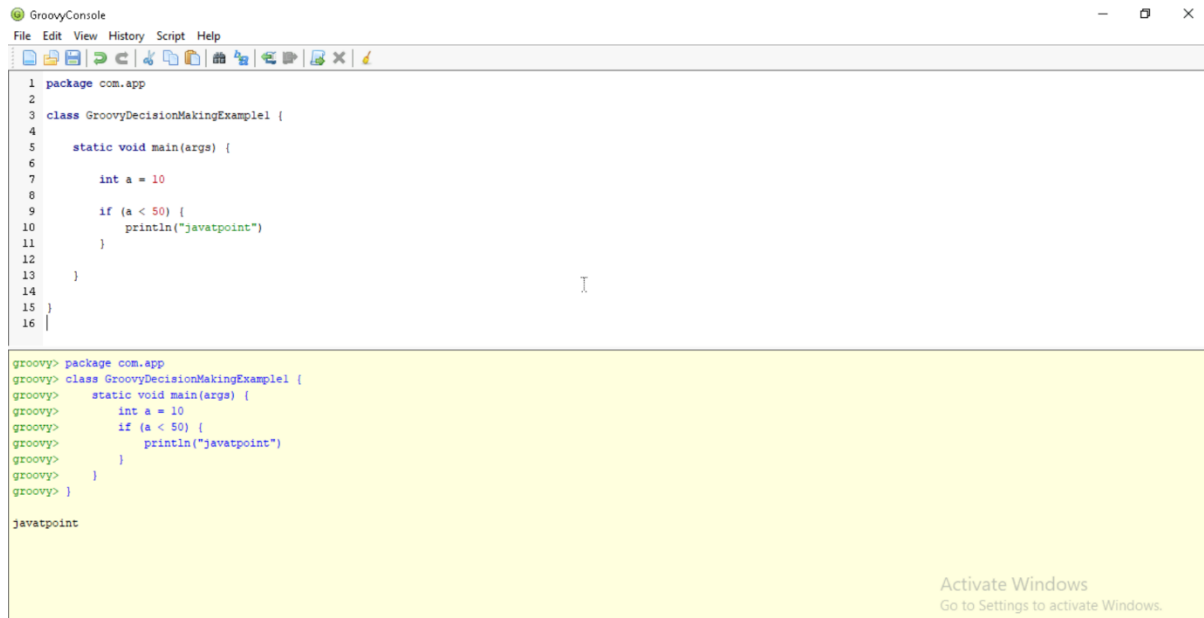


Decision Making in Groovy

1) If statement



The screenshot shows the GroovyConsole application. The top pane contains the following Groovy code:

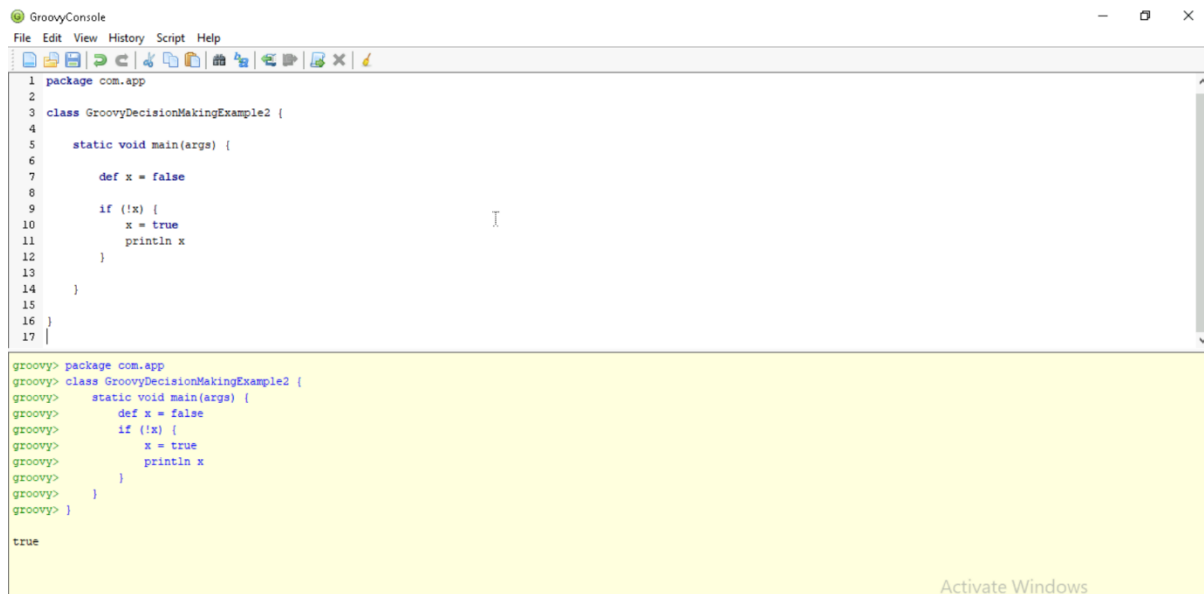
```
1 package com.app
2
3 class GroovyDecisionMakingExample1 {
4
5     static void main(args) {
6
7         int a = 10
8
9         if (a < 50) {
10             println("javatpoint")
11         }
12     }
13 }
14
15
16
```

The bottom pane shows the execution output:

```
groovy> package com.app
groovy> class GroovyDecisionMakingExample1 {
groovy>     static void main(args) {
groovy>         int a = 10
groovy>         if (a < 50) {
groovy>             println("javatpoint")
groovy>         }
groovy>     }
groovy> }
groovy>
javatpoint
```

An "Activate Windows" watermark is visible in the bottom right corner.

2) If statement



The screenshot shows the GroovyConsole application. The top pane contains the following Groovy code:

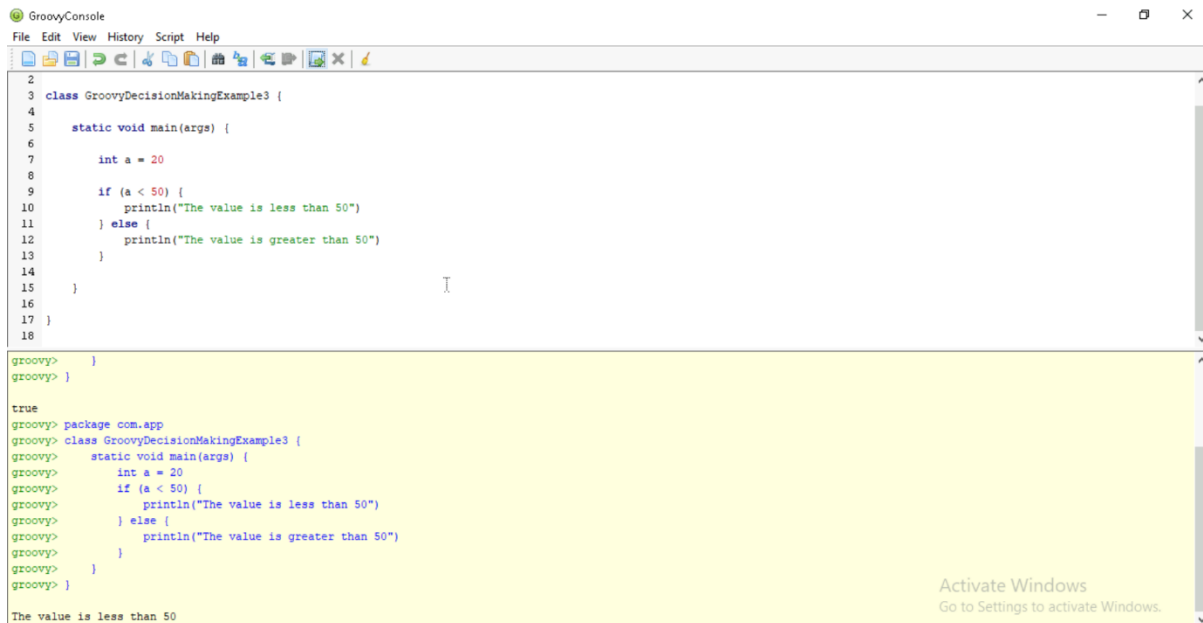
```
1 package com.app
2
3 class GroovyDecisionMakingExample2 {
4
5     static void main(args) {
6
7         def x = false
8
9         if (!x) {
10             x = true
11             println x
12         }
13     }
14 }
15
16
17
```

The bottom pane shows the execution output:

```
groovy> package com.app
groovy> class GroovyDecisionMakingExample2 {
groovy>     static void main(args) {
groovy>         def x = false
groovy>         if (!x) {
groovy>             x = true
groovy>             println x
groovy>         }
groovy>     }
groovy> }
groovy>
true
```

An "Activate Windows" watermark is visible in the bottom right corner.

3) If else statement



The screenshot shows the GroovyConsole application. The top pane contains the following Groovy code:

```
2
3 class GroovyDecisionMakingExample3 {
4
5     static void main(args) {
6
7         int a = 20
8
9         if (a < 50) {
10             println("The value is less than 50")
11         } else {
12             println("The value is greater than 50")
13         }
14     }
15 }
16
17
18
```

The bottom pane shows the execution results:

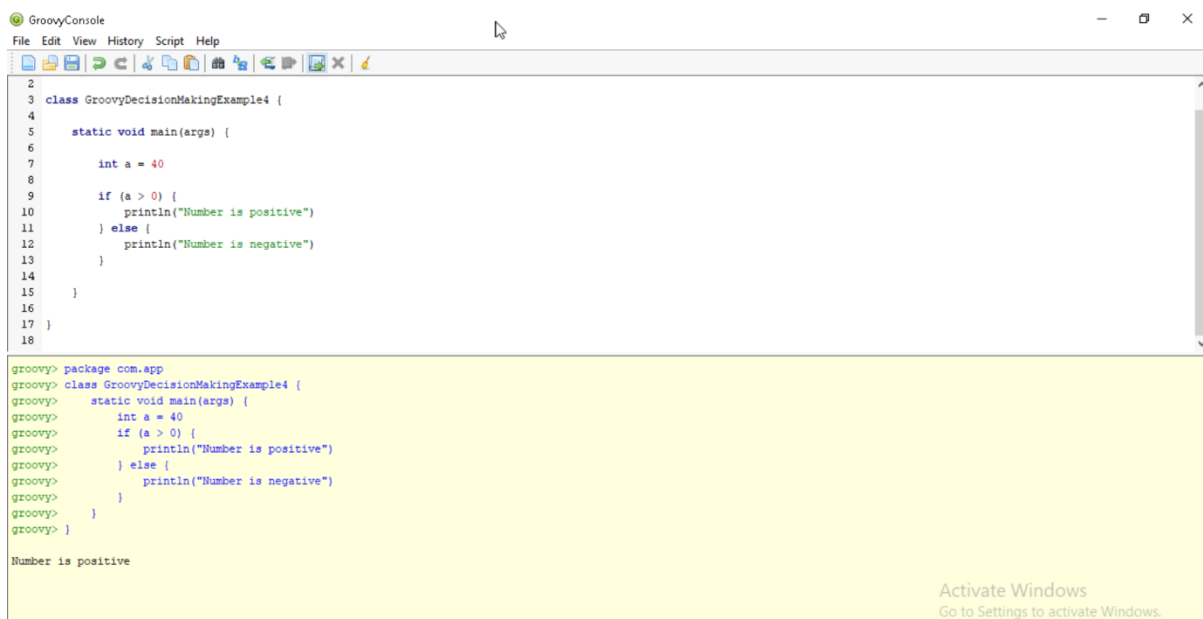
```
groovy>
groovy> ]

true
groovy> package com.app
groovy> class GroovyDecisionMakingExample3 {
groovy>     static void main(args) {
groovy>         int a = 20
groovy>         if (a < 50) {
groovy>             println("The value is less than 50")
groovy>         } else {
groovy>             println("The value is greater than 50")
groovy>         }
groovy>     }
groovy> ]
groovy> ]

The value is less than 50
```

An "Activate Windows" watermark is visible in the bottom right corner of the console window.

4) If else statement



The screenshot shows the GroovyConsole application. The top pane contains the following Groovy code:

```
2
3 class GroovyDecisionMakingExample4 {
4
5     static void main(args) {
6
7         int a = 40
8
9         if (a > 0) {
10             println("Number is positive")
11         } else {
12             println("Number is negative")
13         }
14     }
15 }
16
17
18
```

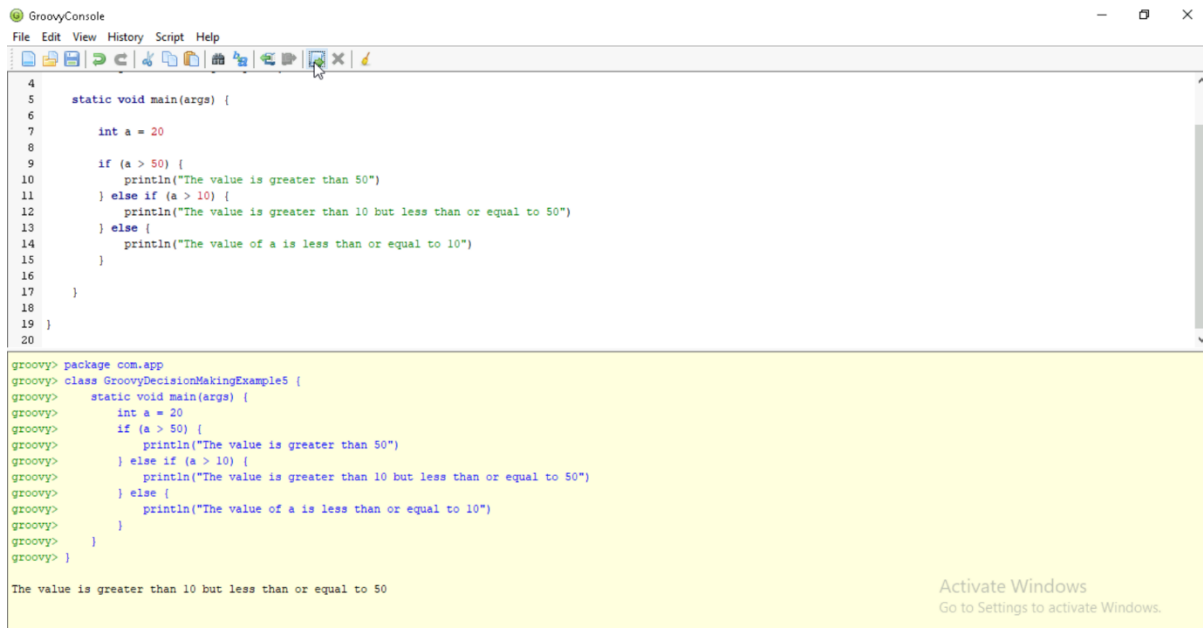
The bottom pane shows the execution results:

```
groovy> package com.app
groovy> class GroovyDecisionMakingExample4 {
groovy>     static void main(args) {
groovy>         int a = 40
groovy>         if (a > 0) {
groovy>             println("Number is positive")
groovy>         } else {
groovy>             println("Number is negative")
groovy>         }
groovy>     }
groovy> ]
groovy> ]

Number is positive
```

An "Activate Windows" watermark is visible in the bottom right corner of the console window.

5) Nested If Statement



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with a nested if statement. The bottom pane shows the command history and the output of the script.

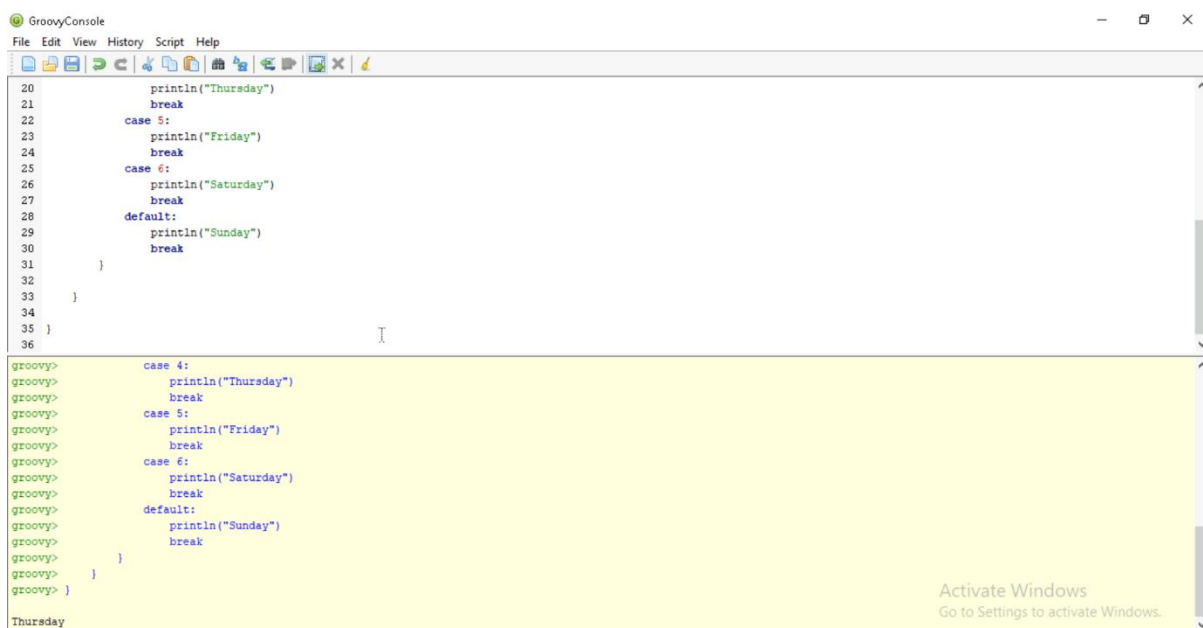
```
4 static void main(args) {
5
6     int a = 20
7
8     if (a > 50) {
9         println("The value is greater than 50")
10    } else if (a > 10) {
11        println("The value is greater than 10 but less than or equal to 50")
12    } else {
13        println("The value of a is less than or equal to 10")
14    }
15 }
16
17
18
19
20
```

```
groovy> package com.app
groovy> class GroovyDecisionMakingExample5 {
groovy>     static void main(args) {
groovy>         int a = 20
groovy>         if (a > 50) {
groovy>             println("The value is greater than 50")
groovy>         } else if (a > 10) {
groovy>             println("The value is greater than 10 but less than or equal to 50")
groovy>         } else {
groovy>             println("The value of a is less than or equal to 10")
groovy>         }
groovy>     }
groovy> }

The value is greater than 10 but less than or equal to 50
```

Activate Windows
Go to Settings to activate Windows.

6) Switch Statement



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with a switch statement. The bottom pane shows the command history and the output of the script.

```
20     println("Thursday")
21     break
22     case 5:
23         println("Friday")
24         break
25     case 6:
26         println("Saturday")
27         break
28     default:
29         println("Sunday")
30         break
31 }
32
33 }
34
35 }
36
```

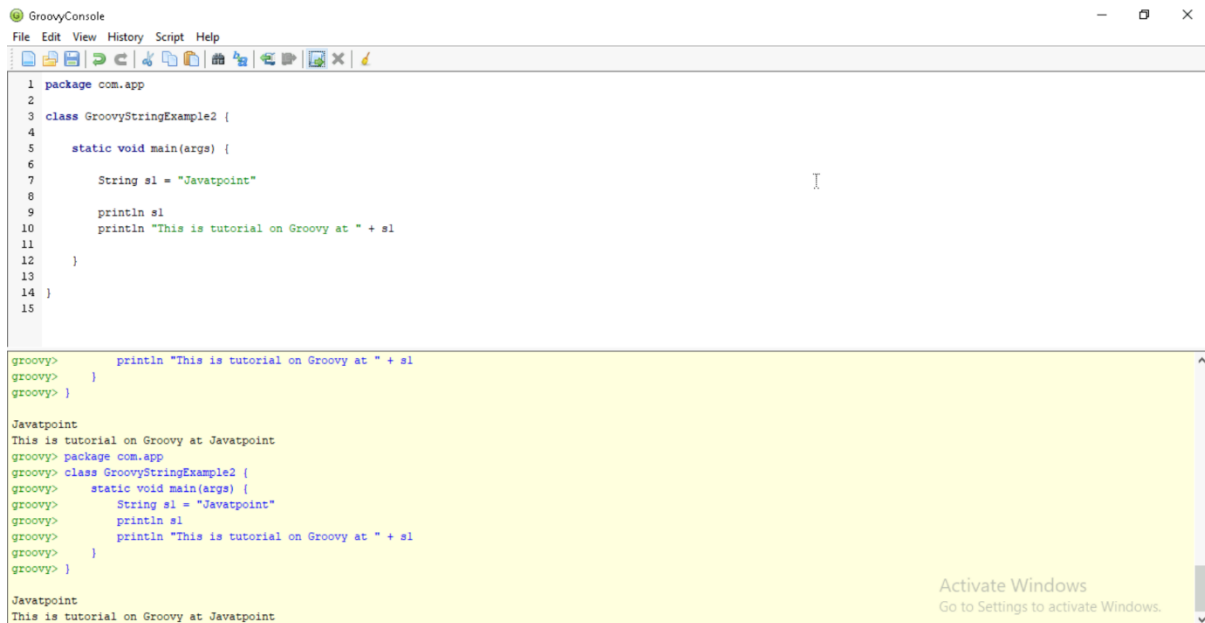
```
groovy> case 4:
groovy>     println("Thursday")
groovy>     break
groovy> case 5:
groovy>     println("Friday")
groovy>     break
groovy> case 6:
groovy>     println("Saturday")
groovy>     break
groovy> default:
groovy>     println("Sunday")
groovy>     break
groovy> }
groovy> }
groovy> }

Thursday
```

Activate Windows
Go to Settings to activate Windows.

String in Groovy

7) Single-quoted string



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with the following code:

```
1 package com.app
2
3 class GroovyStringExample2 {
4
5     static void main(args) {
6
7         String s1 = "Javatpoint"
8
9         println s1
10        println "This is tutorial on Groovy at " + s1
11    }
12 }
13
14 }
15
```

The bottom pane shows the output of the script:

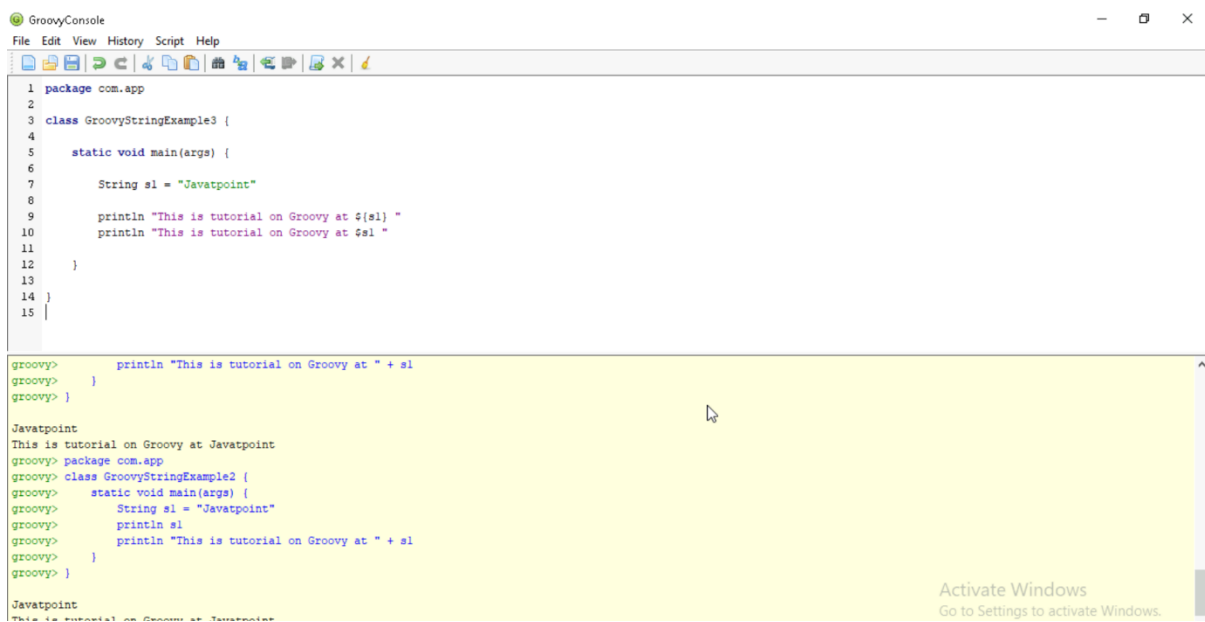
```
groovy>      println "This is tutorial on Groovy at " + s1
groovy>      }
groovy> }

Javatpoint
This is tutorial on Groovy at Javatpoint
groovy> package com.app
groovy> class GroovyStringExample2 {
groovy>     static void main(args) {
groovy>         String s1 = "Javatpoint"
groovy>         println s1
groovy>         println "This is tutorial on Groovy at " + s1
groovy>     }
groovy> }

Javatpoint
This is tutorial on Groovy at Javatpoint
```

An "Activate Windows" watermark is visible in the bottom right corner of the console window.

8) Double-quoted string



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with the following code:

```
1 package com.app
2
3 class GroovyStringExample3 {
4
5     static void main(args) {
6
7         String s1 = "Javatpoint"
8
9         println "This is tutorial on Groovy at ${s1} "
10        println "This is tutorial on Groovy at $s1 "
11    }
12 }
13
14 }
15
```

The bottom pane shows the output of the script:

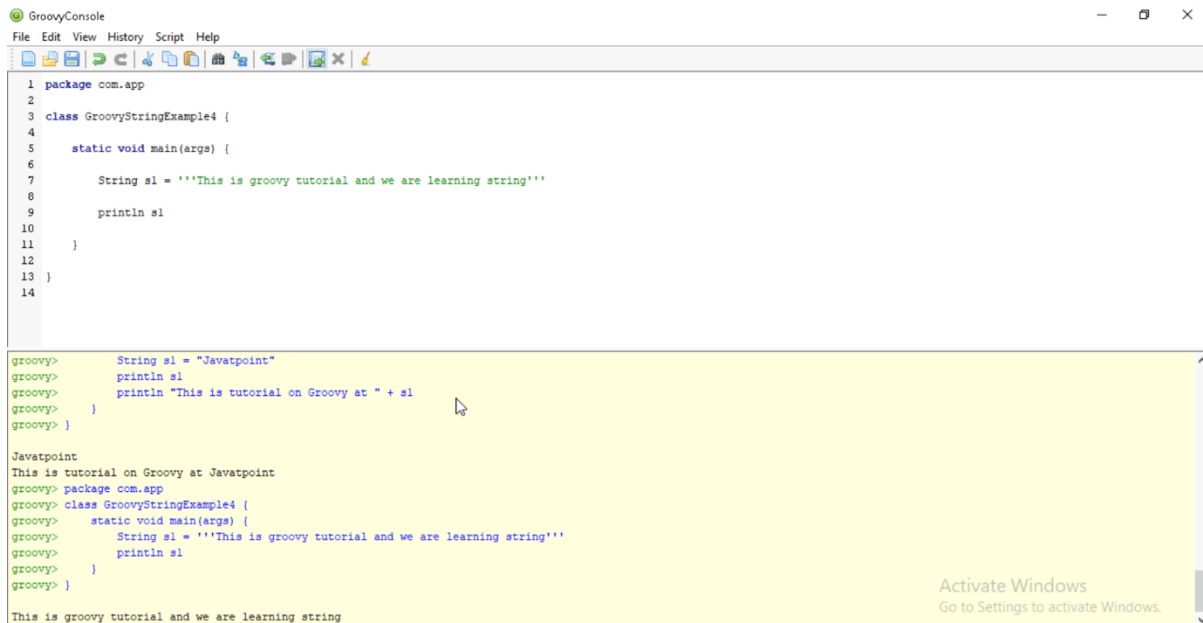
```
groovy>      println "This is tutorial on Groovy at " + s1
groovy>      }
groovy> }

Javatpoint
This is tutorial on Groovy at Javatpoint
groovy> package com.app
groovy> class GroovyStringExample2 {
groovy>     static void main(args) {
groovy>         String s1 = "Javatpoint"
groovy>         println s1
groovy>         println "This is tutorial on Groovy at " + s1
groovy>     }
groovy> }

Javatpoint
This is tutorial on Groovy at Javatpoint
```

An "Activate Windows" watermark is visible in the bottom right corner of the console window.

9) Double-quoted string



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with the following code:

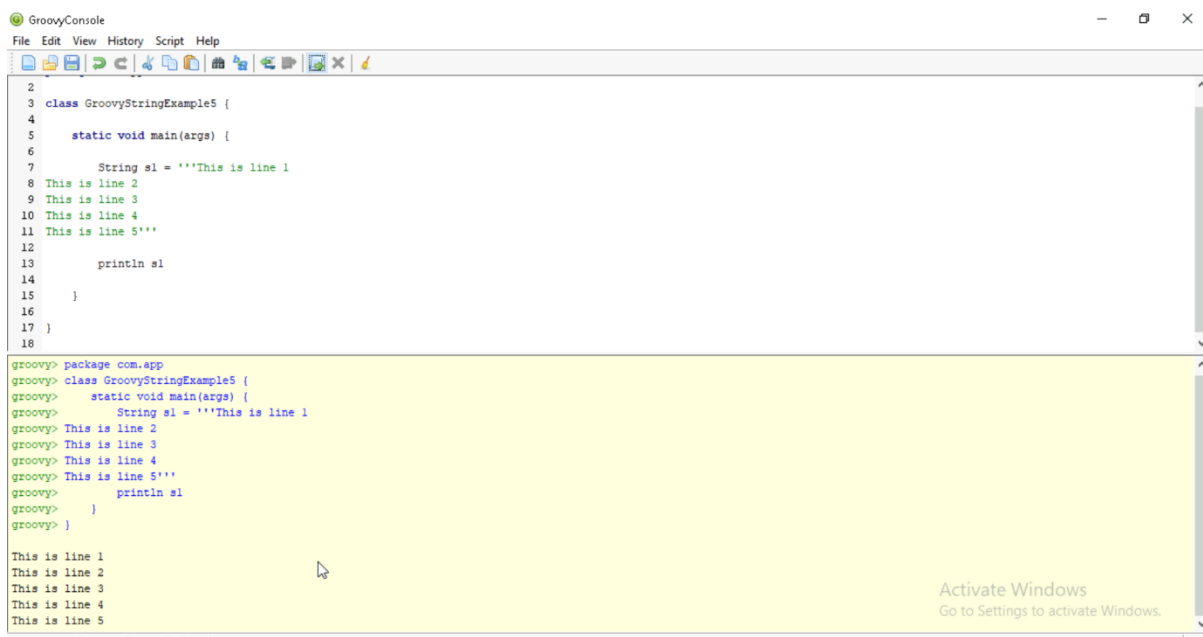
```
1 package com.app
2
3 class GroovyStringExample4 {
4
5     static void main(args) {
6
7         String s1 = "This is groovy tutorial and we are learning string"
8
9         println s1
10
11     }
12 }
13
14
```

The bottom pane shows the execution output:

```
groovy> String s1 = "Javatpoint"
groovy> println s1
groovy> println "This is tutorial on Groovy at " + s1
groovy> ]
Javatpoint
This is tutorial on Groovy at Javatpoint
groovy> package com.app
groovy> class GroovyStringExample4 {
groovy>     static void main(args) {
groovy>         String s1 = "This is groovy tutorial and we are learning string"
groovy>         println s1
groovy>     }
groovy> ]
This is groovy tutorial and we are learning string
```

An "Activate Windows" watermark is visible in the bottom right corner of the console window.

10) Triple-single-quoted string



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with the following code:

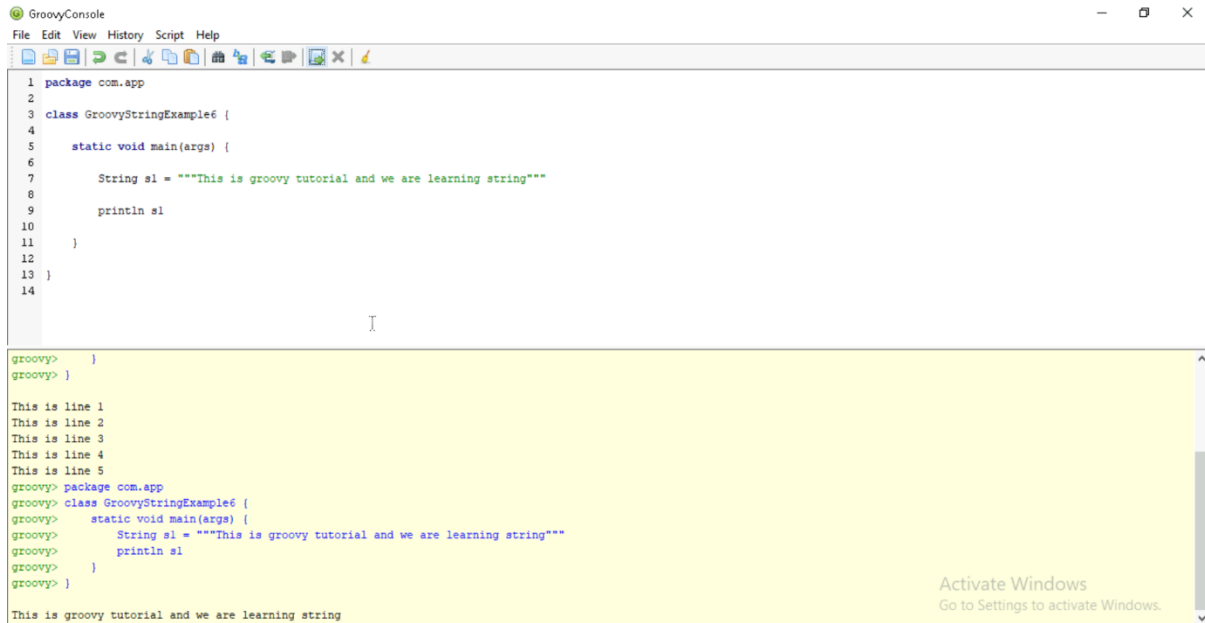
```
2
3 class GroovyStringExample5 {
4
5     static void main(args) {
6
7         String s1 = '''This is line 1
8 This is line 2
9 This is line 3
10 This is line 4
11 This is line 5'''
12
13         println s1
14
15     }
16 }
17
18
```

The bottom pane shows the execution output:

```
groovy> package com.app
groovy> class GroovyStringExample5 {
groovy>     static void main(args) {
groovy>         String s1 = '''This is line 1
groovy> This is line 2
groovy> This is line 3
groovy> This is line 4
groovy> This is line 5'''
groovy>         println s1
groovy>     }
groovy> ]
This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
```

An "Activate Windows" watermark is visible in the bottom right corner of the console window.

11) Triple-single-quoted string



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with a triple-single-quoted string. The bottom pane shows the output of the script, which prints the string on a single line.

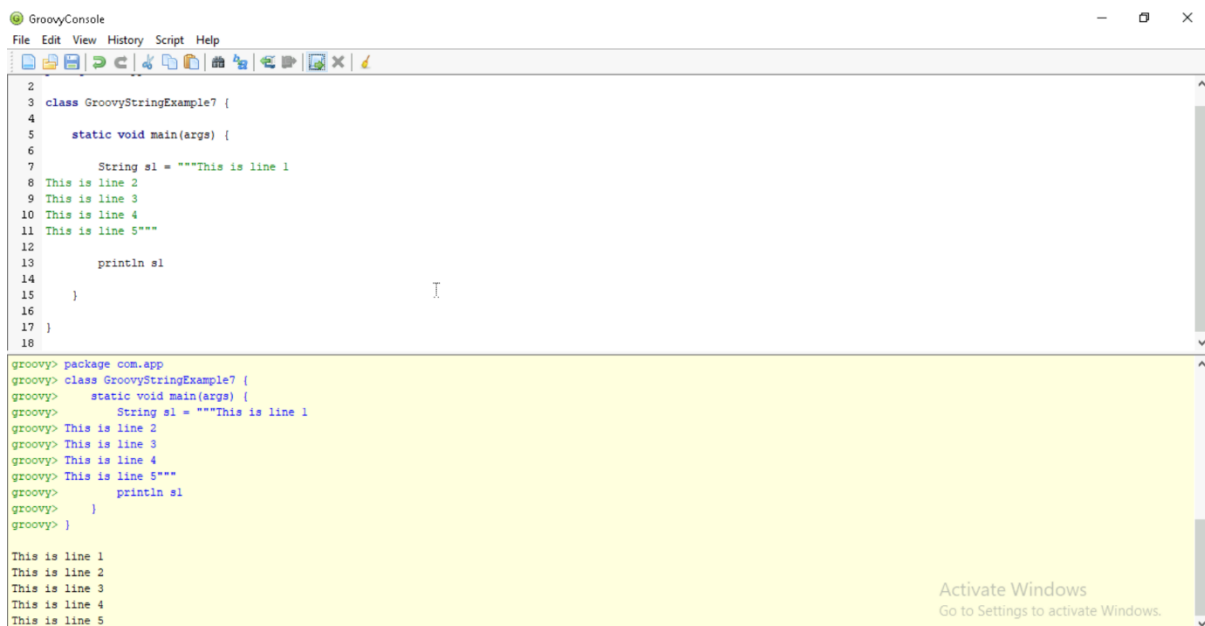
```
1 package com.app
2
3 class GroovyStringExample6 {
4
5     static void main(args) {
6
7         String s1 = """This is groovy tutorial and we are learning string"""
8
9         println s1
10
11     }
12 }
13
14
```

```
groovy> }
groovy> ]

This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
groovy> package com.app
groovy> class GroovyStringExample6 {
groovy>     static void main(args) {
groovy>         String s1 = """This is groovy tutorial and we are learning string"""
groovy>         println s1
groovy>     }
groovy> }
groovy> ]

This is groovy tutorial and we are learning string
```

12) Triple-double-quoted string



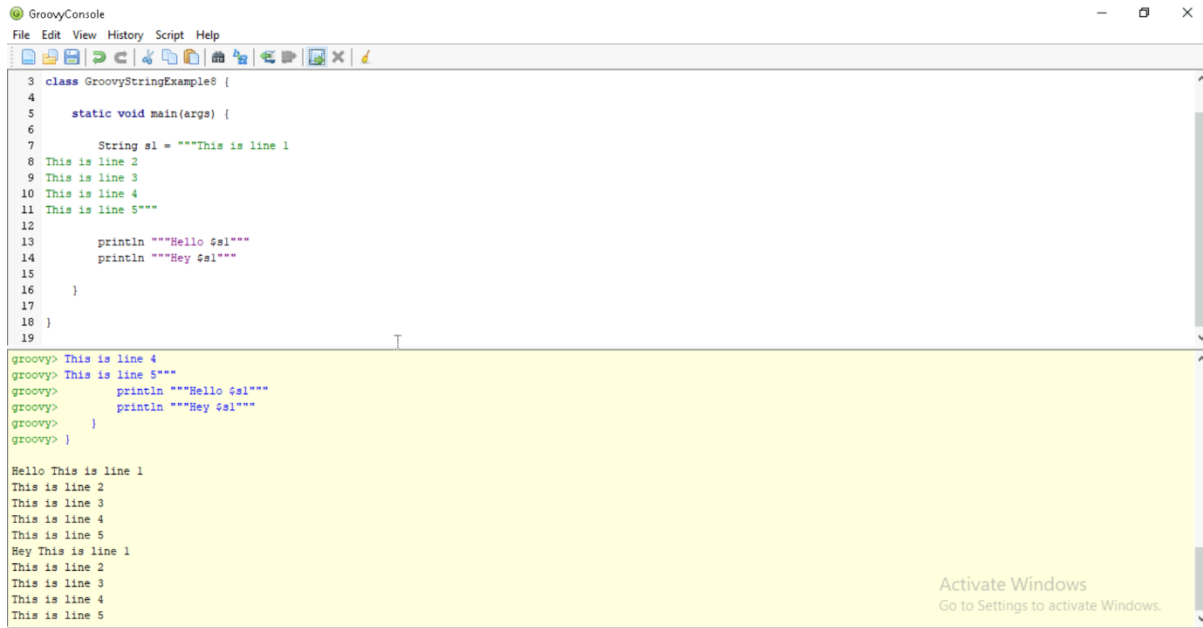
The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with a triple-double-quoted string. The bottom pane shows the output of the script, which prints the string on five lines.

```
2
3 class GroovyStringExample7 {
4
5     static void main(args) {
6
7         String s1 = """This is line 1
8 This is line 2
9 This is line 3
10 This is line 4
11 This is line 5"""
12
13         println s1
14
15     }
16 }
17
18
```

```
groovy> package com.app
groovy> class GroovyStringExample7 {
groovy>     static void main(args) {
groovy>         String s1 = """This is line 1
groovy> This is line 2
groovy> This is line 3
groovy> This is line 4
groovy> This is line 5"""
groovy>         println s1
groovy>     }
groovy> }
groovy> ]

This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
```

13) Triple-double-quoted string



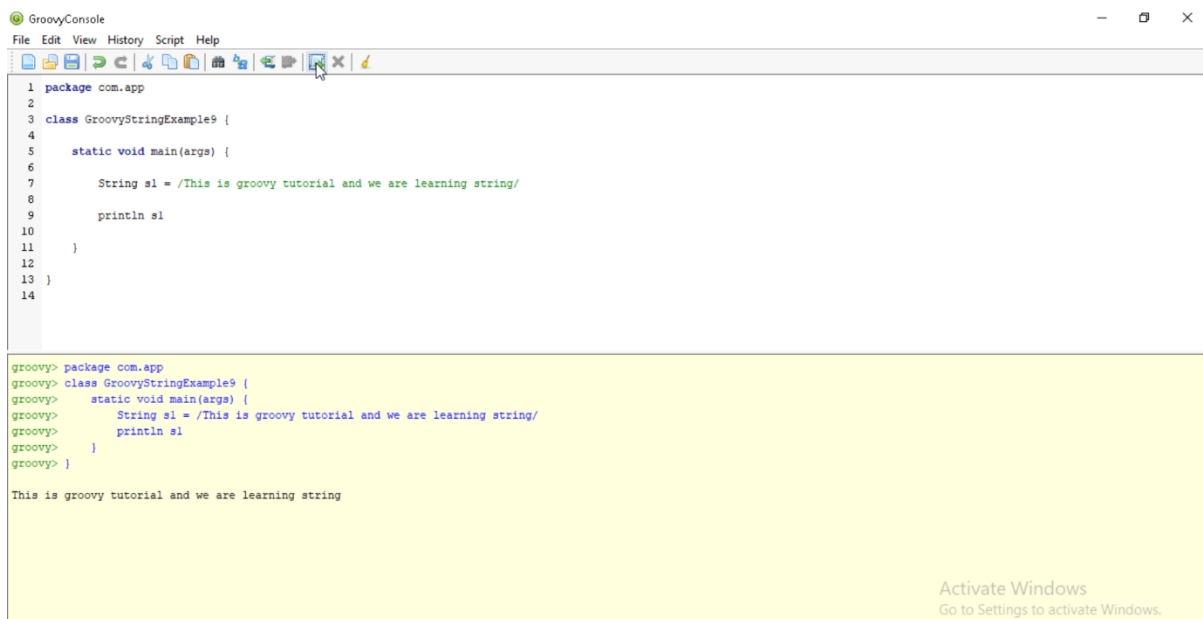
The screenshot shows the GroovyConsole application. The top pane contains a Groovy script for `GroovyStringExample8`. The script defines a `main` method that prints a multi-line string and two formatted strings. The bottom pane shows the output of the script, which matches the code's intent.

```
3 class GroovyStringExample8 {
4
5     static void main(args) {
6
7         String s1 = """This is line 1
8 This is line 2
9 This is line 3
10 This is line 4
11 This is line 5"""
12
13         println ""Hello $s1""
14         println ""Hey $s1""
15     }
16 }
17
18
19
```

```
groovy> This is line 4
groovy> This is line 5"""
groovy>         println ""Hello $s1""
groovy>         println ""Hey $s1""
groovy>     }
groovy> ]

Hello This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
Hey This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
```

14) Slashy string



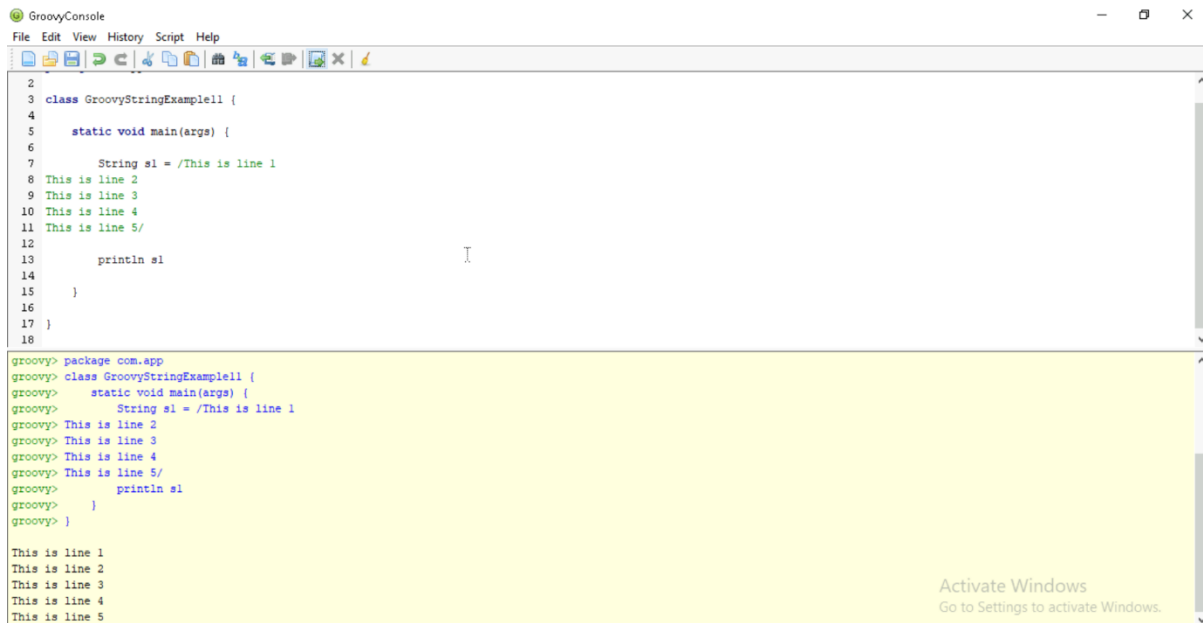
The screenshot shows the GroovyConsole application. The top pane contains a Groovy script for `GroovyStringExample9`. The script defines a `main` method that prints a string using a slashy string. The bottom pane shows the output of the script, which is the string defined in the code.

```
1 package com.app
2
3 class GroovyStringExample9 {
4
5     static void main(args) {
6
7         String s1 = /This is groovy tutorial and we are learning string/
8
9         println s1
10     }
11 }
12
13
14
```

```
groovy> package com.app
groovy> class GroovyStringExample9 {
groovy>     static void main(args) {
groovy>         String s1 = /This is groovy tutorial and we are learning string/
groovy>         println s1
groovy>     }
groovy> }
```

```
This is groovy tutorial and we are learning string
```

15) Slashy string



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with a class `GroovyStringExample11` and a `main` method. The script defines a `String s1` using a slashy string `/"This is line 1` followed by four lines of text. The bottom pane shows the output of the script, which prints each line of the string separately. An "Activate Windows" watermark is visible in the bottom right corner.

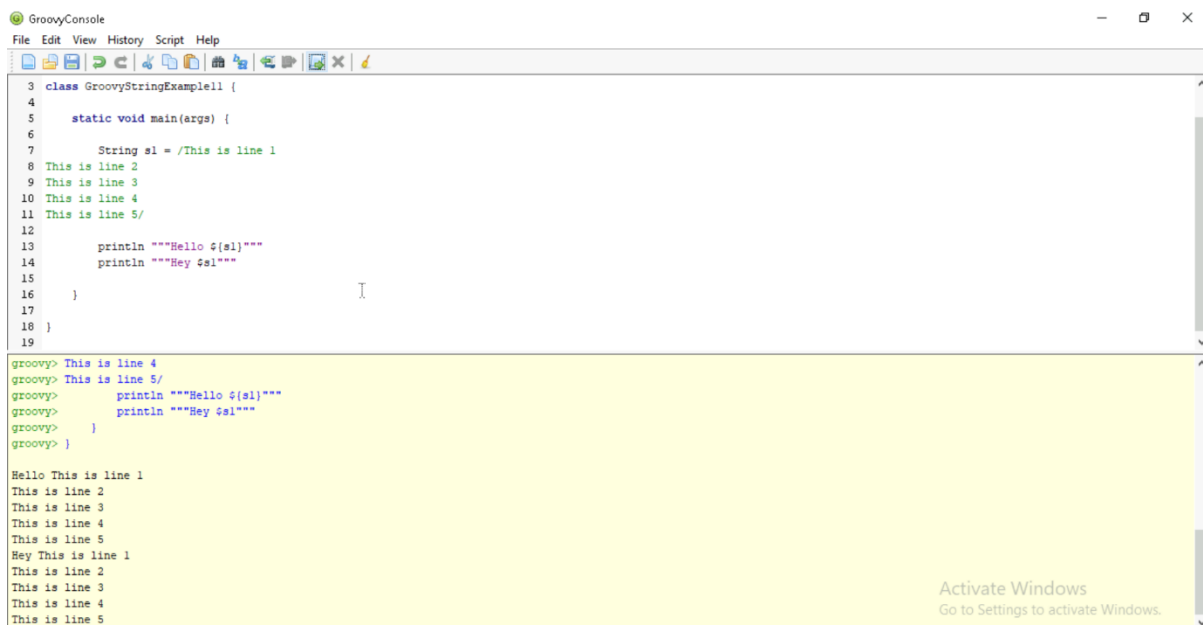
```
2
3 class GroovyStringExample11 {
4
5     static void main(args) {
6
7         String s1 = /"This is line 1
8 This is line 2
9 This is line 3
10 This is line 4
11 This is line 5/
12
13         println s1
14
15     }
16
17 }
18
```

```
groovy> package com.app
groovy> class GroovyStringExample11 {
groovy>     static void main(args) {
groovy>         String s1 = /"This is line 1
groovy> This is line 2
groovy> This is line 3
groovy> This is line 4
groovy> This is line 5/
groovy>         println s1
groovy>     }
groovy> }

This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
```

Activate Windows
Go to Settings to activate Windows.

16) Slashy string



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with a class `GroovyStringExample11` and a `main` method. The script defines a `String s1` using a slashy string `/"This is line 1` followed by four lines of text. The bottom pane shows the output of the script, which prints each line of the string separately. An "Activate Windows" watermark is visible in the bottom right corner.

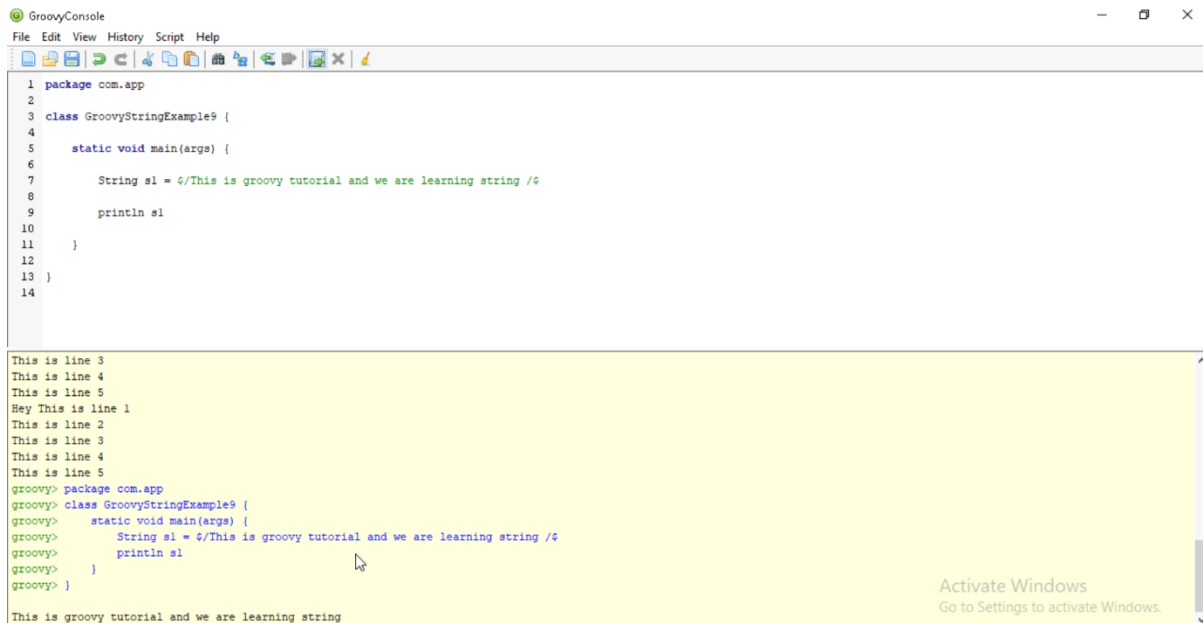
```
3 class GroovyStringExample11 {
4
5     static void main(args) {
6
7         String s1 = /"This is line 1
8 This is line 2
9 This is line 3
10 This is line 4
11 This is line 5/
12
13         println ""Hello ${s1}""
14         println ""Hey ${s1}""
15
16     }
17
18 }
19
```

```
groovy> This is line 4
groovy> This is line 5/
groovy>         println ""Hello ${s1}""
groovy>         println ""Hey ${s1}""
groovy>     }
groovy> }

Hello This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
Hey This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
```

Activate Windows
Go to Settings to activate Windows.

17) Dollar slashy string



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with the following code:

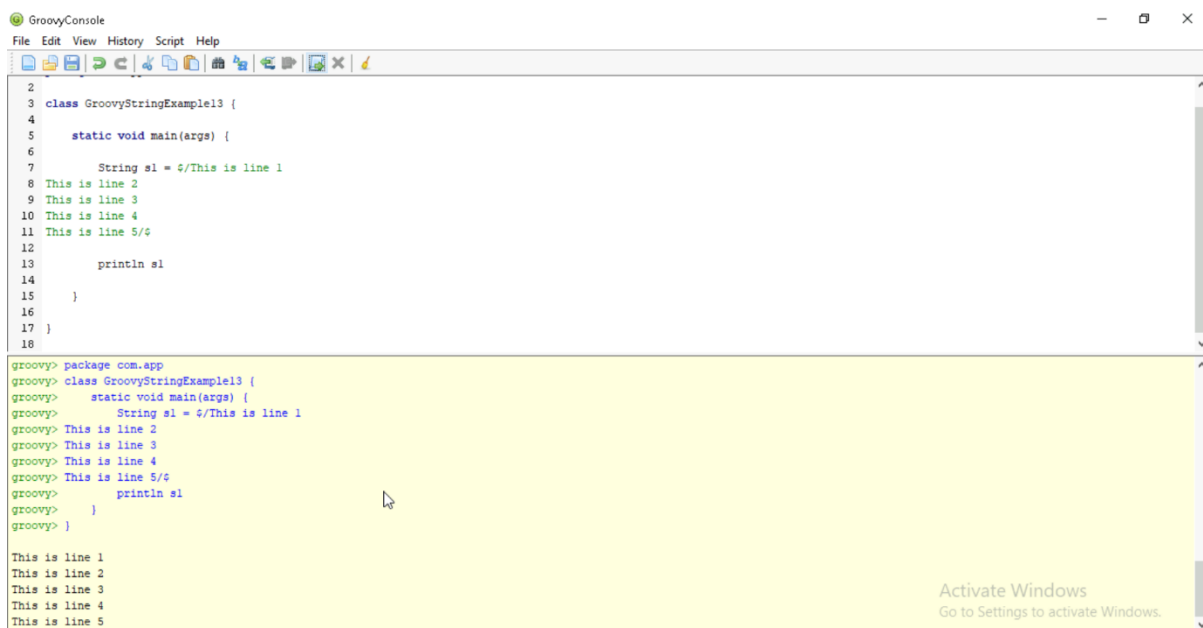
```
1 package com.app
2
3 class GroovyStringExample9 {
4
5     static void main(args) {
6
7         String s1 = "/This is groovy tutorial and we are learning string /"
8
9         println s1
10
11     }
12 }
13
14
```

The bottom pane shows the output of the script:

```
This is line 3
This is line 4
This is line 5
Hey This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
groovy> package com.app
groovy> class GroovyStringExample9 {
groovy>     static void main(args) {
groovy>         String s1 = "/This is groovy tutorial and we are learning string /"
groovy>         println s1
groovy>     }
groovy> }
This is groovy tutorial and we are learning string
```

An "Activate Windows" watermark is visible in the bottom right corner of the console window.

18) Dollar slashy string



The screenshot shows the GroovyConsole application. The top pane contains a Groovy script with the following code:

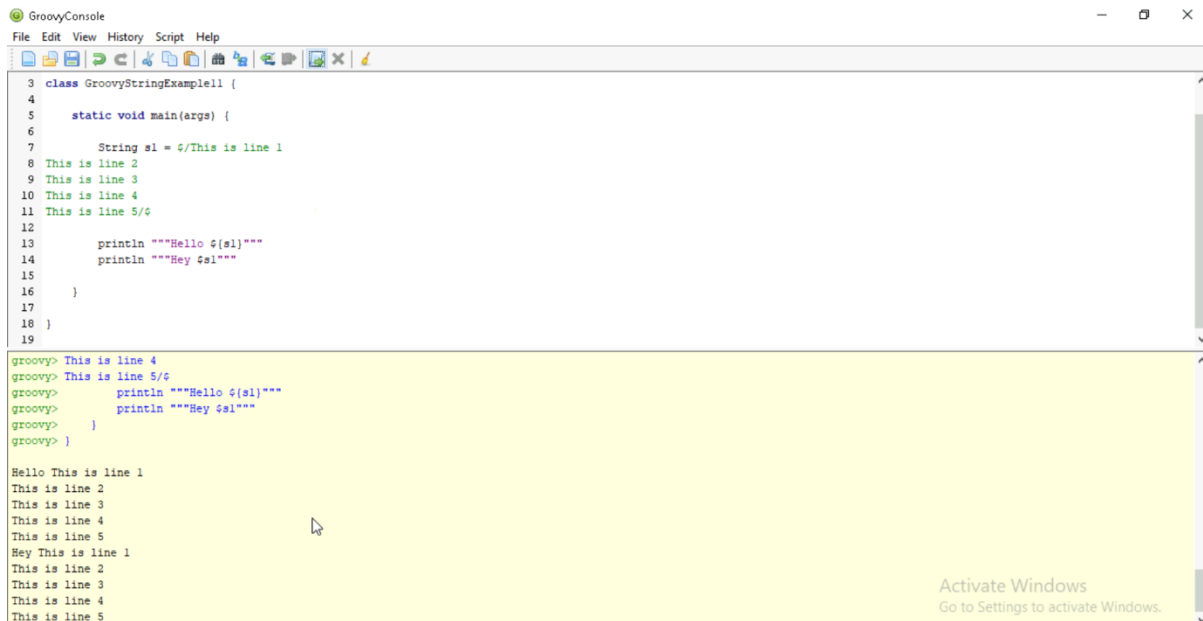
```
2
3 class GroovyStringExample13 {
4
5     static void main(args) {
6
7         String s1 = "/This is line 1
8 This is line 2
9 This is line 3
10 This is line 4
11 This is line 5/"
12
13         println s1
14
15     }
16 }
17
18
```

The bottom pane shows the output of the script:

```
groovy> package com.app
groovy> class GroovyStringExample13 {
groovy>     static void main(args) {
groovy>         String s1 = "/This is line 1
groovy> This is line 2
groovy> This is line 3
groovy> This is line 4
groovy> This is line 5/"
groovy>         println s1
groovy>     }
groovy> }
This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
```

An "Activate Windows" watermark is visible in the bottom right corner of the console window.

19) Dollar slashy string



The screenshot shows the GroovyConsole application window. The top pane contains a Groovy script defining a class `GroovyStringExample11` with a `main` method. The script uses dollar slashy strings (`$/`) to define a string `s1` and to format output. The bottom pane shows the output of the script, which includes the content of `s1` and the formatted strings.

```
3 class GroovyStringExample11 {
4
5     static void main(args) {
6
7         String s1 = $/This is line 1
8         This is line 2
9         This is line 3
10        This is line 4
11        This is line 5/$
12
13        println ""Hello ${s1}""
14        println ""Hey ${s1}""
15    }
16 }
17
18 }
19
```

```
groovy> This is line 4
groovy> This is line 5/$
groovy>         println ""Hello ${s1}""
groovy>         println ""Hey ${s1}""
groovy>     }
groovy> ]
groovy> ]

Hello This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
Hey This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
```

Case Study

Web Scraping with Groovy & Jsoup

Description:

Build a Groovy script that scrapes data from a website and saves it in a CSV file.

Solution:

- Use Jsoup to fetch and parse HTML content.
- Extract required elements using CSS selectors.
- Write the extracted data to a CSV file.

Tech Stack: Groovy, Jsoup, CSV Parsing

Use Case: Data Scraping, Automation

```

1 @Grab('org.jsoup:jsoup:1.16.2')
2
3 import org.jsoup.Jsoup
4 import java.nio.file.*
5
6 def url = "https://example.com"
7 def doc = Jsoup.connect(url).get()
8
9
10 def titles = doc.select("h2")
11 def csvContent = new StringBuilder("UST is a global digital technology company that provides IT and engineering services and solutions. They partner with clients to help them tr
12 titles.each { title ->
13     csvContent.append("${title.text()}\n")
14 }
15
16 def filePath = Paths.get("scraped_data.csv")
17 Files.write(filePath, csvContent.toString().getBytes())
18
19 println "Scraping complete! Data saved to scraped_data.csv"
20

```

Scraping complete! Data saved to scraped_data.csv

```

C:\Users\Administrator\Desktop>groovy -version
Groovy Version: 5.0.0-alpha-11 JVM: 11.0.15.1 Vendor: Oracle Corporation OS: Windows 10

C:\Users\Administrator\Desktop>groovyConsole
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.vmplugin.v9.Java9 (file:/C:/Program%20Files%20(x86)/Groovy/lib/groovy-5.0.0-alpha-11.jar) to method
java.awt.dnd.DropTargetContext.isDataFlavorSupported(java.awt.datatransfer.DataFlavor)
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.vmplugin.v9.Java9
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Terminate batch job (Y/N)? y

C:\Users\Administrator\Desktop>cd C:\Users\Administrator

C:\Users\Administrator>cd C:\Users\Administrator\Desktop

C:\Users\Administrator\Desktop>dir scraped_data.csv
Volume in drive C has no label.
Volume Serial Number is 5600-05A6

Directory of C:\Users\Administrator\Desktop

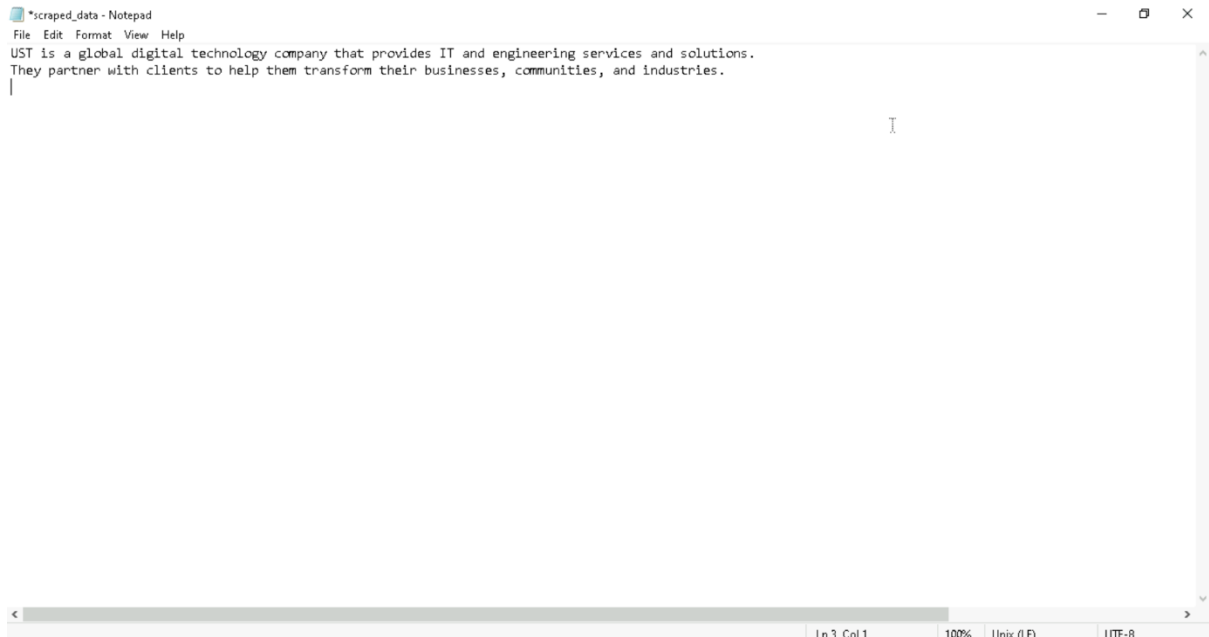
17-02-2025  17:43                197 scraped_data.csv
               1 File(s)                197 bytes
               0 Dir(s)  71,232,761,056 bytes free

C:\Users\Administrator\Desktop>notepad scraped_data.csv

C:\Users\Administrator\Desktop>_

```

Output:



Groovy Script for Log File Analysis

Description:

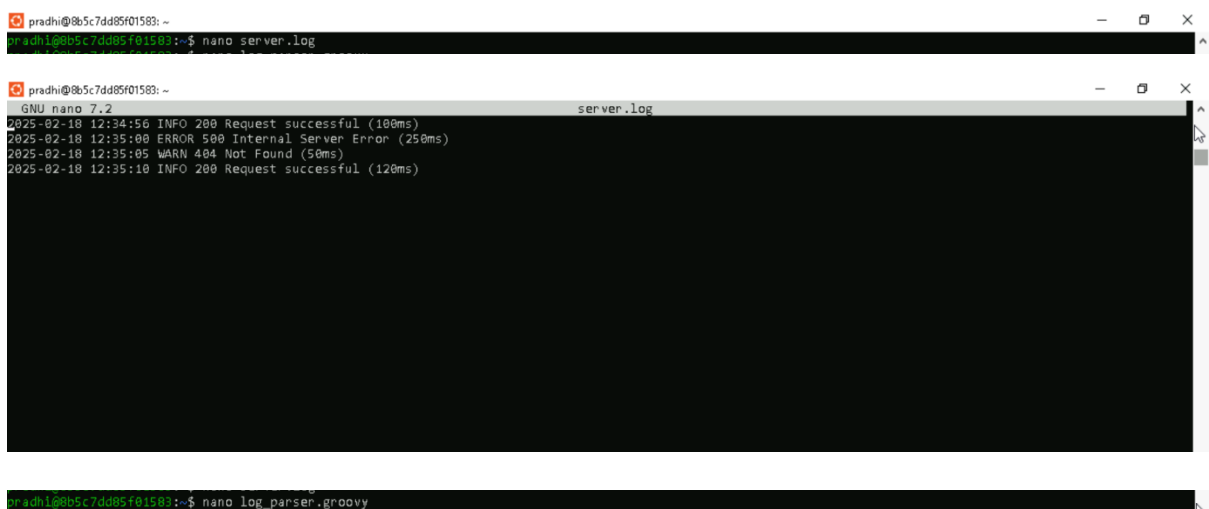
Parse and analyze server log files to extract useful insights like error counts, request trends, and response times.

Solution:

- Read a log file line by line.
- Use regex to extract timestamp, status codes, and messages.
- Aggregate and summarize the data.
- Generate a report.

Tech Stack: Groovy, Regex, File I/O

Use Case: DevOps, System Monitoring



```
pradhi@8b5c7dd85f01583: ~
GNU nano 7.2 log_parser.groovy
import java.nio.file.Files
import java.nio.file.Paths
import java.util.regex.*

def logFilePath = "server.log" // Change this to your actual log file path

// Regex pattern to extract timestamp, status code, and message
def logPattern = ~/(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}) .* (\d{3}) (.+)/

def errorCount = [:].withDefault { 0 }
def statusCount = [:].withDefault { 0 }
def responseTimes = []

println "Processing log file: $logFilePath"

// Read file line by line
Files.lines(Paths.get(logFilePath)).each { line ->
    def matcher = logPattern.matcher(line)
    if (matcher.find()) {
        def timestamp = matcher.group(1)
        def statusCode = matcher.group(2)
        def message = matcher.group(3)

        // Count status codes
        statusCount[statusCode]++

        // Count errors (4xx and 5xx)
        if (statusCode.startsWith("4") || statusCode.startsWith("5")) {
            errorCount[statusCode]++
        }

        // Simulate extracting response time (if available in the log message)
        def responseTimeMatcher = message =~ /(\d+)ms/
        if (responseTimeMatcher.find()) {
            responseTimes << responseTimeMatcher.group(1).toInteger()
        }
    }
}

[ Read 50 lines ]
[ Undo ] [ Redo ] [ Cut ] [ Paste ] [ Copy ] [ Paste ] [ Read 50 lines ]
[ Execute ] [ Justify ] [ Location ] [ Go To Line ] [ M-U Undo ] [ M-E Redo ]
[ M-A Set Mark ] [ M-G Copy ] [ M-I To Bracket ] [ M-O Where Was ]
[ M-X Exit ] [ M-W Write Out ] [ M-R Read File ] [ M-V Where Is ] [ M-N Replace ]
[ M-C Cut ] [ M-P Paste ] [ M-J Justify ] [ M-L Location ] [ M-G Go To Line ]
[ M-U Undo ] [ M-E Redo ] [ M-A Set Mark ] [ M-G Copy ] [ M-I To Bracket ]
[ M-O Where Was ]
Type here to search
ENG 1601
US 18-02-2025
```

Output:

```
pradhi@8b5c7dd85f01583: ~$ groovy log_parser.groovy
Processing log file: server.log

--- Log Analysis Report ---
Total Requests: 4
Status Code Counts: [200:2, 500:1, 404:1]
Error Counts: [500:1, 404:1]
Average Response Time: 130 ms
pradhi@8b5c7dd85f01583: ~$
```