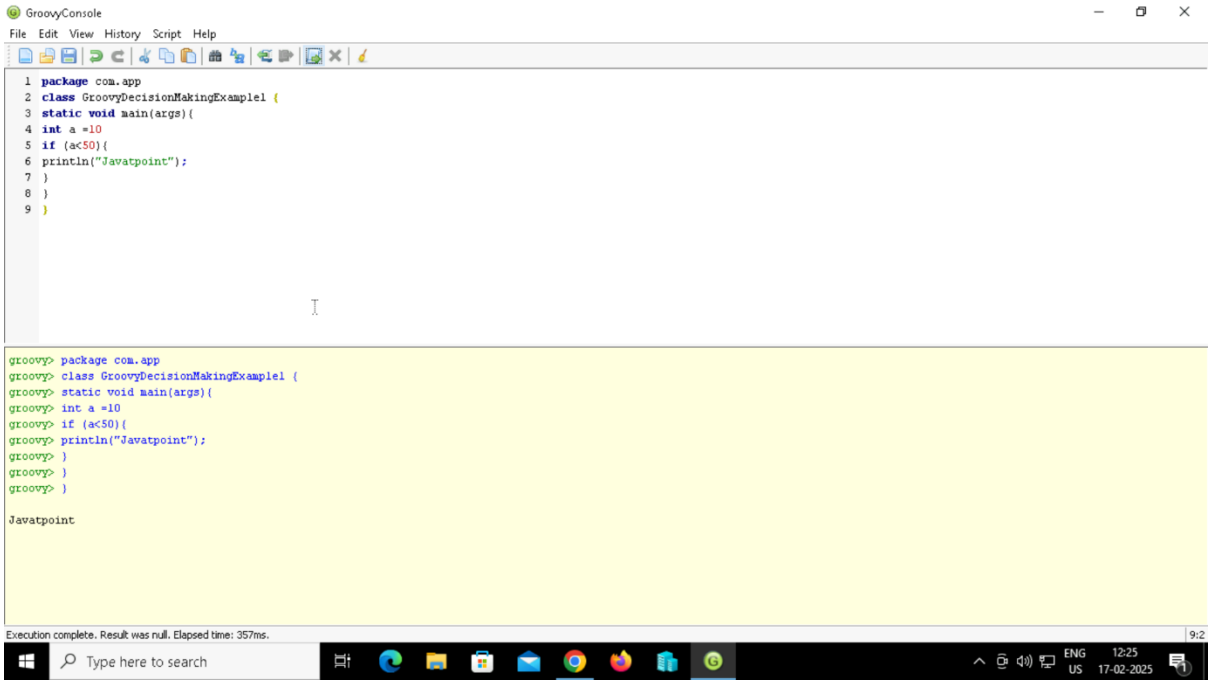
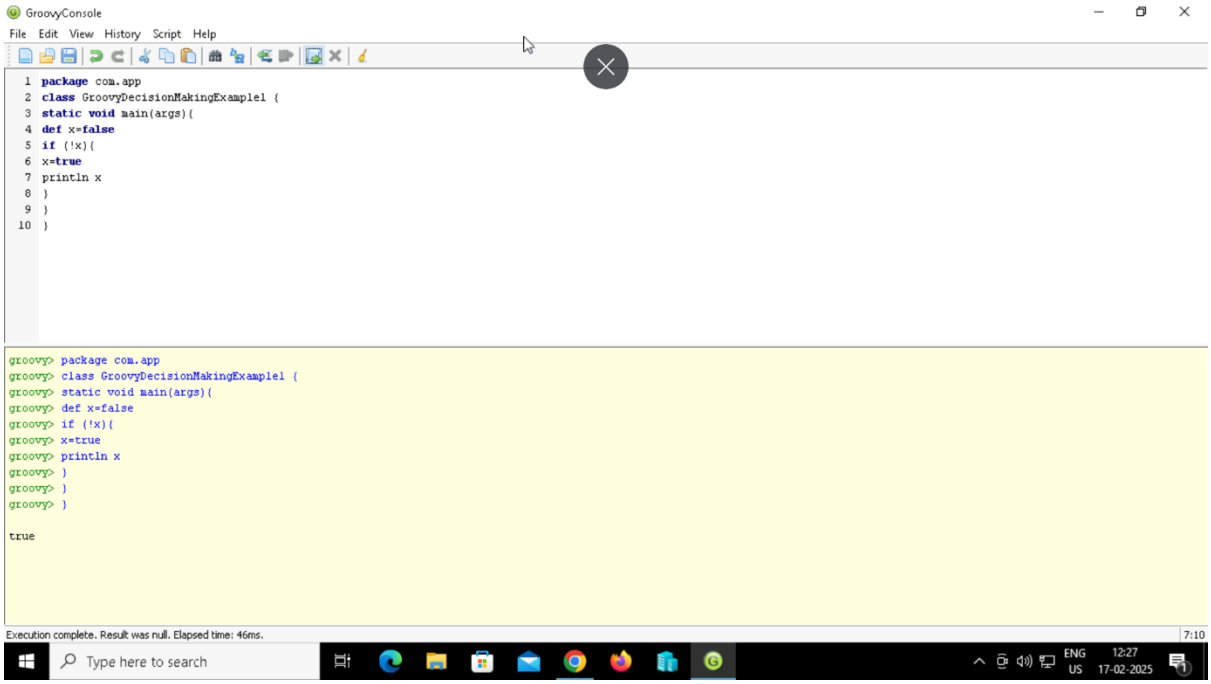
Groovy Project

If Statement

Example 1

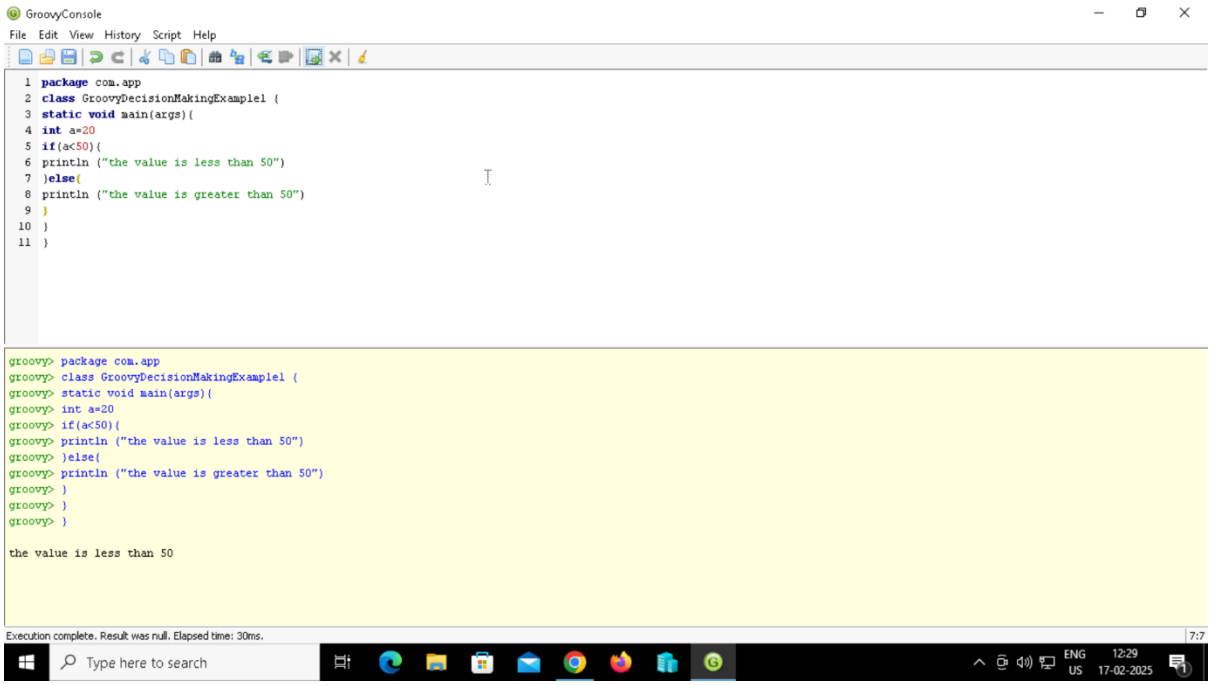


Example 2

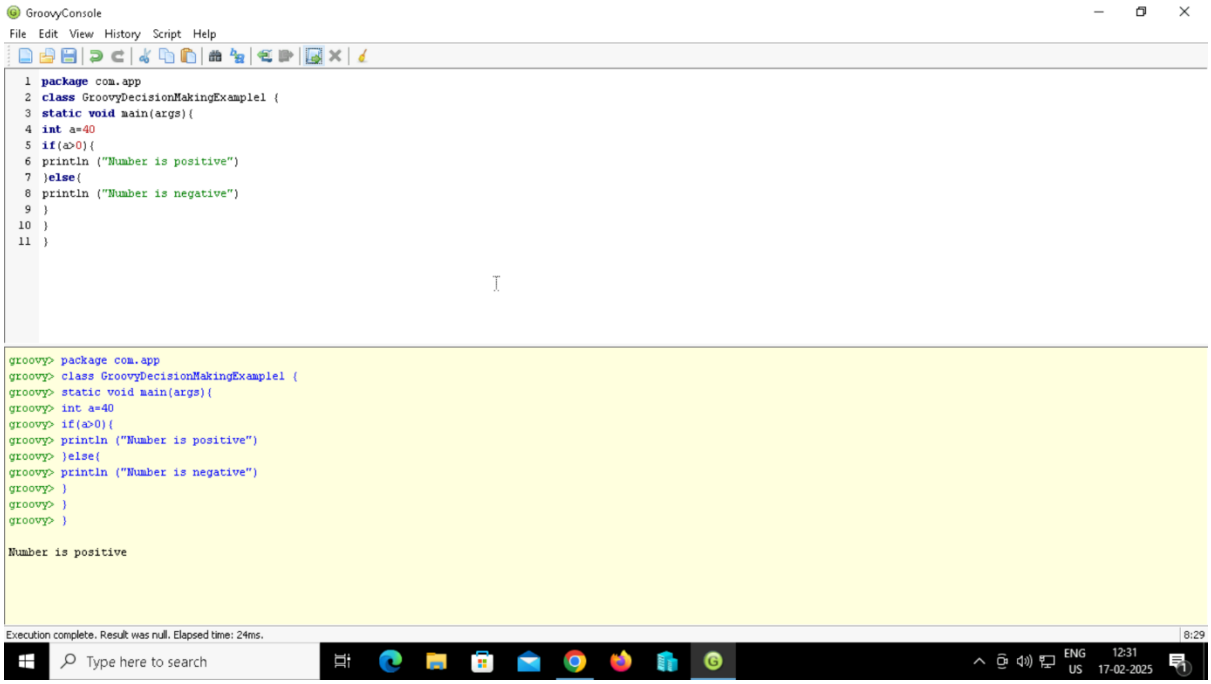


If else Statement

Example 3

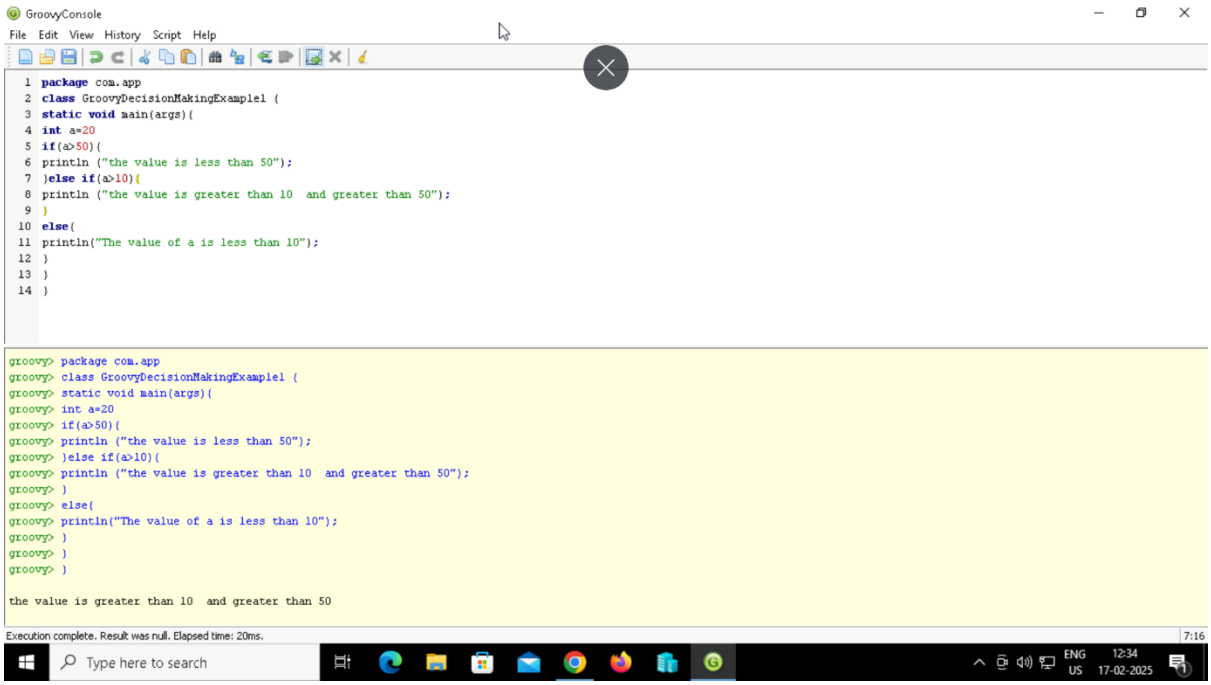


Example 4



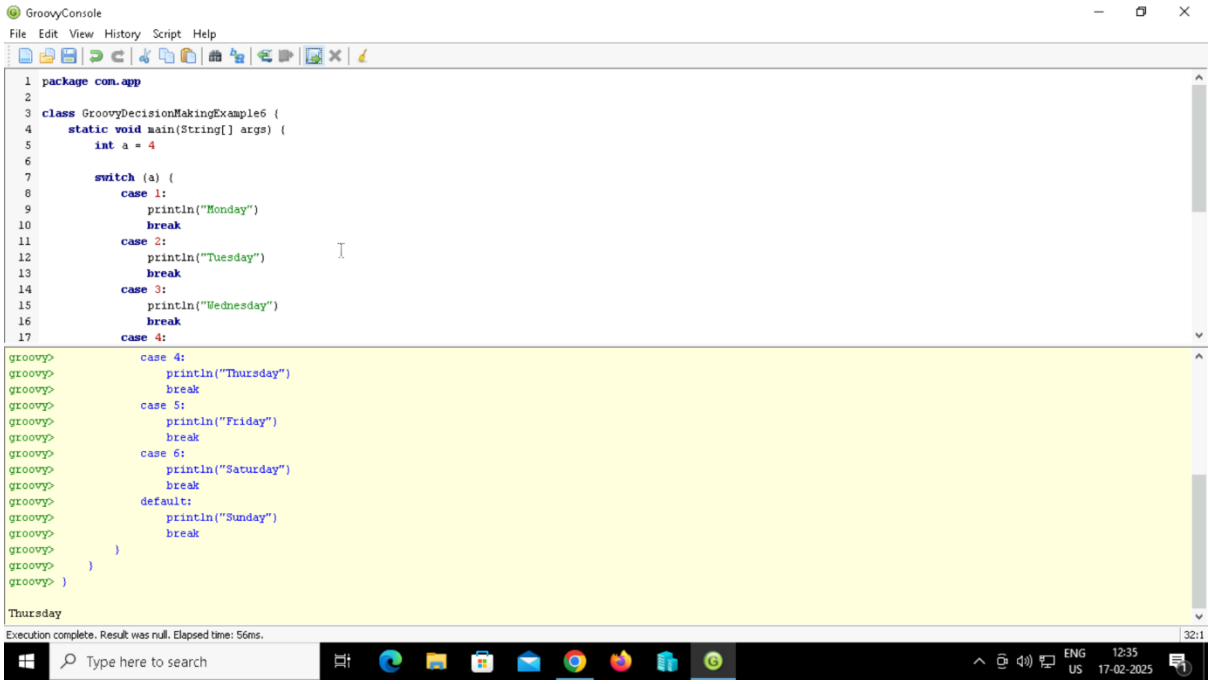
Nested If Statement

Example 5



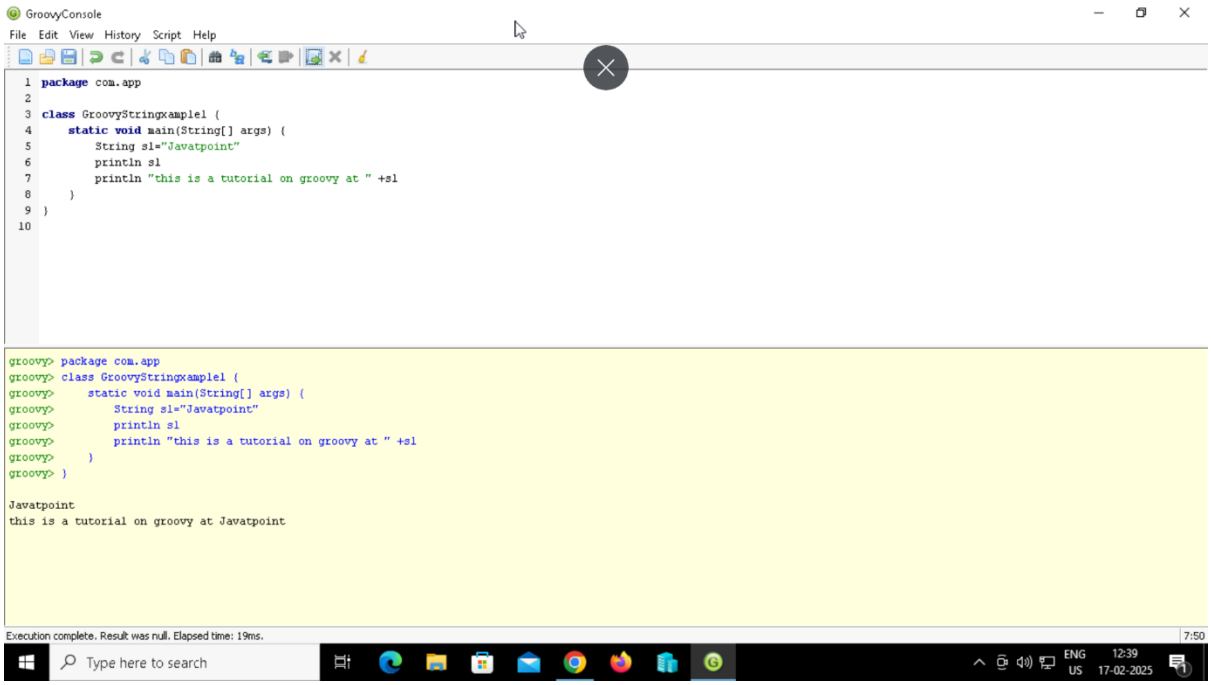
Switch Statement

Example 6



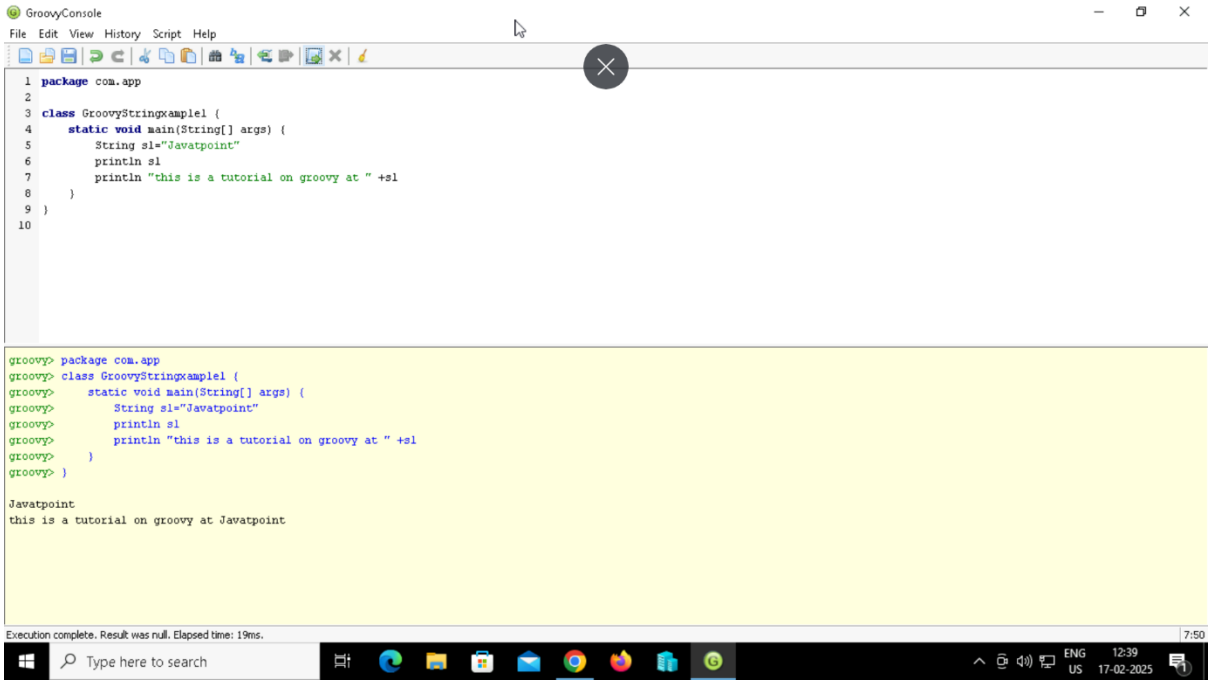
Single-quoted string

Example 1

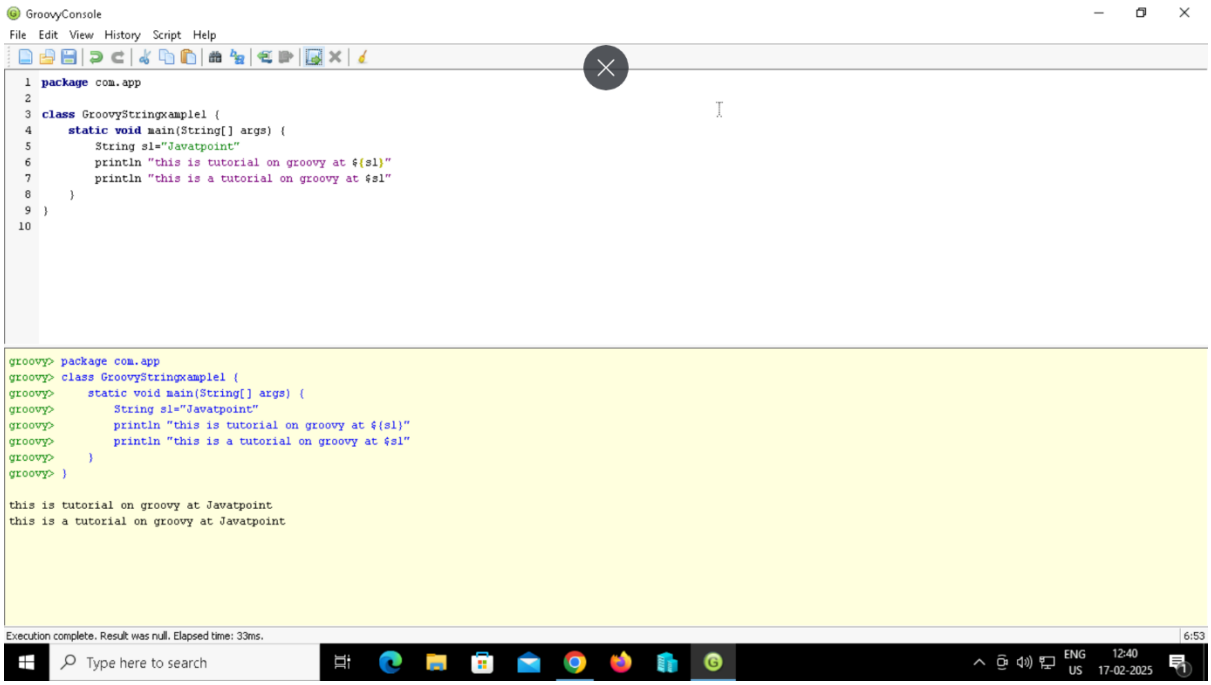


Double-quoted string

Example 2

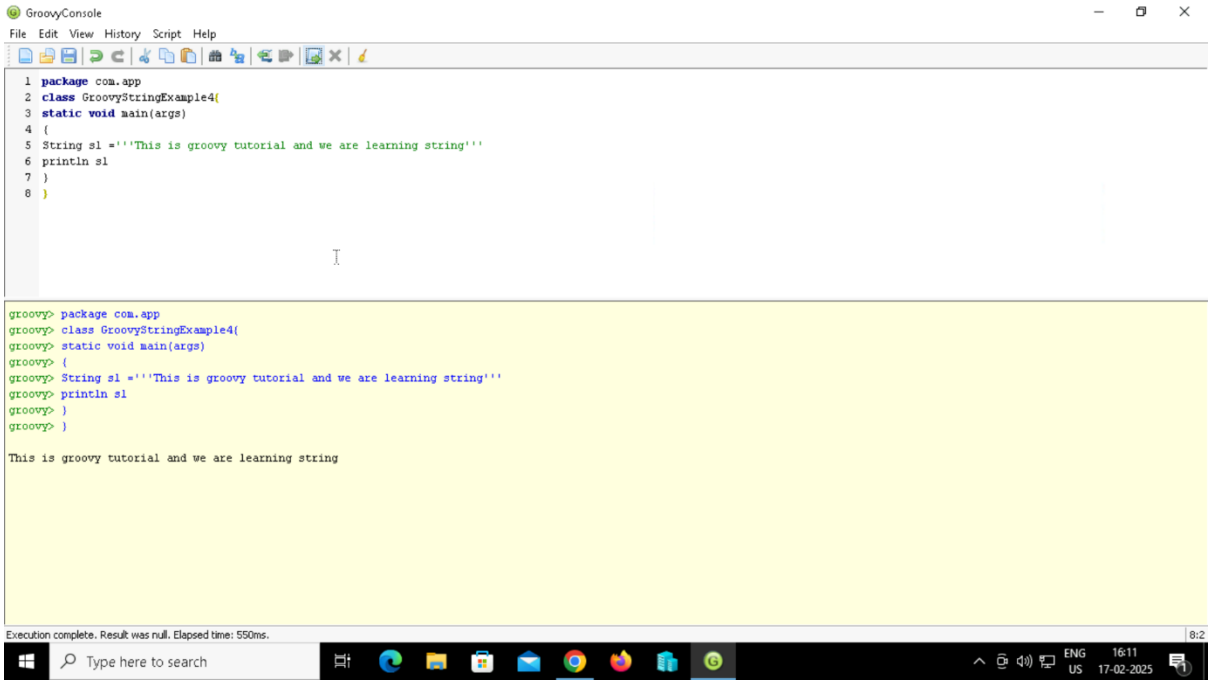


Example 3

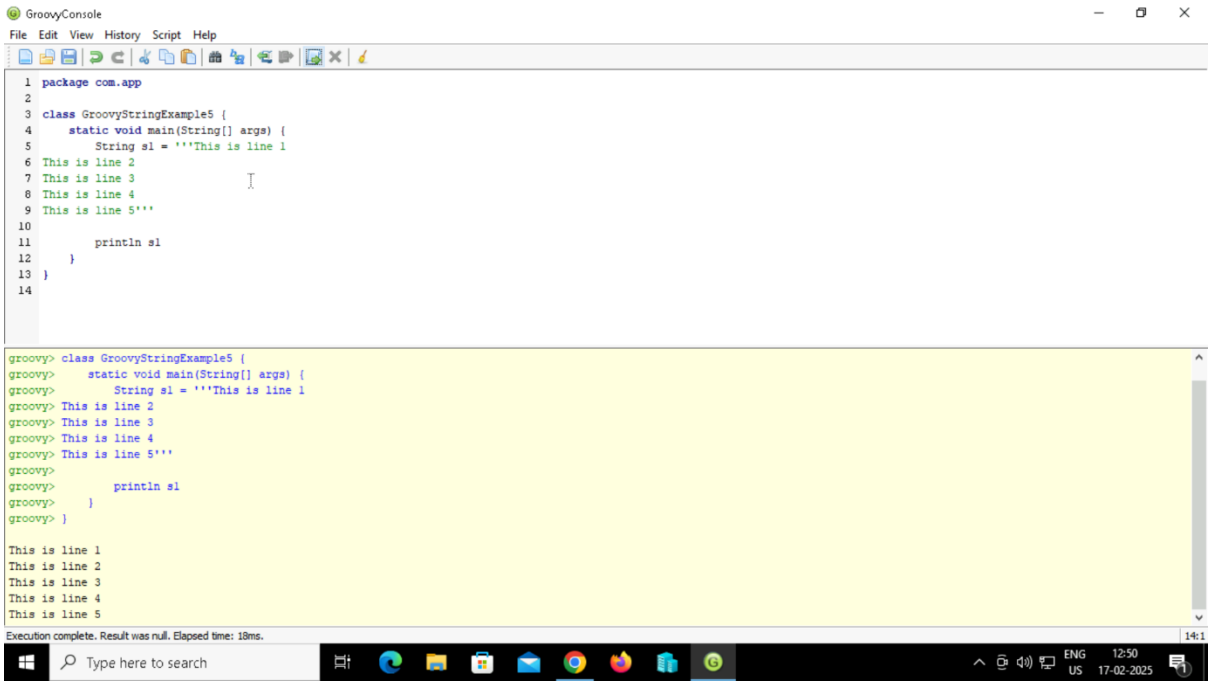


Triple -single-quoted string

Example 4

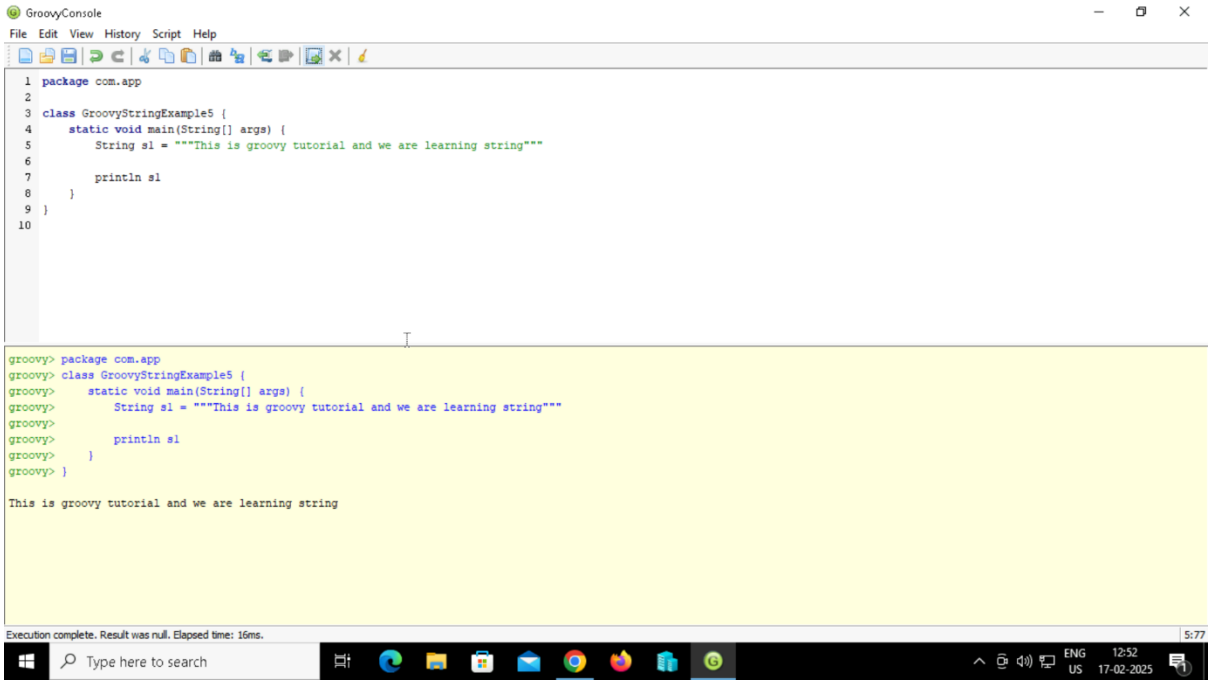


Example 5

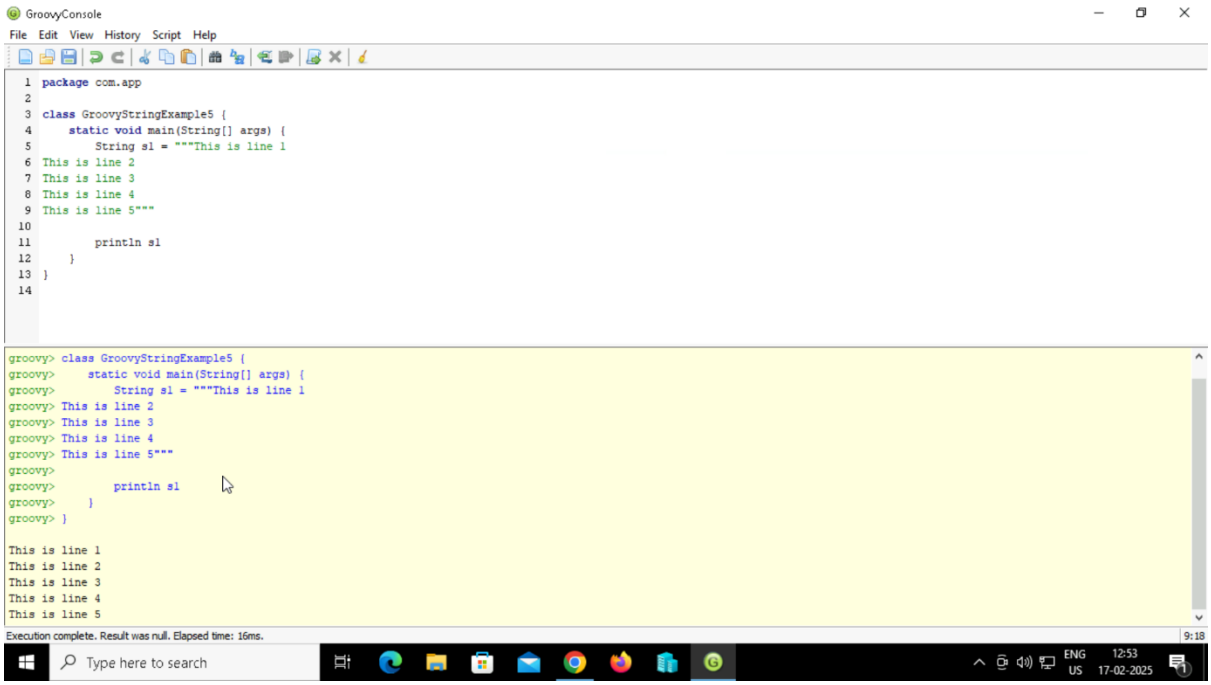


Triple-double-quoted string

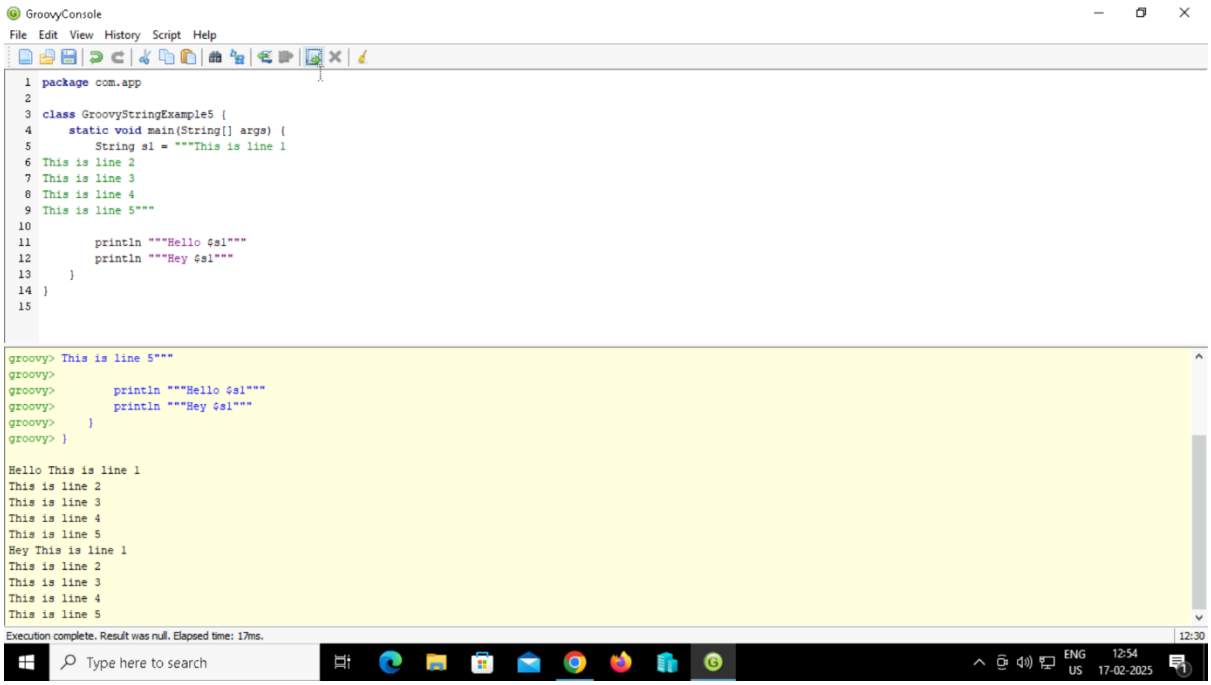
Example 6



Example 7

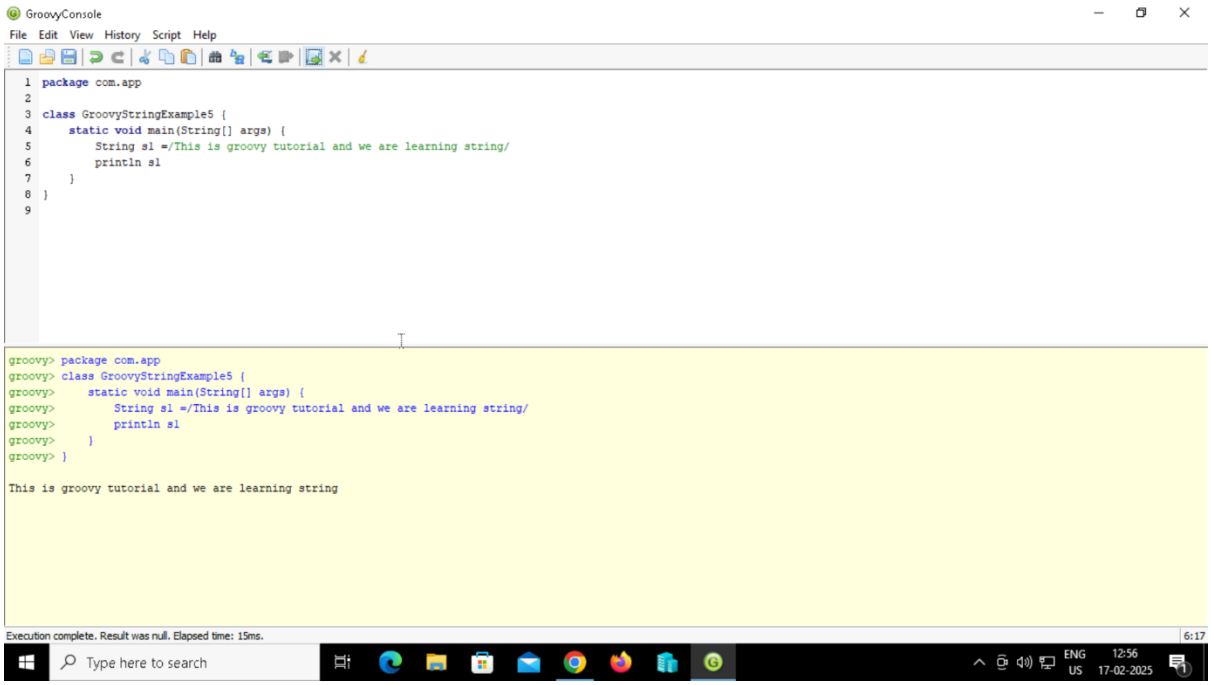


Example 8

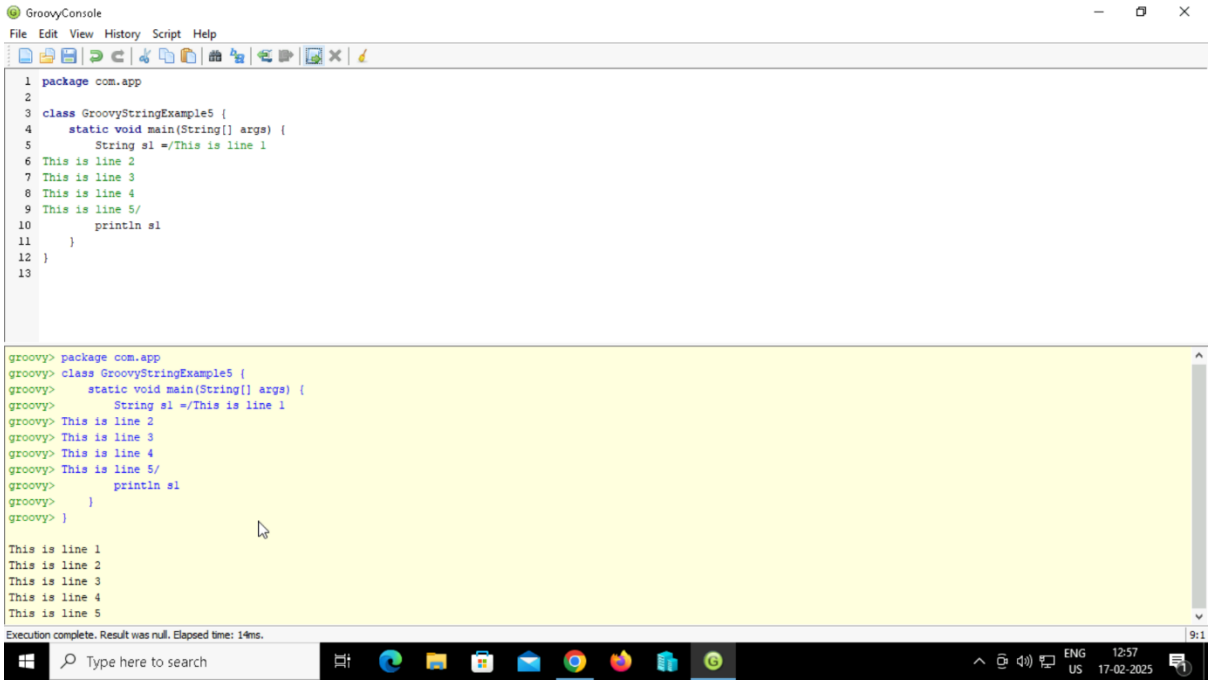


Slashy string

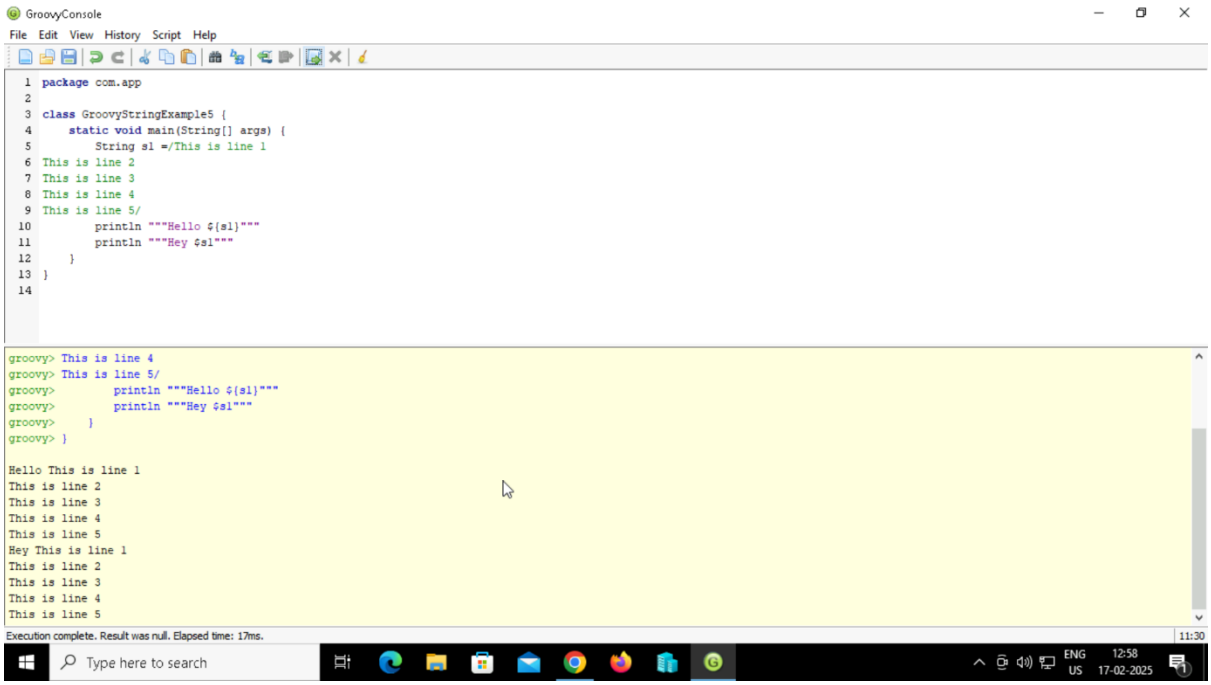
Example 9



Example 10

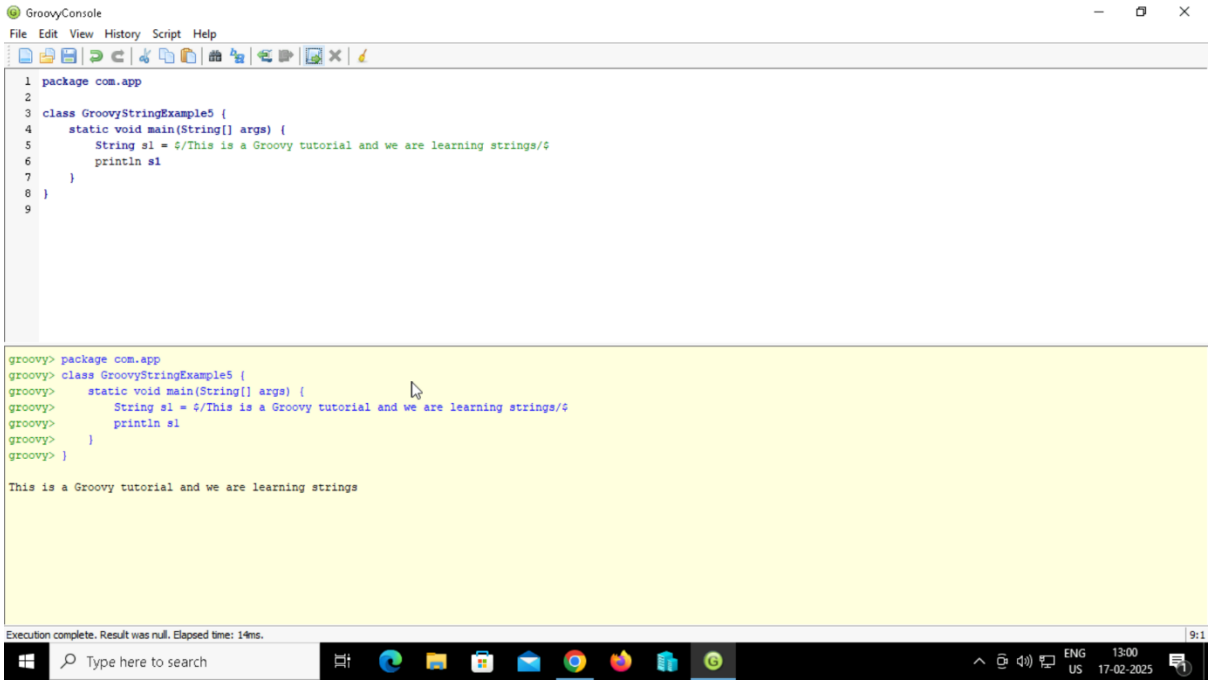


Example 11

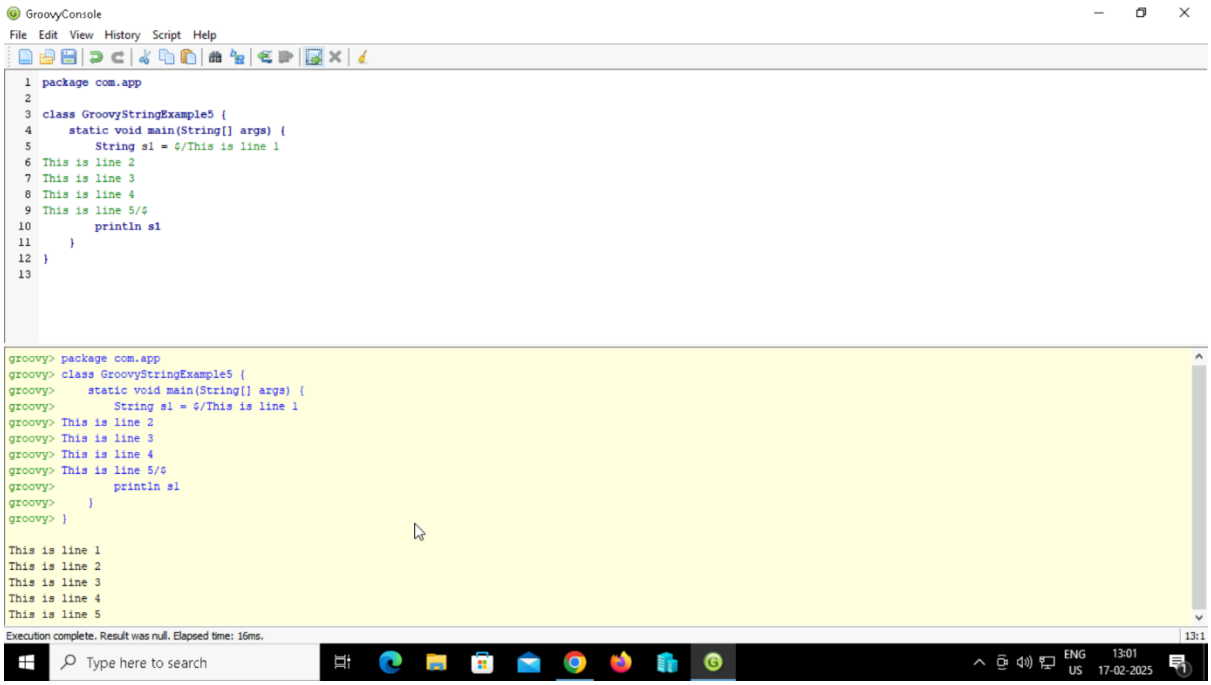


Dollar slashy string

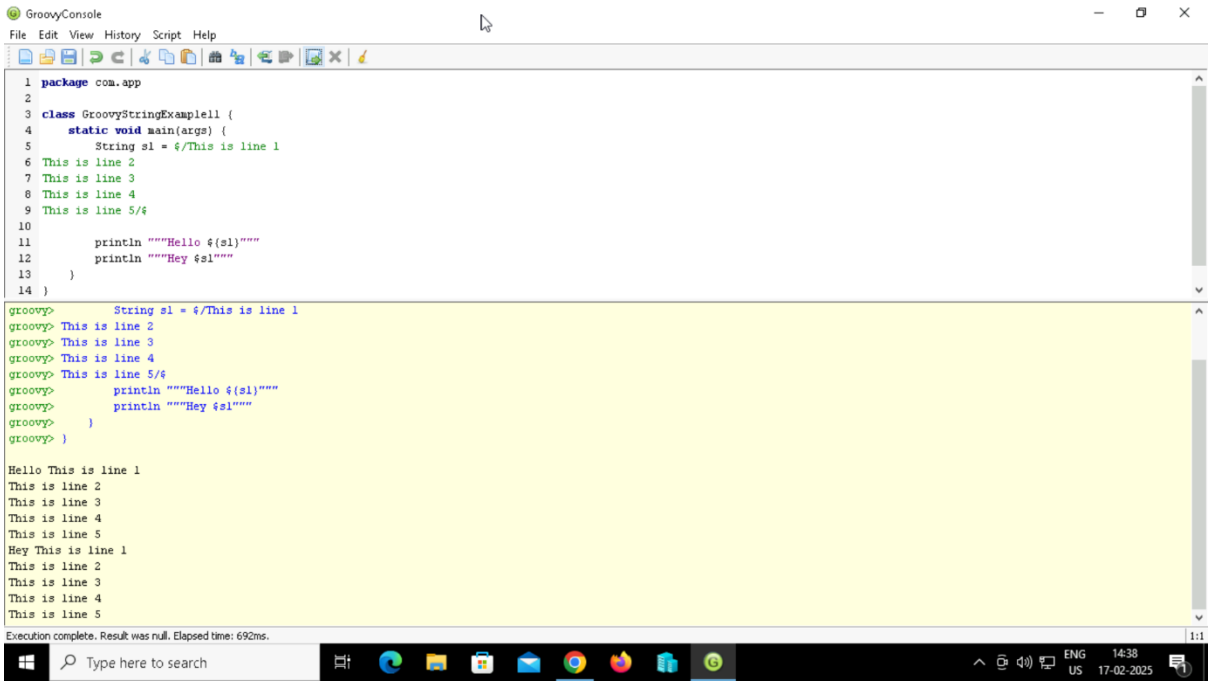
Example 12



Example 13



Example 14



Groovy Project

5. 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.

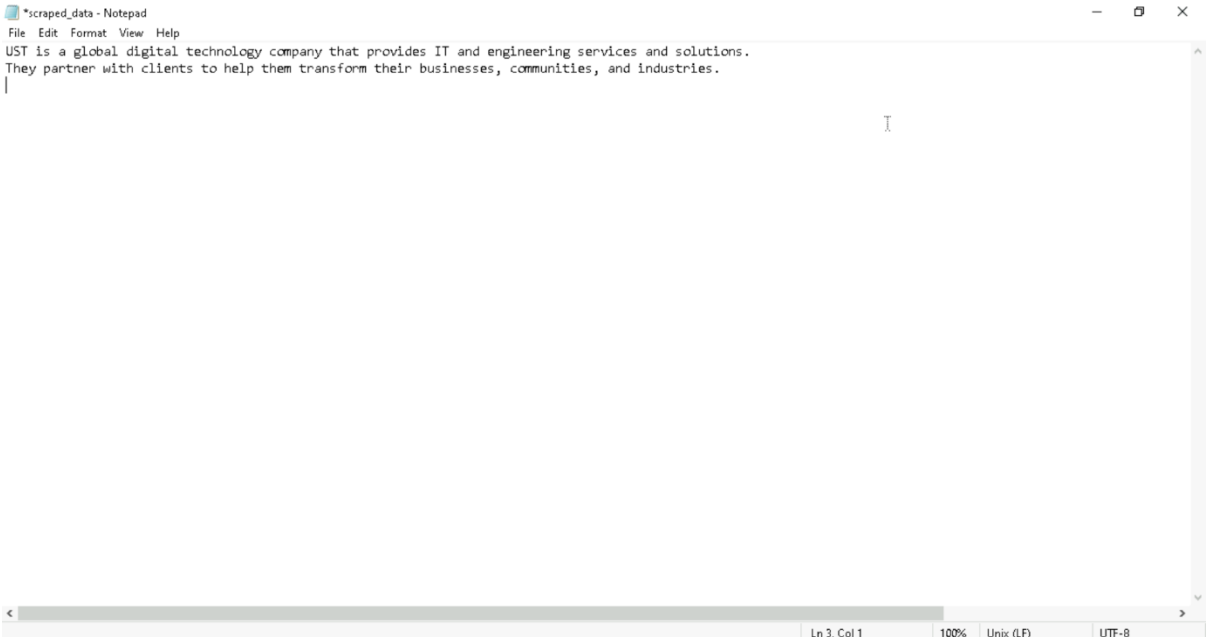
Tech Stack: Groovy, Jsoup, CSV Parsing
Use Case: Data Scraping, Automation

```groovy
@Grab('org.jsoup:jsoup:1.16.2')

import org.jsoup.Jsoup
import java.nio.file.*

def url = "https://example.com"
def doc = Jsoup.connect(url).get()


def titles = doc.select("h2")
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
titles.each { title ->
    csvContent.append("${title.text()}\n")
}

def filePath = Paths.get("scraped_data.csv")
Files.write(filePath, csvContent.toString().getBytes())

println "Scraping complete! Data saved to scraped_data.csv"

```

```
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)  71,232,761,856 bytes free

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

C:\Users\Administrator\Desktop>
```

Output:



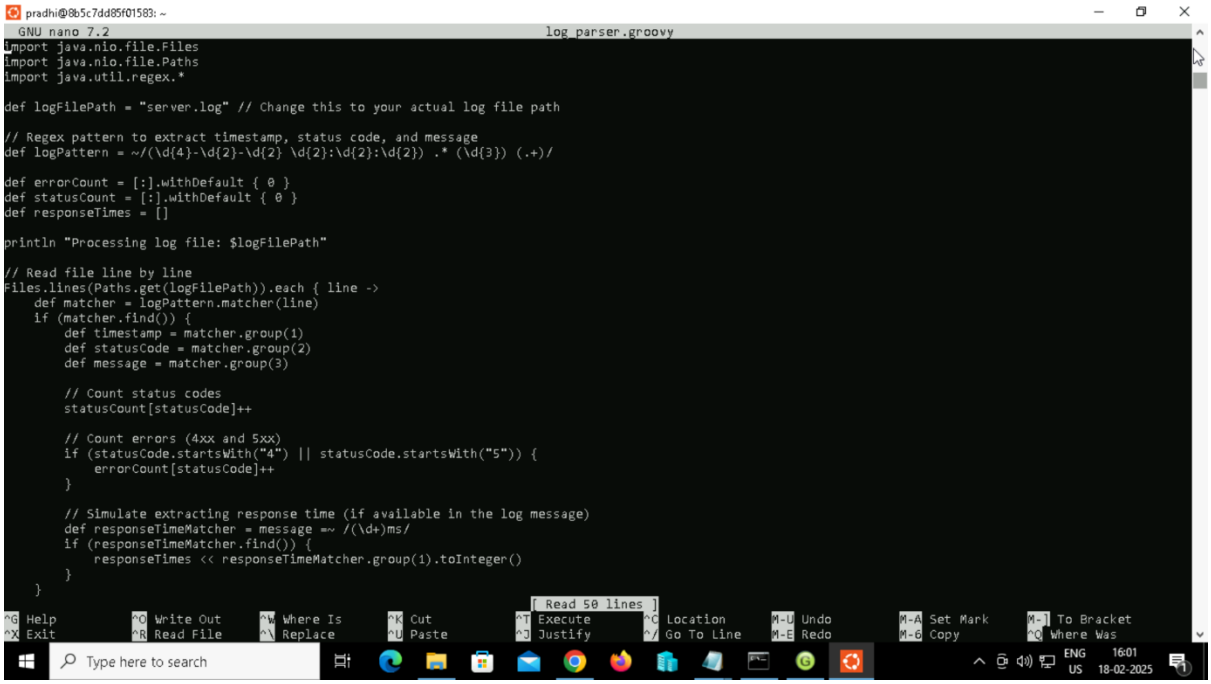
1. Groovy Script for Log File Analysis

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]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-] To Bracket
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy ^Q Where Was
Type here to search
ENG
US
16:01
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:~$