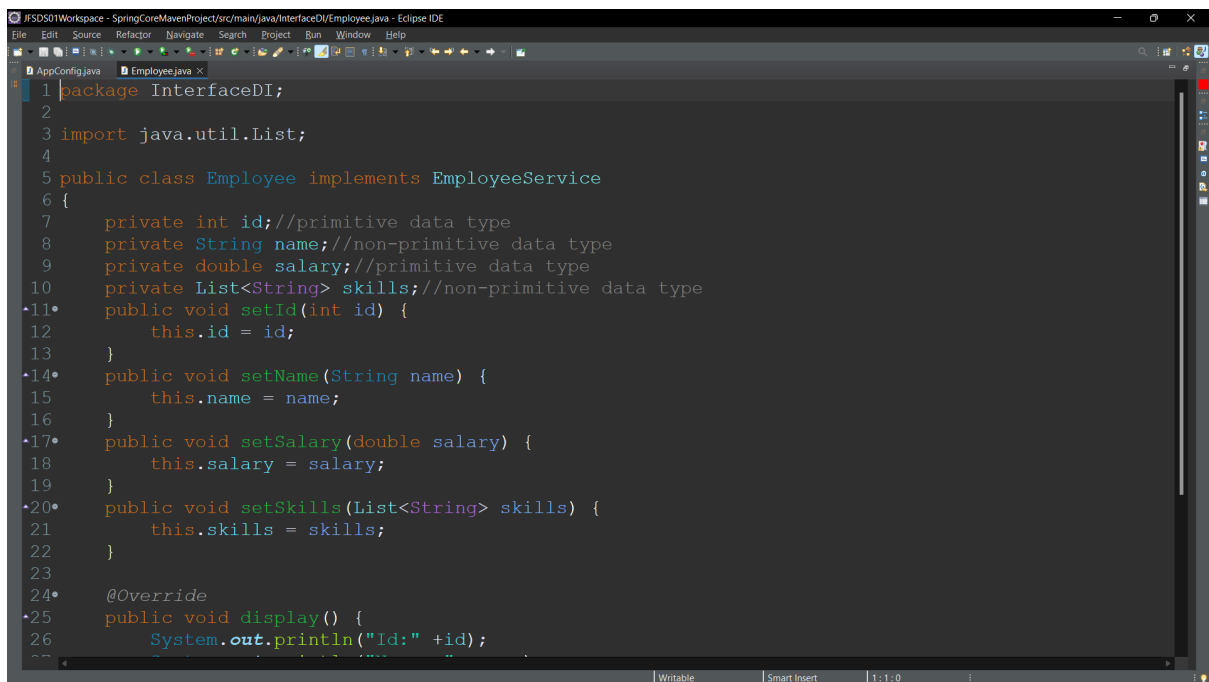
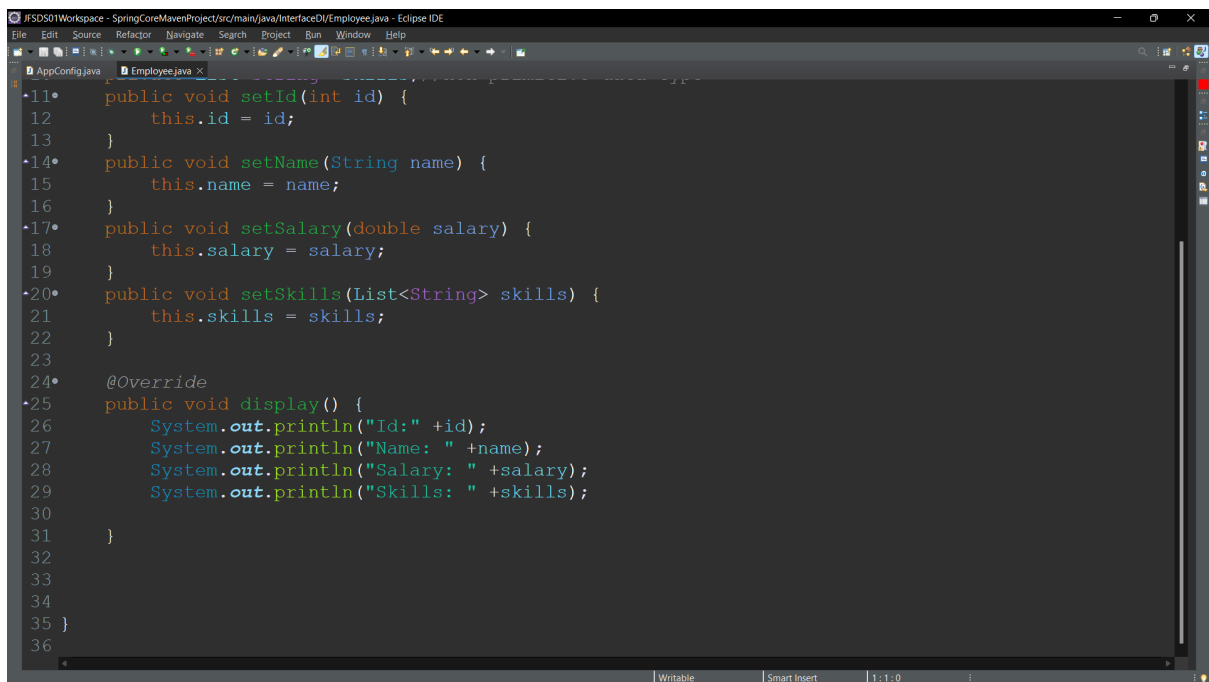


Interface Dependency Injection:

Employee.java

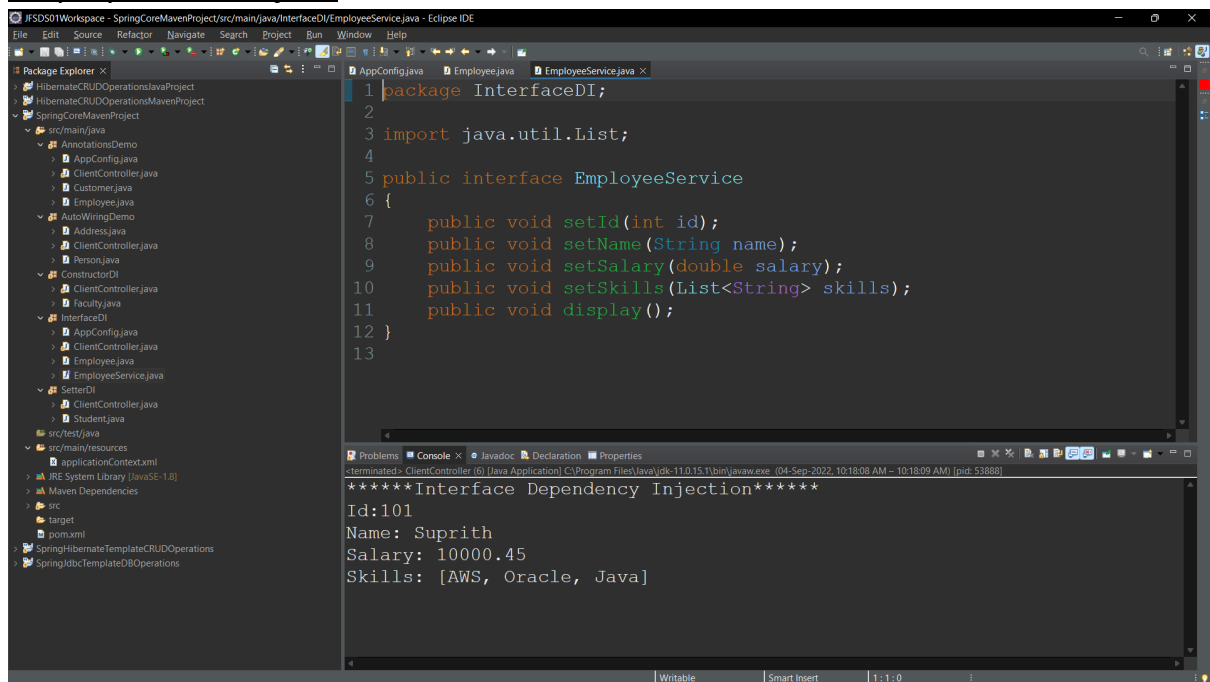
A screenshot of the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The project explorer on the left shows a workspace named 'JFSDS01Workspace' containing a 'SpringCoreMavenProject' with a source folder 'src/main/java/InterfaceDI/Employee.java'. The editor window displays the code for 'Employee.java'. The code defines a package 'InterfaceDI', imports 'java.util.List', and implements the 'EmployeeService' interface. It includes private fields for 'id', 'name', 'salary', and 'skills', along with corresponding setter methods and an overridden 'display()' method.

```
1 package InterfaceDI;
2
3 import java.util.List;
4
5 public class Employee implements EmployeeService
6 {
7     private int id;//primitive data type
8     private String name;//non-primitive data type
9     private double salary;//primitive data type
10    private List<String> skills;//non-primitive data type
11    public void setId(int id) {
12        this.id = id;
13    }
14    public void setName(String name) {
15        this.name = name;
16    }
17    public void setSalary(double salary) {
18        this.salary = salary;
19    }
20    public void setSkills(List<String> skills) {
21        this.skills = skills;
22    }
23
24    @Override
25    public void display() {
26        System.out.println("Id: " +id);
27    }
28 }
```

A continuation of the Eclipse IDE screenshot, showing the same code editor window. The code continues from the previous block, showing the rest of the 'display()' method and the closing braces for the class and the 'display()' method.

```
26        System.out.println("Id: " +id);
27        System.out.println("Name: " +name);
28        System.out.println("Salary: " +salary);
29        System.out.println("Skills: " +skills);
30    }
31 }
32
33
34
35 }
36 }
```

EmployeeService.java



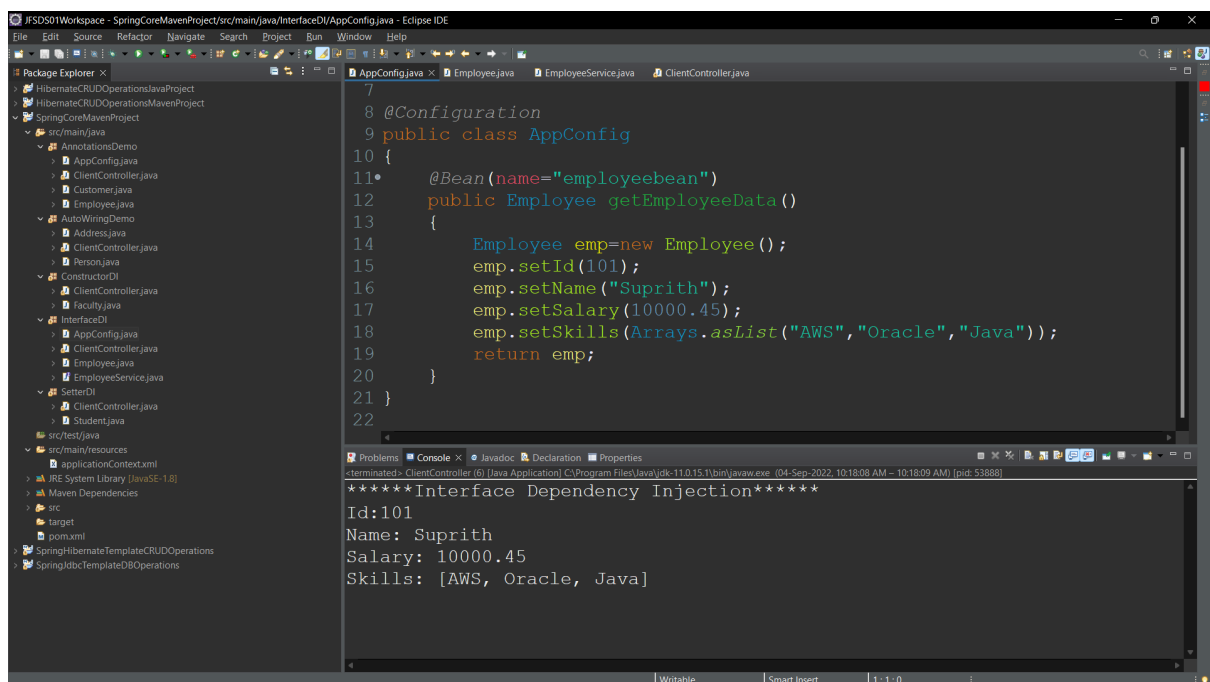
The screenshot shows the Eclipse IDE with the `EmployeeService.java` file open. The Package Explorer on the left shows the project structure, including `src/main/java` and `src/test/java`. The main editor displays the following code:

```
1 package InterfaceDI;
2
3 import java.util.List;
4
5 public interface EmployeeService
6 {
7     public void setId(int id);
8     public void setName(String name);
9     public void setSalary(double salary);
10    public void setSkills(List<String> skills);
11    public void display();
12 }
13
```

The Console window at the bottom shows the output of the application, indicating that the `EmployeeService` interface is being used for dependency injection:

```
*****Interface Dependency Injection*****
Id:101
Name: Suprith
Salary: 10000.45
Skills: [AWS, Oracle, Java]
```

AppConfig.java



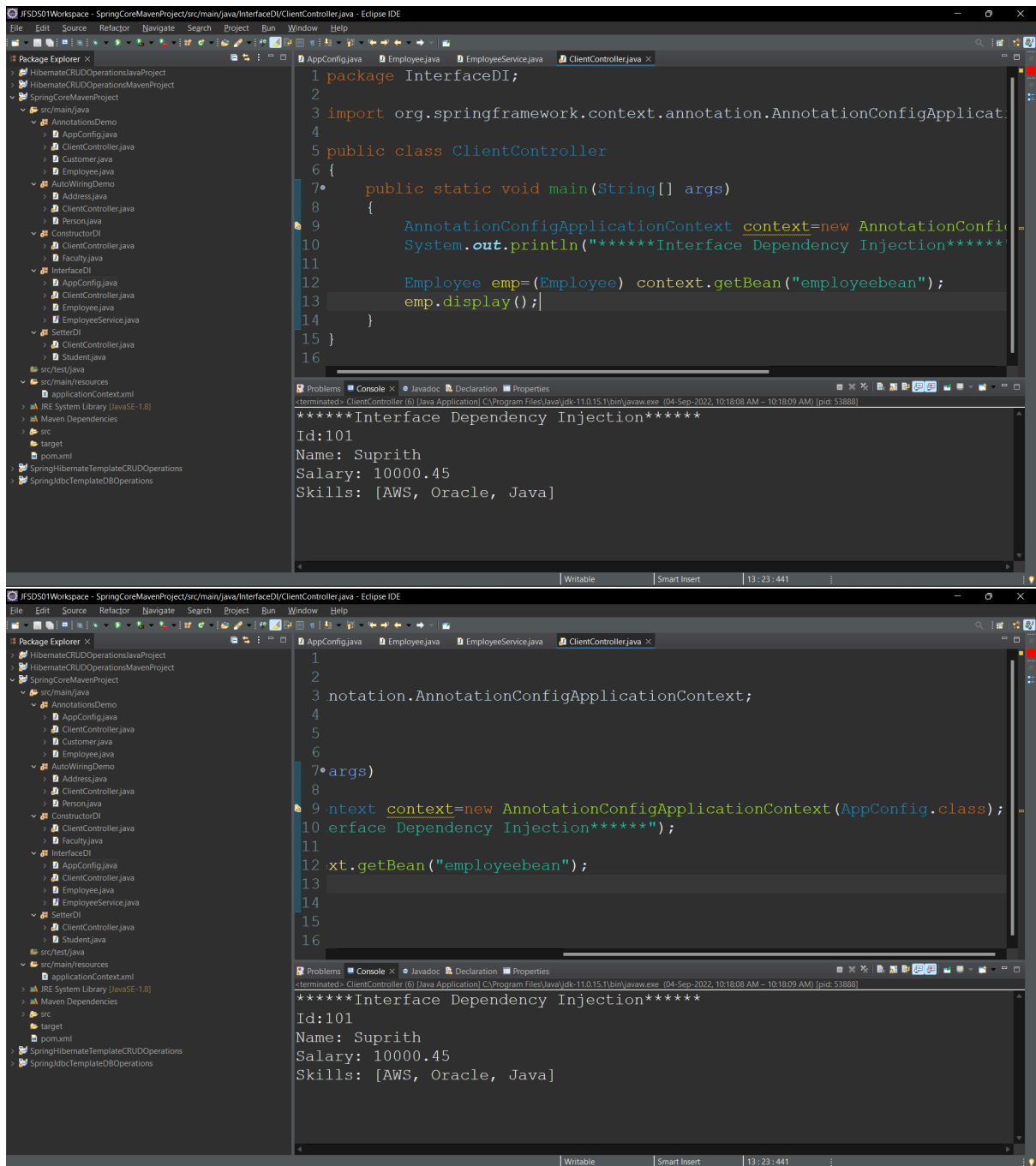
The screenshot shows the Eclipse IDE with the `AppConfig.java` file open. The Package Explorer on the left shows the project structure, including `src/main/java` and `src/test/java`. The main editor displays the following code:

```
7
8 @Configuration
9 public class AppConfig
10 {
11     @Bean(name="employeebean")
12     public Employee getEmployeeData()
13     {
14         Employee emp=new Employee();
15         emp.setId(101);
16         emp.setName("Suprith");
17         emp.setSalary(10000.45);
18         emp.setSkills(Arrays.asList("AWS","Oracle","Java"));
19         return emp;
20     }
21 }
22
```

The Console window at the bottom shows the output of the application, indicating that the `AppConfig` class is being used for dependency injection:

```
*****Interface Dependency Injection*****
Id:101
Name: Suprith
Salary: 10000.45
Skills: [AWS, Oracle, Java]
```

ClientController.java



Output:

