build.gradle

```
// gradle task
task('task1') {
        doFirst {
            println "first"
        }
        doLast {
            println "last"
        }
}

// gradle task
task('task2') {
        doLast {
            println "first"
        }
        doFirst {
            println "last"
        }
}

//gradle task but on project 'scope'
task Project1() {
    println("Hello World")
}

//gradle task but on project 'scope'
task('project2') {
    println "Hello you"
}
```

- https://docs.gradle.org/current/userguide/tutorial_using_tasks.html#sec:projects_and_tasks

CUSTOM JAVA TASKS

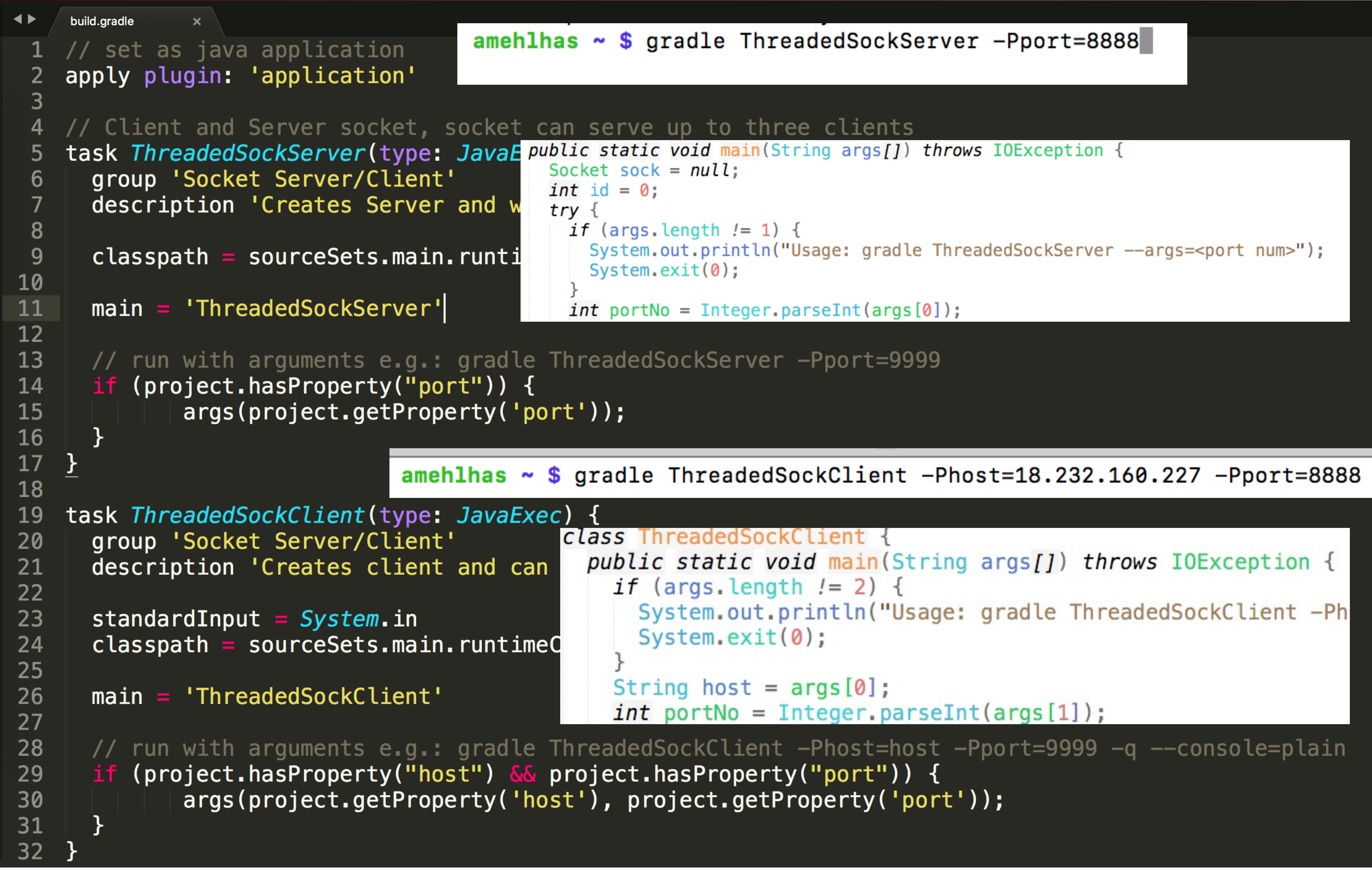


```
amehlhas ~ $ gradle ThreadedSockServer -Pport=8888
```

```
// set as java application
apply plugin: 'application'

// Client and Server socket, socket can serve up to three clients
task ThreadedSockServer(type: JavaExec) {
  group 'Socket Server/Client'
  description 'Creates Server and waits for clients to connect'

  classpath = sourceSets.main.runtimeClasspath

  main = 'ThreadedSockServer'

  // run with arguments e.g.: gradle ThreadedSockServer -Pport=9999
  if (project.hasProperty("port")) {
      args(project.getProperty('port'));
  }
}

task ThreadedSockClient(type: JavaExec) {
  group 'Socket Server/Client'
  description 'Creates client and can send numbers to server'

  standardInput = System.in
  classpath = sourceSets.main.runtimeClasspath

  main = 'ThreadedSockClient'

  // run with arguments e.g.: gradle ThreadedSockClient -Phost=host -Pport=9999 -q --console=plain
  if (project.hasProperty("host") && project.hasProperty("port")) {
      args(project.getProperty('host'), project.getProperty('port'));
  }
}
```

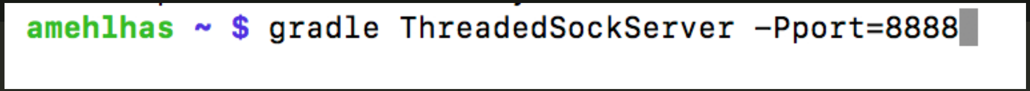
```
amehlhas ~ $ gradle ThreadedSockClient -Phost=18.232.160.227 -Pport=8888
```

CUSTOM JAVA TASKS

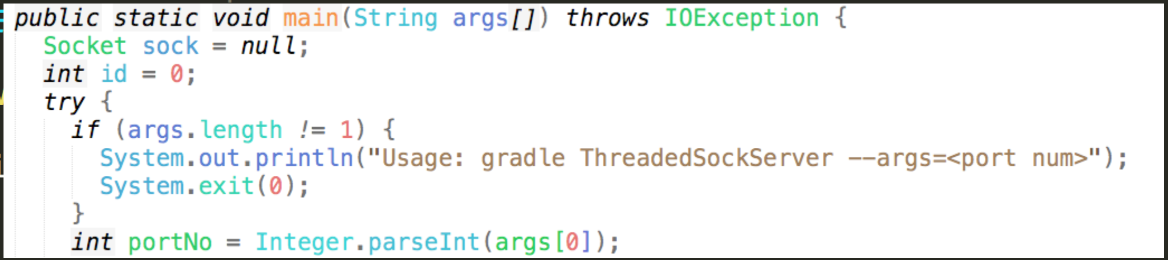


build.gradle

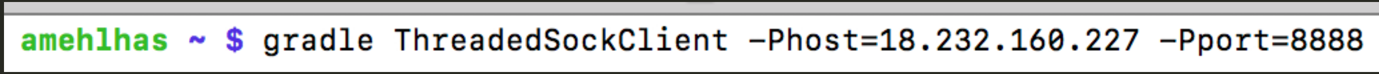
```
// set as java application
apply plugin: 'application'

// Client and Server socket, socket can serve up to three clients
task ThreadedSockServer(type: JavaE
  group 'Socket Server/Client'
  description 'Creates Server and w

  classpath = sourceSets.main.runti

  main = 'ThreadedSockServer'

  // run with arguments e.g.: gradle ThreadedSockServer -Pport=9999
  if (project.hasProperty("port")) {
        args(project.getProperty('port'));
  }
}

task ThreadedSockClient(type: JavaExec) {
  group 'Socket Server/Client'
  description 'Creates client and can

  standardInput = System.in
  classpath = sourceSets.main.runtimeC

  main = 'ThreadedSockClient'

  // run with arguments e.g.: gradle ThreadedSockClient -Phost=host -Pport=9999 -q --console=plain
  if (project.hasProperty("host") && project.hasProperty("port")) {
        args(project.getProperty('host'), project.getProperty('port'));
  }
}
```



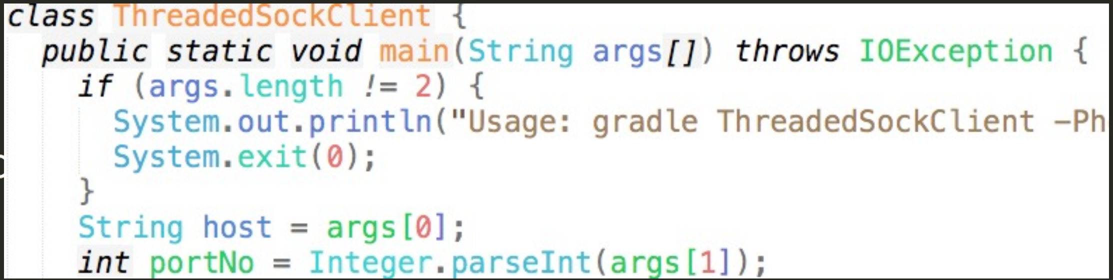
```
amehlhas ~ $ gradle ThreadedSockServer -Pport=8888
```



```
public static void main(String args[]) throws IOException {
  Socket sock = null;
  int id = 0;
  try {
    if (args.length != 1) {
      System.out.println("Usage: gradle ThreadedSockServer --args=<port num>");
      System.exit(0);
    }
    int portNo = Integer.parseInt(args[0]);
```



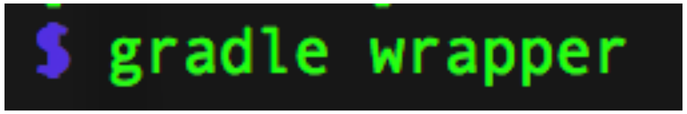
```
amehlhas ~ $ gradle ThreadedSockClient -Phost=18.232.160.227 -Pport=8888
```



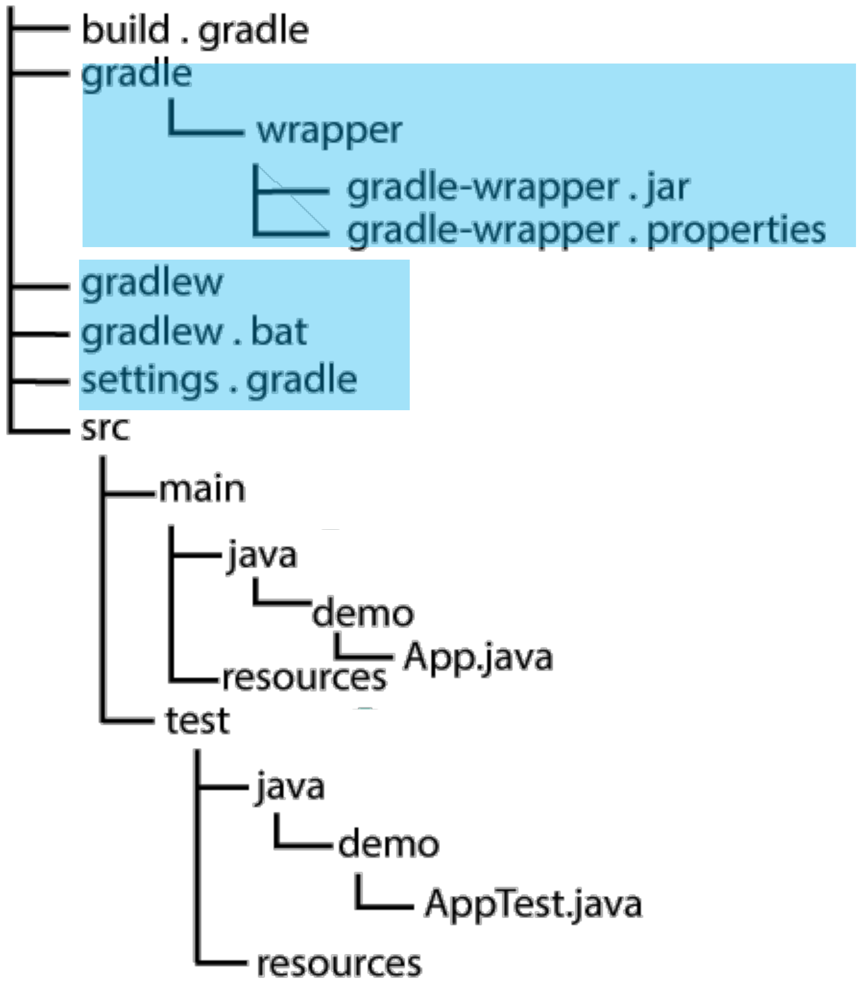
```
class ThreadedSockClient {
  public static void main(String args[]) throws IOException {
    if (args.length != 2) {
      System.out.println("Usage: gradle ThreadedSockClient -Ph
      System.exit(0);
    }
    String host = args[0];
    int portNo = Integer.parseInt(args[1]);
```

GRADLE WRAPPER

Creates

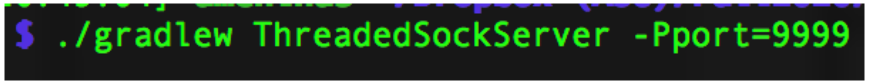


```
$ gradle wrapper
```



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

- Specifies specific Gradle version
- Downloads Gradle version if needed



```
$ ./gradlew ThreadedSockServer -Pport=9999
```

GRADLE SUMMARY

- Custom project structure
- Can create custom tasks with inputs
- Can create wrapper to specify specific Gradle/Java version
- And much much more