

SEPM EXP NO: 5

TO BUILD THE PIPELINE OF JOBS USING MAVEN / GRADLE / ANT IN JENKINS, CREATE A PIPELINE SCRIPT TO TEST AND DEPLOY AN APPLICATION OVER THE TOMCAT SERVER

Zidan Shaikh T22-96 AI&DS

Theory:

1. Introduction to Jenkins and CI/CD

Jenkins is an open-source automation server used for continuous integration and continuous deployment (CI/CD). It helps automate the software development lifecycle by building, testing, and deploying applications.

1.1 CI/CD Concepts

- Continuous Integration (CI): Automatically integrates code changes into a shared repository and runs tests.
- Continuous Deployment (CD): Automates the process of deploying applications to production or staging environments.

1.2 Tools Used

- Jenkins – Automation server.
- Maven/Gradle/Ant – Build automation tools.
- Tomcat – A web server for deploying Java applications.

2. Installing and Configuring Jenkins

1. Download and Install Jenkins:

- o Download from Jenkins official site.
- o Install and start the Jenkins service.
- o Access Jenkins at <http://localhost:8080/>.

2. Install Required Plugins:

- o Go to Manage Jenkins > Plugin Manager.
- o Install Pipeline, Maven Integration, and Deploy to Container plugins.

3. Creating a Jenkins Pipeline

Jenkins pipelines define a series of automated steps for building, testing, and deploying applications.

3.1 Steps to Create a Pipeline

1. Open Jenkins Dashboard and click on New Item.
2. Select Pipeline and provide a project name.
3. Click OK and navigate to the Pipeline section.
4. Write a Pipeline script (Declarative or Scripted) to define the build and deployment process.

4. Writing a Jenkins Pipeline Script

The following script builds a Java application using Maven and deploys it to Tomcat:

```
groovy CopyEdit pipeline {    agent any    stages    {        stage('Checkout Code') {            steps {                git                'https://github.com/your-repository.git'            }        }        stage('Build with Maven') {            steps {                sh 'mvn clean package'            }        }        stage('Deploy to Tomcat') {            steps {                deploy adapters:                [tomcat8(credentialsId: 'tomcat-cred', path: '', url:
```

```
'http://yourtomcat-server:8080')],      war:
'*/*.war'

    }
  }
}
}
```

5. Configuring Jenkins for Deployment

1. Configure Tomcat Server:

- o Install Tomcat and start the server.
- o Set up a user with deployment privileges in conf/tomcatusers.xml: xml CopyEdit

```
<role rolename="manager-gui"/>
```

```
<role rolename="manager-script"/>
```

```
<user username="admin" password="admin" roles="manager-gui,manager-script"/> o
```

Restart Tomcat.

2. Set Up Jenkins Credentials:

- o Go to Manage Jenkins > Credentials.
- o Add a username/password credential for Tomcat deployment.

3. Run the Pipeline in Jenkins:

- o Click Build Now to execute the pipeline.
- o Verify the deployment at <http://your-tomcat-server:8080/your-app>.

Example 1 :

Creating a job:

Start building your software project


Create a job





Naming the job and setting it as freestyle:


Enter an item name


» Required field


**Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

**Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

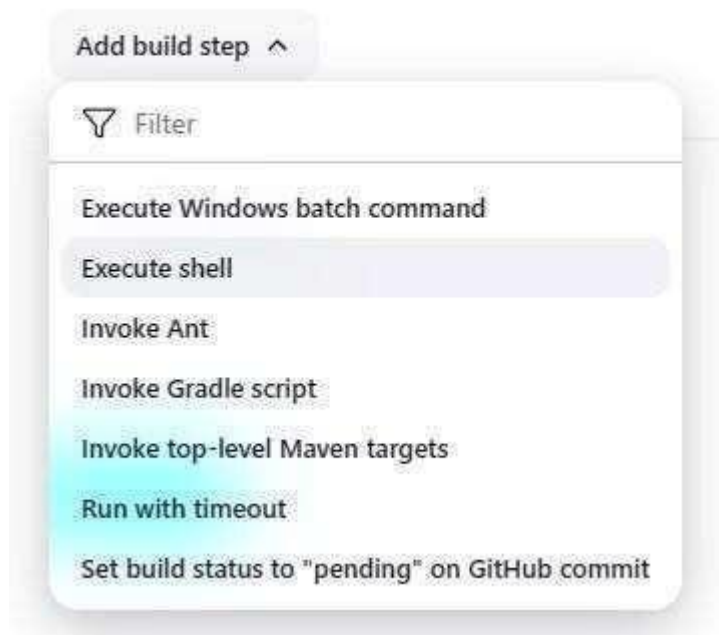
**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

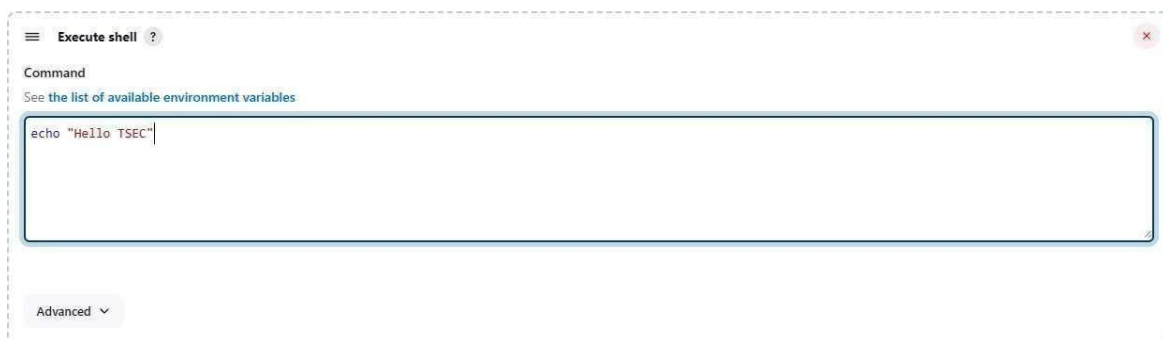
Selecting build type as “Execute shell”:

Build Steps



Entering a simple command for the shell execution:

Build Steps



Applying and saving the project configuration:



Building the project:

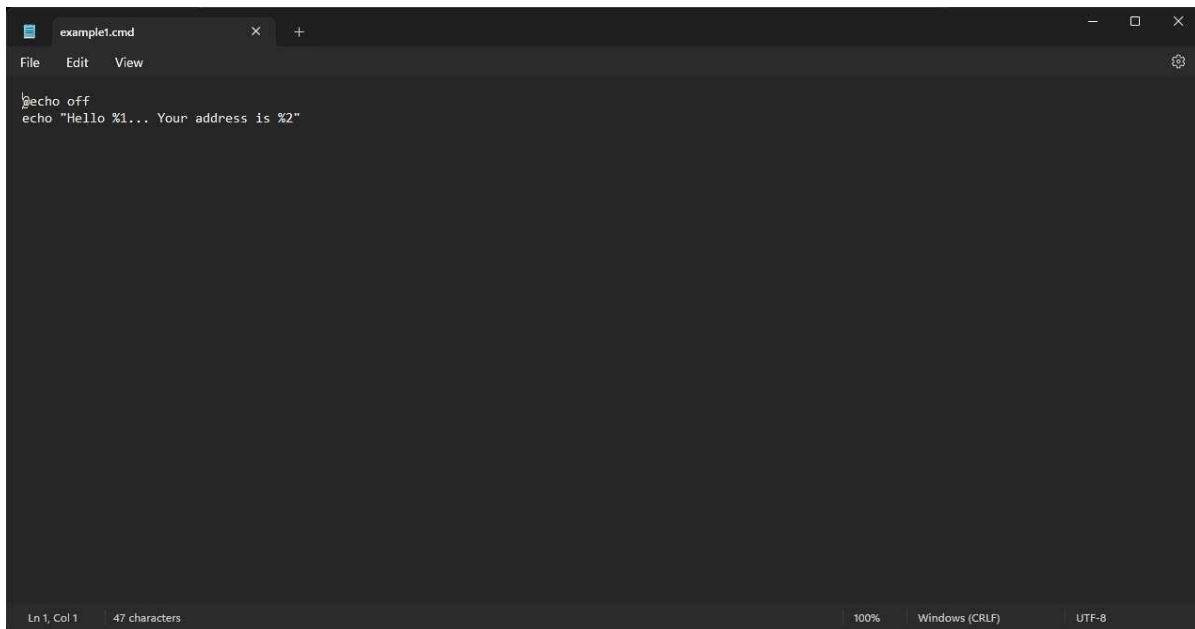


Console output (after building):

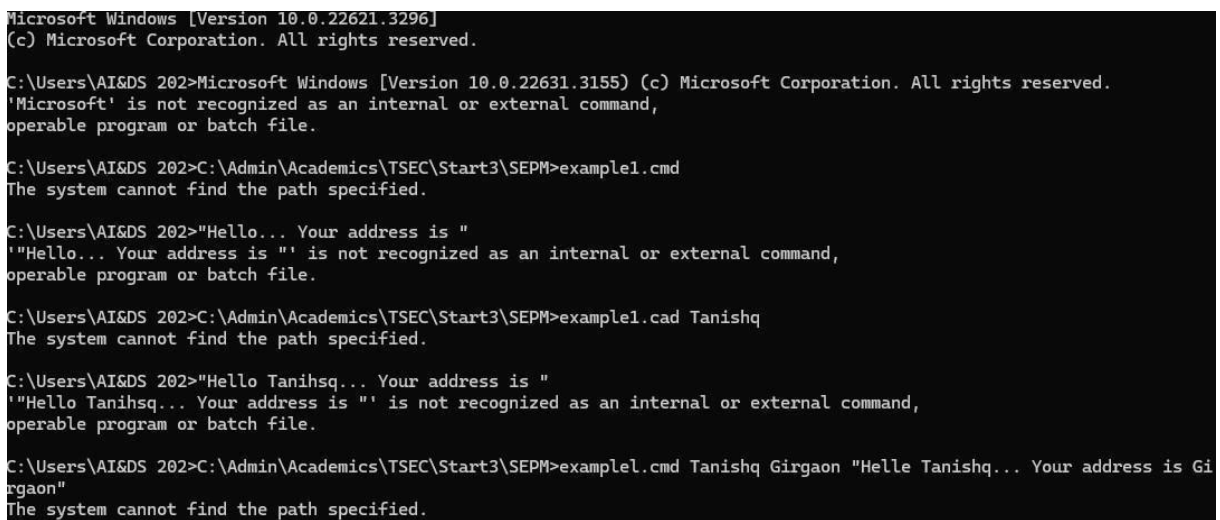


Example 1.2: Taking parameters through files Contents of script

example1.cmd:



Executing script example1.cmd on the terminal:



Modifying the Jenkins project to execute the script while supplying required parameters:

Build Steps

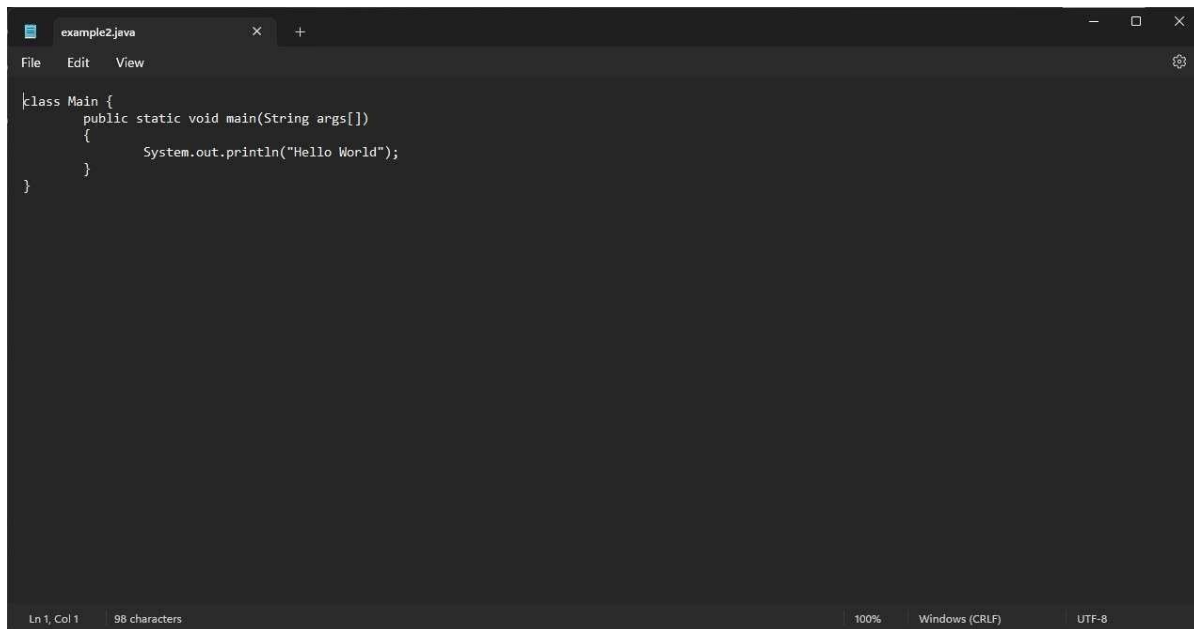


Console output after building the modified project:



Example 2 Example 2.1: Running a Java program under

Jenkins Creating a simple Java program:



Compiling and running the program on the terminal:

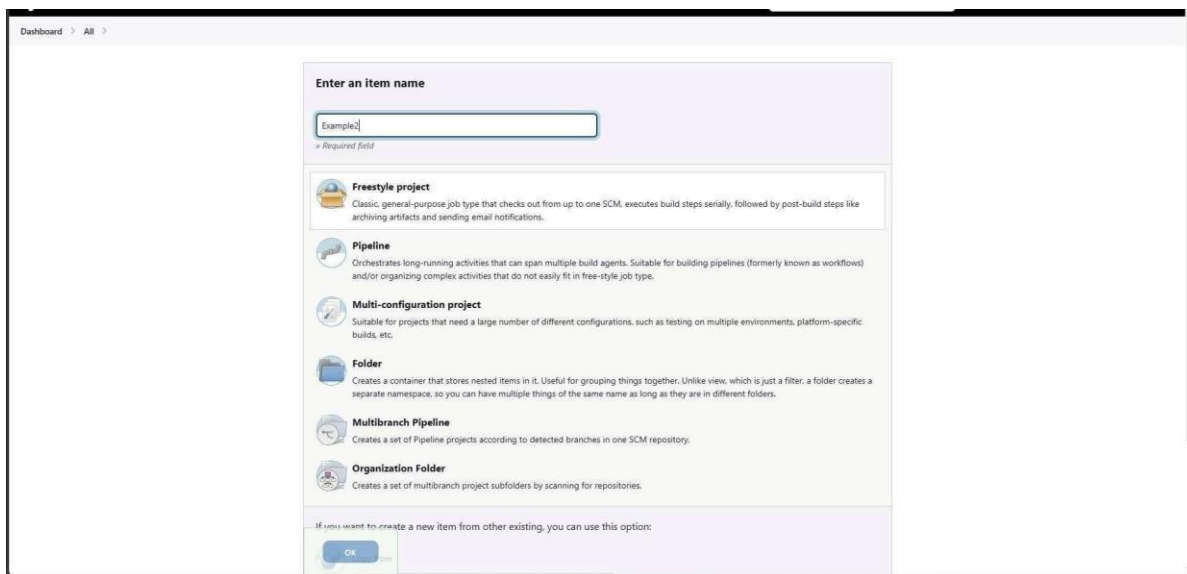
```
Command Prompt
Microsoft Windows [Version 10.0.22631.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Admin\Academics\TSEC\Start3\SEPM>javac example2.java

C:\Admin\Academics\TSEC\Start3\SEPM>java example2.java
Hello World

C:\Admin\Academics\TSEC\Start3\SEPM>
```

Creating a new freestyle project:



Configure new project:

Build Steps

Execute Windows batch command ?

Command

See [the list of available environment variables](#)

```
javac C:\Admin\Academics\TSEC\Start3\SEPH\example2.java
java C:\Admin\Academics\TSEC\Start3\SEPH\example2.java
```

Advanced ▾

Add build step ▾

Console output after building:



Console Output

```
Started by user Siddhant Chetlur
Running as SYSTEM
[EnvInject] - Loading node environment variables.
Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\Example2
[Example2] $ cmd /c call C:\WINDOWS\TEMP\jenkins15296462404998614135.bat

C:\ProgramData\Jenkins\jenkins\workspace\Example2>javac C:\Admin\Academics\TSEC\Start3\SEPH\example2.java

C:\ProgramData\Jenkins\jenkins\workspace\Example2>java C:\Admin\Academics\TSEC\Start3\SEPH\example2.java
Hello World

C:\ProgramData\Jenkins\jenkins\workspace\Example2>exit: 0
Finished: SUCCESS
```


Example 3 Example 3.1: Parameterise build

Creating a new freestyle project:


Enter an item name

Example3


[» Required field](#)

**Freestyle project**


Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

**Pipeline**


Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**


Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**

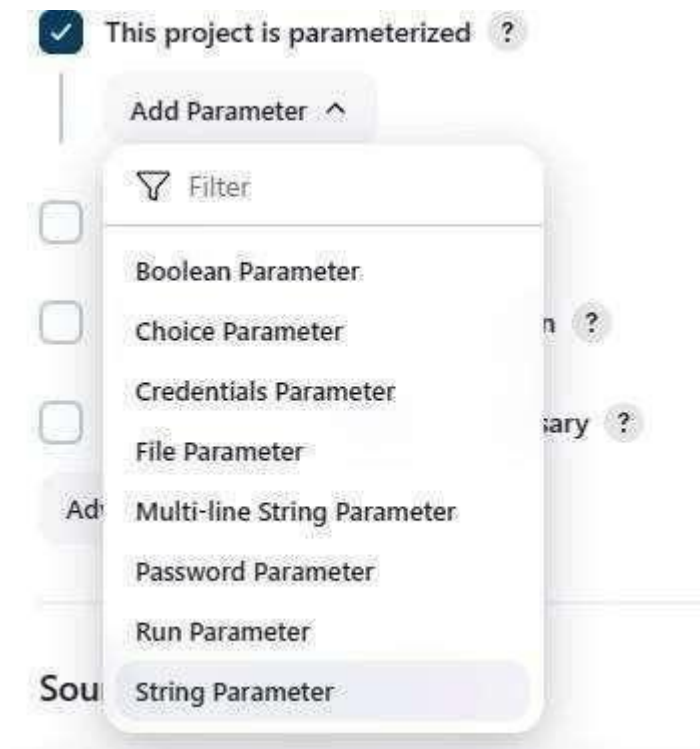
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

OK

Copy from

Enabling parameterisation and adding a String parameter:



Configuring the string parameter as Fname:

A screenshot of a configuration form for a "String Parameter". The form is enclosed in a dashed border and has a title bar with a hamburger menu icon, the text "String Parameter", and a question mark icon. The form contains several fields: a "Name" field with the value "Fname", a "Default Value" field, and a "Description" field. Below these fields is a "Plain text" label with a "Preview" link. At the bottom, there is a checkbox labeled "Trim the string" with a question mark icon. The form is styled with a clean, modern look using light gray borders and a white background.

Adding a choice parameter and configuring it as City with the following choices:

Choice Parameter ?

Name ?

City

Choices ?

Bandra

Kalyan

Dombivali

Churchgate

Thane

Dadar

Description ?

Plain text

Preview

Creating a script which takes 2 arguments for name and city:

```

C:\Users\AI&DS 202>Microsoft Windows [Version 10.0.22631.3155] (c) Microsoft Corporation. All rights reserved.
'Microsoft' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\AI&DS 202>C:\Admin\Academics\TSEC\Start3\SEPH>example3.cnd
The system cannot find the path specified.

C:\Users\AI&DS 202>Hello your name is and your city is
'Hello' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\AI&DS 202>C:\Admin\Academics\TSEC\Start3\SEPH example3.cmd Tanishq
The system cannot find the path specified.

C:\Users\AI&DS 202>Hello your name is Tanishq and your city is
'Hello' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\AI&DS 202>C:\Admin\Academics\TSEC\Start3\SEPM>example3.cmd Tansishq Bandra
The system cannot find the path specified.

C:\Users\AI&DS 202>Hello your name is Tanishq and your city is Bandra
'Hello' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\AI&DS 202>C:\Admin\Academics\TSEC\Start3\SEPH|

```

Configuring build steps:

Build Steps

Execute Windows batch command ?

Command

See the list of available environment variables

C:\Admin\Academics\TSEC\Start3\SEPM\example3.cmd %fname% %city%

Advanced

Add build step

Entering parameters for build:

Project Example3

This build requires parameters:

Fname

City

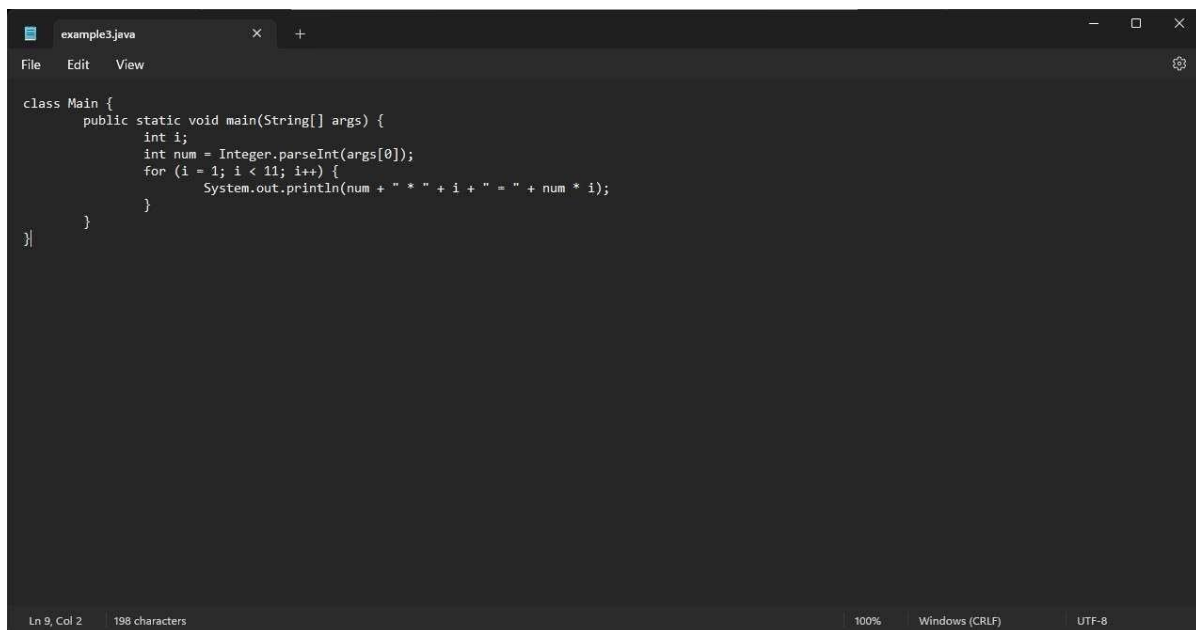
Console output after building:

✓ Console Output

```
Started by user Siddhant Chetlur
Running as SYSTEM
[EnvInject] - Loading node environment variables.
Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\Example3
[Example3] $ cmd /c call C:\WINDOWS\TEMP\jenkins14094536165150986151.bat

C:\ProgramData\Jenkins\jenkins\workspace\Example3>C:\Admin\Academics\TSEC\Start3\SEPM\example3.cmd Siddhant Bandra
Hello your name is Siddhant and your city is Bandra
Finished: SUCCESS
```

Example 3.2: Running a Java program with parameters Creating
a Java program with an input argument:



```
example3.java
File Edit View

class Main {
    public static void main(String[] args) {
        int i;
        int num = Integer.parseInt(args[0]);
        for (i = 1; i < 11; i++) {
            System.out.println(num + " * " + i + " = " + num * i);
        }
    }
}
```

Ln 9, Col 2 198 characters 100% Windows (CRLF) UTF-8

Testing the program on the terminal:

```
Command Prompt
Microsoft Windows [Version 10.0.22631.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Admin\Academics\TSEC\Start3\SEPM>javac example3.java

C:\Admin\Academics\TSEC\Start3\SEPM>java example3.java 4
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40


C:\Admin\Academics\TSEC\Start3\SEPM>
```

Creating a new freestyle project:

Enter an item name


Example4

» Required field




Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.




Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.




Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.




Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.



Organization Folder

Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

OK

om

Parameterise the project by adding a string parameter as follows:

☒ This project is parameterized ?

String Parameter ?

×

Name ?

num

Default Value ?

Description ?

Plain text [Preview](#)

☐ Trim the string ?

Add Parameter ▾

Configure the build steps:

Build Steps

Execute Windows batch command ?

Command

See the list of available environment variables

```
javac C:\Admin\Academics\TSEC\Start3\SEPM\example3.java
java C:\Admin\Academics\TSEC\Start3\SEPM\example3.java %num%
```

Advanced ▾

Add build step ▾

Entering the parameter for the build:

Project Example4

This build requires parameters:

num

25

▶ Build

Cancel

Console output after building:

✓ Console Output

```
Started by user Siddhant Chetlun
Running as SYSTEM
[EnvInject] - Loading node environment variables.
Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\Example4
[Example4] $ cmd /c call C:\WINDOWS\TEMP\jenkins15119185770823247708.bat

C:\ProgramData\Jenkins\jenkins\workspace\Example4>javac C:\Admin\Academics\TSEC\Start3\SEPM\example3.java

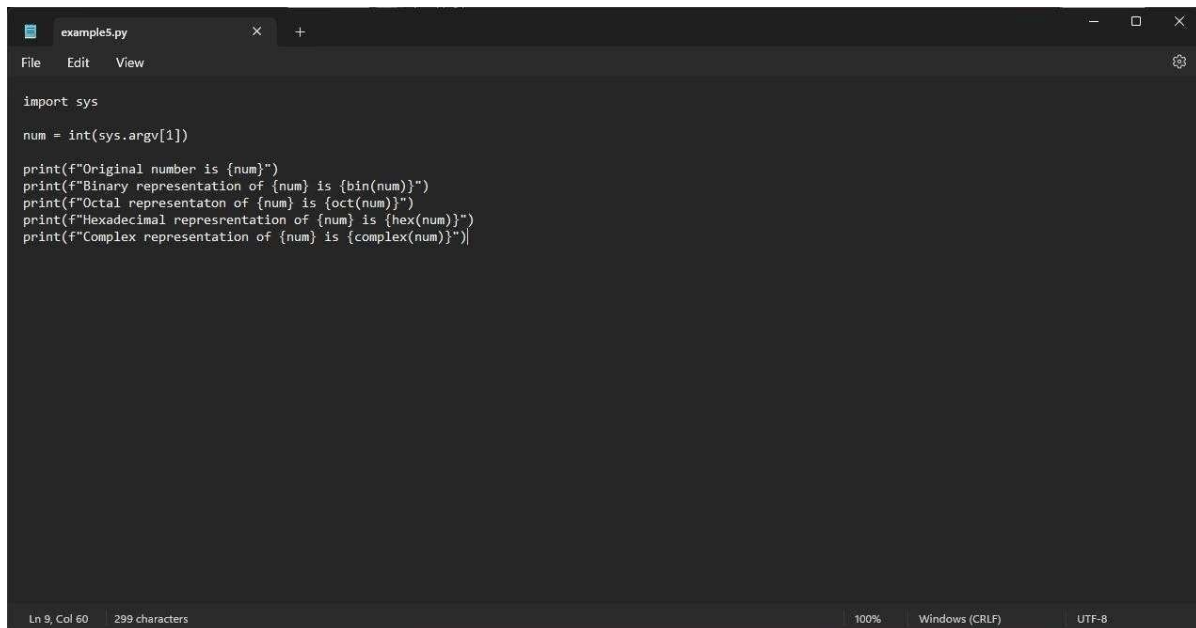
C:\ProgramData\Jenkins\jenkins\workspace\Example4>java C:\Admin\Academics\TSEC\Start3\SEPM\example3.java 25
25 * 1 = 25
25 * 2 = 50
25 * 3 = 75
25 * 4 = 100
25 * 5 = 125
25 * 6 = 150
25 * 7 = 175
25 * 8 = 200
25 * 9 = 225
25 * 10 = 250

C:\ProgramData\Jenkins\jenkins\workspace\Example4>exit 0
Finished: SUCCESS
```

Example 5 Example

5.1: Running a Python program Creating a simple Python

script:



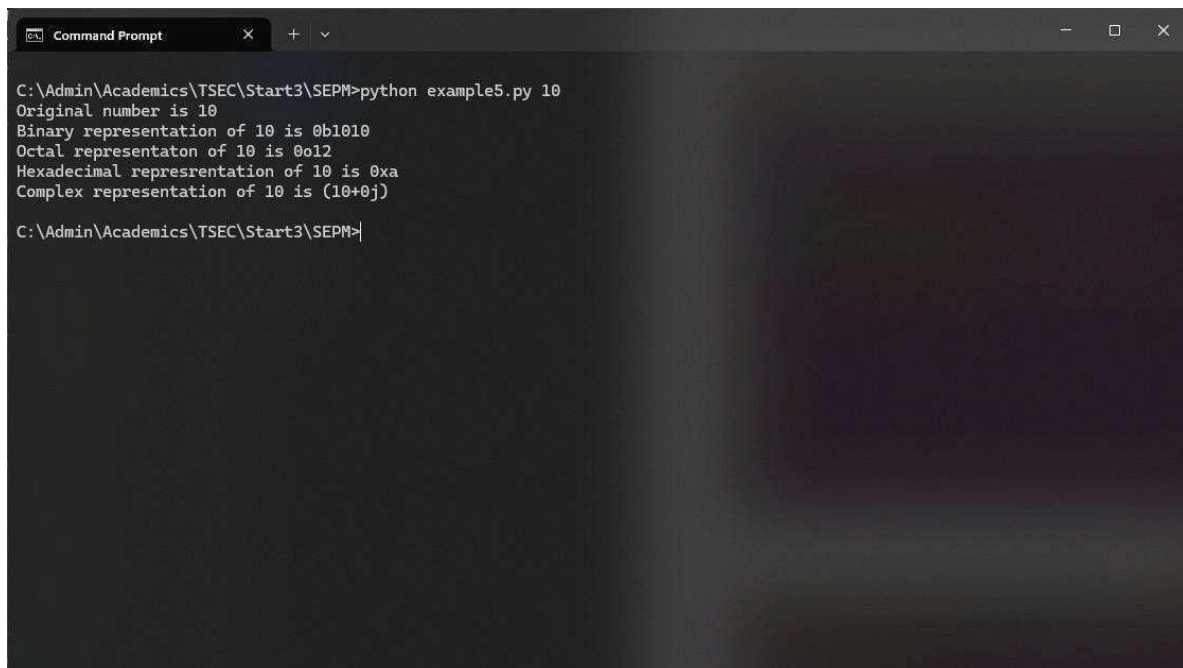
```
import sys

num = int(sys.argv[1])

print(f"Original number is {num}")
print(f"Binary representation of {num} is {bin(num)}")
print(f"Octal representation of {num} is {oct(num)}")
print(f"Hexadecimal representation of {num} is {hex(num)}")
print(f"Complex representation of {num} is {complex(num)}")
```

The screenshot shows a code editor window with a dark theme. The title bar indicates the file is 'example5.py'. The menu bar includes 'File', 'Edit', and 'View'. The status bar at the bottom shows 'Ln 9, Col 60', '299 characters', '100%' zoom, 'Windows (CRLF)' line endings, and 'UTF-8' encoding.

Running the Python script on the terminal:



```
C:\Admin\Academics\TSEC\Start3\SEPM>python example5.py 10
Original number is 10
Binary representation of 10 is 0b1010
Octal representation of 10 is 0o12
Hexadecimal representation of 10 is 0xa
Complex representation of 10 is (10+0j)


C:\Admin\Academics\TSEC\Start3\SEPM>|
```

The screenshot shows a Windows Command Prompt window. The user has executed the command 'python example5.py 10'. The output displays the number 10 in its original, binary, octal, hexadecimal, and complex representations. The prompt is now waiting for the next command.

Creating a new freestyle project:


Enter an item name

» Required field




Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.




Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.




Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.




Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.



Organization Folder

Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Parameterising the project with a string parameter as follows:

☒ This project is parameterized [?](#)

String Parameter [?](#)

Name [?](#)

Default Value [?](#)

Description [?](#)

Plain text [Preview](#)

☐ Trim the string [?](#)

Add Parameter [v](#)

Configuring the build steps:

Build Steps

Execute Windows batch command ?

Command

[See the list of available environment variables](#)

```
python C:\Admin\Academics\TSEC\Start3\SEPM\example5.py %num%
```

Advanced ▾

Add build step ▾

Setting the parameter for the build:

Project Example5

This build requires parameters:

num

▶ Build

Cancel

Console output after building:

✓ Console Output

```
Started by user Siddhant Chetlur
Running as SYSTEM
[EnvInject] - Loading node environment variables.
Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\Example5
[Example5] $ cmd /c call C:\WINDOWS\TEMP\jenkins11157306491994478222.bat

C:\ProgramData\Jenkins\jenkins\workspace\Example5>python C:\Admin\Academics\TSEC\Start3\SEPM\example5.py 10
Original number is 10
Binary representation of 10 is 0b1010
Octal representation of 10 is 0o12
Hexadecimal representation of 10 is 0xa
Complex representation of 10 is (10+0j)

C:\ProgramData\Jenkins\jenkins\workspace\Example5>exit 0
Finished: SUCCESS
```

Conclusion

This experiment demonstrated how to automate a software build and deployment process using Jenkins. By integrating Maven, Gradle, or Ant, we streamlined the compilation and testing of applications, while Jenkins facilitated continuous deployment to a Tomcat server.