



Groovy

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

About the Tutorial

Groovy is an object oriented language which is based on Java platform. Groovy 1.0 was released in January 2, 2007 with Groovy 2.4 as the current major release. Groovy is distributed via the Apache License v 2.0. In this tutorial, we would explain all the fundamentals of Groovy and how to put it into practice.

Audience

This tutorial is going to be extremely useful for all those software professionals who would like to learn the basics of Groovy programming.

Prerequisites

Before proceeding with this tutorial, you should have some hands-on experience of Java or any other object-oriented programming language. No Groovy experience is assumed.

Copyright & Disclaimer

© Copyright 2016 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial.....	i
Audience.....	i
Prerequisites.....	i
Copyright & Disclaimer	i
Table of Contents.....	ii
1. GROOVY – OVERVIEW	1
2. GROOVY – ENVIRONMENT	2
3. GROOVY – BASIC SYNTAX	12
Creating Your First Hello World Program	12
Import Statement in Groovy	12
Tokens in Groovy	13
Comments in Groovy	13
Semicolons.....	13
Identifiers	14
Keywords	14
Whitespaces	15
Literals	15
4. GROOVY – DATA TYPES.....	16
Built-in Data Types.....	16
Bound values	16
Class Numeric Types	17
5. GROOVY – VARIABLES.....	19
Variable Declarations.....	19
Naming Variables.....	20

Printing Variables.....	20
6. GROOVY – OPERATORS.....	22
Arithmetic Operators	22
Relational operators	24
Logical Operators	26
Bitwise Operators	27
Assignment operators.....	28
Range Operators	29
Operator Precedence	30
7. GROOVY – LOOPS	31
while Statement.....	31
for Statement.....	32
for-in Statement	34
Loop Control Statements	36
Continue Statement.....	37
8. GROOVY – DECISION MAKING	39
if Statement	39
if / else Statement	40
Nested If statements.....	42
switch Statements	43
Nested Switch Statements	45
9. GROOVY – METHODS	48
Method Parameters	48
Default Parameters.....	49
Method Return Values.....	50
Instance methods	51
Local and External Parameter Names.....	52

this method for Properties	52
10. GROOVY – FILE I/O.....	54
Reading files	54
Reading the Contents of a File as an Entire String	55
Writing to Files.....	55
Getting the Size of a File.....	55
Testing if a File is a Directory.....	56
Creating a Directory	56
Deleting a File	57
Copying files.....	57
Getting Directory Contents	57
11. GROOVY – OPTIONALS.....	59
12. GROOVY – NUMBERS	61
Number Methods.....	62
13. GROOVY – STRINGS	84
String Indexing	84
Basic String Operations	85
String Repetition	86
String Methods	87
14. GROOVY – RANGES.....	105
contains()	105
get()	106
getFrom()	107
getTo()	107
isReverse()	108
size()	109

subList()	109
15. GROOVY – LISTS.....	111
add()	111
contains()	112
get()	113
isEmpty()	113
minus().....	114
plus().....	115
pop()	116
remove()	116
reverse()	117
size()	118
sort()	118
16. GROOVY – MAPS	120
containsKey()	120
get()	121
keySet().....	121
put().....	122
size()	123
values()	124
17. GROOVY – DATES AND TIMES.....	125
Date().....	125
Date (long millisec)	125
after().....	126
equals()	127
compareTo()	128
toString().....	129

before().....	129
getTime()	130
setTime().....	131
18. GROOVY – REGULAR EXPRESSIONS.....	133
19. GROOVY – EXCEPTION HANDLING	134
Catching Exceptions	135
Multiple Catch Blocks.....	136
Finally Block	137
20. GROOVY – OBJECT ORIENTED.....	141
getter and setter Methods	141
Instance Methods	142
Creating Multiple Objects	143
Inheritance.....	144
Extends	144
Inner Classes	145
Abstract Classes	146
Interfaces.....	147
21. GROOVY – GENERICS.....	149
Generic for Collections	149
Generalized Classes.....	150
22. GROOVY – TRAITS.....	151
Implementing Interfaces	152
Properties	152
Composition of Behaviors	153
Extending Traits	154
23. GROOVY – CLOSURES	156

Formal parameters in closures	156
Closures and Variables	157
Using Closures in Methods	157
Closures in Collections and String.....	158
Methods used with Closures	160
24. GROOVY – ANNOTATIONS	164
Annotation Member Values	165
Closure Annotation Parameters	165
Meta Annotations	165
25. GROOVY – XML.....	167
What is XML?	167
XML Support in Groovy	167
XML Markup Builder	168
XML Parsing	171
26. GROOVY – JMX	174
Monitoring the JVM	174
Monitoring Tomcat	176
27. GROOVY – JSON.....	177
JSON Functions	177
Parsing Data using JsonSlurper	177
JsonOutput	180
28. GROOVY – DSLS	182
29. GROOVY – DATABASES	184
Database Connection	184
Creating Database Table	185
Insert Operation.....	185

READ Operation	187
Update Operation	188
DELETE Operation	188
Performing Transactions	189
Commit Operation	189
Rollback Operation	190
Disconnecting Databases	190
30. GROOVY – BUILDERS	191
Swing Builder	191
Event Handlers	193
DOM Builder	195
JsonBuilder	196
NodeBuilder	197
FileTreeBuilder	197
31. GROOVY – COMMAND LINE	198
Classes and Functions	198
Commands	199
32. GROOVY – UNIT TESTING.....	201
Writing a Simple Junit Test Case	201
The Groovy Test Suite	202
33. GROOVY – TEMPLATE ENGINES.....	203
Simple Templating in Strings	203
Simple Template Engine	203
StreamingTemplateEngine	204
XMLTemplateEngine	205
34. GROOVY – META OBJECT PROGRAMMING.....	206

Missing Properties	206
Missing methods.....	207
Metaclass.....	208
Method Missing	209

1. GROOVY – OVERVIEW

Groovy is an object oriented language which is based on Java platform. Groovy 1.0 was released in January 2, 2007 with Groovy 2.4 as the current major release. Groovy is distributed via the Apache License v 2.0.

Features of Groovy

Groovy has the following features:

- Support for both static and dynamic typing
- Support for operator overloading
- Native syntax for lists and associative arrays
- Native support for regular expressions
- Native support for various markup languages such as XML and HTML
- Groovy is simple for Java developers since the syntax for Java and Groovy are very similar
- You can use existing Java libraries
- Groovy extends the `java.lang.Object`

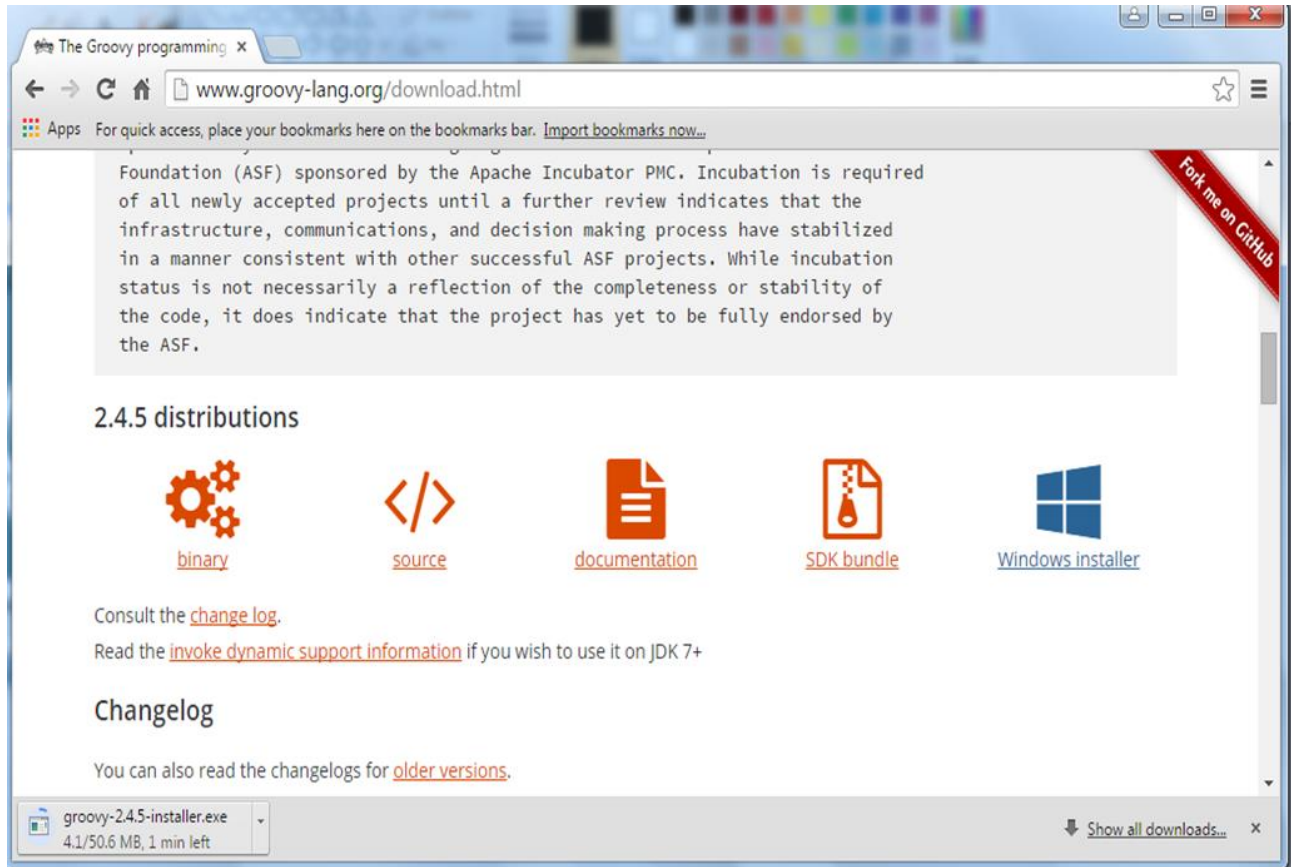
The official website for Groovy is <http://www.groovy-lang.org/>



2. GROOVY – ENVIRONMENT

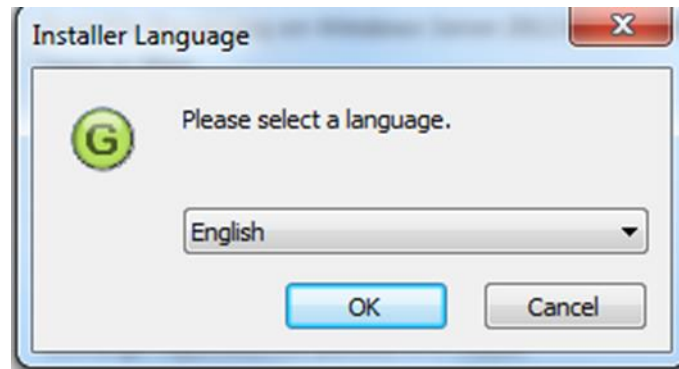
There are a variety of ways to get the Groovy environment setup.

Binary download and installation – Go to the link www.groovy-lang.org/download.html to get the Windows Installer section. Click on this option to start the download of the Groovy installer.



Once you launch the installer, follow the steps given below to complete the installation.

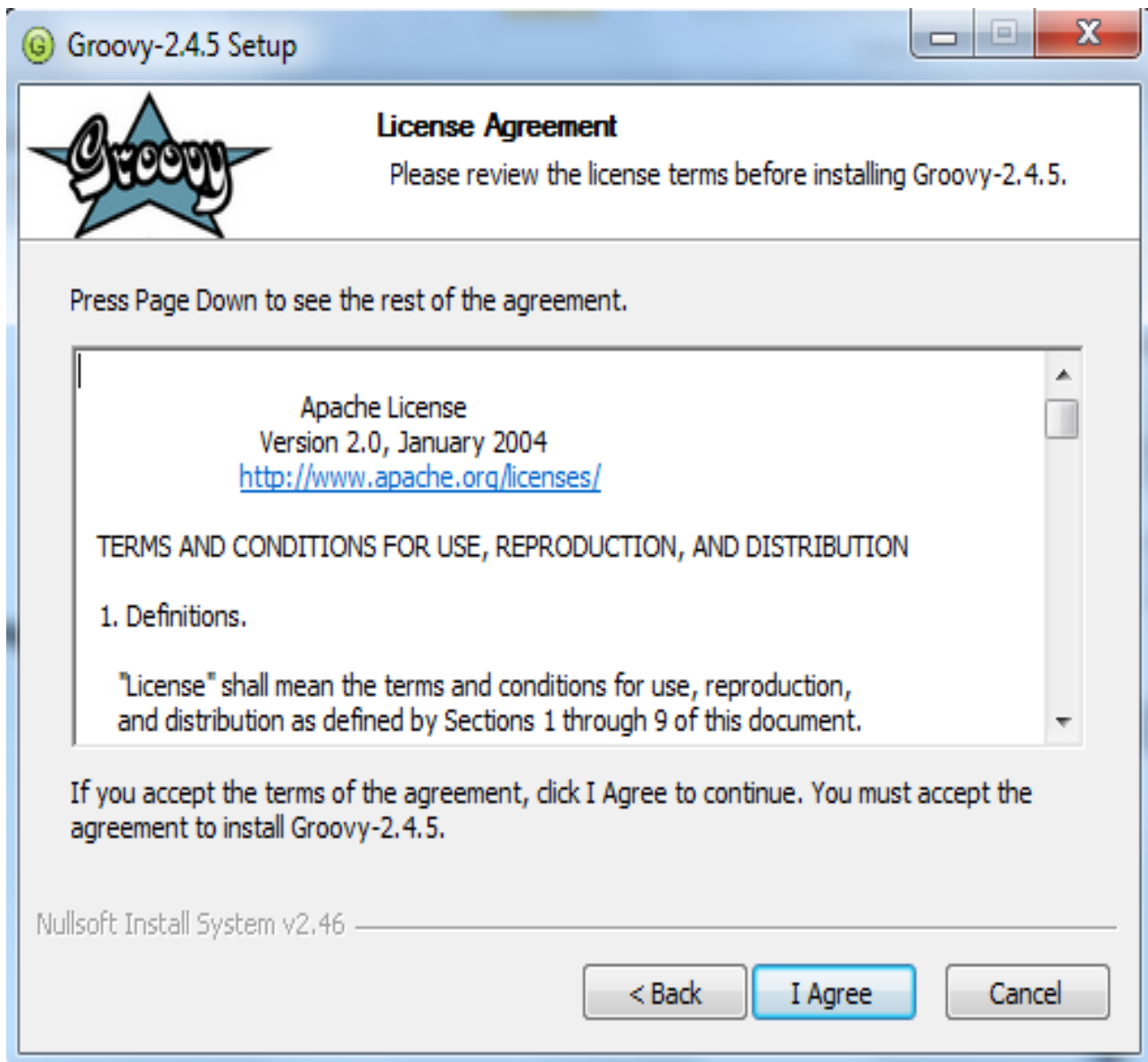
Step 1: Select the language installer



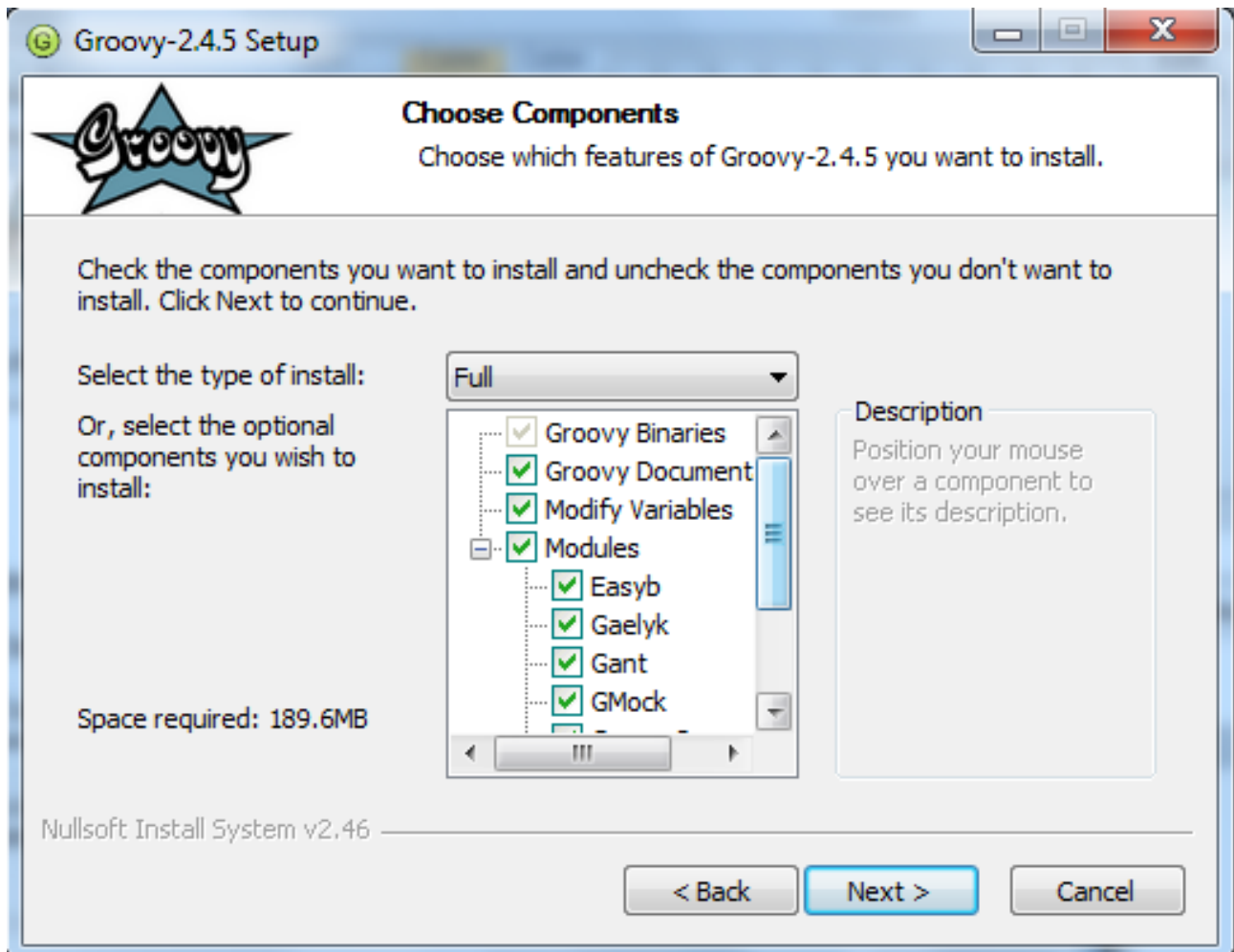
Step 2: Click the Next button in the next screen.



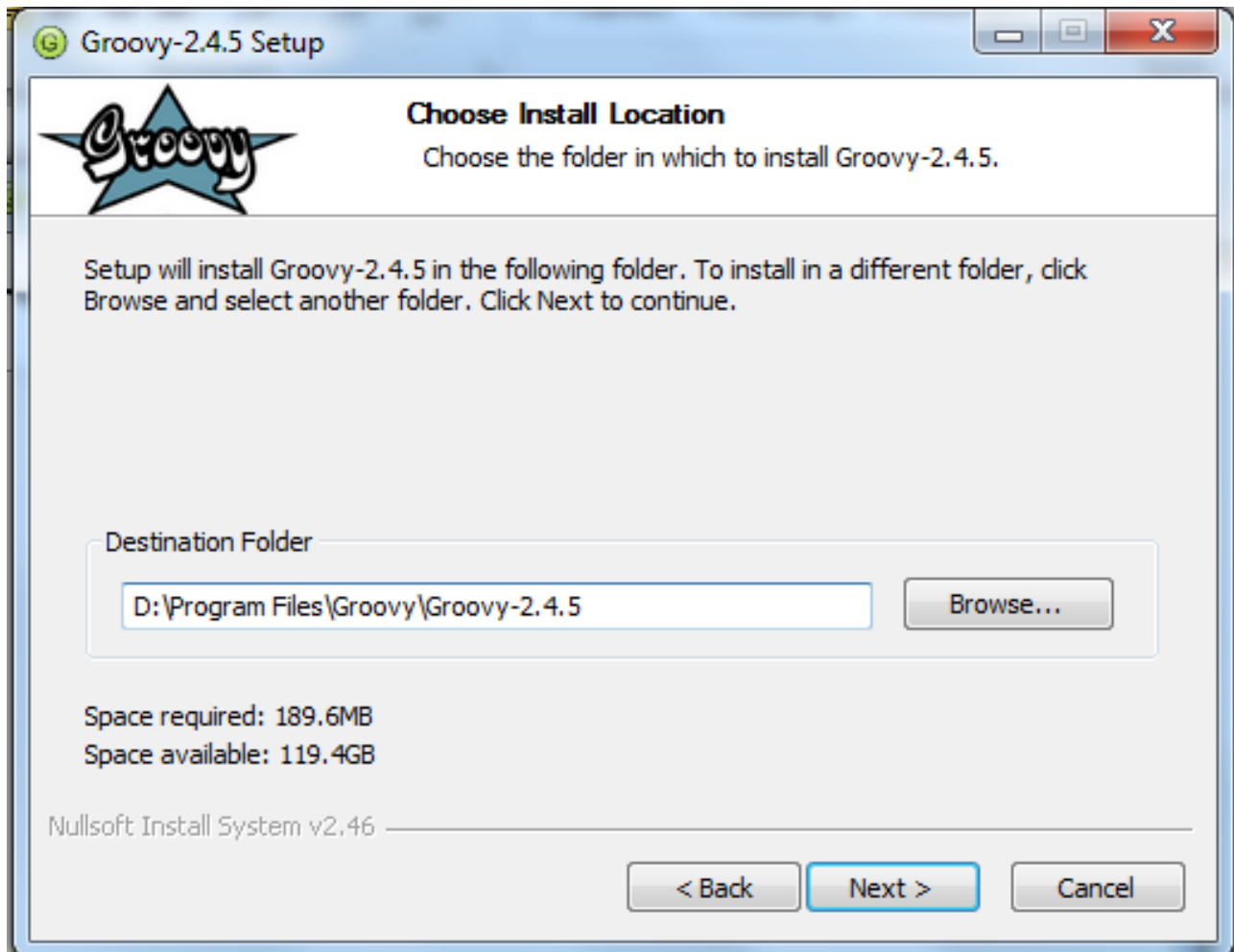
Step 3: Click the 'I Agree' button.



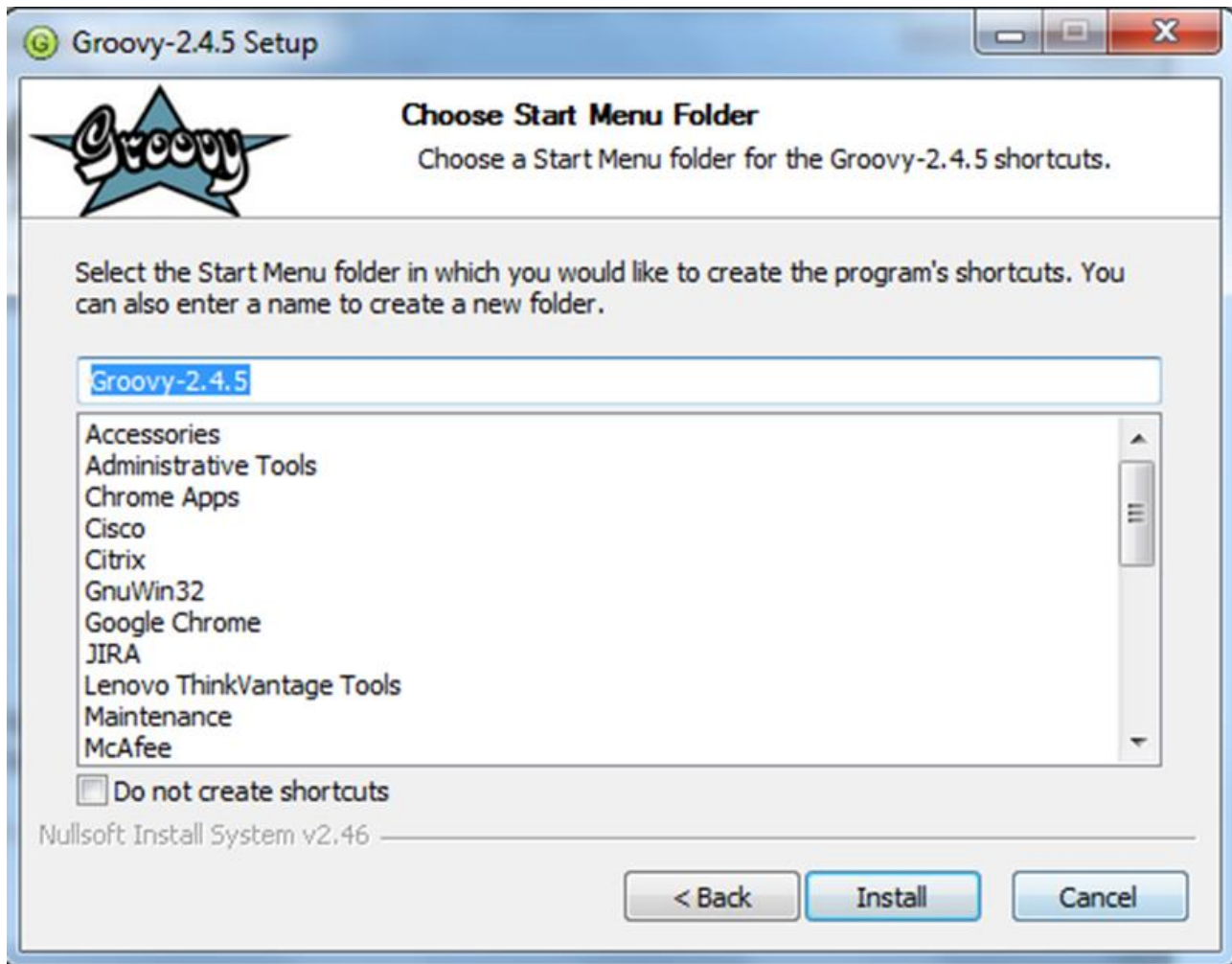
Step 4: Accept the default components and click the Next button.



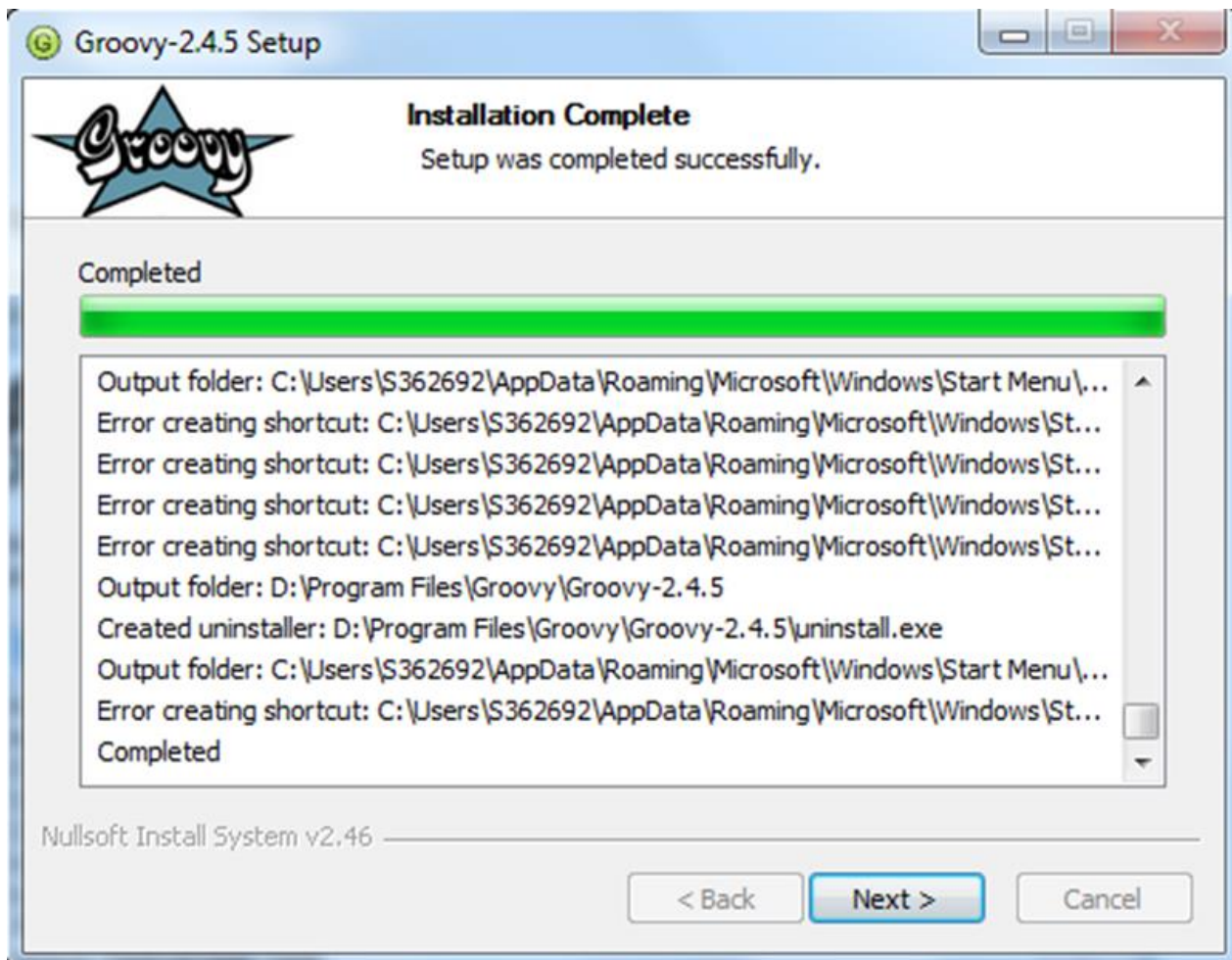
Step 5: Choose the appropriate destination folder and then click the Next button.



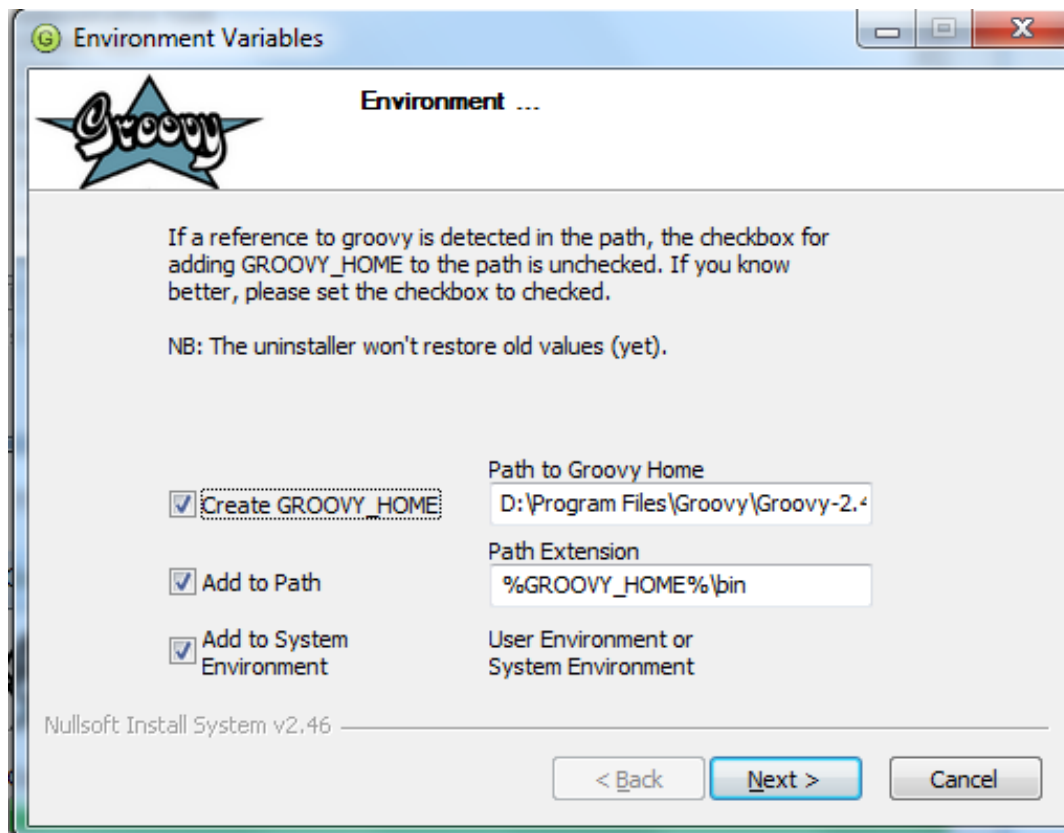
Step 6: Click the Install button to start the installation.



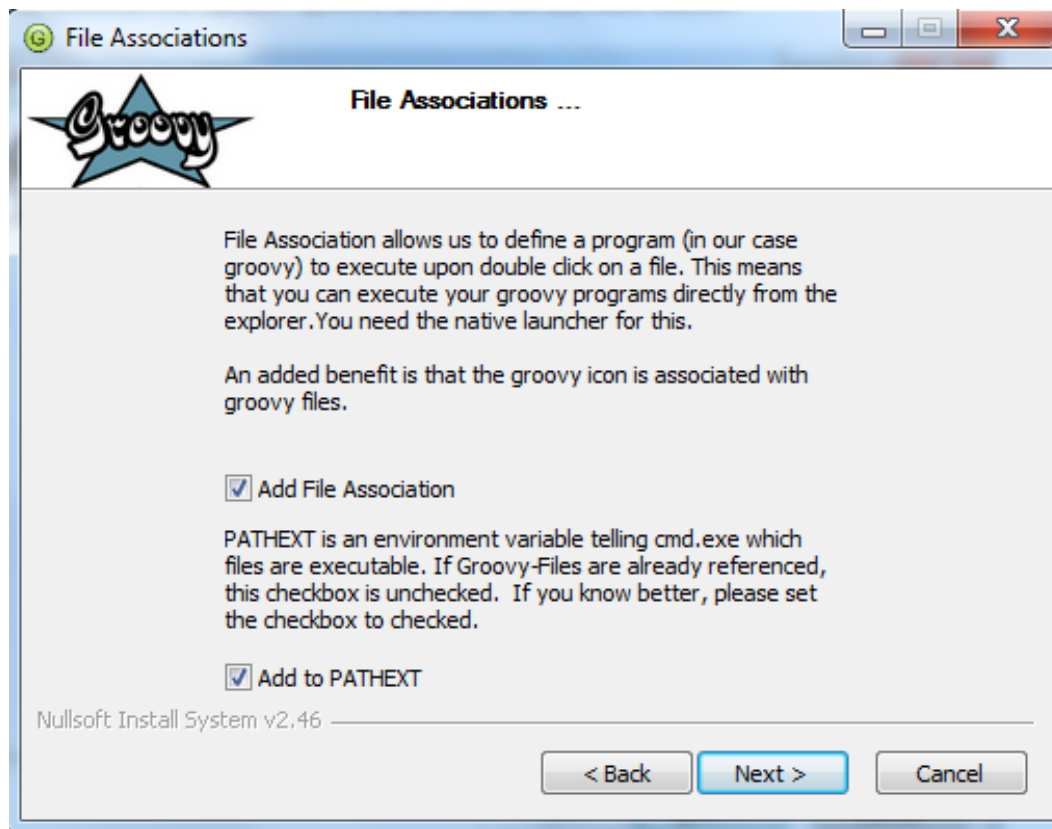
Step 7: Once the installation is complete, click the Next button to start the configuration.



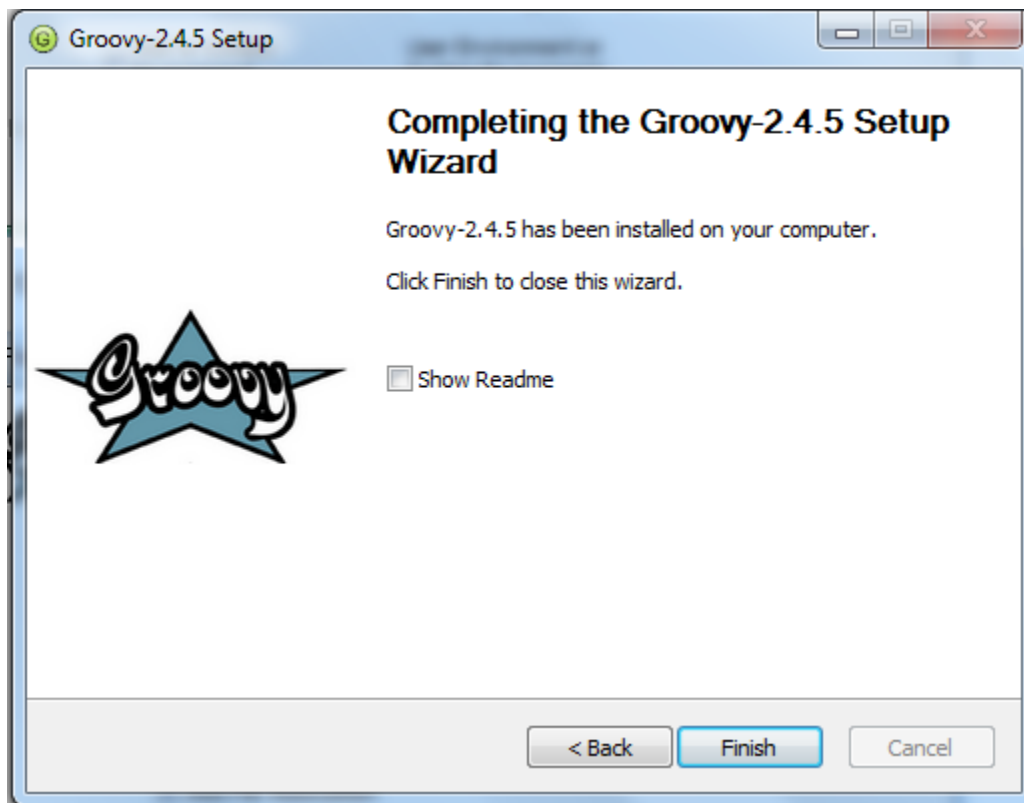
Step 8: Choose the default options and click the Next button.



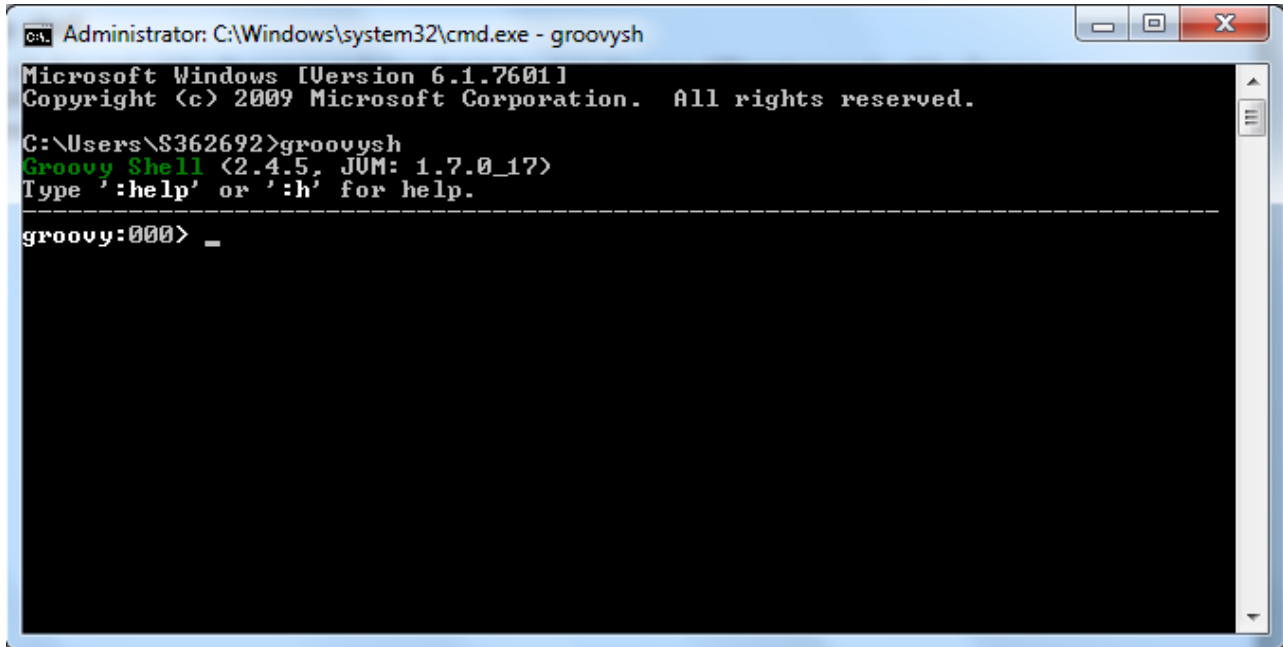
Step 9: Accept the default file associations and click the Next button.



Step 10: Click the Finish button to complete the installation.



Once the above steps are followed, you can then start the groovy shell which is part of the Groovy installation that helps in testing our different aspects of the Groovy language without the need of having a full-fledged integrated development environment for Groovy. This can be done by running the command `groovysh` from the command prompt.



```
Administrator: C:\Windows\system32\cmd.exe - groovysh
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\S362692>groovysh
Groovy Shell (2.4.5, JVM: 1.7.0_17)
Type ':help' or ':h' for help.

-----
groovy:000> _
```

If you want to include the groovy binaries as part of your maven or gradle build, you can add the following lines

Gradle

```
'org.codehaus.groovy:groovy:2.4.5'
```

Maven

```
<groupId>org.codehaus.groovy</groupId>
<artifactId>groovy</artifactId>
<version>2.4.5</version>
```

3. GROOVY – BASIC SYNTAX

In order to understand the basic syntax of Groovy, let's first look at a simple Hello World program.

Creating Your First Hello World Program

Creating your first hello world program is as simple as just entering the following code line:

```
class Example
{
    static void main(String[] args)
    {
        // Using a simple println statement to print output to the console
        println('Hello World');
    }
}
```

When we run the above program, we will get the following result:

```
Hello World
```

Import Statement in Groovy

The import statement can be used to import the functionality of other libraries which can be used in your code. This is done by using the **import** keyword.

The following example shows how to use a simple import of the MarkupBuilder class which is probably one of the most used classes for creating HTML or XML markup.

```
import groovy.xml.MarkupBuilder

def xml=new MarkupBuilder()
```

By default, Groovy includes the following libraries in your code, so you don't need to explicitly import them.

```
import java.lang.*
import java.util.*
```

```
import java.io.*  
import java.net.*  
import groovy.lang.*  
import groovy.util.*  
import java.math.BigInteger  
import java.math.BigDecimal
```

Tokens in Groovy

A token is either a keyword, an identifier, a constant, a string literal, or a symbol.

```
println("Hello World");
```

In the above code line, there are two tokens, the first is the keyword `println` and the next is the string literal of `"Hello World"`.

Comments in Groovy

Comments are used to document your code. Comments in Groovy can be single line or multiline.

Single line comments are identified by using the `//` at any position in the line. An example is shown below:

```
class Example  
{  
    static void main(String[] args)  
    {  
        // Using a simple println statement to print output to the console  
        println('Hello World');  
    }  
}
```

Multiline comments are identified with `/*` in the beginning and `*/` to identify the end of the multiline comment.

```
class Example  
{
```

```
static void main(String[] args)
{
    /* This program is the first program
    This program shows how to display hello world */
    println('Hello World');
}
}
```

Semicolons

Just like the Java programming language, it is required to have semicolons to distinguish between multiple statements defined in Groovy.

```
class Example
{
    static void main(String[] args)
    {
        // One can see the use of a semi-colon after each statement
        def x=5;
        println('Hello World');
    }
}
```

The above example shows semicolons are used to distinguish between different lines of code statements.

Identifiers

Identifiers are used to define variables, functions or other user defined variables. Identifiers start with a letter, a dollar or an underscore. They cannot start with a number. Here are some examples of valid identifiers:

```
def employeename
def student1
def student_name
```

where **def** is a keyword used in Groovy to define an identifier.

Here is a code example of how an identifier can be used in our Hello World program.

```
class Example
{
    static void main(String[] args)
    {
        // One can see the use of a semi-colon after each statement
        def x=5;
        println('Hello World');
    }
}
```

In the above example, the variable **x** is used as an identifier.

Keywords

Keywords as the name suggest are special words which are reserved in the Groovy Programming language. The following table lists the keywords which are defined in Groovy.

as	assert	break	case
catch	class	const	continue
def	default	do	else
enum	extends	false	Finally
for	goto	if	implements
import	in	instanceof	interface
new	pull	package	return
super	switch	this	throw
throws	trait	true	try
while			

Whitespaces

Whitespace is the term used in a programming language such as Java and Groovy to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement.

For example, in the following code example, there is a white space between the keyword **def** and the variable x. This is so that the compiler knows that **def** is the keyword which needs to be used and that x should be the variable name that needs to be defined.

```
def x=5;
```

Literals

A literal is a notation for representing a fixed value in groovy. The groovy language has notations for integers, floating-point numbers, characters and strings. Here are some of the examples of literals in the Groovy programming language:

```
12
```

```
1.45
```

```
'a'
```

```
"aa"
```

4. GROOVY – DATA TYPES

In any programming language, you need to use various variables to store various types of information. Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory to store the value associated with the variable.

You may like to store information of various data types like string, character, wide character, integer, floating point, Boolean, etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

Built-in Data Types

Groovy offers a wide variety of built-in data types. Following is a list of data types which are defined in Groovy:

- **byte** – This is used to represent a byte value. An example is 2.
- **short** – This is used to represent a short number. An example is 10.
- **int** – This is used to represent whole numbers. An example is 1234.
- **long** – This is used to represent a long number. An example is 10000090.
- **float** – This is used to represent 32-bit floating point numbers. An example is 12.34.
- **double** – This is used to represent 64-bit floating point numbers which are longer decimal number representations which may be required at times. An example is 12.3456565.
- **char** – This defines a single character literal. An example is 'a'.
- **Boolean** – This represents a Boolean value which can either be true or false.
- **String** – These are text literals which are represented in **the form** of chain of characters. For example "Hello World".

Bound values

The following table shows the maximum allowed values for the numerical and decimal literals.

byte	-128 to 127
short	-32,768 to 32,767
int	-2,147,483,648 to 2,147,483,647

long	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
float	1.40129846432481707e-45 to 3.40282346638528860e+38
double	4.94065645841246544e-324d to 1.79769313486231570e+308d

Class Numeric Types

In addition to the primitive types, the following object types (sometimes referred to as wrapper types) are allowed:

- java.lang.Byte
- java.lang.Short
- java.lang.Integer
- java.lang.Long
- java.lang.Float
- java.lang.Double

In addition, the following classes can be used for supporting arbitrary precision arithmetic:

Name	Description	Example
java.math.BigInteger	Immutable arbitrary-precision signed integral numbers	30g
java.math.BigDecimal	Immutable arbitrary-precision signed decimal numbers	3.5g

The following code example showcases how the different built-in data types can be used:

```
class Example
{
    static void main(String[] args)
    {
        //Example of a int datatype
        int x=5;
        //Example of a long datatype
        long y=100L;
        //Example of a floating point datatype
        float a=10.56f;
        //Example of a double datatype
        double b=10.5e40;
        //Example of a BigInteger datatype
```

```
        BigInteger bi=30g;
        //Example of a BigDecimal datatype
        BigDecimal bd=3.5g;
        println(x);
        println(y);
        println(a);
        println(b);
        println(bi);
        println(bd);
    }
}
```

When we run the above program, we will get the following result:

```
5
100
10.56
1.05E41
30
3.5
```

5. GROOVY – VARIABLES

Variables in Groovy can be defined in two ways – using the **native syntax** for the data type or the next is **by using the def keyword**. For variable definitions it is mandatory to either provide a type name explicitly or to use "def" in replacement. This is required by the Groovy parser.

There are following basic types of variable in Groovy as explained in the previous chapter:

- **byte** – This is used to represent a byte value. An example is 2.
- **short** – This is used to represent a short number. An example is 10.
- **int** – This is used to represent whole numbers. An example is 1234.
- **long** – This is used to represent a long number. An example is 10000090.
- **float** – This is used to represent 32-bit floating point numbers. An example is 12.34.
- **double** – This is used to represent 64-bit floating point numbers which are longer decimal number representations which may be required at times. An example is 12.3456565.
- **char** – This defines a single character literal. An example is 'a'.
- **Boolean** – This represents a Boolean value which can either be true or false.
- **String** – These are text literals which are represented in **the form** of chain of characters. For example "Hello World".

Groovy also allows for additional types of variables such as arrays, structures and classes which we will see in the subsequent chapters.

Variable Declarations

A variable declaration tells the compiler where and how much to create the storage for the variable.

Following is an example of variable declaration:

```
class Example
{
    static void main(String[] args)
    {
        // x is defined as a variable
```

```
String x="Hello";  
// The value of the variable is printed to the console  
println(x);  
  
}  
}
```

When we run the above program, we will get the following result:

```
Hello
```

Naming Variables

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because Groovy, just like Java is a case-sensitive programming language.

```
class Example  
{  
    static void main(String[] args)  
    {  
        // Defining a variable in lowercase  
        int x=5;  
        // Defining a variable in uppercase  
        int X=6;  
        // Defining a variable with the underscore in it's name  
        def _Name="Joe";  
        println(x);  
        println(X);  
        println(_Name);  
    }  
}
```

When we run the above program, we will get the following result:

```
5  
6
```

Joe

We can see that **x** and **X** are two different variables because of case sensitivity and in the third case, we can see that `_Name` begins with an underscore.

Printing Variables

You can print the current value of a variable with the `println` function. The following example shows how this can be achieved.

```
class Example
{
    static void main(String[] args)
    {
        //Initializing 2 variables
        int x=5;
        int X=6;
        //Printing the value of the variables to the console
        println("The value of x is " + x + "The value of X is " + X);
    }
}
```

When we run the above program, we will get the following result:

The value of x is 5 The value of X is 6

End of ebook preview
If you liked what you saw...
Buy it from our store @ **<https://store.tutorialspoint.com>**