

Build

Compile

Run Tests

Package Jar War

Deploy and Run

Ant

build.xml

Tasks

Customize

Ivy

Convention Over Configuration

Gradle
DSL
Groovy or Kotlin
build.gradle



Maven

MyProject

src/main/java

src/main/resources

src/test/java

src/test/resources

In-Built pom.xml

Ant is project to project differs when migrate we have to learn a lot then implement

Maven is great on the other hand but customization is that easy

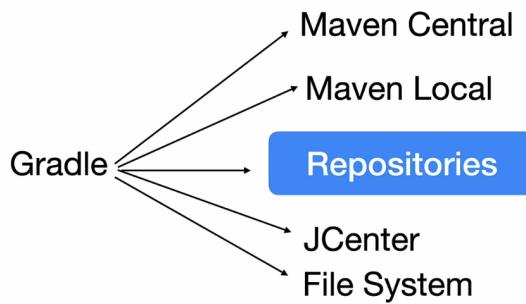
Gradle covers both disadvantages.

Why

Java Swift

C++ Groovy

Kotlin



Gradle pulls dependencies from these remote repositories and cache them locally.

It provides custom scope. Whereas maven provides fixed scope

Incremental Builds

Build Cache

Daemon

Due to the above points, gradle is fast. It compiles only recently changed files.

Plugin

- jar
- war
- jacoco
- bootRun

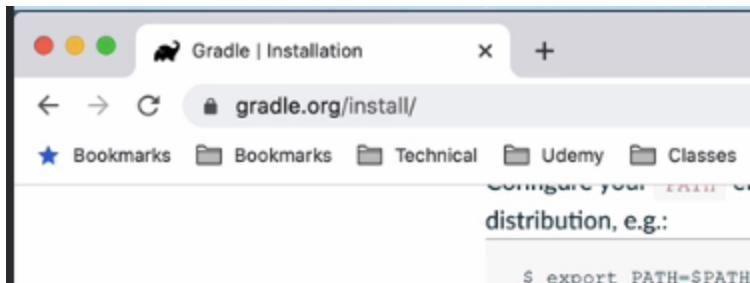
DSL

build.gradle groovy/kotlin

CLI

IDE

Install Gradle: the below site give all possible options (manual/automated/windows/mac/ linux)



```
Terminal Shell Edit View Window Help firstJavaProject -- bash -- 90x24
bharaths-MacBook-Pro:firstJavaProject bharaththippireddy$ gradle -v
-----
Gradle 6.5.1
-----
Build time: 2020-06-30 06:32:47 UTC
Revision: 66bc713f7169626a7f0134bf452abde51550ea0a

Kotlin: 1.3.72
Groovy: 2.5.11
Ant: Apache Ant(TM) version 1.10.7 compiled on September 1 2019
JVM: 14.0.1 (Oracle Corporation 14.0.1+14)
OS: Mac OS X 10.15.6 x86_64
bharaths-MacBook-Pro:firstJavaProject bharaththippireddy$
```

How to create a build.gradle file.

```
firstproject -- bash -- 90x24
bharaths-MacBook-Pro:gradle bharaththippireddy$ pwd
/Users/bharaththippireddy/Documents/gradle
bharaths-MacBook-Pro:gradle bharaththippireddy$ cd firstproject/
bharaths-MacBook-Pro:firstproject bharaththippireddy$ gradle init
```

It will ask few questions

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

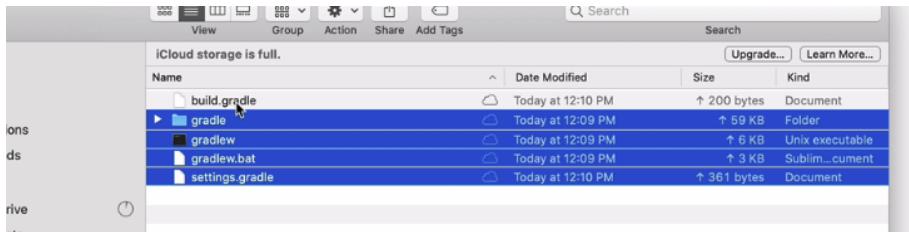
Select build script DSL:
1: Groovy
2: Kotlin
Enter selection (default: Groovy) [1..2] 1

Project name (default: firstproject):

> Task :init
Get more help with your project: https://guides.gradle.org/creating-new-gradle-builds

BUILD SUCCESSFUL in 53s
2 actionable tasks: 2 executed
bharaths-MacBook-Pro:firstproject bharaththippireddy$
```

It creates lot of files

A screenshot of a Sublime Text editor window titled 'build.gradle'. The menu bar includes 'File', 'Edit', 'Selection', 'Find', 'View', 'Goto', 'Tools', 'Project', 'Window', and 'Help'. The status bar at the bottom right shows 'Thu 12:11 PM'. The code in the editor is:

```
1 /*
2  * This file was generated by the Gradle 'init' task,
3  *
4  * This is a general purpose Gradle
5  * Learn how to create Gradle builds at https://guides.gradle.org/creating-new-gradle-builds
6  */
7
8 task firstTask{
9     println 'Gradle Rocks!!'
```

Create a task as follow

A screenshot of a Sublime Text editor window titled 'build.gradle'. The code now includes a new 'task firstTask' block:

```
1 /*
2  * This file was generated by the G
3  *
4  * This is a general purpose Gradle
5  * Learn how to create Gradle builds
6  */
7 task firstTask{
8     println 'Gradle Rocks!!'
9 }
```

Run the task as follow

```
eddy$ gradle tasks --all
```

A screenshot of a terminal window. The title bar says 'Other tasks'. The main area shows the command 'gradle tasks --all' being run, followed by the output:

```
-----  
firstTask  
prepareKotlinBuildScriptModel  
  
BUILD SUCCESSFUL : - 100ms
```

Or run as follow

```
terminal Shell Edit View Window Help
firstproject — bash — 90x24
bharaths-MacBook-Pro:firstproject bharaththippiredy$ gradle firstTask

> Configure project :
Gradle Rocks!!

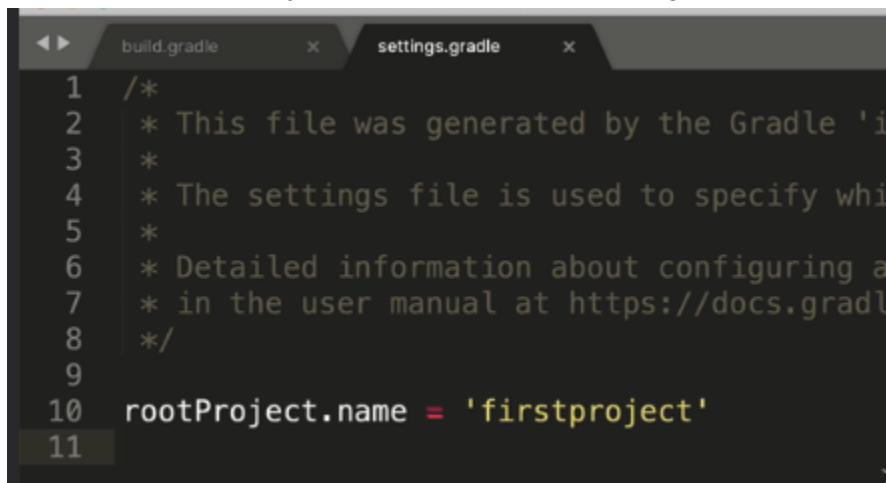
BUILD SUCCESSFUL in 601ms
bharaths-MacBook-Pro:firstproject bharaththippiredy$
```

Or Task Abbreviated form

```
bharaths-MacBook-Pro:firstproject bharaththippiredy$ gradle fT

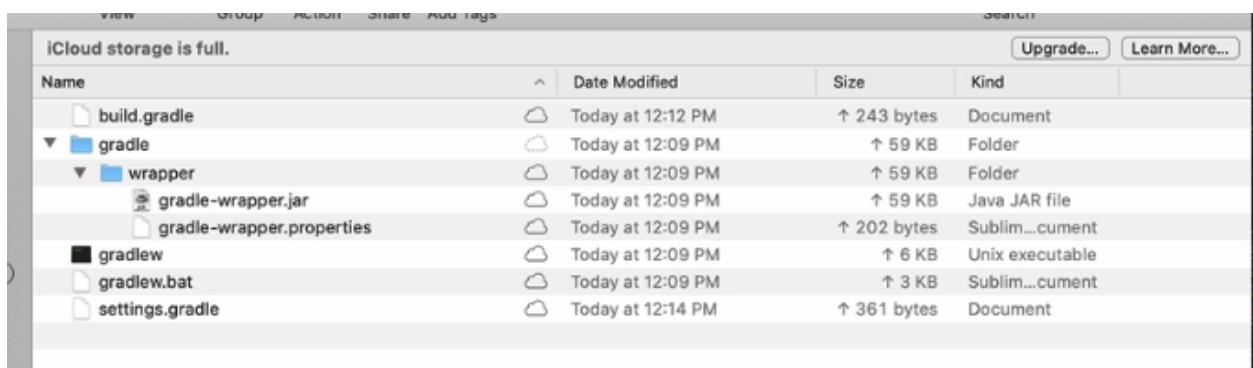
> Configure project :
Gradle Rocks!!
```

Where we find the project name. We can also change its value.

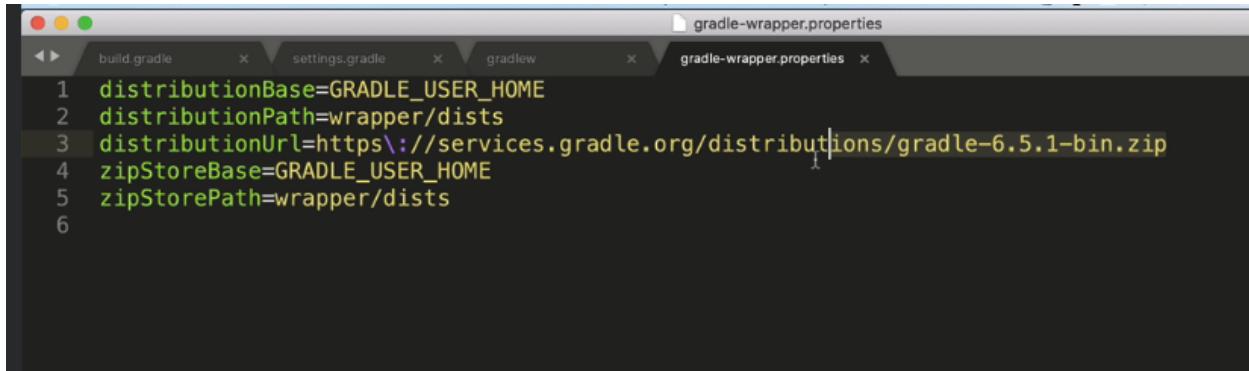


The screenshot shows a code editor with two tabs: 'build.gradle' and 'settings.gradle'. The 'build.gradle' tab contains the following code:

```
1 /**
2  * This file was generated by the Gradle 'init' plugin.
3  *
4  * The settings file is used to specify which
5  *
6  * Detailed information about configuring a
7  * in the user manual at https://docs.gradle.org/5.0/userguide/tutorial_gradle_getting_started.html#sec:hello_world_settings
8  */
9
10 rootProject.name = 'firstproject'
11
```



Gradlew unix executable and gradlew.bat windows executable



```
gradle-wrapper.properties
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-6.5.1-bin.zip
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
```

Here is the place where the gradle version of the project is maintained. In the PCF and servers whatever gradle version is available will be ignored and pull the project specific version using gradle-wrapper.jar file.

Similarly create Java Project

```
bharaththippireddy@bharaths-MacBook-Pro-2 gradle % cd firstJavaProject
bharaththippireddy@bharaths-MacBook-Pro-2 firstJavaProject % gradle init

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

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] 1

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] 1

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

Generate build using new APIs and behavior (some features may change in the next minor re

Select test framework:
 1: JUnit 4
 2: TestNG
 3: Spock
 4: JUnit Jupiter
Enter selection (default: JUnit Jupiter) [1..4] 1

Project name (default: firstJavaProject):
Source package (default: firstJavaProject): com.bharath.gradle
```



```

1 /*
2  * This file was generated by the Gradle
3  *
4  * The settings file is used to specify
5  *
6  * Detailed information about configuri
7  * in the user manual at https://docs.g
8  */
9
10 rootProject.name = 'firstJavaProject'
11 include('app')
12

```

We can change if we want another name like the following.

```

8 /*
9
10 rootProject.name = 'firstJavaProject'
11 include('firstJavaProject')
12

```

Then in that case we need rename folder "app" to "firstJavaProject".

The below command lists all the tasks.

```

firstJavaProject — docker@minikube: ~ - zsh — 89x26
@minikube: ~ — -zsh
~/localgitrepo — -zsh
ls-MacBook-Pro-2 firstJavaProject % gradle tasks --all

```

```
~/Documents/gradle/firstJavaProject -- docker@minikube: ~ - zsh
dependencyInsight - Displays the insight into a specific dependency 'firstJavaProject'.
firstJavaProject:dependencyInsight - Displays the insight into project ':firstJavaProject'.
help - Displays a help message.
firstJavaProject:help - Displays a help message.
javaToolchains - Displays the detected java toolchains.
firstJavaProject:javaToolchains - Displays the detected java outgoingVariants - Displays the outgoing variants of root project 'firstJavaProject'.
firstJavaProject:outgoingVariants - Displays the outgoing variants of root project 'firstJavaProject'.
projects - Displays the sub-projects of root project 'firstJavaProject'.
firstJavaProject:projects - Displays the sub-projects of properties - Displays the properties of root project 'firstJavaProject'.
firstJavaProject:properties - Displays the properties of tasks - Displays the tasks runnable from root project 'firstJavaProject' (runnable tasks may belong to subprojects).
firstJavaProject:tasks - Displays the tasks runnable from project 'firstJavaProject'.

Verification tasks
-----
firstJavaProject:check - Runs all checks.
firstJavaProject:test - Runs the test suite.

Other tasks
-----
```

All java tasks clean build and test are all as part of the plugin below.

```
7
8
9 plugins {
10   ... // Apply the application plugin to add support for building a CLI application
11   id 'application'
12 }
13
```

And also we have an application node where is the main method reside by running gradle run that will be executed.

```
20
21
22 application {
23   // Define the main class for the application.
24   mainClass = 'com.bharath.gradle.App'
25 }
26
27
28
29
30 }
```

```
bharaththippireddy@bharaths-MacBook-Pro-2 firstJavaProject % gradle compileJava  
  
BUILD SUCCESSFUL in 558ms  
1 actionable task: 1 executed  
bharaththippireddy@bharaths-MacBook-Pro-2 firstJavaProject % gradle clean  
  
BUILD SUCCESSFUL in 453ms  
1 actionable task: 1 executed  
bharaththippireddy@bharaths-MacBook-Pro-2 firstJavaProject % gradle test  
  
BUILD SUCCESSFUL in 1s  
3 actionable tasks: 3 executed  
bharaththippireddy@bharaths-MacBook-Pro-2 firstJavaProject %
```

```
bharaththippireddy@bharaths-MacBook-Pro-2 firstJavaProject % gradle jar  
  
BUILD SUCCESSFUL in 477ms  
2 actionable tasks: 1 executed, 1 up-to-date  
bharaththippireddy@bharaths-MacBook-Pro-2 firstJavaProject % gradle run  
  
> Task :firstJavaProject:run  
Hello World!  
  
BUILD SUCCESSFUL in 515ms  
2 actionable tasks: 1 executed, 1 up-to-date  
bharaththippireddy@bharaths-MacBook-Pro-2 firstJavaProject %
```

```
13  
14 repositories {  
15     // Use Maven Central for resolving dependencies.  
16     mavenCentral()  
17 }  
18  
19 dependencies {  
20     // Use JUnit test framework.  
21     testImplementation 'junit:junit:4.13.2'  
22  
23     // This dependency is used by the application.  
24     implementation 'com.google.guava:guava:30.1.1-jre'  
25 }  
26
```

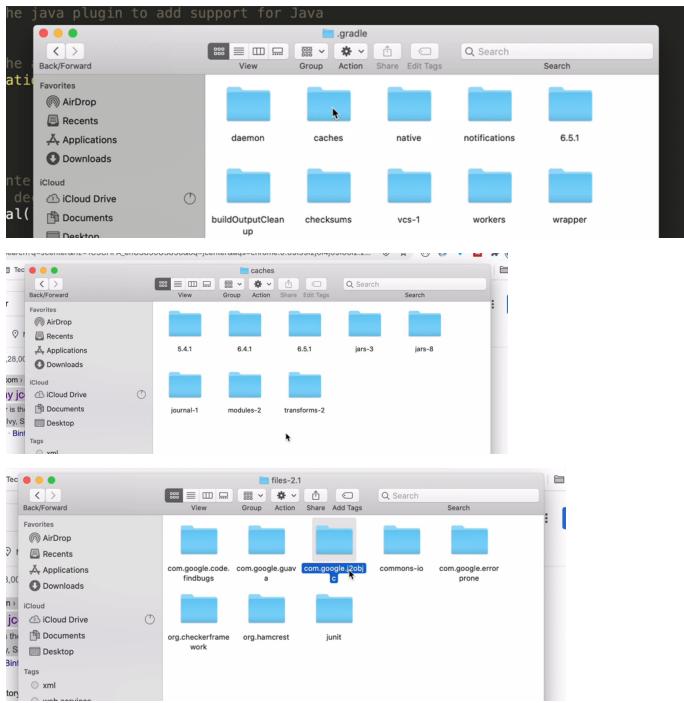
mavenCentral() is the place we download our dependencies.

implementation is for compile and run

testImplementation is for only to run the test.

```
16  
17  
18 repositories {  
19  
20     // Use jcenter for resolving dependencies.  
21     // You can declare any Maven/Ivy/file repository here.  
22     mavenCentral{  
23         url=""  
24     }  
25 }  
26
```

Caching location where dependencies and its transitive dependencies are stored.



MavenLocal is our .m2 folder -

It will check dependencies first in mavenCentral if it could not find it, then it will get from jcenter then followed by mavenLocal.

```

16
17
18 repositories {
19
20     // Use jcenter for resolving dependencies.
21     // You can declare any Maven/Ivy/file repository here.
22     mavenCentral()
23
24     jcenter()
25
26     mavenLocal()
27
28 }
```

Configurations

Scopes

`compile`

`provided`

`runtime`

`test`

`implementation` `api (compile)`

`compileOnly`

`runtimeOnly`

`testImplementation`

`testCompileOnly`

`testRuntimeOnly`

Vs

maven scope

gradle configuration

Mostly used granular level configuration provided below for reference.

compileOnly	runtimeOnly
Dozer	logging-api
Lombok	logging-impl
JMapper	
testCompileOnly	testRuntimeOnly
Junit	Jasmine
	Jupiter
Mockito	Jasmine Runtime

api/compile vs implementation

If we use api/compile as follow the 27 line dependencies were available to project 29 as well.
It is memory leak

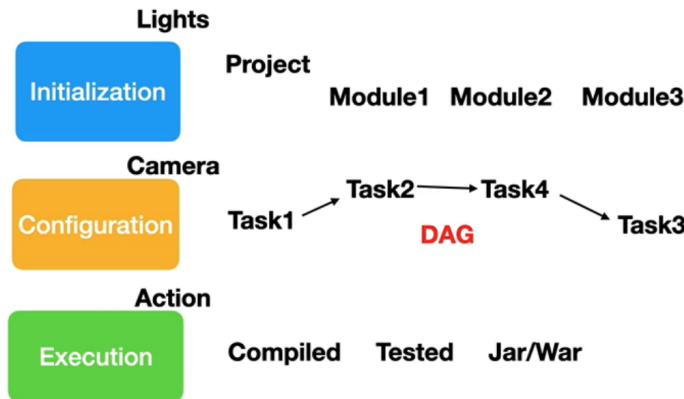
```
23
24
25 dependencies {
26     // This dependency is used by the application.
27     compile 'com.google.guava:guava:29.0-jre'
28
29     ...
30 }
```

api/compile working as transitive dependencies. Whereas implementation work as compile and run.

All the above configuration comes from java plugin

```
8
9 plugins {
10     // Apply the java plugin to add support for Java
11     id 'java'
12 }
```

Gradle has 3 phases as shown below.



```
/Users/bharaththippireddy/Documents/gradle/firstproject
bharaths-MacBook-Pro:firstproject bharaththippireddy$ gradle fT

> Configure project :
Gradle Rocks!!
```

When we run our earlier project we see one phase configuration. In that phase it will print all the logs statements

Every task has two methods as shown below.

```
/*
 * This file was generated by the Gradle build tool.
 * This is a general purpose Gradle build script.
 * Learn how to create Gradle builds at https://docs.gradle.org/6.2/userguide/tutorial_building_a_gradle_project.html
 */
task firstTask{
    println 'Gradle Rocks!!'
    doFirst(){
        println 'doFirst'
    }
    doLast(){
        println 'doLast'
    }
}
```

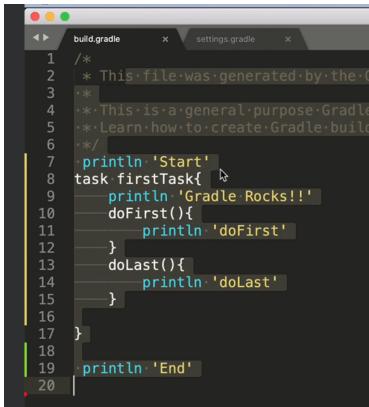
And its output.

```
BUILD SUCCESSFUL in 560ms
bharaths-MacBook-Pro:firstproject bharaththippireddy$ gradle fT

> Configure project :
Gradle Rocks!!

↓ Task :firstTask
doFirst
doLast
```

We wont normally use doFirst



A screenshot of a code editor showing two files: build.gradle and settings.gradle. The build.gradle file contains the following Groovy code:

```
1 /*
2  * This file was generated by the Gradle 'init' plugin.
3  * This is a general-purpose Gradle build file.
4  * Learn how to create Gradle build files at:
5  * https://docs.gradle.org/6.7/userguide/tutorial_tasks.html
6  */
7 task firstTask {
8     println 'Start'
9     doFirst{
10         println 'Gradle Rocks!!!'
11         println 'doFirst'
12     }
13     doLast(){
14         println 'doLast'
15     }
16 }
17
18 println 'End'
19
20
```

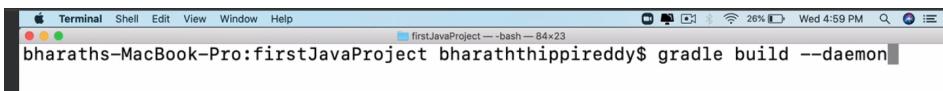
add print statement out of task blocks and notice its output.

```
1 actionable task: 1 executed
bharaths-MacBook-Pro:firstproject bharaththippireddy$ gradle fT

> Configure project :
Start
Gradle Rocks!!
End

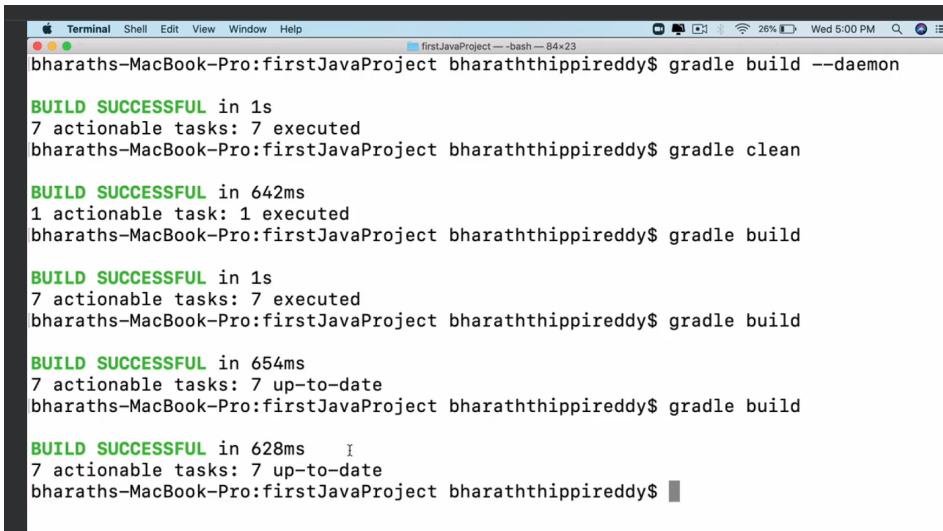
> Task :firstTask
doFirst
doLast
```

This will cache all the information



A screenshot of a terminal window showing the command "gradle build --daemon". The output shows the project configuration and the execution of the firstTask.

Subsequent builds happen very fast due to cache of information about the project.



A screenshot of a terminal window showing multiple Gradle build commands: "gradle build --daemon", "gradle clean", "gradle build", "gradle build", "gradle build", and "gradle build". Each command results in a "BUILD SUCCESSFUL" message, indicating that the build is fast because of the cached information.

We also have not to apply daemon as shown below.

```

Terminal Shell Edit View Window Help firstJavaProject — bash — 84x23
bharaths-MacBook-Pro:firstJavaProject bharaththippireddy$ gradle build --no-daemon
To honour the JVM settings for this build a new JVM will be forked. Please consider
using the daemon: https://docs.gradle.org/6.5.1/userguide/gradle_daemon.html.
Daemon will be stopped at the end of the build stopping after processing

BUILD SUCCESSFUL in 4s
7 actionable tasks: 7 up-to-date
bharaths-MacBook-Pro:firstJavaProject bharaththippireddy$ gradle build --no-daemon

```

Maven vs Gradle

We have plugin section in both pom.xml and build.gradle

```

24
25   <build>
26     <plugins>
27       <plugin>
28         <groupId>org.jvnet.jaxb2.maven2</groupId>
29         <artifactId>maven-jaxb2-plugin</artifactId>
30         <version>0.14.0</version>
31         <executions>
32           <execution>
33             <goals>
34               <goal>generate</goal>
35             </goals>
36           </execution>
37         </executions>
38       </plugin>
39     </plugins>
40   </build>

8
9   <plugins>
10    <!-- Apply the Java plugin to add support for Java -->
11    <!-- id 'java' -->
12    <!-- Apply the application plugin to add support for building a CLI application. -->
13    <!-- id 'application' -->
14  </plugins>
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

```

Similarly dependencies and scope

```

14   <url>http://maven.apache.org</url>
15
16   <dependencies>
17     <dependency>
18       <groupId>junit</groupId>
19       <artifactId>junit</artifactId>
20       <version>4.8.2</version>
21       <scope>test</scope>
22     </dependency>
23   </dependencies>
24
25
26
27
28
29
30
31
32

```

Repository

```

12  ...
13  ...<name>Maven Demo</name>...
14  ...<url>http://maven.apache.org</url>...
15  ...

```

```
15  
16  
17 repositories {  
18     // Use jcenter for resolving dependencies.  
19     // You can declare any Maven/Ivy/file repository here.  
20     mavenCentral()  
21 }  
22  
23 }
```

 Good job!

Question 1:

Which of the following plugin gives tasks to run the java application

java

run

application

main

 Good job!

Question 2:

The java compile task is added by which of the following plugin

compile

jvm

application

java

 Good job!

Question 3:

What should be the dependency scope for junit

runtime

implementation

testcompile

testImplementation

 Good job!

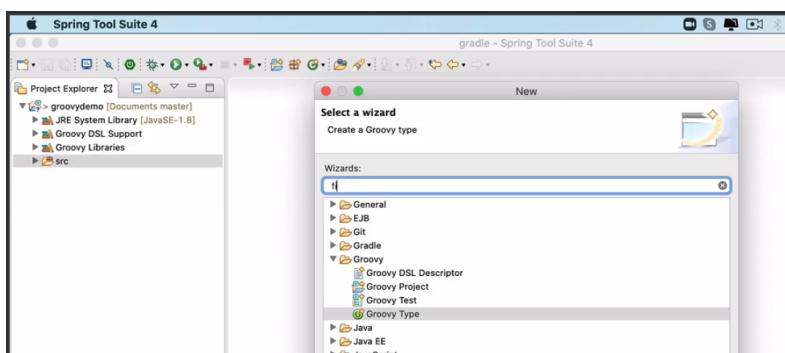
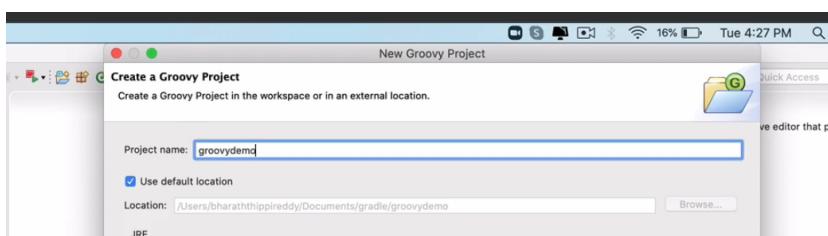
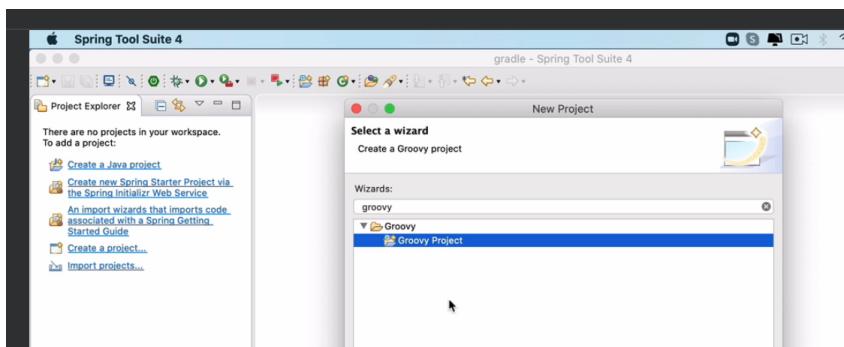
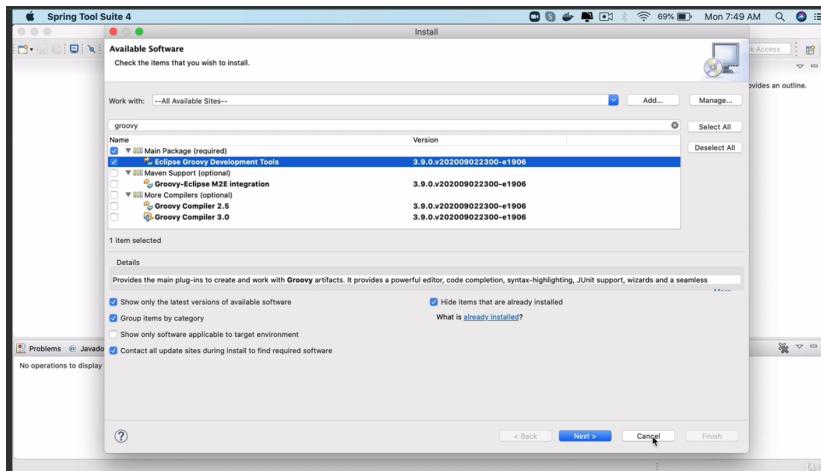
Question 4:

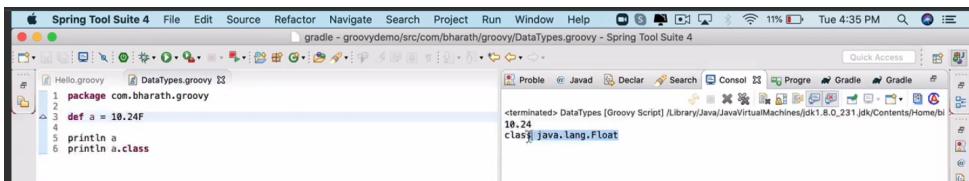
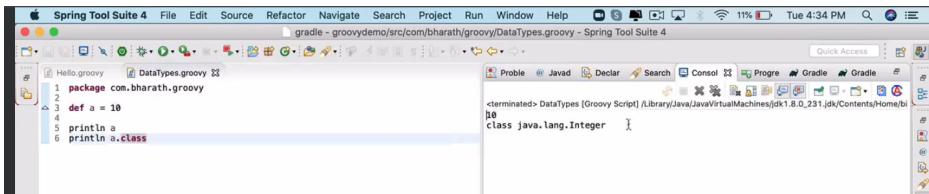
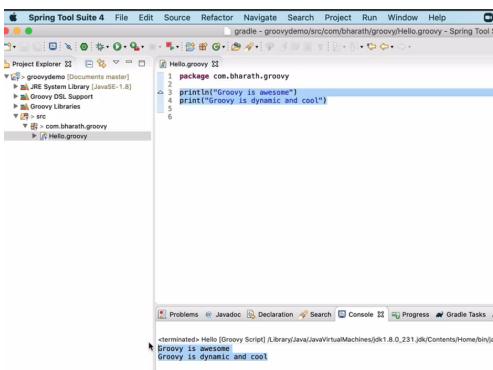
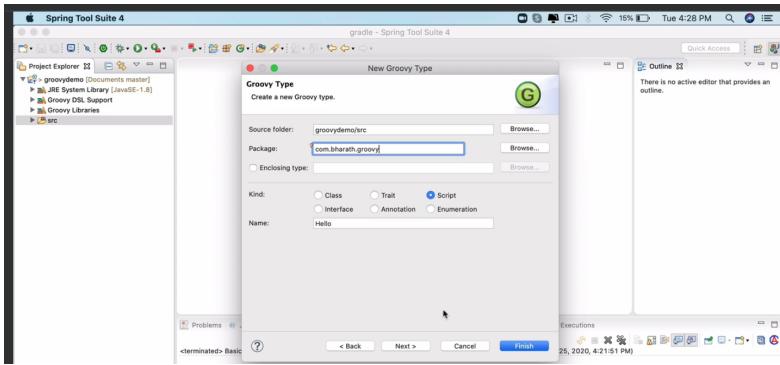
Gradle will run the compile task every-time even if the code has not changed

True

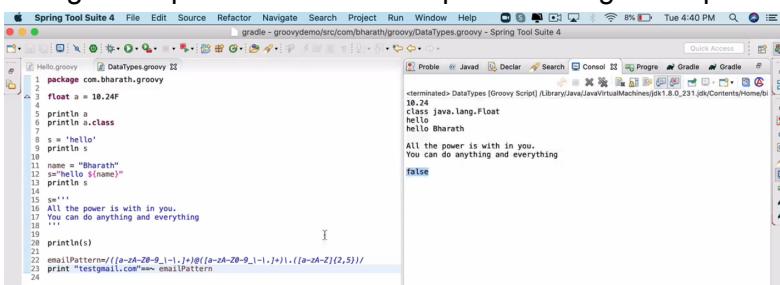
False

Groovy installation on IDE





Groovy string at line number 12 and Multiline string starting at line 15 and line number 22 and 23 regular expression. `==~` is find/pattern in regular expression.



Closure : Lambda expression

The screenshot shows the Spring Tool Suite 4 interface. On the left, there are three files: Hello.groovy, DataTypes.groovy, and ClosuresDemo.groovy. The ClosuresDemo.groovy file contains the following code:

```
1 package com.bharath.groovy
2
3 @c ={
4     println("Closures are super simple")
5 }
6
7 c.call()
```

The right side of the interface shows the 'Console' tab with the output of the script's execution:

```
<terminated> ClosuresDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Home
Closures are super simple
```

The screenshot shows the Spring Tool Suite 4 interface. On the left, there are three files: Hello.groovy, DataTypes.groovy, and ClosuresDemo.groovy. The ClosuresDemo.groovy file contains the following code:

```
1 package com.bharath.groovy
2
3 @c={ n>
4     println(n%2==0?"even":"odd")
5 }
6
7 c.call(10)
```

Default value to n = 2

The screenshot shows the Spring Tool Suite 4 interface. On the left, there are three files: Hello.groovy, DataTypes.groovy, and ClosuresDemo.groovy. The ClosuresDemo.groovy file contains the following code:

```
1 package com.bharath.groovy
2
3 @c={ n2=>
4     println(n2%2==0?"even":"odd")
5 }
6
7 c.call()
```

The right side of the interface shows the 'Console' tab with the output of the script's execution:

```
<terminated> ClosuresDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Home
even
```

If you do not pass anything then we will get null pointer exception

The screenshot shows the Spring Tool Suite 4 interface. On the left, there are three files: Hello.groovy, DataTypes.groovy, and ClosuresDemo.groovy. The ClosuresDemo.groovy file contains the following code:

```
1 package com.bharath.groovy
2
3 @c={ n->
4     println(n%2==0?"even":"odd")
5 }
6
7 c.call()
```

The right side of the interface shows the 'Console' tab with the output of the script's execution, which includes an error message:

```
<terminated> ClosuresDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Home
Caught: java.lang.NullPointerException: Cannot invoke method mod() on null object
java.lang.NullPointerException: Cannot invoke method mod() on null object
at com.bharath.groovy.ClosuresDemo$$_run_closure1.doCall(ClosuresDemo.groovy:1)
at com.bharath.groovy.ClosuresDemo.run(ClosuresDemo.groovy:1)
```

times

The screenshot shows the Spring Tool Suite 4 interface. On the left, there are three files: Hello.groovy, DataTypes.groovy, and ClosuresDemo.groovy. The ClosuresDemo.groovy file contains the following code:

```
1 package com.bharath.groovy
2
3 @c={ n2=>
4     println(n2%2==0?"even":"odd")
5 }
6
7 c.call()
8
9 4.times { n->print n }
```

The right side of the interface shows the 'Console' tab with the output of the script's execution:

```
<terminated> ClosuresDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Home
even
0123
```

It is same as this keyword in java

The screenshot shows the Spring Tool Suite 4 interface. In the left panel, there are several files: Hello.groovy, DataTypes.groovy, ClosuresDemo.groovy, and another unnamed file. The ClosuresDemo.groovy file contains the following code:

```
1 package com.bharath.groovy
2
3 c={ n=2->
4     println(n%2==0?"even":"odd")
5 }
6
7 c.call()
8
9 4.times { n->println n }
10 4.times { println it }
```

In the right panel, the 'Console' tab shows the output of the script:

```
<terminated> ClosuresDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Ho
even
0
1
2
3
0
1
2
3
```

The screenshot shows the Spring Tool Suite 4 interface. In the left panel, there are several files: Hello.groovy, DataTypes.groovy, ClosuresDemo.groovy, and CollectionDemo.groovy. The CollectionDemo.groovy file contains the following code:

```
1 package com.bharath.groovy
2
3 l = [1,2,3]
4 println l
5 println l.class
```

In the right panel, the 'Console' tab shows the output of the script:

```
<terminated> CollectionDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Ho
[1, 2, 3]
class java.util.ArrayList
```

The screenshot shows the Spring Tool Suite 4 interface. In the left panel, there are several files: Hello.groovy, DataTypes.groovy, ClosuresDemo.groovy, and CollectionDemo.groovy. The CollectionDemo.groovy file contains the following code:

```
1 package com.bharath.groovy
2
3 LinkedList l = [1,2,3]
4 println l
5 println l.class
```

In the right panel, the 'Console' tab shows the output of the script:

```
<terminated> CollectionDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Ho
[1, 2, 3]
class java.util.LinkedList
```

<< operator same as ruby.

The screenshot shows the Spring Tool Suite 4 interface. In the left panel, there are several files: Hello.groovy, DataTypes.groovy, ClosuresDemo.groovy, and CollectionDemo.groovy. The CollectionDemo.groovy file contains the following code:

```
1 package com.bharath.groovy
2
3 LinkedList l = [1,2,3]
4 println l
5 println l.class
6
7 l << 4
8
9 println l
```

In the right panel, the 'Console' tab shows the output of the script:

```
<terminated> CollectionDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Ho
[1, 2, 3]
class java.util.LinkedList
[1, 2, 3, 4]
```

plus

The screenshot shows the Spring Tool Suite 4 interface. In the left panel, there are several files: Hello.groovy, DataTypes.groovy, ClosuresDemo.groovy, and CollectionDemo.groovy. The CollectionDemo.groovy file contains the following code:

```
1 package com.bharath.groovy
2
3 LinkedList l = [1,2,3]
4 println l
5 println l.class
6
7 l << 4
8
9 println l
10
11 l=+[5,6,7]
12
13 println l
```

In the right panel, the 'Console' tab shows the output of the script:

```
<terminated> CollectionDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Ho
[1, 2, 3]
class java.util.LinkedList
[1, 2, 3, 4]
[1, 2, 3, 4, 5, 6, 7]
```

minus

The screenshot shows the Spring Tool Suite 4 interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help, and a system status bar showing battery level (38%), time (Tue 5:11 PM), and search icon.

The left sidebar displays project files: Hello.groovy, DataTypes.groovy, ClosuresDemo.groovy, and CollectionDemo.groovy.

The main editor area contains the following Groovy code:

```
1 package com.bharath.groovy
2
3 LinkedList l = [1,2,3]
4 println l
5 println l.class
6
7 l << 4
8
9 println l
10
11 l+= [5,6,7]
12
13 println l
14
15 println l-[3,5]
```

The right side features a tool palette with icons for Problem, Javadoc, Declar, Search, Console, Progress, Gradle, and another Gradle icon. Below the palette is a terminal window showing the output of the executed script:

```
<terminated> CollectionDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Content
[1, 2, 3]
class: java.util.LinkedList
[1, 2, 3, 4]
[1, 2, 3, 4, 5, 6, 7]
[1, 2, 4, 6, 7]
```

each



The screenshot shows the Spring Tool Suite 4 interface with the title bar "gradle - groovydemo/src/com/bharath/groovy/CollectionDemo.groovy - Spring Tool Suite 4". The left pane displays the Groovy script "CollectionDemo.groovy" containing code related to Java's LinkedList class. The right pane shows the output of the script execution, which includes the definition of the LinkedList class and its methods, followed by the printed values of lists l and l-3, and finally the results of the l.each loop.

```
1 package com.bharath.groovy
2
3 LinkedList l = [1,2,3]
4 println l
5 println l.class
6
7 l << 4
8
9 println l
10
11 l=+[5,6,7]
12
13 println l
14
15 println l-[3,5]
16
17 l.each{ println it }
```

```
<terminated> CollectionDemo [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Ho
[1, 2, 3]
class java.util.LinkedList
[1, 2, 3, 4]
[1, 2, 3, 4, 5, 6, 7]
[1, 2, 4, 6, 7]
1
2
3
4
5
6
7
```

The screenshot shows the Spring Tool Suite 4 interface. The title bar reads "Spring Tool Suite 4 gradle - groovydemo/src/com/bharath/groovy/CollectionDemo.groovy - Spring Tool Suite 4". The left sidebar is the "Project Explorer" showing a hierarchy of Groovy files under "groovydemo". The main editor area displays the following Groovy code:

```
1 package com.bharath.groovy
2
3 LinkedList l = [1,2,3]
4 println l
5 println l.class
6
7 l << 4
8
9 println l
10
11 l=l+[5,6,7]
12
13 println l
14
15 println l-[3,5]
16
17 l.each { println it }
18 l.reverseEach { println it }
```

All permutations and similarly for set we have explicitly specify as shown below otherwise it will be treated as arraylist and map.

The screenshot shows the Spring Tool Suite 4 interface with the title bar "gradle - groovydemo/src/com/bharath/groovy/CollectionDemo.groovy - Spring Tool Suite 4". The code editor displays a Groovy script named "CollectionDemo.groovy" containing code to demonstrate various Java collections and their methods. The output window on the right shows the results of the script execution.

```
1 package com.bharath.groovy
2
3 LinkedList l = [1,2,3]
4 println l
5 println l.class
6
7 l << 4
8
9 println l
10
11 l<=[5,6,7]
12
13 println l
14
15 println l-[3,5]
16
17 l.each { println it }
18 l.reverseEach { println it }
19 l.eachPermutation{ println it }
20
21
22 s=['java','js','python','js'] as Set
23 println s
24 println s.class
25
26 m=[[courseName:'Gradle',rating:5,price:20]
27 println(m)
```

```
<terminated> CollectionDemo [Groovy Script] ./Library/Java/JavaVirtualMachine
[7, 6, 5, 1, 4, 3, 2]
[7, 6, 5, 2, 1, 3, 4]
[7, 6, 5, 2, 1, 4, 3]
[7, 6, 5, 2, 3, 1, 4]
[7, 6, 5, 2, 3, 4, 1]
[7, 6, 5, 2, 4, 1, 3]
[7, 6, 5, 2, 4, 3, 1]
[7, 6, 5, 3, 1, 2, 4]
[7, 6, 5, 3, 1, 4, 2]
[7, 6, 5, 3, 2, 1, 4]
[7, 6, 5, 3, 2, 4, 1]
[7, 6, 5, 3, 4, 1, 2]
[7, 6, 5, 3, 4, 2, 1]
[7, 6, 5, 4, 1, 2, 3]
[7, 6, 5, 4, 1, 3, 2]
[7, 6, 5, 4, 2, 1, 3]
[7, 6, 5, 4, 2, 3, 1]
[7, 6, 5, 4, 3, 1, 2]
[7, 6, 5, 4, 3, 2, 1]
[java, js, python]
class java.util.LinkedHashSet
[courseName:Gradle, rating:5, price:20]
```

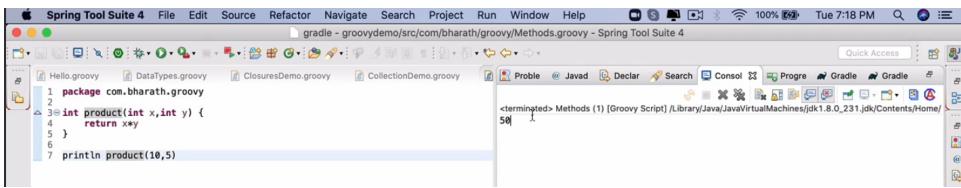
```

5   println l
10  l+=[5,6,7]
11
12  println l
13
14  println l-[3,5]
15
16  l.each { println it }
17  l.reverseEach { println it }
18  l.eachPermutation{ println it }
19
20
21
22  set=['java','js','python','js'] as Set
23  println s
24  println s.class
25
26  m=[courseName:'Gradle',rating:5,price:20]
27  println(m)
28  m.each { k,v->
29    println k
30    println v
31  }
32
33  println m.courseName
34  println m['courseName']
35  println m.get('courseName')
36
37  m['review']="Its Awesome"
38  println m.get('review')

```

[7, 6, 5, 4, 2, 3, 1]
[7, 6, 5, 4, 3, 1, 2]
[7, 6, 5, 4, 3, 2, 1]
[java, js, python]
class java.util.LinkedHashSet
[courseName:Gradle, rating:5, price:20]
courseName
Gradle
rating
5
price
20
Gradle
Gradle
Gradle
Gradle
Its Awesome]

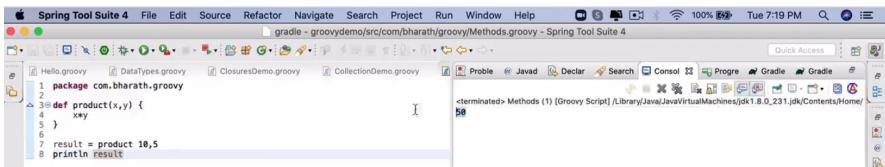
Methods



```

1 package com.bharath.groovy
2
3 int product(int x,int y) {
4     return x*y
5 }
6
7 println product(10,5)

```

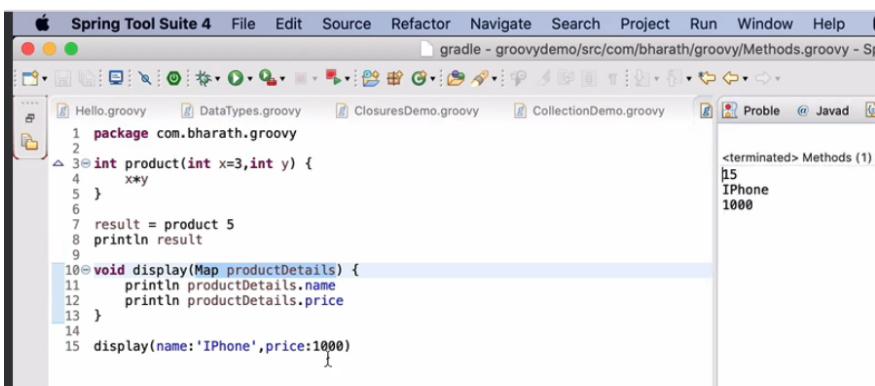


```

<terminated> Methods (1) [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Home/58

```

Default value passing to method and passing map example.



```

1 package com.bharath.groovy
2
3 int product(int x=3,int y) {
4     x*y
5 }
6
7 result = product 5
8 println result
9
10 void display(Map productDetails) {
11     println productDetails.name
12     println productDetails.price
13 }
14
15 display(name:'IPhone',price:1000)

```

Without parentheses also we can pass

```

gradle - groovydemo/src/com/bharath/groovy/Methods.groovy - Spring Tool Suite 4

Hello.groovy  DataTypes.groovy  ClosuresDemo.groovy  CollectionDemo.groovy
1 package com.bharath.groovy
2
3 int product(int x=3,int y) {
4     x*y
5 }
6
7 result = product 5
8 println result
9
10 void display(Map productDetails) {
11     println productDetails.name
12     println productDetails.price
13 }
14
15 display price:1000,name:'IPhone'

```

Console Output:

```

<terminated> Methods (1) [Groovy Script] /Library/Java/JavaVirtualMachines/jdk1
15
IPhone
1000

```

Classes and objects

We do not need to create argument constructor and setters and getters explicitly.

How a static variable is declared and can be called is also provided in the example below.

```

gradle - groovydemo/src/com/bharath/groovy/ClassesDemo.groovy - Spring Tool Suite 4

Hello.groovy  DataTypes.groovy  ClosuresDemo.groovy  CollectionDemo.groovy
1 package com.bharath.groovy
2
3 class Patient {
4     def firstName,lastName,age
5     static hospitalCode="Best Hospital"
6
7     void setLastName(lastName) {
8         if(lastName==null) {
9             throw new IllegalArgumentException("Last Name can not be null");
10        }
11        this.lastName=lastName
12    }
13
14     static void display() {
15         println hospitalCode
16     }
17
18
19 p=new Patient(firstName:'John',lastName:'Bailey',age:40)
20 p.setLastName("Buffer")
21 println p.firstName+" "+p.lastName+" "+p.age
22
23 Patient.display()
24

```

Console Output:

```

<terminated> ClassesDemo [Groovy Script] /Library/Java/JavaVirtualMac
John Buffer 40
Best Hospital

```

Good job!

Question 1:

GStrings can be initialized using which of the following

Single Quotes

//

triple single quotes

double quotes

Good job!

Question 2:

Which of the following holds the value of the parameter passed to a closure

this

that

var

it

 Good job!

Question 3:

Closures are similar to which of the following in java

classes

interfaces

lambdas

abstract classes

 Good job!

Question 4:

Which of the following is used to define a List type

[]

[]

{}

<>

 Good job!

Question 5:

Which of the following should be used between each key and value in a map

,

:

;

/

 Good job!

Question 6:

Groovy supports named parameters using which of the following

lists

sets

tree

map