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

Yatish Shah

T21-91

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. 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.
  2. Tools Used
     + Jenkins – Automation server.
     + Maven/Gradle/Ant – Build automation tools.
     + Tomcat – A web server for deploying Java applications.

1. Installing and Configuring Jenkins
2. Download and Install Jenkins:

* Download from Jenkins official site. o Install and start the Jenkins service.
* Access Jenkins at [http://localhost:8080/.](http://localhost:8080/)

1. Install Required Plugins:

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

1. Creating a Jenkins Pipeline

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

* 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.

1. 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')],](http://yourtomcat-server:8080/)

war: '\*\*/\*.war'

}

}

}

}

1. Configuring Jenkins for Deployment
2. Configure Tomcat Server:
   * 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.

1. Set Up Jenkins Credentials:

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

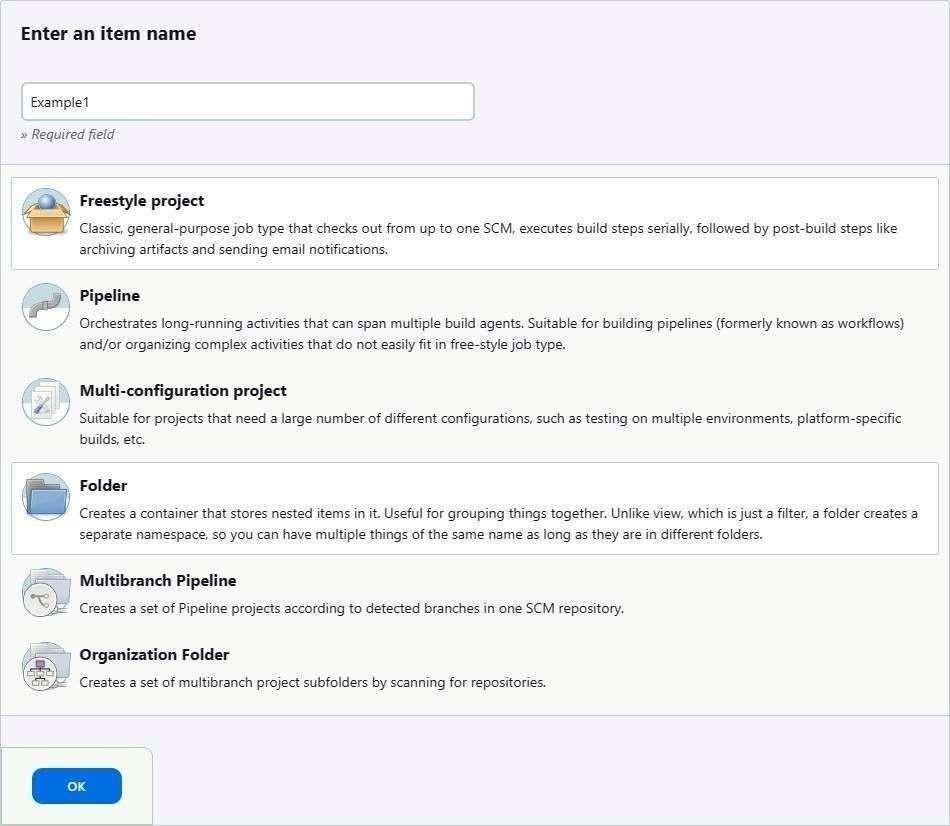
1. Run the Pipeline in Jenkins:
   * Click Build Now to execute the pipeline. o Verify the deployment at [http://your-tomcat-server:8080/your-app.](http://your-tomcat-server:8080/your-app)

Example 1 :

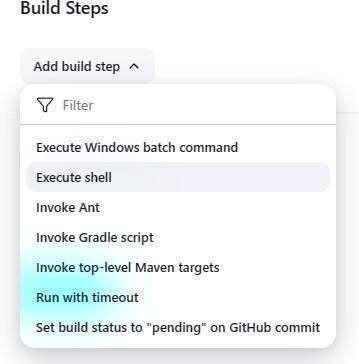
Creating a job:



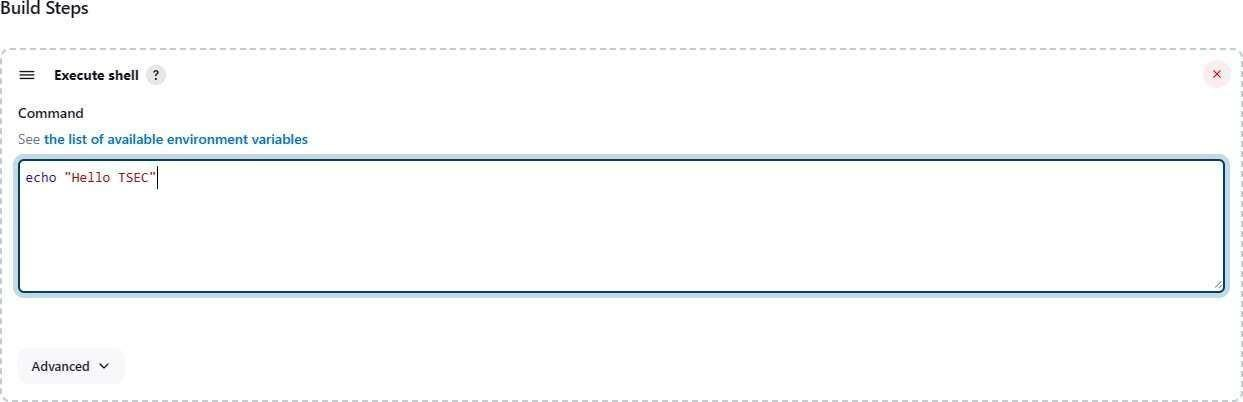
Naming the job and setting it as freestyle:



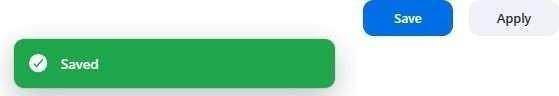
Selecting build type as “Execute shell”:



Entering a simple command for the shell execution:



Applying and saving the project configuration:



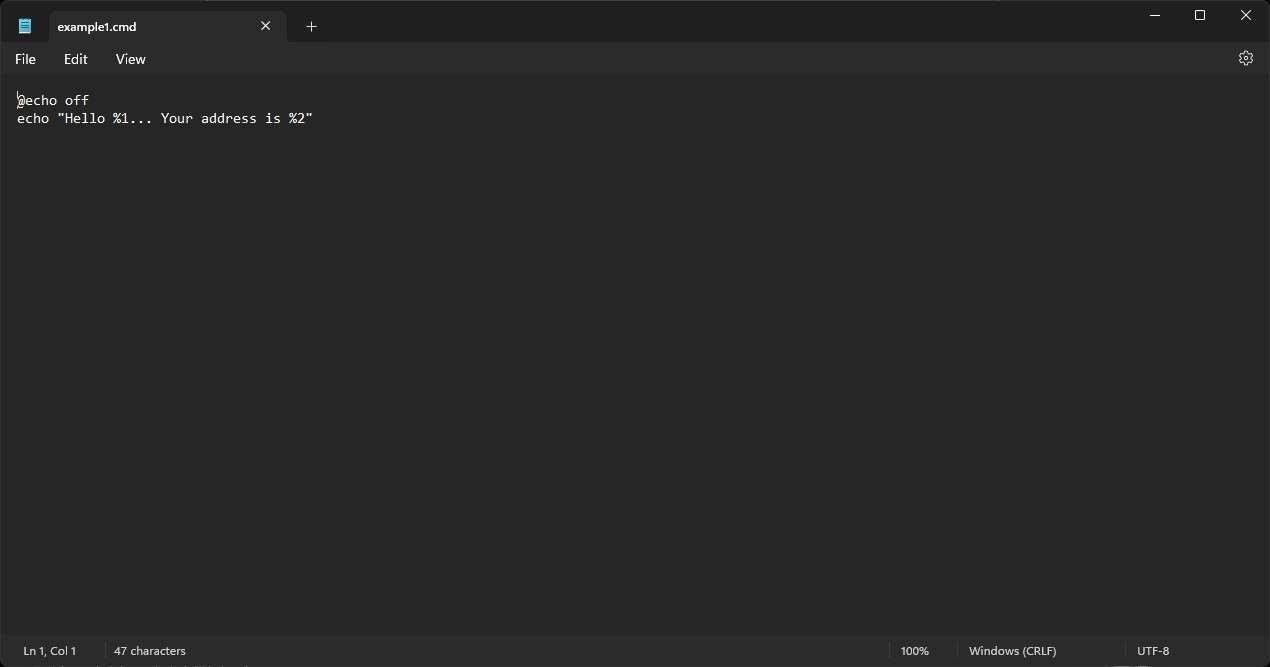
Building the project:



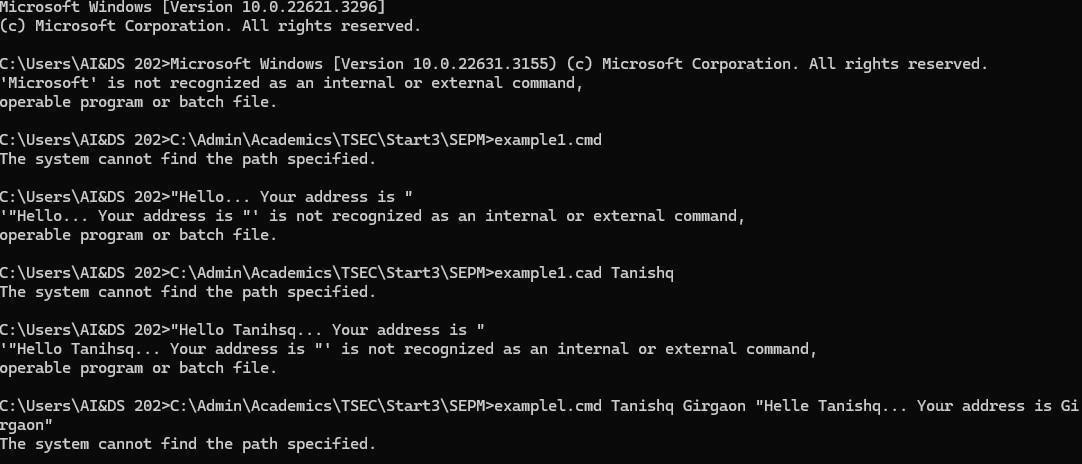
Console output (after building):



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:

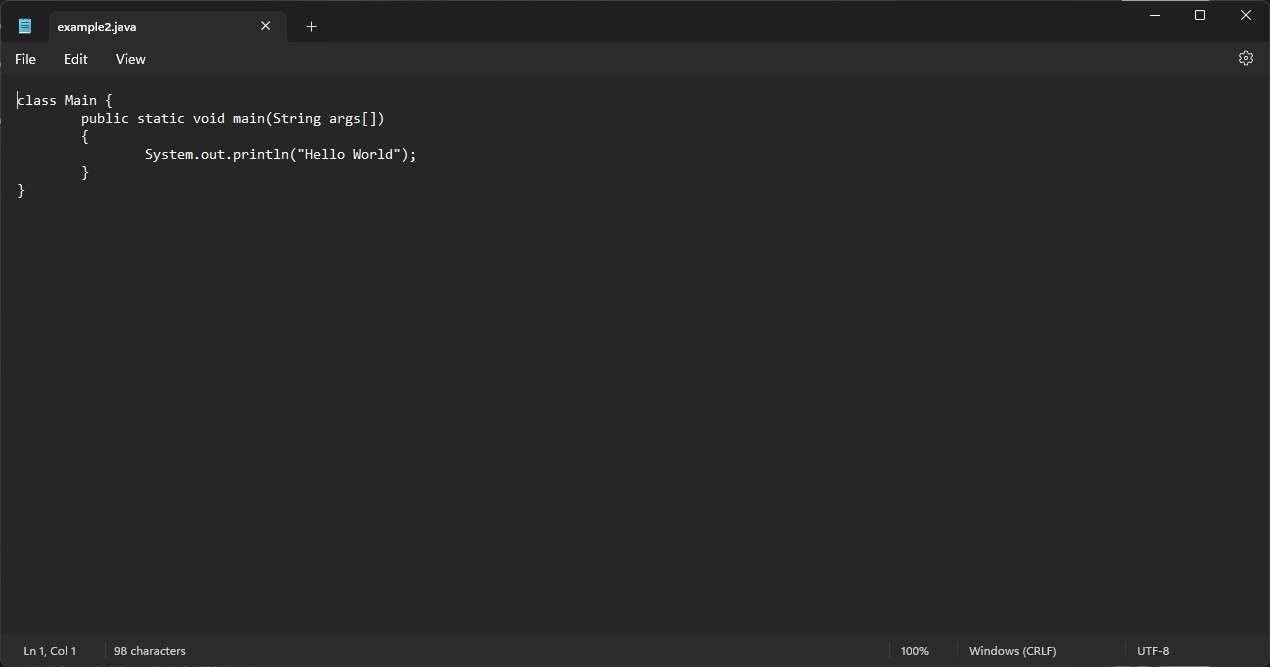


Console output after building the modified project:

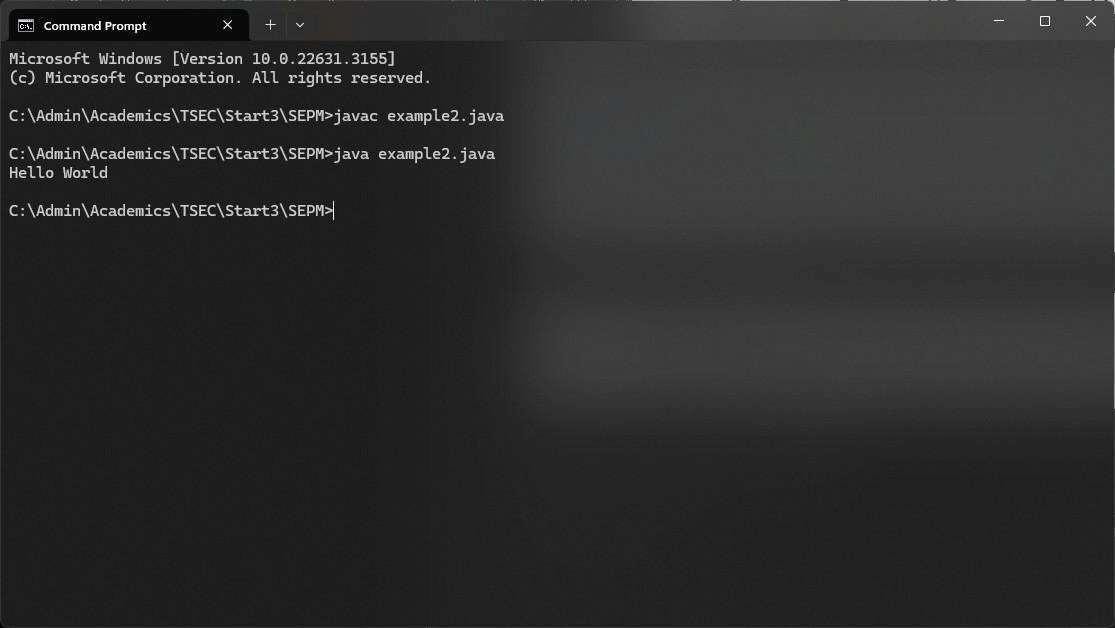


Example 2 Example 2.1: Running a Java program under

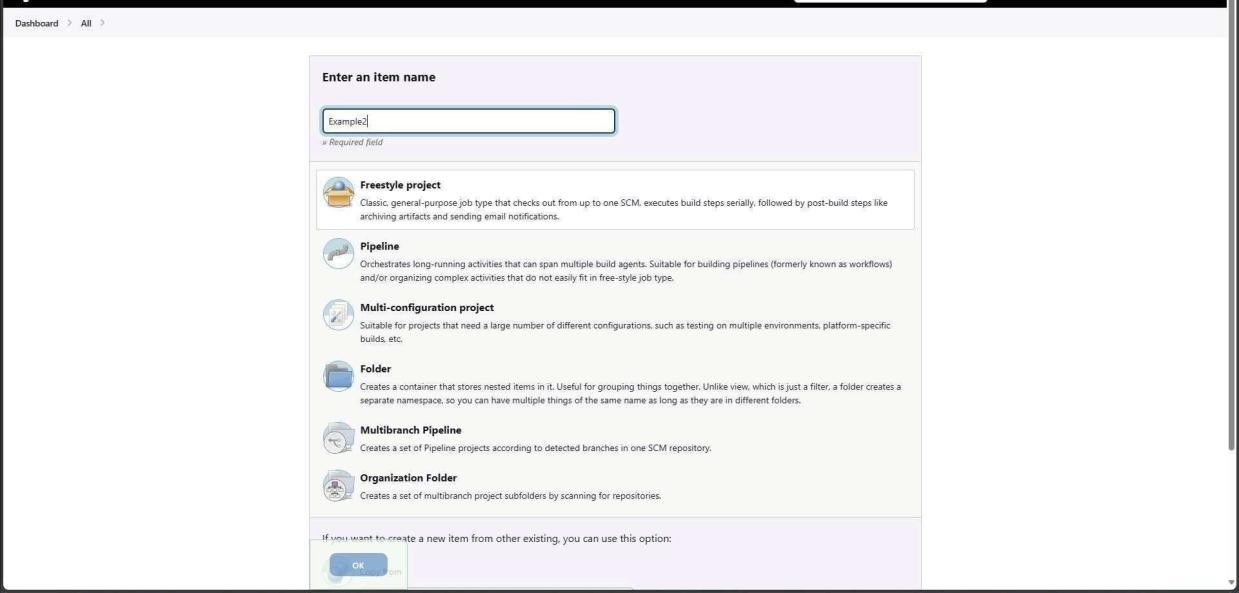
Jenkins Creating a simple Java program:



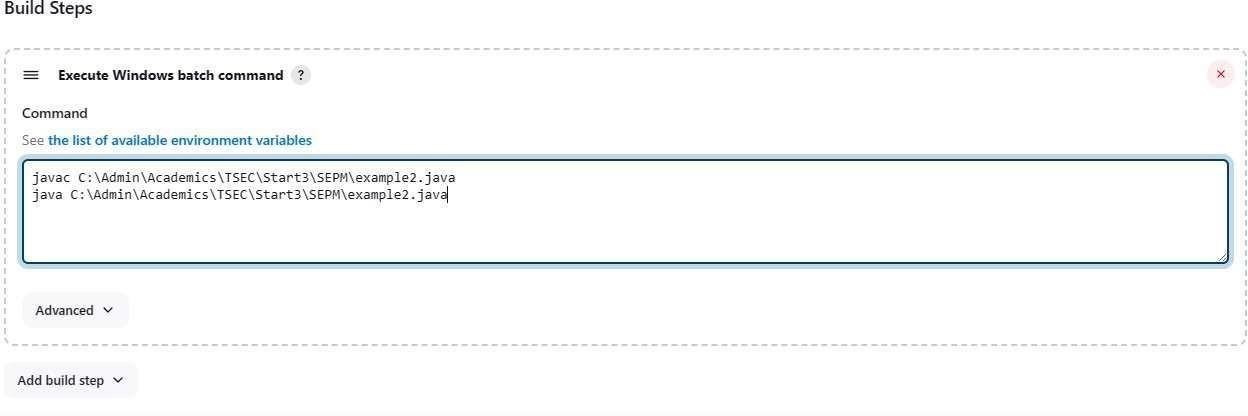
Compiling and running the program on the terminal:



Creating a new freestyle project:



Configure new project:

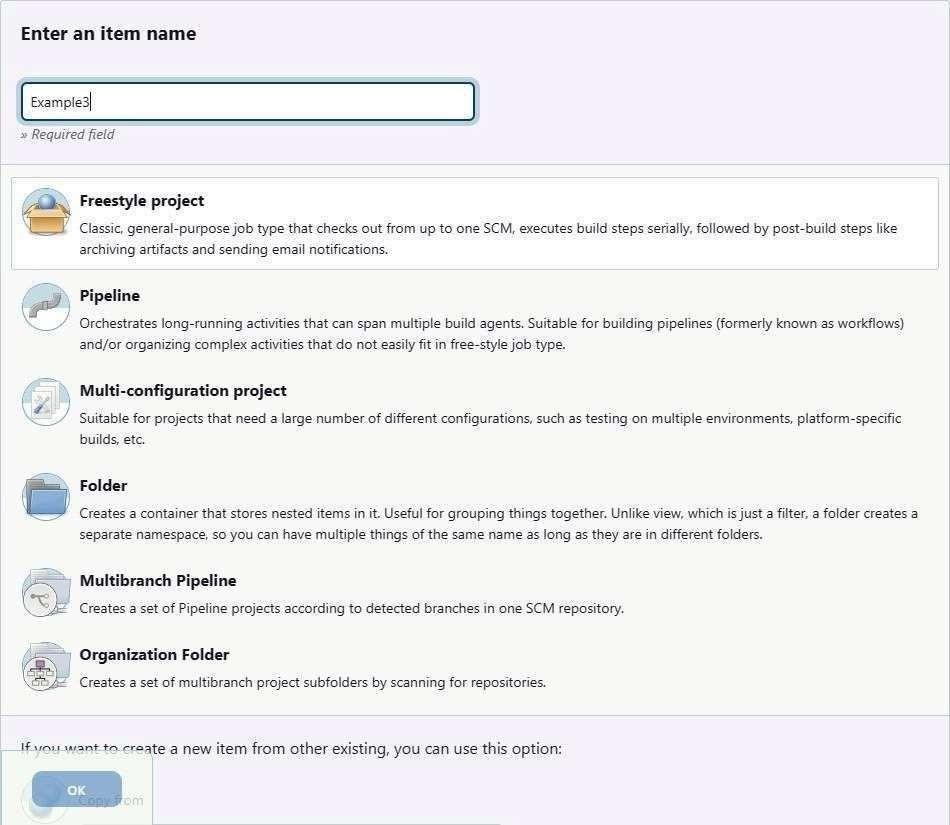


Console output after building:

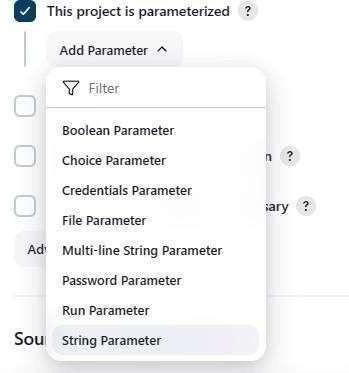


Example 3 Example 3.1: Parameterise build

Creating a new freestyle project:



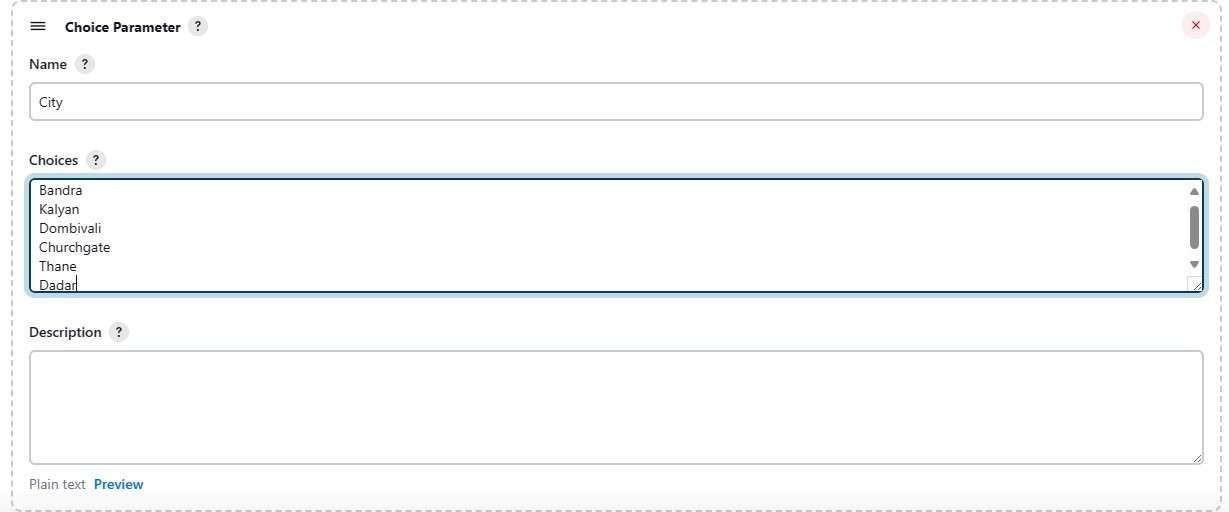
Enabling parameterisation and adding a String parameter:



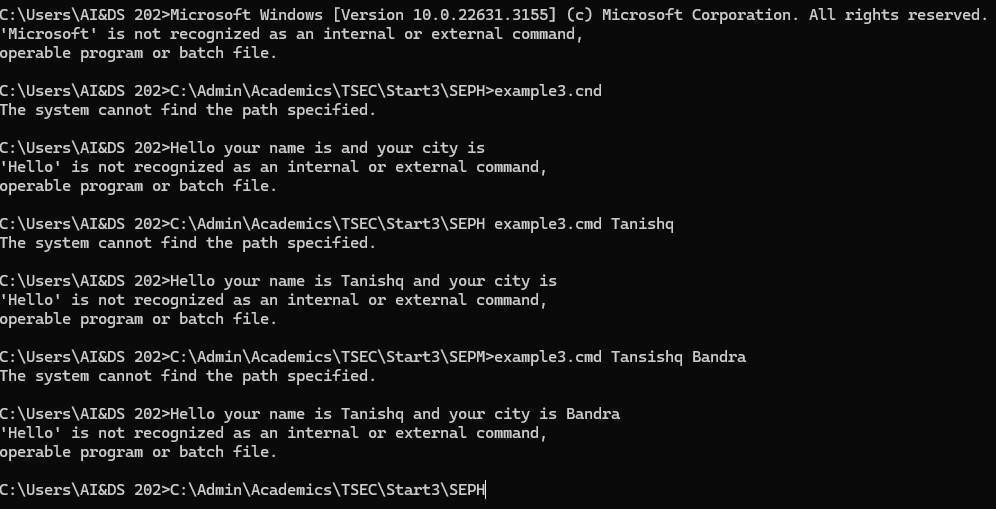
Configuring the string parameter as Fname:



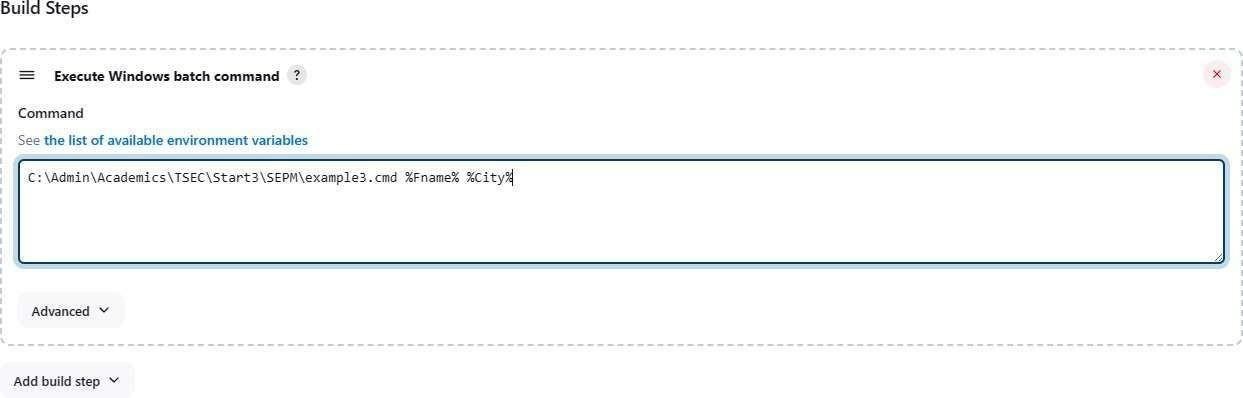
Adding a choice parameter and configuring it as City with the following choices:



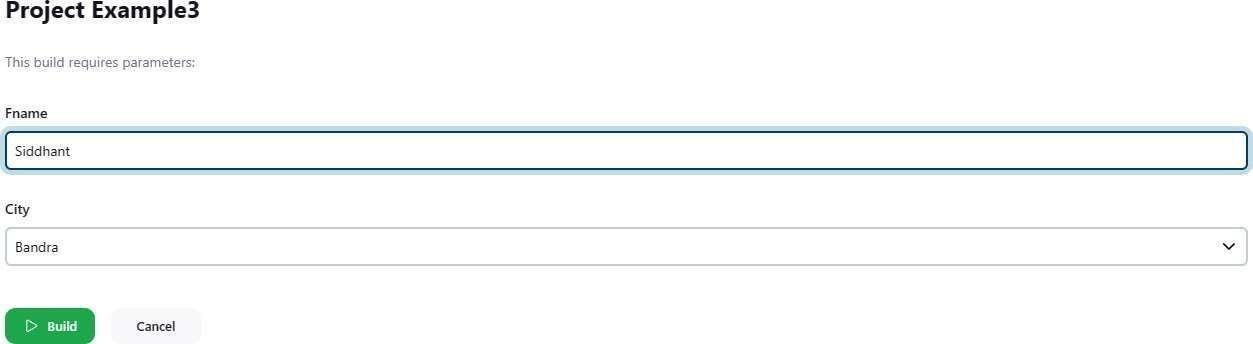
Creating a script which takes 2 arguments for name and city:



Configuring build steps:



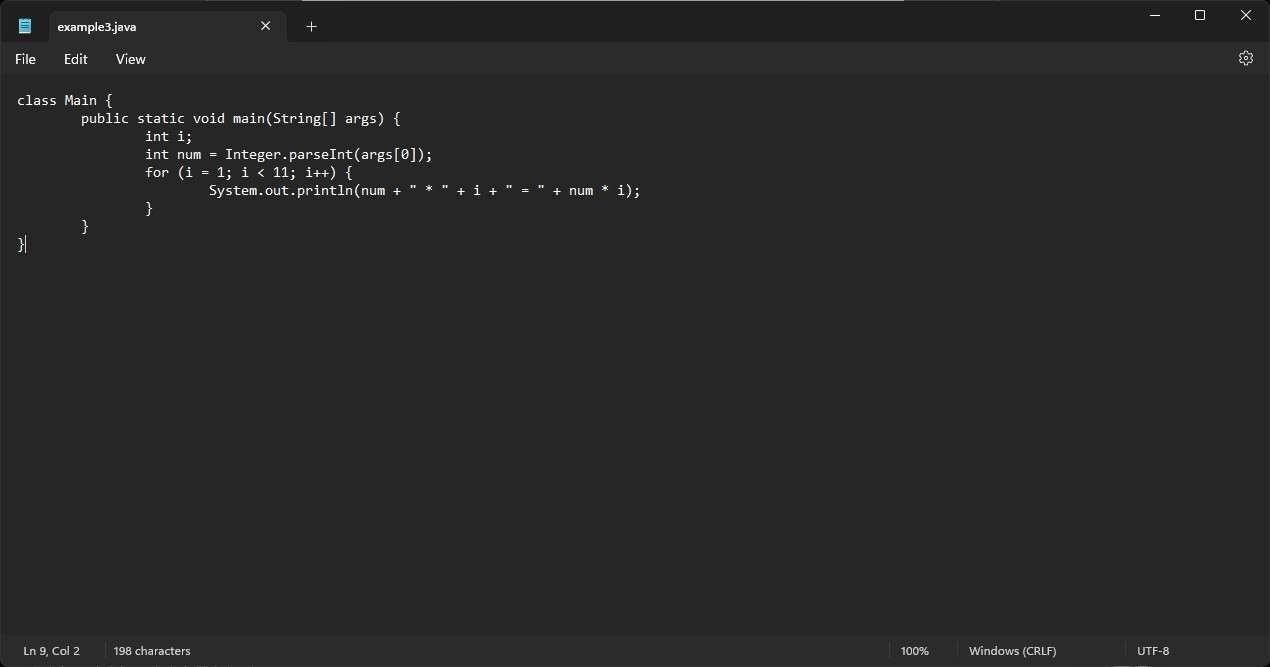
Entering parameters for build:



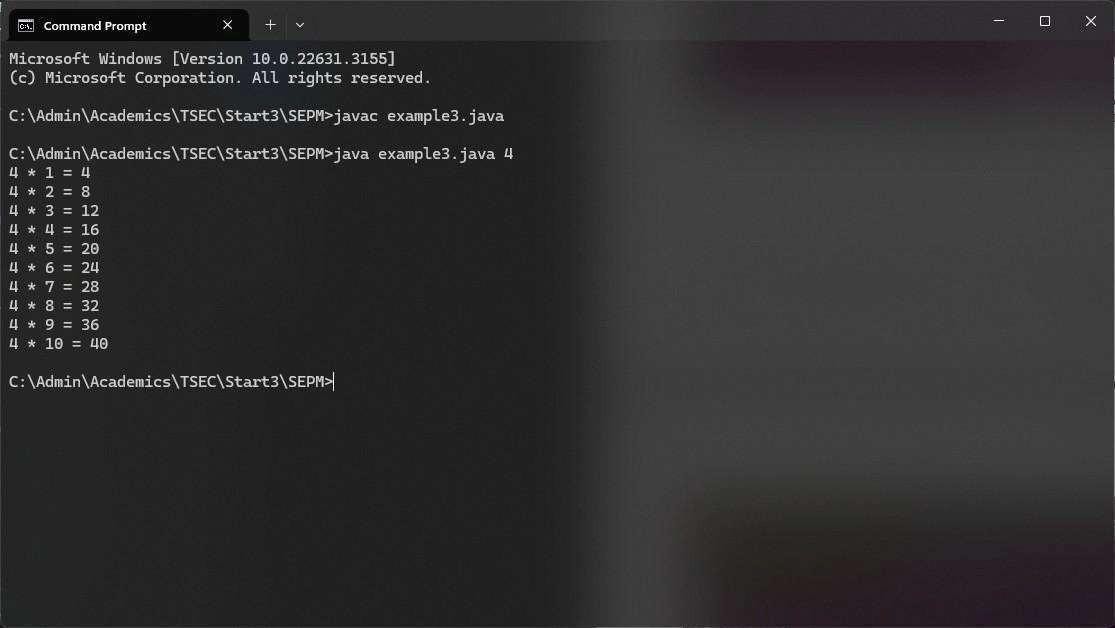
Console output after building:



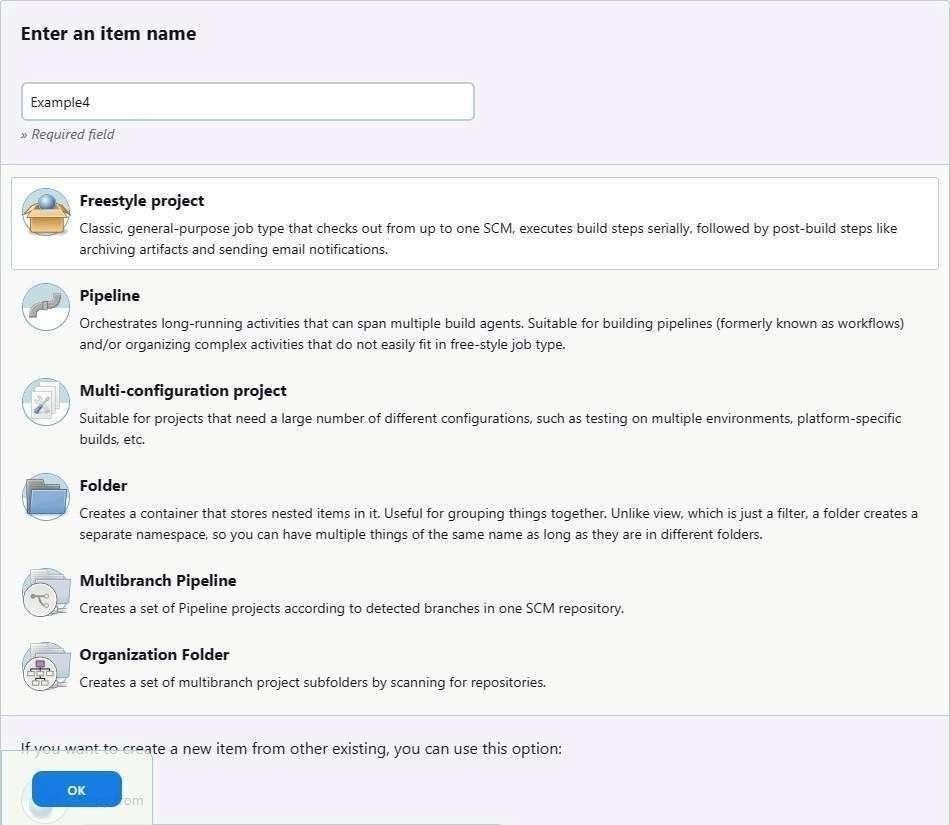
Example 3.2: Running a Java program with parameters Creating a Java program with an input argument:



Testing the program on the terminal:



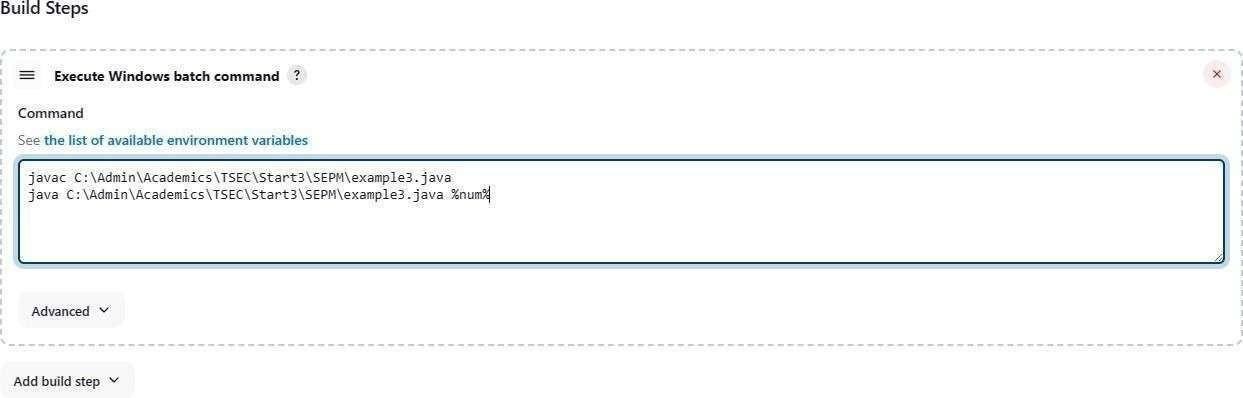
Creating a new freestyle project:



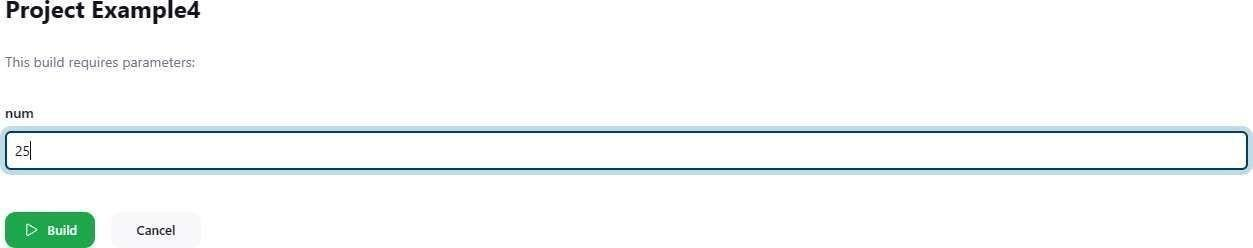
Parameterise the project by adding a string parameter as follows:



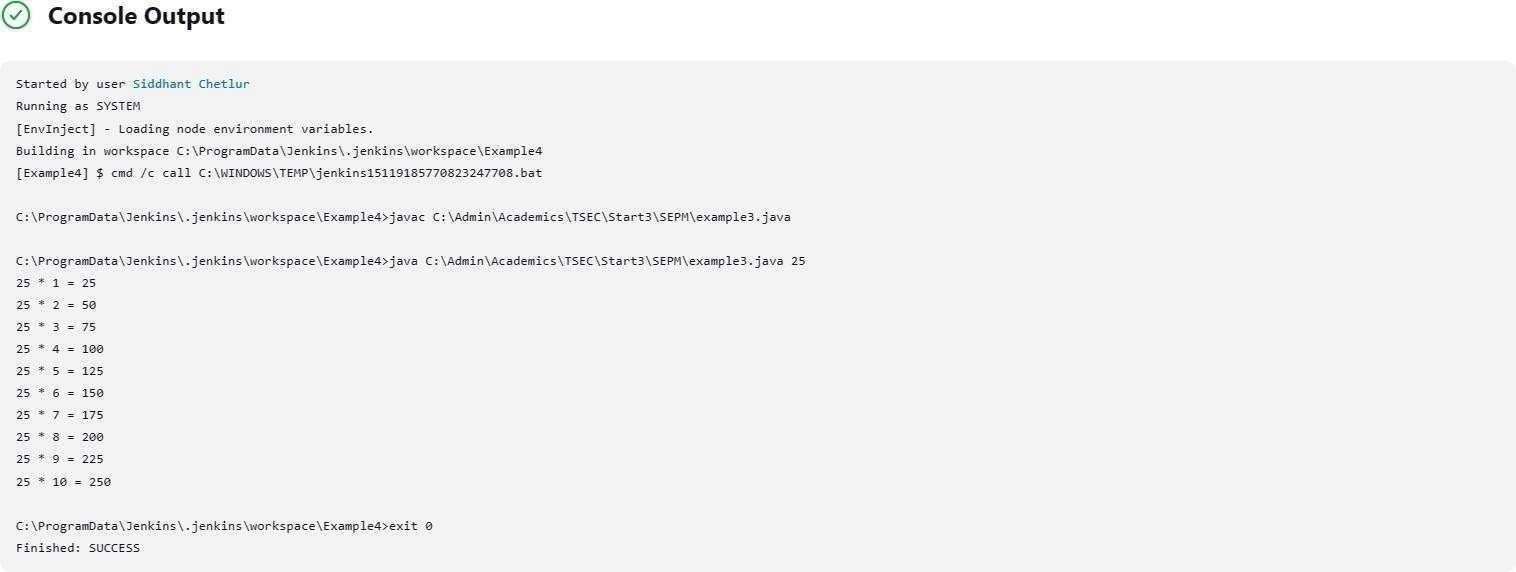
Configure the build steps:



Entering the parameter for the build:



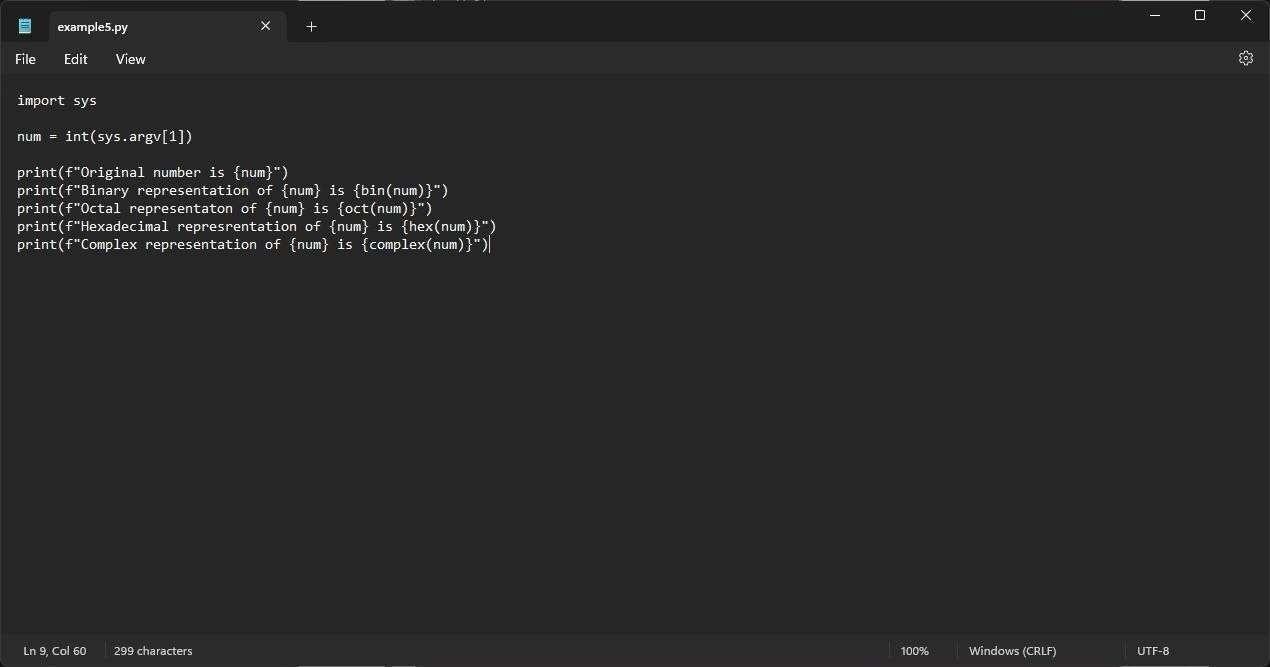
Console output after building:



Example 5 Example

5.1: Running a Python program Creating a simple Python

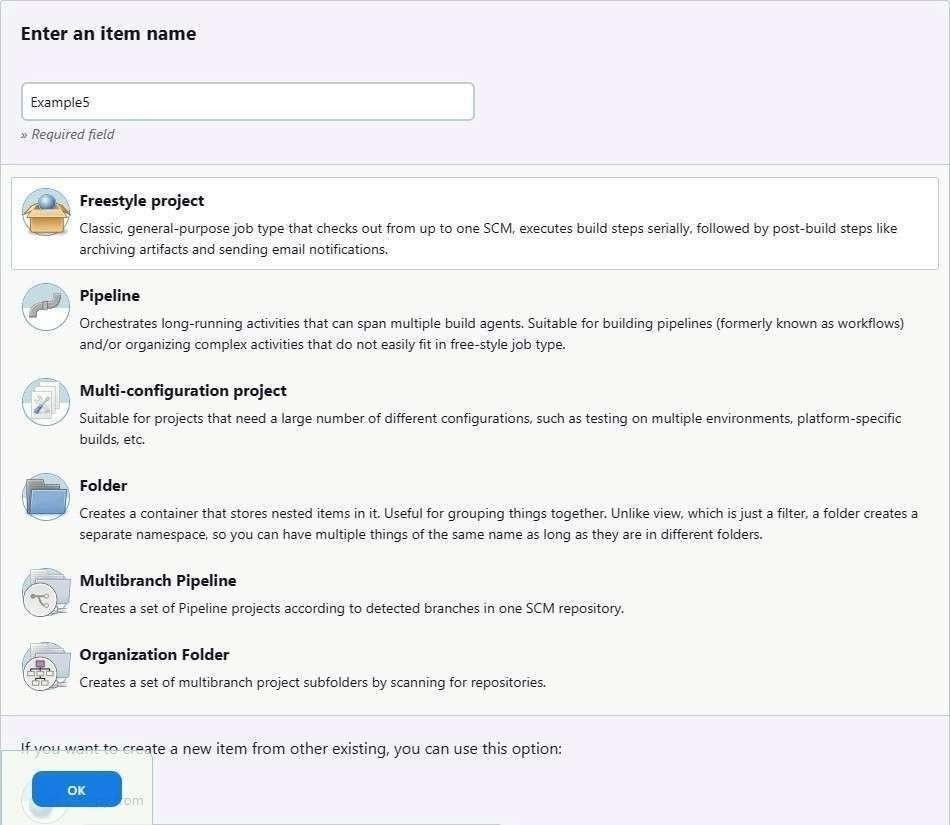
script:



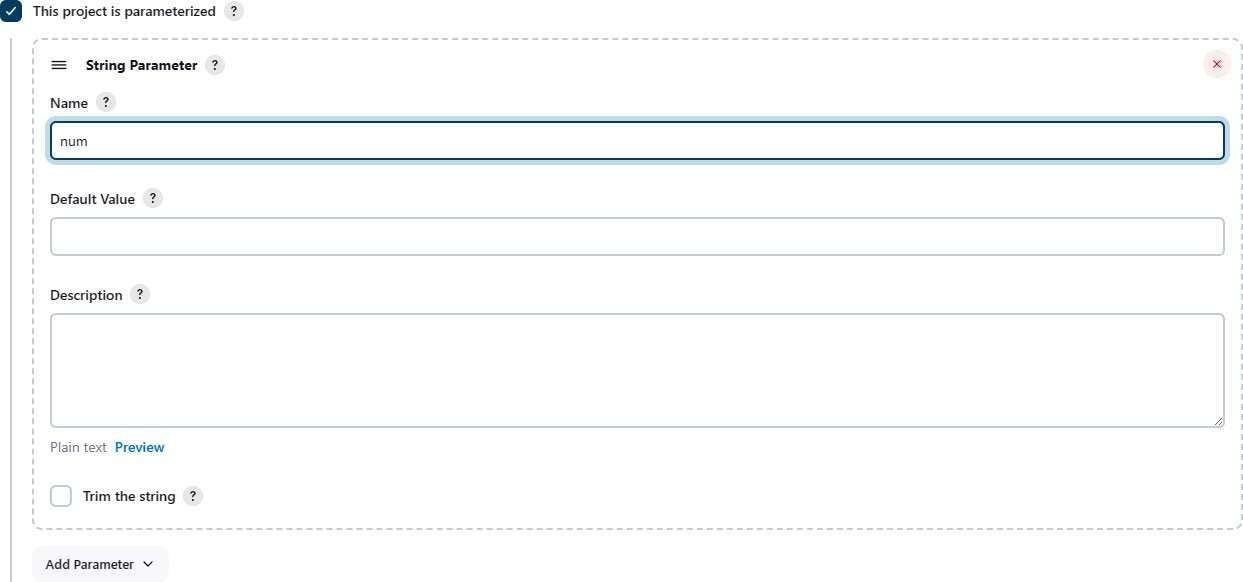
Running the Python script on the terminal:



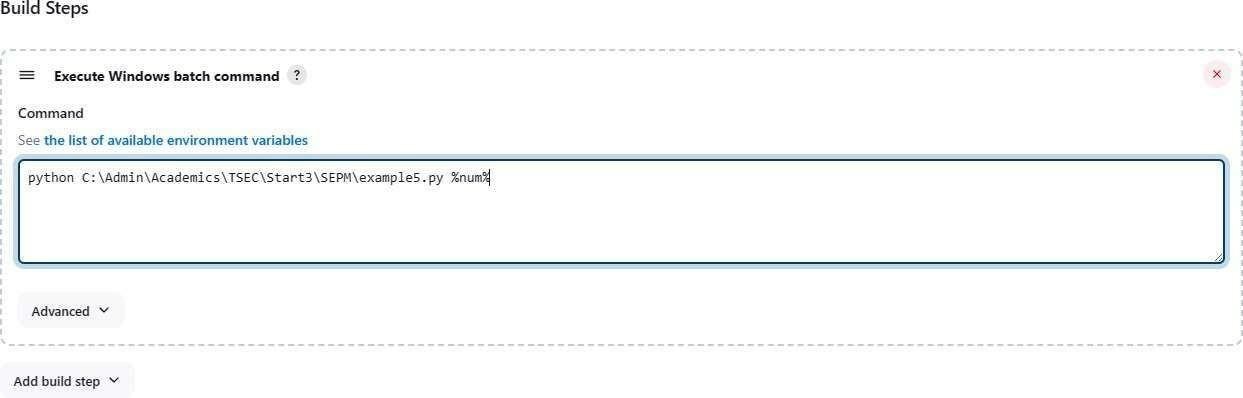
Creating a new freestyle project:



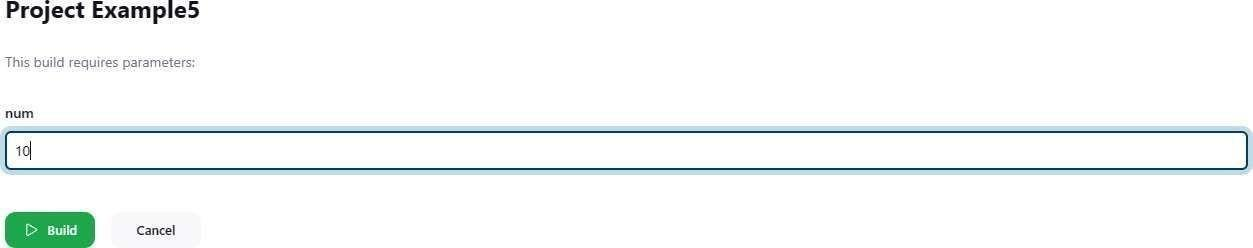
Parameterising the project with a string parameter as follows:



Configuring the build steps:



Setting the parameter for the build:



Console output after building:



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.