Write a program that compares two strings and checks for substring presence.

**Program:**

package Practicals;

public class String\_Comparison {

public static void main(String[] args) {

// TODO Auto-generated method stub

String str1 = "Hello World";

String str2 = "World";

if (str1.contains(str2)) {

System.out.println(str2 + " is a substring of " + str1);

} else {

System.out.println(str2 + " is not a substring of " + str1);

}

}

}

Program to encode characters to their Unicode representations and decode them back.

**Program:**

package Practicals;

import java.nio.charset.StandardCharsets;

import java.util.\*;

public class UnicodeEncodeDecode {

public static void main(String arr[]) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter string to encode: ");

String inputString = sc.nextLine();

for (int i = 0; i < inputString.length(); i++) {

int codePoint = inputString.codePointAt(i);

System.out.printf("Codepoint: '%c' => Unicode: '\\u%04x'\n", inputString.charAt(i), codePoint);

}

byte[] utf8Byte = inputString.getBytes(StandardCharsets.UTF\_8);

System.out.println("\nString in UTF 8 encoding: ");

for (byte b : utf8Byte) {

System.out.printf("%02X ", b);

}

System.out.println("\nDecoded String is: ");

String decodedString = new String(utf8Byte, StandardCharsets.UTF\_8);

System.out.println(decodedString);

sc.close();

}

}

Write a program that takes user input for multiple strings and appends them using StringBuilder.

**Program:**

package Practicals;

import java.util.Scanner;

public class StringBuilderAppend {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

StringBuilder stringBuilder = new StringBuilder();

System.out.println("Enter First String:");

while (true) {

String input = sc.nextLine();

if (input.equalsIgnoreCase("exit")) {

break;

}

stringBuilder.append(input).append(" ");

}

System.out.println("String is " + stringBuilder.toString());

sc.close();

}

}

Write a program to split a paragraph into individual sentences.

**Program:**

package Practicals;

public class ParagraphSplitter {

public static void main(String[] args) {

String paragraph = "This is the first sentence. This is the second one. And here is the third!";

String[] sentences = paragraph.split("\\.\\s\*|\\!\\s\*|\\?\\s\*");

for (String sentence : sentences) {

System.out.println(sentence.trim());

}

}

}

Write a program to convert a date object to a string in a specific format.

**Program:**

package Practicals;

import java.text.SimpleDateFormat;

import java.util.Date;

public class DateFormatter {

public static void main(String[] args) {

Date date = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");

String formattedDate = formatter.format(date);

System.out.println("Formatted Date: " + formattedDate);

}

}

Write a program to insert a substring into a string at a specific position using StringBuilder.

**Program:**

package Practicals;

public class SubstringInserter {

public static void main(String[] args) {

String original = "Hello Anamika Mam!";

String toInsert = "Intelligent ";

int position = 6;

StringBuilder builder = new StringBuilder(original);

builder.insert(position, toInsert);

System.out.println("Result: " + builder.toString());

}

}

Write a program to remove null values from an array of strings

**Program:**

package Practicals;

import java.util.ArrayList;

public class RemoveNull {

public static void main(String[] args) {

String[] originalArray = { "Apple", null, "banana", "Orange", null, "Grapes" };

ArrayList<String> nonNullList = new ArrayList<>();

for (String element : originalArray) {

if (element != null) {

nonNullList.add(element);

}

}

String[] nonNullArray = nonNullList.toArray(new String[0]);

System.out.print("After removing null values\n");

for (String e : nonNullArray) {

System.out.println(e);

}

}

}

Write a program to find all occurrences of a pattern in a string using Pattern and Matcher.

**Program:**

package Practicals;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class Pattern\_Matcher {

public static void main(String[] args) {

String input = "Hello World";

String pattern = "ello";

Pattern p = Pattern.compile(pattern);

Matcher m = p.matcher(input);

while (m.find()) {

System.out.println("Pattern found at index: " + m.start());

}

}

}

Write a program to count the number of vowels in a string using the Character class.

**Program:**

package Practicals;

public class VowelCounter {

public static void main(String[] args) {

String inputString = "Hello World";

int vowelCount = 0;

inputString = inputString.toLowerCase();

for (int i = 0; i < inputString.length(); i++) {

char currentChar = inputString.charAt(i);

if (Character.isLetter(currentChar) && isVowel(currentChar)) {

vowelCount++;

}

}

System.out.println("Vowel count: " + vowelCount);

}

private static boolean isVowel(char c) {

return c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u';

}

}

1. Write a program to check whether a given string is a palindrome or not by using:
2. StringBuffer Class
3. String Class

**Program A: StringBuffer Class**

package Practicals;

public class PalindromeCheckerWithBuffer {

public static void main(String[] args) {

String inputString = "madam";

StringBuffer sb = new StringBuffer(inputString);

StringBuffer reversedString = sb.reverse();

if (inputString.equals(reversedString.toString())) {

System.out.println("String is palindrome");

} else {

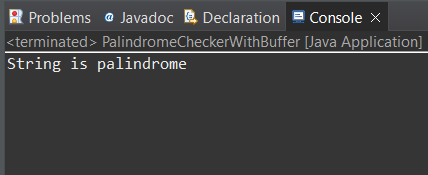
System.out.println("String is not palindrome");

}

}

}

**Output:**

****

**Program B: String Class**

package Practicals;

public class StringPalindrome {

public static void main(String[] args) {

String inputString = "madam";

String reversedString = "";

for (int i = inputString.length() - 1; i >= 0; i--) {

reversedString += inputString.charAt(i);

}

if (inputString.equals(reversedString)) {

System.out.println("String is palindrome");

} else {

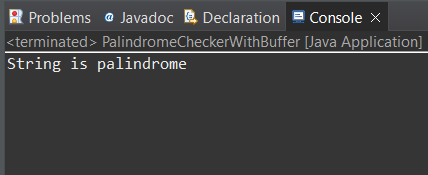
System.out.println("String is not palindrome");

}

}

}

**Output:**



1. Write a Java program to create List containing list of items and use ListIterator interface to print items present in the list. Also print the list in reverse/ backward direction

**CODE:**package collections;

import java.util.ArrayList;

import java.util.List;

import java.util.ListIterator;

public class List\_1 {

public static void main(String[] args) {

// TODO Auto-generated method stub

List<Integer> list1 = new ArrayList<Integer>();

list1.add(1);

list1.add(2);

list1.add(3);

list1.add(4);

ListIterator<Integer> li = list1.listIterator();

System.*out*.println("Forward direction iteration");

while(li.hasNext()) {

System.*out*.println(li.next());

}

System.*out*.println("Previous direction iteration");

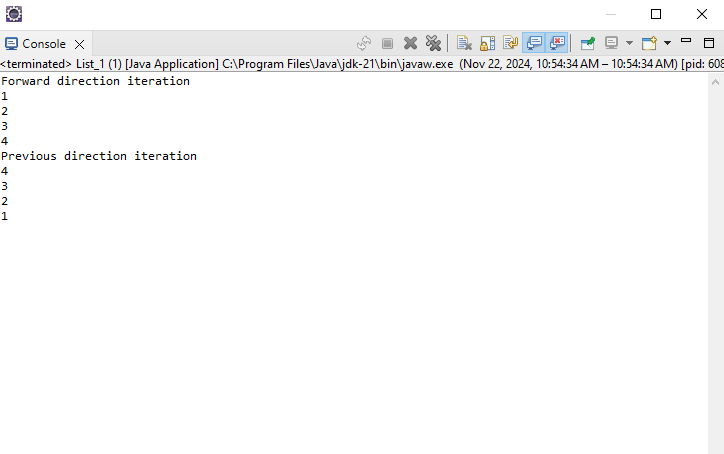
while(li.hasPrevious()) {

System.*out*.println(li.previous());

}

}

}

**OUTPUT:**

1. Write a Java program to create a Set containing list of items of type String and print the items in the list using the Iterator interface. Also print the list in reverse/ backward direction

**CODE:package collections;**

**import java.util.Set;**

**import java.util.TreeSet;**

**import java.util.Iterator;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.util.Collections;**

**public class Set\_2 {**

**public static void main(String[] args) {**

**// TODO Auto-generated method stub**

**Set<String> set1 = new TreeSet<String>();**

**set1.add("java");**

**set1.add("C++");**

**set1.add("Alpha");**

**Iterator<String> iterator = set1.iterator();**

**System.*out*.println("Forward Direction");**

**while(iterator.hasNext()) {**

**System.*out*.println(iterator.next());**

**}**

**List<String> l1= new ArrayList<String>(set1);**

**Collections.*reverse*(l1);**

**System.*out*.println("Backward Direction");**

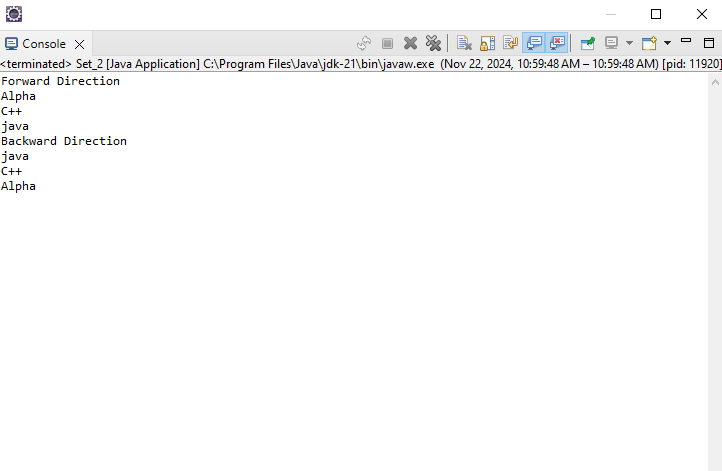
**for(String s1:l1) {**

**System.*out*.println(s1);**

**}**

**}**

**}**

**OUTPUT:**

1. Write a Java program using Set interface containing list of items and

perform the following operations:

a. Add items in the set.

b. Insert items of one set into another set.

c. Remove items from the set

d. Search the specified item in the set

**CODE:**package collections;

import java.util.Set;

import java.util.TreeSet;

public class Set\_3 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Set<String> langSet = new TreeSet<String>();

langSet.add("Riya");

langSet.add("bot");

Set<String> langSet2 = new TreeSet<String>();

langSet2.add("pooja");

langSet2.add("yogesh");

langSet2.add("Riya");

langSet.addAll(langSet2);

for(String l1: langSet){

System.*out*.println(l1);

}

System.*out*.println();

if(langSet.contains("pooja")) {

langSet.remove("pooja");

}

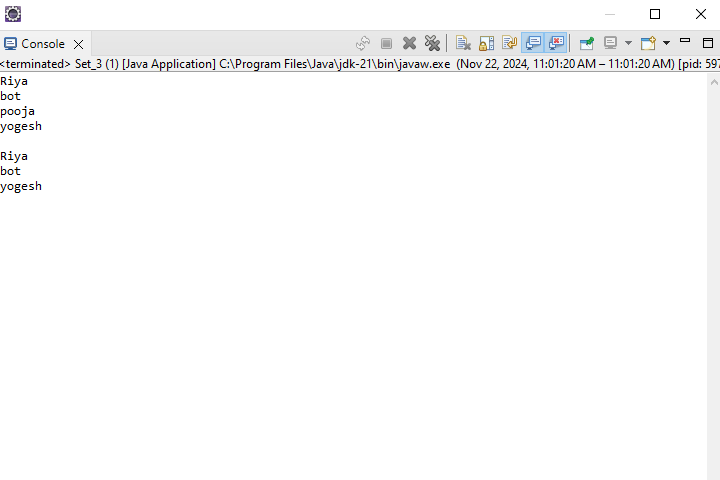
for(String l1:langSet) {

System.*out*.println(l1);

}

}

}

**OUTPUT:**

1. Write a Java program using Map interface containing list of items having keys and associated values and perform the following operations:

a.Add items in the map.

b.Remove items from the map

c.Search specific key from the map

d.Get value of the specified key

e.Insert map elements of one map into another map.

f. Print all keys and values of the map.

**CODE:**package collections;

import java.util.HashMap;

import java.util.Map;

public class Map\_4 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Map<Integer,String> map = new HashMap<Integer,String>();

map.put(1, "Riya");

map.put(2, "Ram");

map.put(3, "Richa");

map.forEach( (k,v) -> System.*out*.println(k+":"+v) );

if(map.containsKey(1)) {

map.remove(1);

}

System.*out*.println();

System.*out*.println("Map after removing Key");

map.forEach( (k,v) -> System.*out*.println(k+":"+v) );

System.*out*.println();

System.*out*.println("Get value of Key 3: "+map.get(3));

Map<Integer,String> map2 = new HashMap<Integer,String>();

map2.put(4, "Rose");

map2.put(5, "Maria");

map2.put(1, "Lily");

map2.put(2, "Chameli");

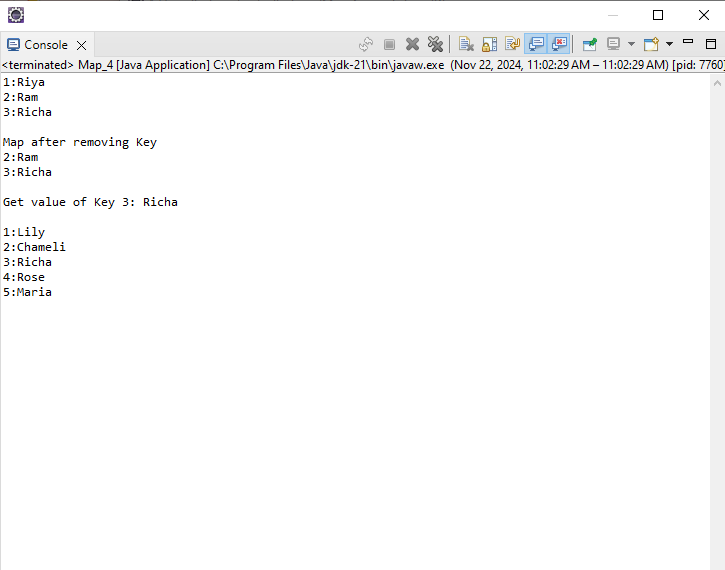
map.putAll(map2);

System.*out*.println();

map.forEach( (k,v) -> System.*out*.println(k+":"+v) );

}

}



1. WAP using Lambda Expression with multiple parameters to print addition of two numbers.

**CODE:**package collections;

interface Arithmetic{

public int add(int n1,int n2);

}

public class Lambda\_5 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Arithmetic arth = (a,b) -> a+b;

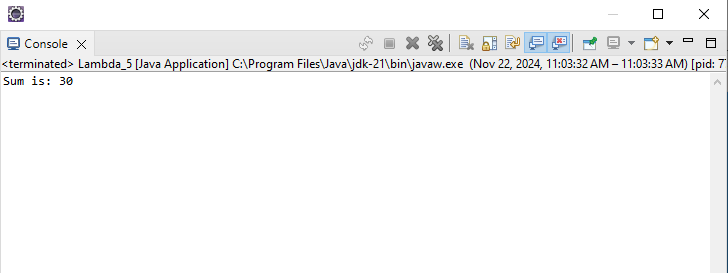
int sum ;

sum =arth.add(10, 20);

System.*out*.println("Sum is: "+sum);

}

}

**OUTPUT:**

1. WAP using Lambda Expression to calculate following:

a) Convert Fahrenheit to Celsius

b) Convert Kilometers to Meters

**CODE:package** collections;

**interface** Temperature{

**public** **void** ftoc(**double** f);

}

**interface** Distance{

**public** **int** kmtom(**int** km);

}

**public** **class** Lambda\_6 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Temperature t1 = (f) -> {

**double** c = (f-32)\*(5.0/9.0);

System.***out***.println("Temperature in celcius: "+c);

};

Distance d1 = (km)->{

**int** m = km\*1000;

**return** m;

};

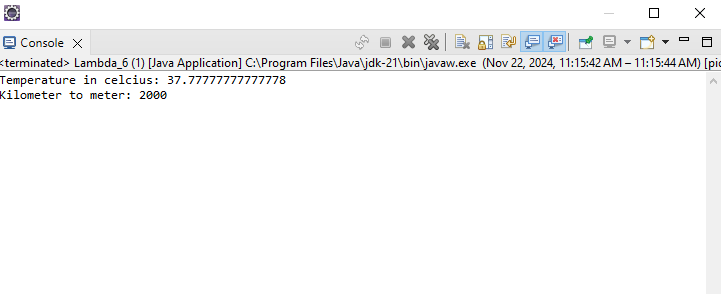
t1.ftoc(100);

System.***out***.println("Kilometer to meter: "+d1.kmtom(2));

}

}

**OUTPUT:**



1.Write a Spring application that demonstrates Dependency Injection using the constructor, implemented through XML-based configuration or annotation-based configuration

**CODE &OUTPUT:**

**App.java**

package module4;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

// TODO Auto-generated method stub

ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");

Student s = context.getBean("student", Student.class);

System.out.println(s.toString());

}

}

**Student.java**

**package module4;**

**public class Student {**

**private Fullname fullname;**

**private int id;**

**public Student(Fullname fullname, int id) {**

**this.fullname = fullname;**

**this.id = id;**

**}**

***@Override***

**public String toString() {**

**return "Student [fullname=" + fullname + ", id=" + id + "]";**

**}**

**}**

**Fullname.java**

**package module4;**

**public class Fullname {**

**private String firstname;**

**private String lastname;**

**public Fullname() {**

**super();**

**// TODO Auto-generated constructor stub**

**this.firstname = firstname;**

**this.lastname = lastname;**

**}**

**public String getFirstname() {**

**return firstname;**

**}**

**public void setFirstname(String firstname) {**

**this.firstname = firstname;**

**}**

**public String getLastname() {**

**return lastname;**

**}**

**public void setLastname(String lastname) {**

**this.lastname = lastname;**

**}**

***@Override***

**public String toString() {**

**return "Fullname [firstname=" + firstname + ", lastname=" + lastname + "]";**

**}**

**}**

**Config.xml**

**<?xml version=*"1.0"* encoding=*"UTF-8"*?>**

**<beans**

**xmlns=*"http://www.springframework.org/schema/beans"***

**xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"***

**xmlns:p=*"http://www.springframework.org/schema/p"***

**xsi:schemaLocation=*"http://www.springframework.org/schema/beans***

***http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>**

**<bean id = *"Fullname"* class = *"module4.Fullname"*>**

**<constructor-arg index = *"0"* value = *"yogesh"*></constructor-arg>**

**<constructor-arg index = *"1"* value = *"yadav"*></constructor-arg>**

**</bean>**

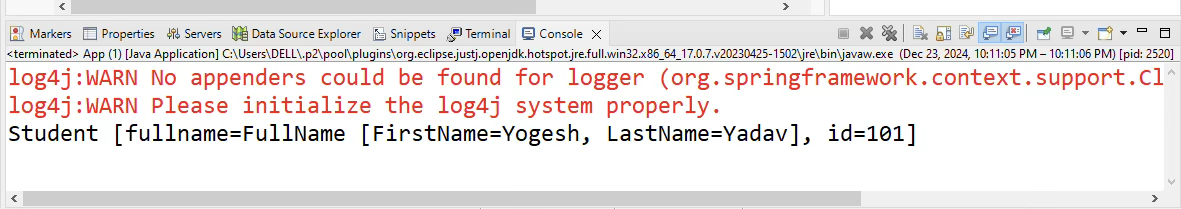
**<bean id = *"Student"* class = *"module4.Student"*>**

**<constructor-arg index= *"0"* ref = *"fullname"*></constructor-arg>**

**<constructor-arg index = *"1"* value = *"101"*></constructor-arg>**

**</bean>**

**</beans>**



2.Write a Spring application that demonstrates Dependency Injection using the setter method, implemented through XML-based configuration or annotation-based configuration

**CODE &OUTPUT:**

**Account.java**

package Spring.mod3;

public class Account {

private int accNo;

private String accName;

public int getAccNo() {

return accNo;

}

public void setAccNo(int accNo) {

this.accNo = accNo;

}

public String getAccName() {

return accName;

}

public void setAccName(String accName) {

this.accName = accName;

}

}

App.java

package Spring.mod3;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

// TODO Auto-generated method stub

ApplicationContext context=new ClassPathXmlApplicationContext("config.xml");

Account acc=context.getBean("account",Account.class);

System.out.println("Account No: "+acc.getAccNo());

System.out.println("Account Holder Name: "+acc.getAccName());

}

}

**Config.xml**

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans**

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<**bean** id = *"account"* class=*"Spring.mod3.Account"*>

<**property** name=*"accNo"* value=*"112"*></**property**>

<**property** name=*"accName"* value=*"Riya Maurya"*></**property**>

</**bean**>

</**beans**>

Write a program to insert, update and delete records from the Movie table.(Movie Application) Movie table(movie\_id , movie\_name)

**Movie.java**

package com.jdbc;

public class Movie {

private int id;

private String name;

// Generating getter setter and constructor

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Movie(int id, String name) {

super();

this.id = id;

this.name = name;

}

}

**MovieDAO.java**

**package com.jdbc;**

**import org.springframework.jdbc.core.JdbcTemplate;**

**public class MovieDAO {**

**private JdbcTemplate jdbcTemplate;**

**// Setter method for JdbcTemplate**

**public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {**

**this.jdbcTemplate = jdbcTemplate;**

**}**

**// To insert data in movie table**

**public int insertMovie(Movie m) {**

**String query = "INSERT INTO movie (id, name) VALUES (?, ?)";**

**return jdbcTemplate.update(query, m.getId(), m.getName());**

**}**

**// To update data in movie table**

**public int updateMovie(Movie m) {**

**String query = "UPDATE movie SET name=? WHERE id=?";**

**return jdbcTemplate.update(query, m.getName(), m.getId());**

**}**

**// To delete data from movie table**

**public int deleteMovie(Movie m) {**

**String query = "DELETE FROM movie WHERE id=?";**

**return jdbcTemplate.update(query, m.getId());**

**}**

**}**

**App.java**

**package com.jdbc;**

**import org.springframework.context.ApplicationContext;**

**import org.springframework.context.support.ClassPathXmlApplicationContext;**

**public class App {**

**public static void main(String[] args) {**

**// TODO Auto-generated method stub**

**ApplicationContext context=new ClassPathXmlApplicationContext("config.xml");**

**MovieDAO mdao = context.getBean("moviedao", MovieDAO.class);**

**Movie m = new Movie(1, "abc");**

**Movie m2 = new Movie(2, "xyz");**

**Movie m3 = new Movie(3, "hij");**

**System.out.println("Insertion of movie");**

**System.out.println(mdao.insertMovie(m));**

**System.out.println(mdao.insertMovie(m2));**

**System.out.println(mdao.insertMovie(m3));**

**System.out.println();**

**System.out.println("Updation of movie");**

**Movie m4 = new Movie(3, "xyw");**

**System.out.println(mdao.updateMovie(m4));**

**//System.out.println();**

**System.out.println("Deletion of Movie");**

**System.out.println(mdao.deleteMovie(m));**

**}**

**}**

Create Database

CREATE DATABASE movie\_db;

USE movie\_db;

CREATE TABLE movie (id int,name varchar(20));

Select \* from movie

**Config .xml**

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<**bean** id=*"ds"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<**property** name=*"driverClassName"* value=*"com.mysql.cj.jdbc.Driver"*></**property**>

<**property** name=*"url"* value=*"jdbc:mysql://localhost:3306/film"*></**property**>

<**property** name=*"username"* value=*"root"*></**property**>

<**property** name=*"password"* value=*"Riya@1234"*></**property**>

</**bean**>

<**bean** id=*"jdbcTemplate"* class=*"org.springframework.jdbc.core.JdbcTemplate"*>

<**property** name=*"dataSource"* ref=*"ds"*></**property**>

</**bean**>

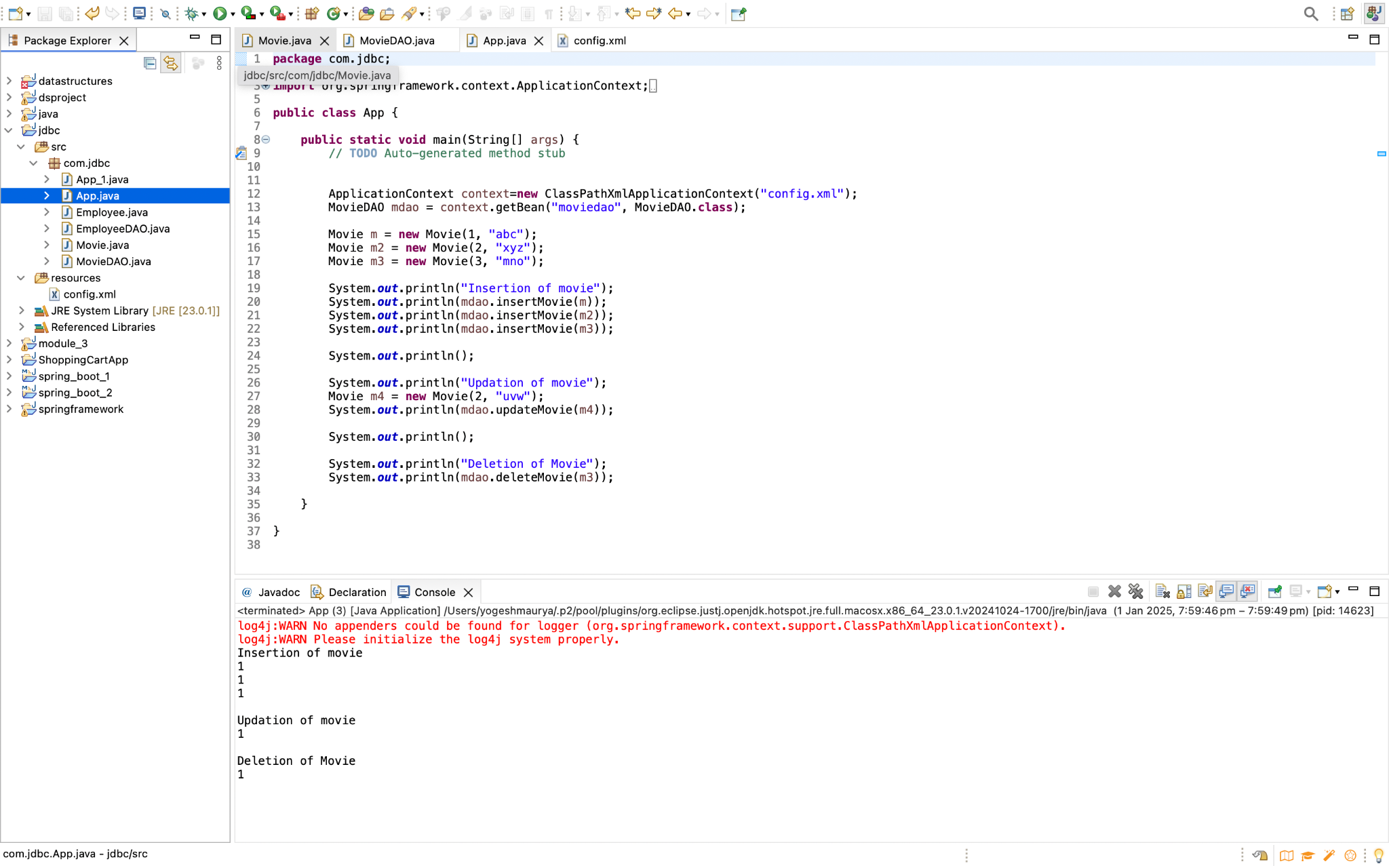
<**bean** id =*"moviedao"* class = *"com.jdbc.MovieDAO"*>

<**property** name=*"jdbcTemplate"* ref=*"jdbcTemplate"*></**property**>

</**bean**>

</**beans**>

**output:**

****

Write a program to demonstrate PreparedStatement in Spring JdbcTemplate.

Write a program to demonstrate the ResultSetExtractor interface to fetch the records from the database.

Write a program to demonstrate the RowMapper interface to fetch the records from the database.

**Create Database**

create database film;

use film;

create table employee(id int,name varchar(20),email varchar(20));

**Employee.java**

**package com.jdbc;**

**public class Employee {**

**private int id;**

**private String name;**

**private String email;**

**//Generate getter setter**

**public int getId() {**

**return id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public String getEmail() {**

**return email;**

**}**

**public void setEmail(String email) {**

**this.email = email;**

**}**

**public Employee(int id, String name, String email) {**

**super();**

**this.id = id;**

**this.name = name;**

**this.email = email;**

**}**

**public Employee() {**

**// TODO Auto-generated constructor stub**

**}**

**}**

**EmployeeDAO.java**

package com.jdbc;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import org.springframework.dao.DataAccessException;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.PreparedStatementCallback;

import org.springframework.jdbc.core.ResultSetExtractor;

import org.springframework.jdbc.core.RowMapper;

public class EmployeeDAO {

private JdbcTemplate jdbctemplate;

// Generate only Setter

public void setJdbctemplate(JdbcTemplate jdbctemplate) {

this.jdbctemplate = jdbctemplate;

}

// To insert data in employee table

public Boolean insertEmployee(Employee e) {

String query = "insert into employee values(?, ?, ?)";

return jdbctemplate.execute(query, new PreparedStatementCallback<Boolean>() {

@Override

public Boolean doInPreparedStatement(PreparedStatement arg0) throws SQLException, DataAccessException {

// TODO Auto-generated method stub

arg0.setInt(1, e.getId());

arg0.setString(2, e.getName());

arg0.setString(3, e.getEmail());

return arg0.execute();

}

});

}

public List<Employee> getAllEmployee() {

String query = "select \* from employee";

return jdbctemplate.query(query, new ResultSetExtractor<List<Employee>>() {

@Override

public List<Employee> extractData(ResultSet arg0) throws SQLException, DataAccessException {

// TODO Auto-generated method stub

List<Employee> employees = new ArrayList<Employee>();

while(arg0.next()) {

Employee e = new Employee();

e.setId(arg0.getInt(1));

e.setName(arg0.getString(2));

e.setEmail(arg0.getString(3));

employees.add(e);

}

return employees;

}

});

}

public List<Employee> getEmployees() {

String query = "select \* from employee";

List<Employee> employees = jdbctemplate.query(query, new RowMapper<Employee>() {

@Override

public Employee mapRow(ResultSet arg0, int arg1) throws SQLException{

Employee e = new Employee();

e.setId(arg0.getInt(1));

e.setName(arg0.getString(2));

e.setEmail(arg0.getString(3));

return e;

}

});

return employees;

}

}

**App.java**

package com.jdbc;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App\_1 {

public static void main(String[] args) {

// TODO Auto-generated method stub

ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");

EmployeeDAO edao = context.getBean("employeedao", EmployeeDAO.class);

Employee e = new Employee(1, "efg", "abc@gmail.com");

System.out.println("Adding Employee:");

System.out.println(edao.insertEmployee(e));

System.out.println();

System.out.println("Employee Data using ResultSetExtractor:");

List<Employee> ep1 = edao.getAllEmployee();

for(Employee e1: ep1) {

System.out.println(e1.getId()+ " : " +e1.getName()+ " : " +e1.getEmail());

}

System.out.println();

System.out.println("Employee Data using Rowmapper:");

List<Employee> ep2 = edao.getEmployees();

for(Employee e1: ep2) {

System.out.println(e1.getId()+ " : " +e1.getName()+ " : " +e1.getEmail());

}

}

}

**Config .xml**

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<**bean** id=*"ds"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<**property** name=*"driverClassName"* value=*"com.mysql.cj.jdbc.Driver"*></**property**>

<**property** name=*"url"* value=*"jdbc:mysql://localhost:3306/film"*></**property**>

<**property** name=*"username"* value=*"root"*></**property**>

<**property** name=*"password"* value=*"Riya@1234"*></**property**>

</**bean**>

<**bean** id=*"jdbcTemplate"* class=*"org.springframework.jdbc.core.JdbcTemplate"*>

<**property** name=*"dataSource"* ref=*"ds"*></**property**>

</**bean**>

<**bean** id = *"employeedao"* class =*"com.jdbc.EmployeeDAO"*>

<**property** name=*"jdbctemplate"* ref=*"jdbcTemplate"*></**property**>

</**bean**>

</**beans**>

Write a program to create a simple Spring Boot application that prints a message.

**SpringBoot1Application.java**

package com.example.spring\_boot\_1;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBoot1Application {

public static void main(String[] args) {

SpringApplication.run(SpringBoot1Application.class, args);

}

}

**MessageController.java**

package com.example.spring\_boot\_1;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

*@Controller*

public class MessageController {

*@GetMapping*("/message")

public String getMessage(Model m) {

m.addAttribute("msg", "Hello! welcome to spring boot");

return "message";

}

}

**Message.html**

<!DOCTYPE html>

<html>

<head>

<meta charset = "UTF-8"

<title>Insert title here</title>

</head>

<body>

<h1 th:text="${msg}">

</h1>

</body>

</html>

Write a program to demonstrate RESTful Web Services with spring boot.(Product Application)

**Product.java**

package com.example.spring\_boot\_2;

public class Product {

private String id;

private String name;

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**ProductController.java**

package com.example.spring\_boot\_2;

import java.util.HashMap;

import java.util.Map;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class ProductController {

private static Map<String,Product> products\_List= new HashMap<String,Product>();

static {

Product honey= new Product();

honey.setId("1");

honey.setName("Honey");

products\_List.put(honey.getId(), honey);

}

@RequestMapping(value="/products")

public ResponseEntity<Object> getProduct(){

return new ResponseEntity<Object>(products\_List.values(),HttpStatus.OK);

}

@RequestMapping(value = "/products",method = RequestMethod.POST)

public ResponseEntity<Object> addProduct(@RequestBody Product p){

products\_List.put(p.getId(), p);

return new ResponseEntity<Object>("product is added sucessfully",HttpStatus.OK);

}

@RequestMapping(value="/products/{id}",method = RequestMethod.PUT)

public ResponseEntity<Object> updateProduct(@PathVariable("id") String id,@RequestBody Product p){

products\_List.put(id, p);

return new ResponseEntity<Object>("Product is updated sucessfully",HttpStatus.OK);

}

@RequestMapping(value="/products/{id}",method = RequestMethod.DELETE)

public ResponseEntity<Object> deleteProduct(@PathVariable("id") String id){

products\_List.remove(id);

return new ResponseEntity<Object> ("Product is deleted sucessfully",HttpStatus.OK);

}

}

**SpringBoot2Application.java**

package com.example.spring\_boot\_2;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBoot2Application {

public static void main(String[] args) {

SpringApplication.run(SpringBoot2Application.class, args);

}

}

GET- localhost:8080/products

POST-localhost:8080/products

PUT-localhost:8080/products/id

DELETE-localhost:8080/products/id

Write a program demonstrating how to set up a CRUD (Create, Read, Update,Delete) application using Spring Framework and Hibernate ORM with annotations for entity mapping and dependency injection.

**HibernatedemoApplication.java**

package com.example.hibernatedemo\_1;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Hibernatedemo1Application {

public static void main(String[] args) {

SpringApplication.run(Hibernatedemo1Application.class, args);

}

}

**Employee.java**

package com.example.hibernatedemo\_1;

import jakarta.persistence.\*;

*@Entity*

*@Table*(name = "employee")

public class Employee {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private Long id;

*@Column* (name = "name")

private String name;

*@Column* (name = "department")

private String department;

*@Column* (name = "salary")

private Double salary;

public Long getId() {

return id;

}

//GENERATE GETTER SETTER

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

public Double getSalary() {

return salary;

}

public void setSalary(Double salary) {

this.salary = salary;

}

public void setId(Long id) {

this.id = id;

}

}

**EmployeeRepository**

package com.example.hibernatedemo\_1;

import java.util.List;

import org.springframework.stereotype.Repository;

import jakarta.persistence.EntityManager;

import jakarta.persistence.PersistenceContext;

import jakarta.transaction.Transactional;

*@Repository*

*@Transactional*

public class EmployeeRepository {

*@PersistenceContext*

private EntityManager entityManager;

//create and update

public Employee save(Employee e) {

if (e.getId()==null) {

entityManager.persist(e);

}

else {

entityManager.merge(e);

}

return e;

}

//get employee by id

public Employee findById(Long id) {

return entityManager.find(Employee.class,id);

}

//get all Employee Data

public List<Employee> findAll(){

return entityManager.createQuery("from Employee",Employee.class).getResultList();

}

//Delete employee

public void deleteById(Long id) {

Employee e = entityManager.find(Employee.class, id);

if(e!=null) {

entityManager.remove(e);

}

}

}

**EmployeeService.java**

package com.example.hibernatedemo\_1;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

*@Service*

public class EmployeeService {

*@Autowired*

private EmployeeRepository er;

//create and update

public Employee saveEmployee(Employee e) {

return er.save(e);

}

//get by id

public Employee getEmployeeById(Long id) {

return er.findById(id);

}

//get all employees

public List<Employee> getAllEmployee(){

return er.findAll();

}

//Delete all Employees

public void deleteEmployee(Long id) {

er.deleteById(id);

}

}

**EmployeeController.java**

package com.example.hibernatedemo\_1;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

*@RestController*

*@RequestMapping*("/api/employees")

public class EmployeeController {

*@Autowired*

private EmployeeService es;

//create and update

*@PostMapping*

public ResponseEntity<Employee> saveEmployee(*@RequestBody* Employee e) {

return ResponseEntity.*ok*(es.saveEmployee(e));

}

//Get By Id

*@GetMapping*

public ResponseEntity<Employee> getEmployeeById(*@PathVariable* Long id) {

Employee e = es.getEmployeeById(id);

if(e!=null) {

return ResponseEntity.*ok*(e);

}

else {

return ResponseEntity.*notFound*().build();

}

}

//get all employee

public ResponseEntity<List<Employee>> getAllEmployees() {

return ResponseEntity.*ok*(es.getAllEmployee());

}

//delete employee

*@DeleteMapping*("/{id}")

public ResponseEntity<Void> deleteEmployee(*@PathVariable* Long id){

es.deleteEmployee(id);

return ResponseEntity.*noContent*().build();

}

}

**Application.properties**

spring.application.name=hibernatedemo\_1

spring.datasource.url = jdbc:mysql://localhost:3306/college

spring.datasource.username = root

spring.datasource.password = Riya@1234

spring.jpa.hibernate.ddl-auto = update

spring.jpa.show-sql = true

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQLDialect

**Database**

**create database college;**

**use college;**

**create table employee(id int AUTO\_INCREMENT PRIMARY KEY,name varchar(20),department varchar(20),salary float(10,2));**

**insert into employee values(null,'Sandhya','MCA',1000.00);**