Name: Kunal Sanjay Ekare

Batch-RollNo : B2-37

Subject :Software Lab-2

Date: 11-march-2025

Aim: Create class Address, Student, College, CollegeNameMapping with data members as followed:
Address private String body,
private String city,
private String state;
private int pin;
Student-private String name;
private int roll:
private Address addr,
private ArrayList<Address> alt address;
private ArrayList String name_of_siblings;

College
• String cid;
List Address offices;
College NameMapping-HashMap<String, College mapping:
b1.xml-Create one bean each using setter and constructor injection
b2.xml-Use autowiring byName and by Type to autowire the given classes
b3.xml.-Create class Employee with data members company, dept, name, salary

Create 5 beans with company RCOEM and dept
• CSE
Use bean inheritance using setter injection to reduce code redundancy.

Thoery: Class Design
• Address represents a location with attributes: body, city, state, and pin.
• Student has personal details (name, roll) and a relationship with Address. It also includes a list of alternate addresses and sibling names.
• College maintains an id (cid) and a list of Address objects as office locations.
• CollegeNameMapping uses a HashMap<String, College> to store and retrieve college objects by name.

Spring XML Configurations

• b1.xml: Uses both setter and constructor injection to create and initialize beans.
• b2.xml: Implements autowiring to automatically inject dependencies using byName and byType, reducing manual bean configuration.
• b3.xml: Introduces an Employee class with properties company, dept, name, and salary.
o Defines five employee beans under RCOEM and CSE.
o Uses bean inheritance with setter injection to avoid redundant property definitions.

Code File

Student.java

```
package in.beans;

import java.util.ArrayList;

public class Student {
    private String name;
    private int roll;
    private Address addr;
    private ArrayList<Address> altAddress;
    private ArrayList<String> nameOfSiblings;
    // Default constructor (needed for setter-based injection)
    public Student (){}

    // Constructor for constructor-based injection
    public Student(String name,int roll,Address 
addr,ArrayList<Address>altAddress,ArrayList<String> nameOfSiblings){
        this.name = name;
        this.roll = roll;
        this.addr = addr;
        this.altAddress = altAddress;
        this.nameOfSiblings = nameOfSiblings;
    }
    // Setters
    public void setName(String name) { this.name = name; }
    public void setRoll(int roll) { this.roll = roll; }
    public void setAddr(Address addr) { this.addr = addr; }
    public void setAltAddress(ArrayList<Address> altAddress) { this.altAddress =
altAddress; }
    public void setNameOfSiblings(ArrayList<String> nameOfSiblings) {
this.nameOfSiblings = nameOfSiblings; }

    @Override
    public String toString(){
        return "Student: " + name + ", Roll: " + roll + ", Address: " + addr;
    }
}
```

Address.java

```
package in.beans;

public class Address {
    private String Body;
    private String City;
    private String state;
    private int pin;

    public Address() {
    }

    public Address(String Body, String City, String state, int pin) {
        this.Body = Body;
        this.City = City;
        this.state = state;
        this.pin = pin;
    }
    // Getters and Setters
    public void setBody(String Body) {
        this.Body = Body;
    }

    public void setCity(String City) {
        this.City = City;
    }

    public void setState(String state) {
        this.state = state;
    }

    public void setPin(int pin) {
        this.pin = pin;
    }

    @Override
    public String toString() {
        return Body + "," + City + "," + state + "-" + pin;
    }
}
```

College.java

```
 package in.beans;

import java.util.List;

public class College {
    private String cid;
    private List<Address>offices;
    // Default constructor (needed for setter-based injection)
    public College() {
       // System.out.println("College default constructor called.");
    }

    // Constructor for constructor-based injection
    public College(String cid, List<Address> offices) {
        this.cid = cid;
        this.offices = offices;
    }

    // Getters and setters
    public String getCid() {
        return cid;
    }
    public List<Address> getOffices() {
        return offices;
    }
    public void setCid(String cid){
        this.cid=cid;
    }
    public void setOffices(List<Address>offices){
        this.offices=offices;
    }
    @Override
    public String toString(){
        return "College ID: "+cid+" Offices : "+offices;
    }
}
```

CollegeNameMapping.java

```
package in.beans;

import java.util.HashMap;

public class CollegeNameMapping {
    private HashMap<String,College>mapping;
    public void setMapping(HashMap<String,College>mapping){
        this.mapping=mapping;
    }
    @Override
    public String toString(){
        return "CollegeNameMapping: "+mapping;
    }

}
```

Employee.java

```
package in.beans;
```

```java
public class Employee {
    private String company;
    private String dept;
    private String name;
    private double salary;

    public void setCompany(String company) { this.company = company; }
    public void setDept(String dept) { this.dept = dept; }
    public void setName(String name) { this.name = name; }
    public void setSalary(double salary) { this.salary = salary; }

    @Override
    public String toString() {
        return "Employee: " + name + ", Dept: " + dept + ", Company: " + company +
", Salary: " + salary;
    }
}
```

B1.xml

```xml
<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.xsd">

       <!-- Addresss Beans -->
       <bean id="address1" class="in.beans.Address">
           <property name="body" value="123 Street"/>
           <property name="city" value="Nagpur"/>
           <property name="state" value="Maharastra"/>
           <property name="pin" value="440013"/>
       </bean>

    <!-- Student Beans-->
    <bean id="student1" class="in.beans.Student">
        <property name="name" value="Kunal Ekare"/>
        <property name="roll" value="37"/>
        <property name="addr" ref="address1"/>
    </bean>
    <!--Constructor injection-->
    <bean id="college1" class="in.beans.College">
        <constructor-arg value="ekareks@rknec.edu"/>
        <constructor-arg>
            <list>
                <ref bean="address1"/>
            </list>
        </constructor-arg>
    </bean>
    <bean id="collegeMapping" class="in.beans.CollegeNameMapping">
        <property name="mapping">
                <map>
                    <entry key="RCOEM" value-ref="college1"/>
                </map>
        </property>
    </bean>
</beans>
```

B2.xml

```
<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.xsd">

    <!-- Address Bean -->
    <bean id="address" class="in.beans.Address">
        <property name="body" value="Gittikhadan Road"/>
        <property name="city" value="Noida"/>
        <property name="state" value="Noida"/>
        <property name="pin" value="110001"/>
    </bean>

    <!-- Student Bean (Autowiring byName) -->
    <bean id="student2" class="in.beans.Student" autowire="byName">
```

```
        <property name="name" value="Kunal Ekare"/>
        <property name="roll" value="37"/>
    </bean>

    <!-- College Bean (Autowiring byType) -->
    <bean id="college2" class="in.beans.College" autowire="byType"/>
</beans>
```

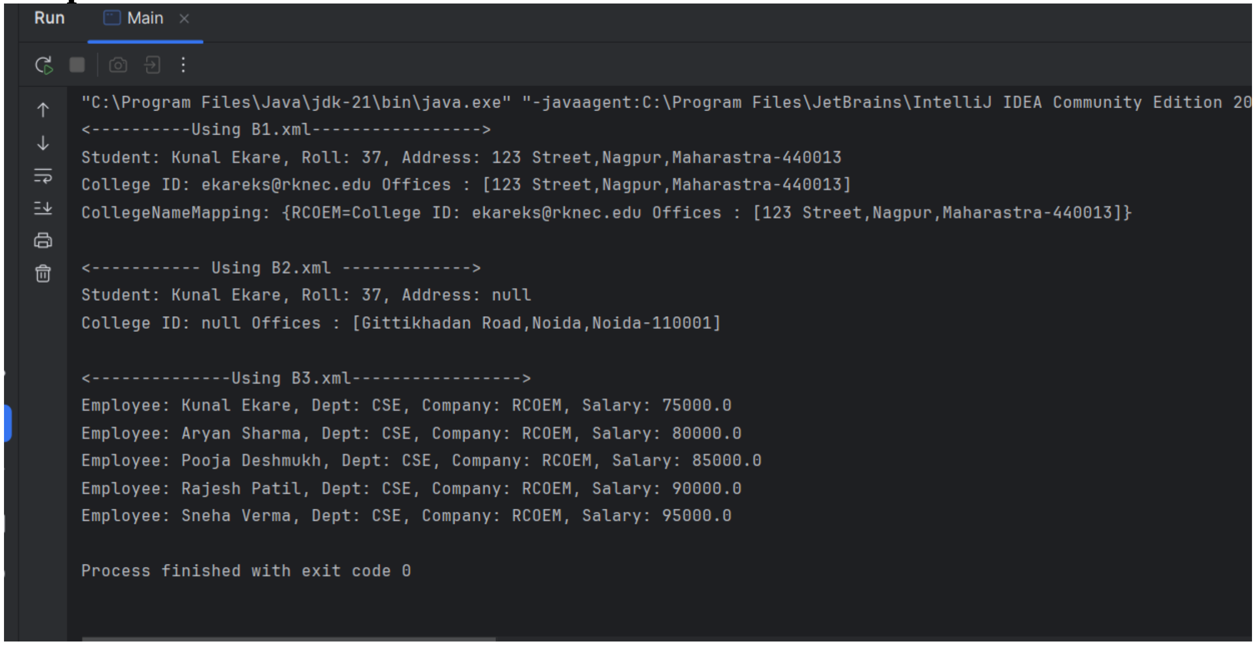
B3.xml

```
<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.xsd">

    <bean id="baseEmployee" class="in.beans.Employee" abstract="true">
        <property name="company" value="RCOEM"/>
        <property name="dept" value="CSE"/>
    </bean>

    <bean id="employee1" parent="baseEmployee">
        <property name="name" value="Kunal Ekare"/>
        <property name="salary" value="75000"/>
    </bean>

    <bean id="employee2" parent="baseEmployee">
        <property name="name" value="Aryan Sharma"/>
        <property name="salary" value="80000"/>
    </bean>

    <bean id="employee3" parent="baseEmployee">
        <property name="name" value="Pooja Deshmukh"/>
        <property name="salary" value="85000"/>
    </bean>

    <bean id="employee4" parent="baseEmployee">
        <property name="name" value="Rajesh Patil"/>
        <property name="salary" value="90000"/>
    </bean>
    <bean id="employee5" parent="baseEmployee">
        <property name="name" value="Sneha Verma"/>
        <property name="salary" value="95000"/>
    </bean>
</beans>
```

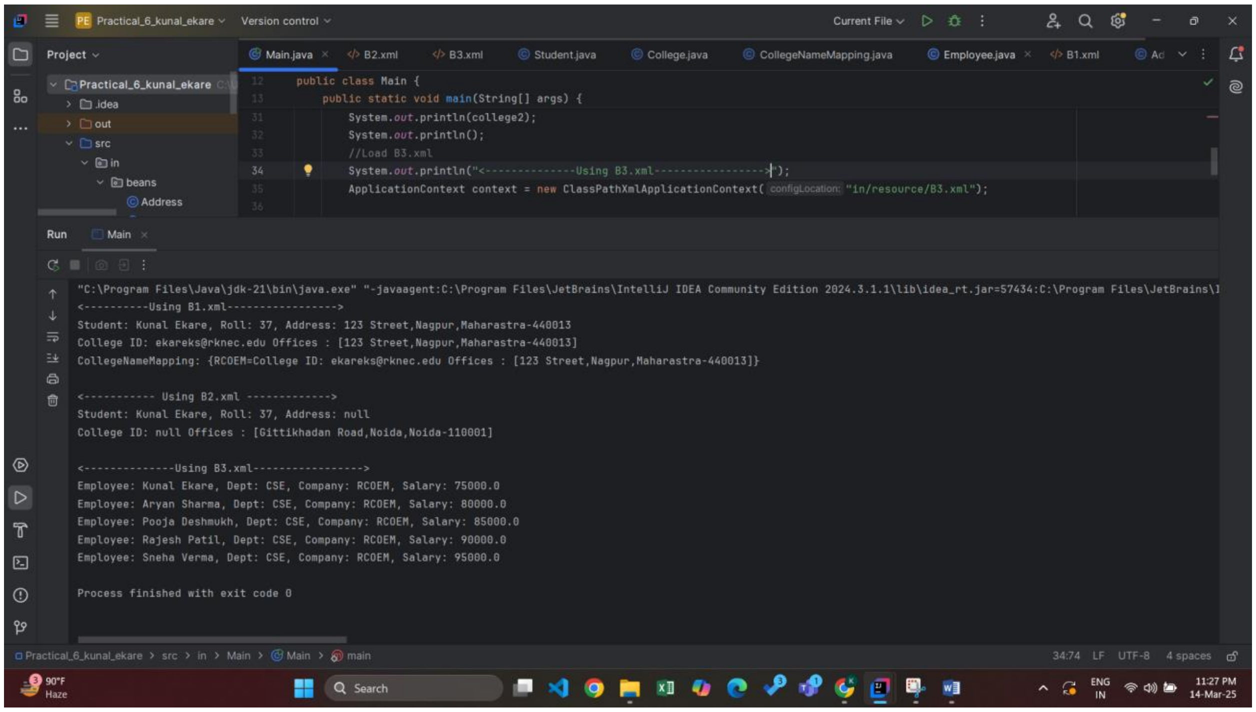
Main.java

```
package in.Main;

import in.beans.College;
import in.beans.CollegeNameMapping;
import in.beans.Student;
import in.beans.Employee;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        ApplicationContext context1=new
ClassPathXmlApplicationContext("/in/resource/B1.xml");
        Student student1=(Student) context1.getBean("student1");
        College college1=(College) context1.getBean("college1");
        CollegeNameMapping mapping=(CollegeNameMapping)
context1.getBean("collegeMapping");

        System.out.println("<----------Using B1.xml---------------->");
        System.out.println(student1);
        System.out.println(college1);
        System.out.println(mapping);

        // Load B2.xml (Autowiring)
        ApplicationContext context2 = new
ClassPathXmlApplicationContext("/in/resource/B2.xml");
        Student student2 = (Student) context2.getBean("student2");
        College college2 = (College) context2.getBean("college2");
        System.out.println("\n<----------- Using b2.xml ------------->");
        System.out.println(student2);
        System.out.println(college2);

        //Load B3.xml
        ApplicationContext context = new
ClassPathXmlApplicationContext("in/resource/B3.xml");

        // Retrieving Employee Beans
        Employee emp1 = (Employee) context.getBean("employee1");
        Employee emp2 = (Employee) context.getBean("employee2");
        Employee emp3 = (Employee) context.getBean("employee3");
        Employee emp4 = (Employee) context.getBean("employee4");
        Employee emp5 = (Employee) context.getBean("employee5");

        // Printing Employees
        System.out.println(emp1);
        System.out.println(emp2);
        System.out.println(emp3);
        System.out.println(emp4);
        System.out.println(emp5);

    }
}
```

Output:



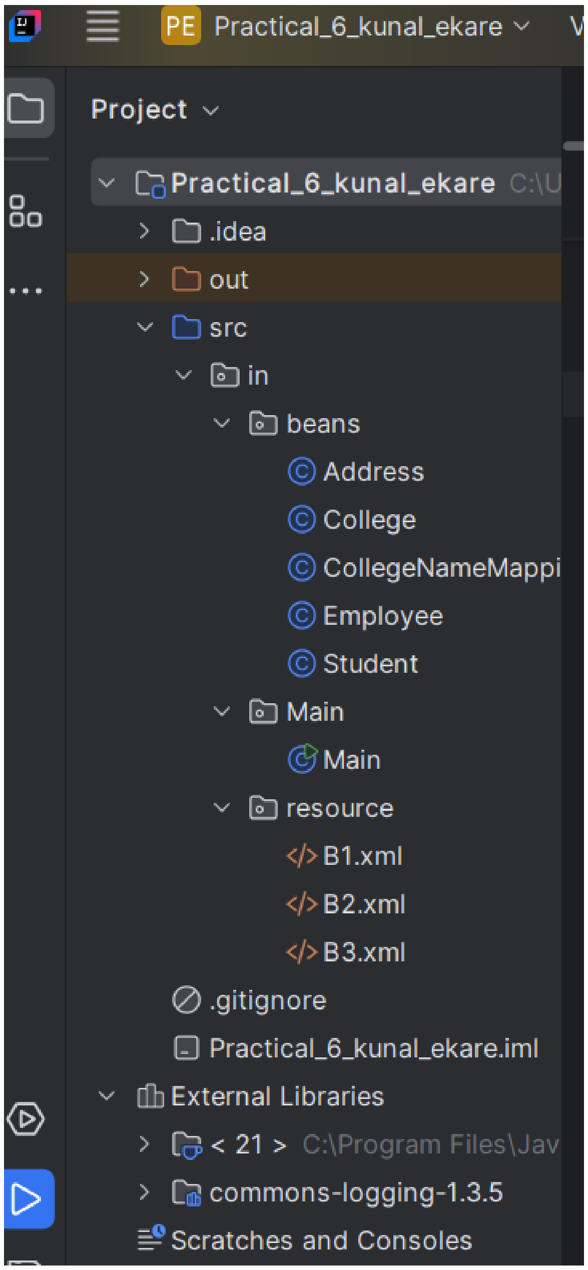
```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 20
<----------Using B1.xml---------------->
Student: Kunal Ekare, Roll: 37, Address: 123 Street,Nagpur,Maharastra-440013
College ID: ekareks@rknec.edu Offices : [123 Street,Nagpur,Maharastra-440013]
CollegeNameMapping: {RCOEM=College ID: ekareks@rknec.edu Offices : [123 Street,Nagpur,Maharastra-440013]}

<----------- Using B2.xml ------------->
Student: Kunal Ekare, Roll: 37, Address: null
College ID: null Offices : [Gittikhadan Road,Noida,Noida-110001]

<--------------Using B3.xml----------------->
Employee: Kunal Ekare, Dept: CSE, Company: RCOEM, Salary: 75000.0
Employee: Aryan Sharma, Dept: CSE, Company: RCOEM, Salary: 80000.0
Employee: Pooja Deshmukh, Dept: CSE, Company: RCOEM, Salary: 85000.0
Employee: Rajesh Patil, Dept: CSE, Company: RCOEM, Salary: 90000.0
Employee: Sneha Verma, Dept: CSE, Company: RCOEM, Salary: 95000.0

Process finished with exit code 0
```



```java
public class Main {
    public static void main(String[] args) {
        System.out.println(college2);
        System.out.println();
        //Load B3.xml
        System.out.println("<--------------Using B3.xml----------------->");
        ApplicationContext context = new ClassPathXmlApplicationContext( configLocation: "in/resource/B3.xml");
```

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.1.1\lib\idea_rt.jar=57434:C:\Program Files\JetBrains\I
<----------Using B1.xml---------------->
Student: Kunal Ekare, Roll: 37, Address: 123 Street,Nagpur,Maharastra-440013
College ID: ekareks@rknec.edu Offices : [123 Street,Nagpur,Maharastra-440013]
CollegeNameMapping: {RCOEM=College ID: ekareks@rknec.edu Offices : [123 Street,Nagpur,Maharastra-440013]}

<----------- Using B2.xml ------------->
Student: Kunal Ekare, Roll: 37, Address: null
College ID: null Offices : [Gittikhadan Road,Noida,Noida-110001]

<--------------Using B3.xml----------------->
Employee: Kunal Ekare, Dept: CSE, Company: RCOEM, Salary: 75000.0
Employee: Aryan Sharma, Dept: CSE, Company: RCOEM, Salary: 80000.0
Employee: Pooja Deshmukh, Dept: CSE, Company: RCOEM, Salary: 85000.0
Employee: Rajesh Patil, Dept: CSE, Company: RCOEM, Salary: 90000.0
Employee: Sneha Verma, Dept: CSE, Company: RCOEM, Salary: 95000.0

Process finished with exit code 0
```

Project Structure:


```
Practical_6_kunal_ekare
Project
Practical_6_kunal_ekare C:\U
  .idea
  out
  src
    in
      beans
        Address
        College
        CollegeNameMappi
        Employee
        Student
      Main
        Main
      resource
        B1.xml
        B2.xml
        B3.xml
  .gitignore
  Practical_6_kunal_ekare.iml
External Libraries
  < 21 > C:\Program Files\Jav
  commons-logging-1.3.5
Scratches and Consoles
```

Conclusion: This implementation demonstrates efficient dependency injection in Spring using setter, constructor, and autowiring techniques. It optimizes bean management through inheritance and reduces redundancy, ensuring modular and maintainable code.