PRACTICAL 7(B) – SPRING DEPENDENCY INJECTION USING SETTER METHOD

AIM

To demonstrate Dependency Injection in Spring Framework using Setter Method.

THEORY

Dependency Injection (DI) is a design pattern used in Spring Framework where objects are provided with their dependencies rather than creating them internally. In Setter Injection, the Spring container injects dependency using setter methods.

SOURCE CODE – Address.java

```
package springdemo;

public class Address {
    private String city;

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

    public String getCity() {
        return city;
    }
}
```

SOURCE CODE – Student.java

```
package springdemo;

public class Student {
    private int id;
    private Address address;

    public void setId(int id) {
        this.id = id;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public void display() {
        System.out.println("Student ID: " + id);
        System.out.println("City: " + address.getCity());
    }
}
```

SOURCE CODE – applicationContext.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.xsd">

    <bean id="address" class="springdemo.Address">
        <property name="city" value="Mumbai"/>
    </bean>

    <bean id="student" class="springdemo.Student">
        <property name="id" value="101"/>
        <property name="address" ref="address"/>
    </bean>
</beans>
```

SOURCE CODE – MainApp.java

```
package springdemo;

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

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationContext.xml");

        Student s = (Student) context.getBean("student");
        s.display();
    }
}
```

NETBEANS 7 – EXECUTION STEPS

1. Use same Spring project and libraries from Practical 7(A)
2. Create Address.java and Student.java inside springdemo package
3. Update applicationContext.xml
4. Run MainApp.java using Shift + F6

OUTPUT

```
Student ID: 101 City: Mumbai
```

RESULT

Thus, dependency injection using setter method was successfully implemented in Spring Framework.