Kartik Verma

CSE-2C

# Java Lab File

**Lab 7:-** Construct java program using Java I/O package.

**ALGORITHM:**

1. **Set Up Your Project Structure**
   - Create a directory structure for your project.
2. **Create the Main Program**
   - Define the main class and methods for reading from and writing to files.
3. **Implement File Reading**
   - Use Java I/O classes to read content from a file.
4. **Implement File Writing**
   - Use Java I/O classes to write content to a file.

**CODE:**

```
import java.io.*;

public class Main {     public static

void main(String[] args) {

        String inputFilePath = "data/input.txt";

        String outputFilePath = "data/output.txt";


try {

            String content = readFile(inputFilePath);

writeFile(outputFilePath, content);

            System.out.println("File content copied successfully.");
```

```
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
    private static String readFile(String filePath) throws IOException {
        StringBuilder content = new StringBuilder();
        BufferedReader reader = new BufferedReader(new
FileReader(filePath));
        String line;        while ((line =
reader.readLine()) != null) {
content.append(line).append("\n");
        }
        reader.close();
return content.toString();
    }
    private static void writeFile(String filePath, String content) throws
IOException {        BufferedWriter writer = new BufferedWriter(new
FileWriter(filePath));        writer.write(content);        writer.close();
    }
}
```

**OUTPUT**

```
Hello, this is a test file.
It contains multiple lines of text.
End of file.
```

Kartik Verma

CSE-2C

# Java Lab File

**Lab 8:-** Create industry oriented application using Spring Framework.

**Software Used:-** VS Code

**Code:-**

**ALGORITHM:**

1. **Set Up Your Development Environment**
   - Install Java Development Kit (JDK)
   - Install an Integrated Development Environment (IDE)
   - Set up a build tool (Maven or Gradle)
2. **Initialize the Spring Boot Project**
   - Use Spring Initializr to create a new Spring Boot project
3. **Create the Project Structure**
   - Define the necessary packages and classes
4. **Set Up the Application Properties**
   - Configure application properties
5. **Create the Domain Model**
   - Define the entities and data models
6. **Set Up the Repository Layer**
   - Create repositories for data access
7. **Implement the Service Layer**
   - Define the business logic
8. **Create the Controller Layer**

- Implement RESTful endpoints

9. **Test the Application**

   - Write unit and integration tests

10. **Run the Application**

    - Run the application and test the endpoints

11. **Package and Deploy the Application**

    - Package the application into a deployable unit
    - Deploy to a web server or cloud platform

**CODE:**

```
import org.springframework.boot.SpringApplication;

import
org.springframework.boot.autoconfigure.SpringBootApplicati
on;

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

import
org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

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

import javax.persistence.*;

import java.util.List;

@SpringBootApplication public
class DemoApplication {     public
static void main(String[] args) {
```

```
        SpringApplication.run(DemoApplication.class, args);
    }
}
// Domain
Model
@Entity class
User {
    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)     private Long id;
private String name;     private String email;

    // Getters and setters     public Long getId() {
return id; }     public void setId(Long 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;}
```

Now, // Repository Layer @Repository interface UserRepository extends JpaRepository<User, Long> { }

```
// Service Layer
@Service class
UserService {
@Autowired     private
UserRepository
userRepository;
```

```
    public List<User>
getAllUsers() {
return
userRepository.findAll();
    }
    public User getUserById(Long
id) {        return
userRepository.findById(id).orEls
e(null);
    }
    public User
saveUser(User user) {
return
userRepository.save(user)
;
    }
    public void deleteUser(Long id) {
userRepository.deleteById(id);
    }
}
// Controller Layer
@RestController
@RequestMappin
g("/users") class
UserController {
@Autowired
private
UserService
userService;
```

```
    @GetMapping
    public List<User>
    getAllUsers() {
    return
    userService.getAllUs
    ers();


    }
    @GetMapping("/{id}")    public
    User getUserById(@PathVariable
    Long id) {        return
    userService.getUserById(id);
    }

    @PostMapping    public User
    createUser(@RequestBody User
    user) {        return
    userService.saveUser(user);
    }


    @DeleteMapping("/{id}")
    public void
    deleteUser(@PathVariable Long
    id) {
    userService.deleteUser(id);
    }
}
```