Name: Ayush Agrawal

Date: 03-02-2022

**Spring Core Assignments**

1) Create an Address class with the following attributes: - street, city, state, zip, country

Create an Customer class with the following attributes: - customerId, customer Name, customerContact, customer Address.

Inject the Address bean into Customer bean using setter injection

Create a Test class with main() method, get Customer bean from ApplicationContext object and print details of Customer.

Also write the JUnit Test cases for above program.

- Modify the above application and inject the bean using constructor injection

- Use XML based Configuraion.

**Answer:**

**Address.java:**

package org.example.AssignmentQ1;  
  
public class Address {  
 private String street;  
 private String city;  
 private String state;  
 private long zip;  
 private String country;  
  
 public Address() {  
 super();  
 }  
  
 public Address(String street, String city, String state, long zip, String country) {  
 this.street = street;  
 this.city = city;  
 this.state = state;  
 this.zip = zip;  
 this.country = country;  
 }  
  
 public String getStreet() {  
 return street;  
 }  
  
 public void setStreet(String street) {  
 this.street = street;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public String getState() {  
 return state;  
 }  
  
 public void setState(String state) {  
 this.state = state;  
 }  
  
 public long getZip() {  
 return zip;  
 }  
  
 public void setZip(long zip) {  
 this.zip = zip;  
 }  
  
 public String getCountry() {  
 return country;  
 }  
  
 public void setCountry(String country) {  
 this.country = country;  
 }  
  
 @Override  
 public String toString() {  
 return "Address: [" +  
 "street='" + street + '\'' +  
 ", city='" + city + '\'' +  
 ", state='" + state + '\'' +  
 ", zip=" + zip +  
 ", country='" + country + '\'' +  
 ']';  
 }  
}

**Customer.java:**

package org.example.AssignmentQ1;  
  
public class Customer {  
 private int customerId;  
 private String customerName;  
 private long customerContact;  
 private Address address;  
  
 public Customer() {  
 super();  
 }  
  
 public Customer(int customerId, String customerName, long customerContact, Address address) {  
 this.customerId = customerId;  
 this.customerName = customerName;  
 this.customerContact = customerContact;  
 this.address = address;  
 }  
  
 public int getCustomerId() {  
 return customerId;  
 }  
  
 public void setCustomerId(int customerId) {  
 this.customerId = customerId;  
 }  
  
 public String getCustomerName() {  
 return customerName;  
 }  
  
 public void setCustomerName(String customerName) {  
 this.customerName = customerName;  
 }  
  
 public long getCustomerContact() {  
 return customerContact;  
 }  
  
 public void setCustomerContact(long customerContact) {  
 this.customerContact = customerContact;  
 }  
  
 public Address getAddress() {  
 return address;  
 }  
  
 public void setAddress(Address address) {  
 this.address = address;  
 }  
  
 @Override  
 public String toString() {  
 return "Customer: [" +  
 "customerId=" + customerId + "\n" +  
 ", customerName='" + customerName + "\n" +  
 ", customerContact=" + customerContact + "\n" +  
 ", address=" + address +  
 ']';  
 }  
}

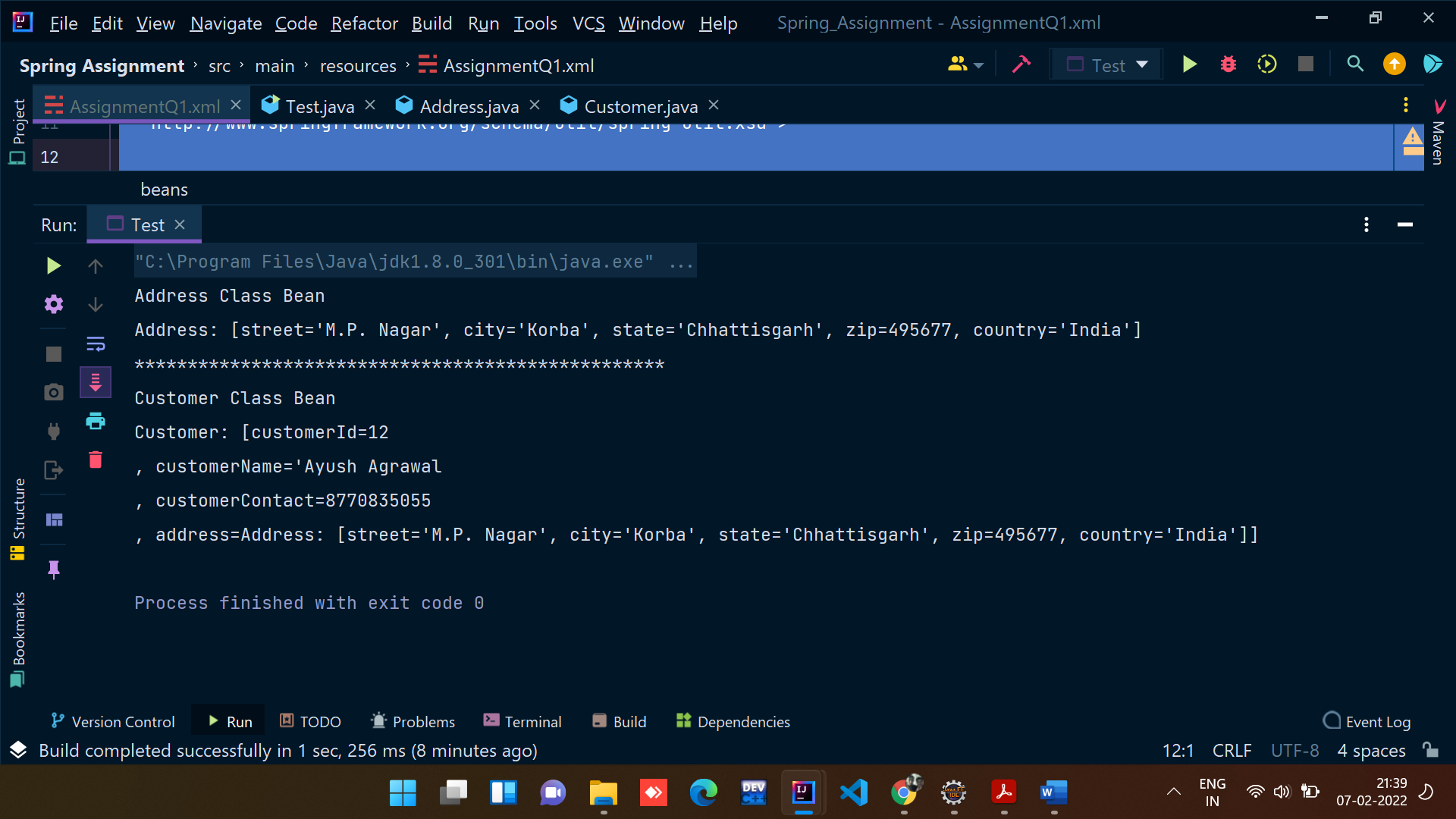
**XML Configuration:**

<?*xml version*="1.0" *encoding*="UTF-8"?>  
<beans *xmlns:xsi*="http://www.w3.org/2001/XMLSchema-instance"  
 *xmlns:context*="http://www.springframework.org/schema/context"  
 *xmlns*="http://www.springframework.org/schema/beans"  
 *xmlns:util*="http://www.springframework.org/schema/util"  
 *xsi:schemaLocation*="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd  
 http://www.springframework.org/schema/util  
 http://www.springframework.org/schema/util/spring-util.xsd">  
  
*<!-- creating bean for Address class object-->* <bean *class*="org.example.AssignmentQ1.Address" *name*="address1">  
 <property *name*="street">  
 <value>M.P. Nagar</value>  
 </property>  
 <property *name*="city">  
 <value>Korba</value>  
 </property>  
 <property *name*="state">  
 <value>Chhattisgarh</value>  
 </property>  
 <property *name*="zip">  
 <value>495677</value>  
 </property>  
 <property *name*="country">  
 <value>India</value>  
 </property>  
 </bean>  
  
*<!-- creating bean for customer class object-->* <bean *name*="customer1" *class*="org.example.AssignmentQ1.Customer">  
 <property *name*="customerId">  
 <value>12</value>  
 </property>  
 <property *name*="customerName">  
 <value>Ayush Agrawal</value>  
 </property>  
 <property *name*="customerContact">  
 <value>8770835055</value>  
 </property>  
 <property *name*="address">  
 <ref *bean*="address1"/>  
 </property>  
 </bean>  
  
*<!-- creating beans using constructor injections-->  
<!-- creating bean for address-->* <bean *class*="org.example.AssignmentQ1.Address" *name*="address2">  
 <constructor-arg *name*="street" *value*="Mahadev Ghat"/>  
 <constructor-arg *name*="city" *value*="Raipur"/>  
 <constructor-arg *name*="state" *value*="Chhattisgarh"/>  
 <constructor-arg *name*="zip" *value*="491102"/>  
 <constructor-arg *name*="country" *value*="India"/>  
 </bean>  
  
*<!-- creating bean for customer-->* <bean *name*="customer2" *class*="org.example.AssignmentQ1.Customer">  
 <constructor-arg *name*="customerId" *value*="14"/>  
 <constructor-arg *name*="customerName" *value*="Ritik Dixit"/>  
 <constructor-arg *name*="customerContact" *value*="7000061030"/>  
 <constructor-arg *name*="address" *ref*="address2"/>  
 </bean>  
</beans>

**Test.java:**

package org.example.AssignmentQ1;  
  
import org.springframework.context.*ApplicationContext*;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Test {  
 public static void main(String[] args) {  
 *ApplicationContext* context = new ClassPathXmlApplicationContext("AssignmentQ1.xml");  
  
 Address address1 = context.getBean("address1", Address.class);  
 System.out.println("Address Class Bean");  
 System.out.println(address1);  
  
 System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
  
 Customer customer1 = context.getBean("customer1", Customer.class);  
 System.out.println("Customer Class Bean ");  
 System.out.println(customer1);  
 }  
}

**Output:**



**Constructor Injection Method:**

**XML Code:**

*<!-- creating beans using constructor injections-->  
<!-- creating bean for address-->* <bean *class*="org.example.AssignmentQ1.Address" *name*="address2">  
 <constructor-arg *name*="street" *value*="Mahadev Ghat"/>  
 <constructor-arg *name*="city" *value*="Raipur"/>  
 <constructor-arg *name*="state" *value*="Chhattisgarh"/>  
 <constructor-arg *name*="zip" *value*="491102"/>  
 <constructor-arg *name*="country" *value*="India"/>  
 </bean>  
  
*<!-- creating bean for customer-->* <bean *name*="customer2" *class*="org.example.AssignmentQ1.Customer">  
 <constructor-arg *name*="customerId" *value*="14"/>  
 <constructor-arg *name*="customerName" *value*="Ritik Dixit"/>  
 <constructor-arg *name*="customerContact" *value*="7000061030"/>  
 <constructor-arg *name*="address" *ref*="address2"/>  
 </bean>

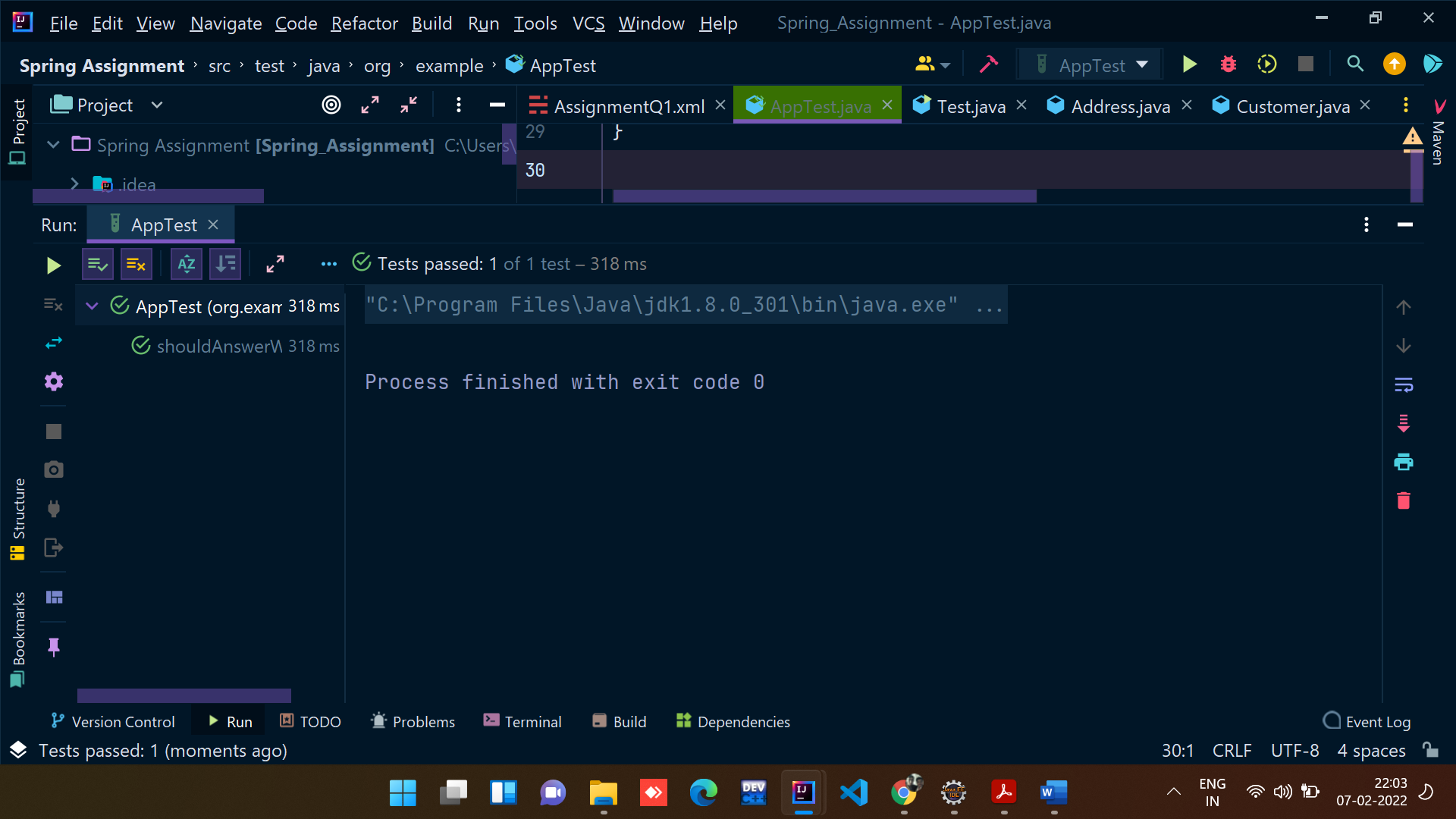
**Test.java**

package org.example.AssignmentQ1;  
  
import org.springframework.context.*ApplicationContext*;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Test {  
 public static void main(String[] args) {  
 *ApplicationContext* context = new ClassPathXmlApplicationContext("AssignmentQ1.xml");  
  
 System.out.println("Creating Address class Bean using constructor injection method");  
 Address address2 = context.getBean("address2",Address.class);  
 System.out.println(address2);  
  
 System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 System.out.println("Creating Customer Class Bean using constructor injection method");  
 Customer customer2 = context.getBean("customer2",Customer.class);  
 System.out.println(customer2);  
 }  
}

**Junit Testing code:**

package org.example;  
  
import static org.junit.Assert.*assertTrue*;  
  
import org.example.AssignmentQ1.Customer;  
import org.junit.Assert;  
import org.junit.Before;  
import org.junit.Test;  
import org.springframework.context.*ApplicationContext*;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class AppTest   
{  
 private Customer customer1;  
  
 @Before  
 public void init(){  
 *ApplicationContext* context = new ClassPathXmlApplicationContext("AssignmentQ1.xml");  
 customer1 = context.getBean("customer1",Customer.class);  
 }  
 @Test  
 public void shouldAnswerWithTrue()  
 {  
 Assert.*assertEquals*(customer1.toString(),"Customer: [customerId=12\n" +  
 ", customerName='Ayush Agrawal\n" +  
 ", customerContact=8770835055\n" +  
 ", address=Address: [street='M.P. Nagar', city='Korba', state='Chhattisgarh', zip=495677, country='India']]");  
 }  
}

**Junit Testing Output:**



2) Example of Injecting collections (List, Set and Map)

Create a class Question with following attributes: questionId, question, answers.

**There are 3 cases for above program.**

a. Write a program where answers is of type List<String> or String []

b. Write a program where answers is of type Set<String>

c. Write a program where answers is of type Map<Integer, String>

In case of Map, Integer value represents answer’s sequence number.

d. Create a Test class with main() method, get Question bean from ApplicationContext object and print question and its answers.

e. Also write the JUnit Test cases for above program.

- Use XML based configuration.

**Answer:**

**Question1.java:**

package org.example.AssignmentQ2;  
  
import java.util.*List*;  
  
public class Question1 {  
 private int questionId;  
 private String question;  
 private *List*<String> answers;  
  
 public Question1() {  
 super();  
 }  
  
 public int getQuestionId() {  
 return questionId;  
 }  
  
 public void setQuestionId(int questionId) {  
 this.questionId = questionId;  
 }  
  
 public String getQuestion() {  
 return question;  
 }  
  
 public void setQuestion(String question) {  
 this.question = question;  
 }  
  
 public *List*<String> getAnswers() {  
 return answers;  
 }  
  
 public void setAnswers(*List*<String> answers) {  
 this.answers = answers;  
 }  
  
 @Override  
 public String toString() {  
 return "Question1: [" +  
 "questionId=" + questionId + '\n' +  
 ", question='" + question + '\n' +  
 ", answers=" + answers + '\n' +  
 ']';  
 }  
}

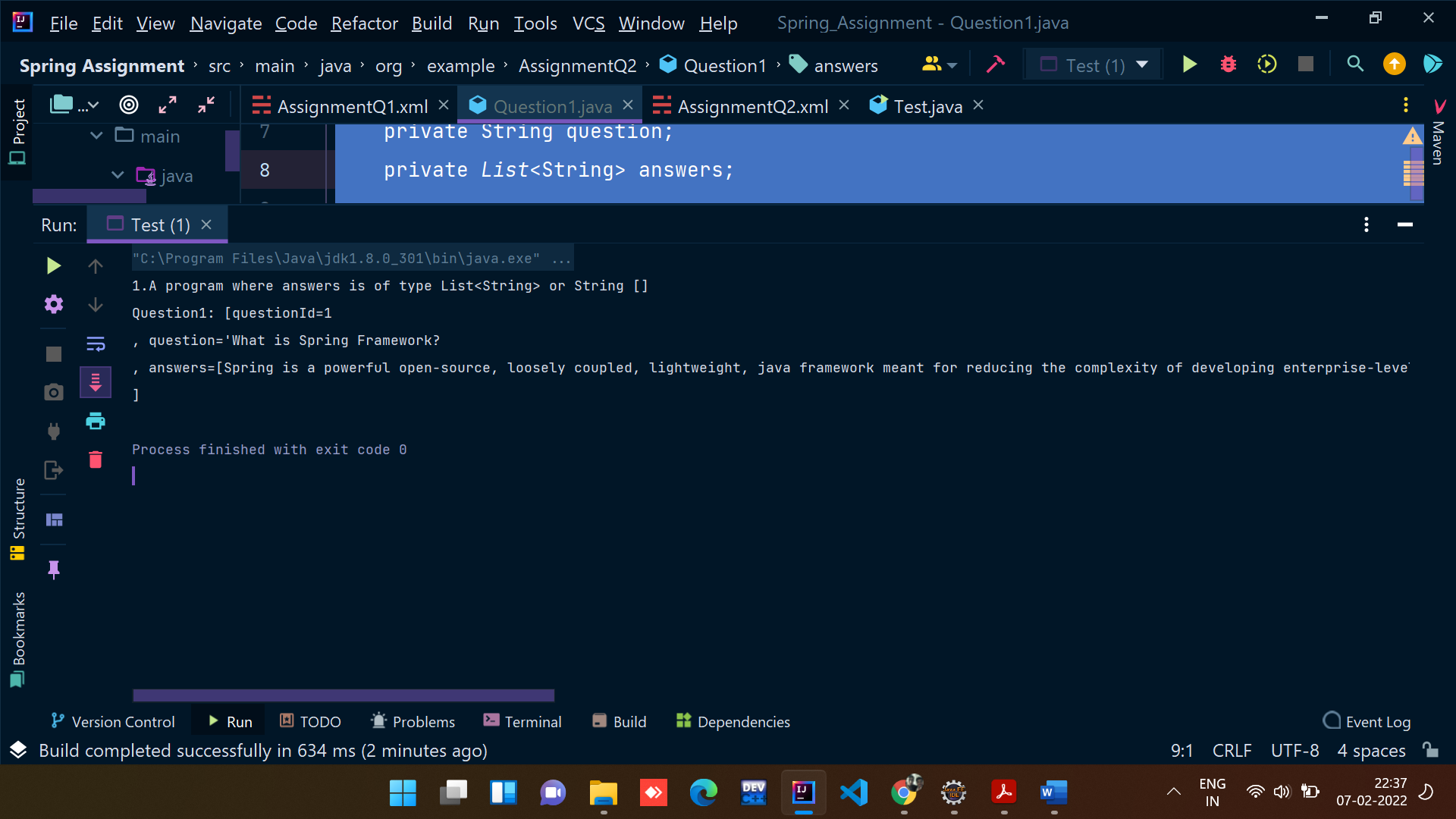
**XML Configuration:**

*<!-- creating bean for class Question1-->* <bean *class*="org.example.AssignmentQ2.Question1" *name*="question1">  
 <property *name*="questionId" *value*="1"/>  
 <property *name*="question" *value*="What is Spring Framework?"/>  
 <property *name*="answers">  
 <list>  
 <value>Spring is a powerful open-source, loosely coupled, lightweight, java framework meant for reducing the complexity of developing enterprise-level applications.</value>  
 <value>This framework is also called the “framework of frameworks” as spring provides support to various other important frameworks like JSF, Hibernate, Structs, EJB, etc.</value>  
 <value>Spring handles all the infrastructure-related aspects which lets the programmer focus mostly on application development.</value>  
 </list>  
 </property>  
 </bean>

**Test.java**

package org.example.AssignmentQ2;  
  
import org.springframework.context.*ApplicationContext*;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Test {  
 public static void main(String[] args) {  
 *ApplicationContext* context = new ClassPathXmlApplicationContext("AssignmentQ2.xml");  
  
 Question1 question1 = context.getBean("question1",Question1.class);  
 System.out.println("1.A program where answers is of type List<String> or String []");  
  
 System.out.println(question1);  
  
  
 System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
  
 Question2 question2 = context.getBean("question2",Question2.class);  
 System.out.println("A program where answers is of type Set<String>");  
 System.out.println(question2);  
  
 System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
  
 Question3 question3 = context.getBean("question3",Question3.class);  
 System.out.println("A program where answers is of type Map<Integer, String>");  
 System.out.println(question3);  
 }  
}

**Output for Test->Question1.java**



**Question2.java**

package org.example.AssignmentQ2;  
  
import java.util.*Set*;  
  
public class Question2 {  
 private int questionId;  
 private String question;  
 private *Set*<String> answers;  
  
 public Question2() {  
 super();  
 }  
  
 public int getQuestionId() {  
 return questionId;  
 }  
  
 public void setQuestionId(int questionId) {  
 this.questionId = questionId;  
 }  
  
 public String getQuestion() {  
 return question;  
 }  
  
 public void setQuestion(String question) {  
 this.question = question;  
 }  
  
 public *Set*<String> getAnswers() {  
 return answers;  
 }  
  
 public void setAnswers(*Set*<String> answers) {  
 this.answers = answers;  
 }  
  
 @Override  
 public String toString() {  
 return "Question2: [" +  
 "questionId=" + questionId +  
 ", question='" + question + '\'' +  
 ", answers=" + answers +  
 ']';  
 }  
}

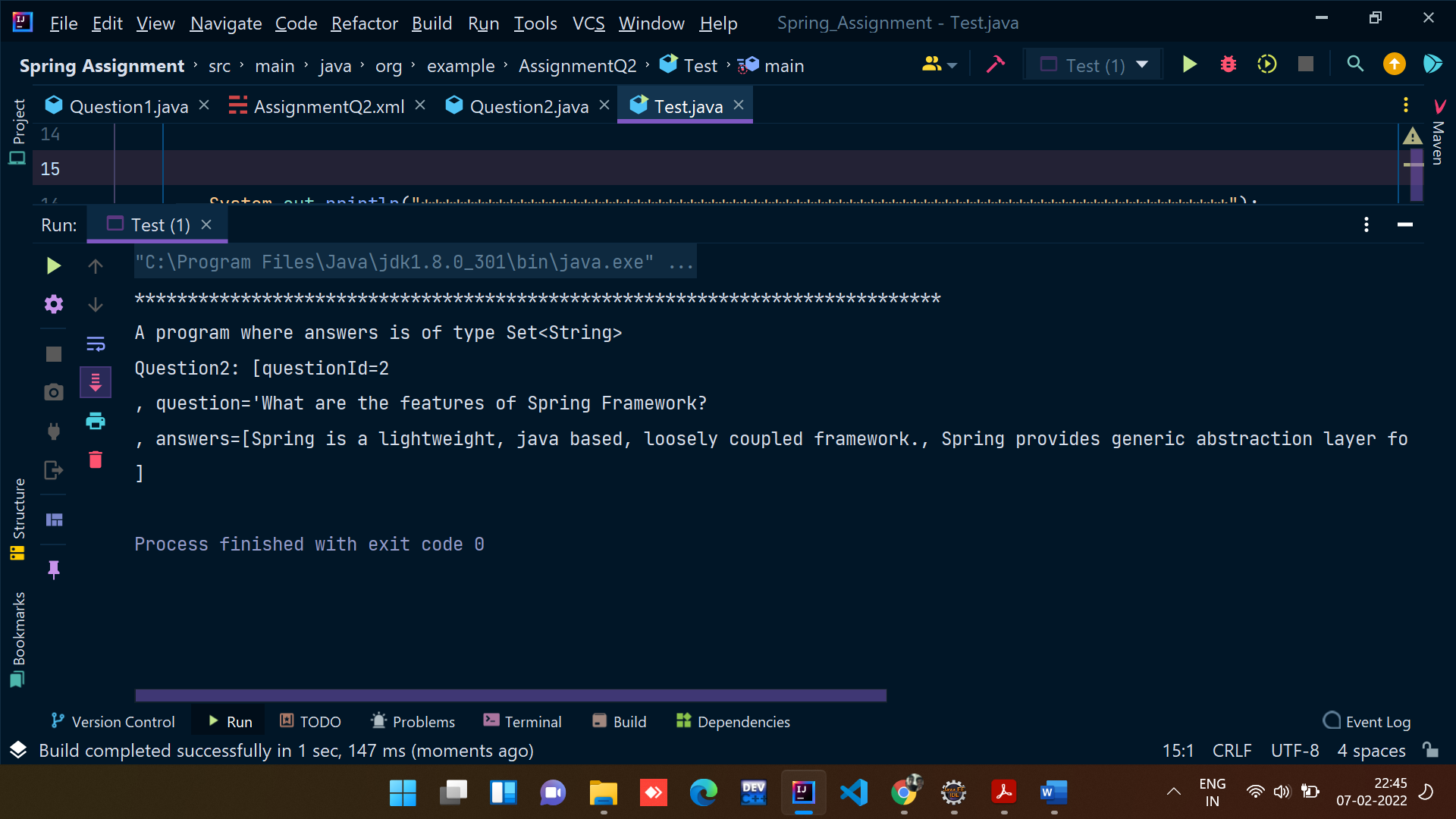
**Question2.java**

package org.example.AssignmentQ2;  
  
import java.util.*Set*;  
  
public class Question2 {  
 private int questionId;  
 private String question;  
 private *Set*<String> answers;  
  
 public Question2() {  
 super();  
 }  
  
 public int getQuestionId() {  
 return questionId;  
 }  
  
 public void setQuestionId(int questionId) {  
 this.questionId = questionId;  
 }  
  
 public String getQuestion() {  
 return question;  
 }  
  
 public void setQuestion(String question) {  
 this.question = question;  
 }  
  
 public *Set*<String> getAnswers() {  
 return answers;  
 }  
  
 public void setAnswers(*Set*<String> answers) {  
 this.answers = answers;  
 }  
  
 @Override  
 public String toString() {  
 return "Question2: [" +  
 "questionId=" + questionId + '\n' +  
 ", question='" + question + '\n' +  
 ", answers=" + answers + '\n' +  
 ']';  
 }  
}

**XML Configuration:**

*<!-- creating a bean for class Question2-->* <bean *class*="org.example.AssignmentQ2.Question2" *name*="question2">  
 <property *name*="questionId" *value*="2"/>  
 <property *name*="question" *value*="What are the features of Spring Framework?"/>  
 <property *name*="answers">  
 <set>  
 <value>Spring is a lightweight, java based, loosely coupled framework.</value>  
 <value>Spring provides generic abstraction layer for transaction management that is also very useful for container-less environments.</value>  
 </set>  
 </property>  
 </bean>

**Output for Test->Question2.java:**



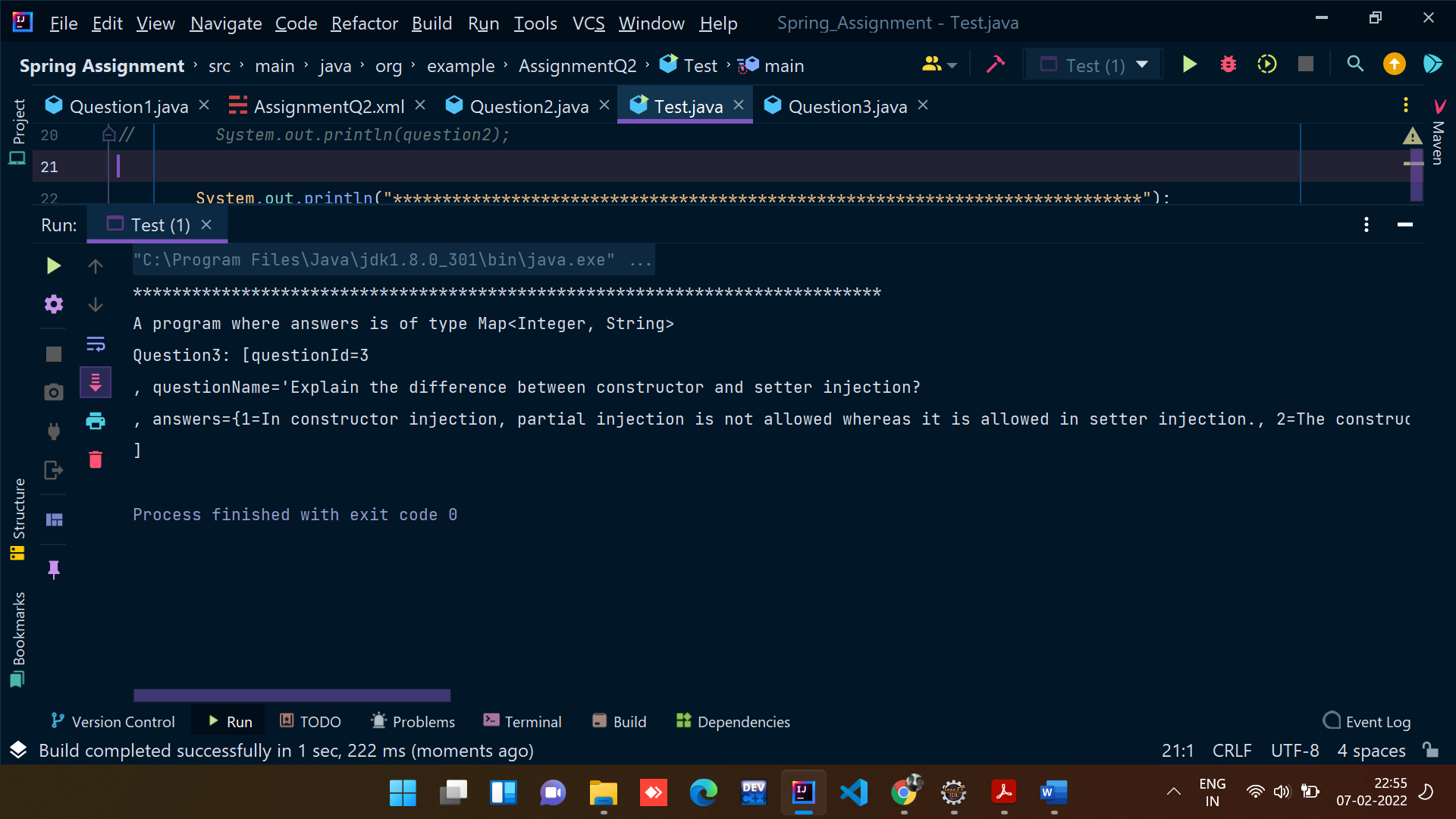
**Question3.java:**

package org.example.AssignmentQ2;  
  
import java.util.*Map*;  
  
public class Question3 {  
 private int questionId;  
 private String questionName;  
 private *Map*<Integer,String> answers;  
  
 public Question3() {  
 super();  
 }  
  
 public int getQuestionId() {  
 return questionId;  
 }  
  
 public void setQuestionId(int questionId) {  
 this.questionId = questionId;  
 }  
  
 public String getQuestionName() {  
 return questionName;  
 }  
  
 public void setQuestionName(String questionName) {  
 this.questionName = questionName;  
 }  
  
 public *Map*<Integer, String> getAnswers() {  
 return answers;  
 }  
  
 public void setAnswers(*Map*<Integer, String> answers) {  
 this.answers = answers;  
 }  
  
 @Override  
 public String toString() {  
 return "Question3: [" +  
 "questionId=" + questionId +'\n' +  
 ", questionName='" + questionName + '\n' +  
 ", answers=" + answers +'\n' +  
 ']';  
 }  
}

**XML Configuration:**

*<!-- creating a bean for class Question3-->* <bean *class*="org.example.AssignmentQ2.Question3" *name*="question3">  
 <property *name*="questionId" *value*="3"/>  
 <property *name*="questionName" *value*="Explain the difference between constructor and setter injection?"/>  
 <property *name*="answers">  
 <map>  
 <entry *key*="1">  
 <value>In constructor injection, partial injection is not allowed whereas it is allowed in setter injection.</value>  
 </entry>  
 <entry *key*="2">  
 <value>The constructor injection doesn’t override the setter property whereas the same is not true for setter injection.</value>  
 </entry>  
 <entry *key*="3">  
 <value>Constructor injection creates a new instance if any modification is done. The creation of a new instance is not possible in setter injection.</value>  
 </entry>  
 <entry *key*="4">  
 <value>In case the bean has many properties, then constructor injection is preferred. If it has few properties, then setter injection is preferred.</value>  
 </entry>  
  
 </map>  
 </property>  
 </bean>

**Output for Test->Question3.java:**



**3) Example on autowiring**

**Design and Develop a Banking Application as follows:**

1. Create a BankAccount class with following attributes: accountId, accountHolderName, accountType, accountBalance.

**BankAccount.java class:**

package org.example.AssignmentQ3;  
  
public class BankAccount {  
 private int accountId;  
 private String accountHolderName;  
 private String accountType;  
 private double accountBalance;  
  
 public BankAccount() {  
 super();  
 }  
  
 public int getAccountId() {  
 return accountId;  
 }  
  
 public void setAccountId(int accountId) {  
 this.accountId = accountId;  
 }  
  
 public String getAccountHolderName() {  
 return accountHolderName;  
 }  
  
 public void setAccountHolderName(String accountHolderName) {  
 this.accountHolderName = accountHolderName;  
 }  
  
 public String getAccountType() {  
 return accountType;  
 }  
  
 public void setAccountType(String accountType) {  
 this.accountType = accountType;  
 }  
  
 public double getAccountBalance() {  
 return accountBalance;  
 }  
  
 public void setAccountBalance(double accountBalance) {  
 this.accountBalance = accountBalance;  
 }  
  
 @Override  
 public String toString() {  
 return "BankAccount: [" + '\n' +  
 "accountId=" + accountId + '\n' +  
 ", accountHolderName='" + accountHolderName + '\n' +  
 ", accountType='" + accountType + '\n' +  
 ", accountBalance=" + accountBalance + '\n' +  
 ']';  
 }  
}

1. Create an interface BankAccountRepository with following methods:

public double getBalance(long accountId)

public double updateBalance(long accountId, double newBalance):

**Interface BankAccountRepository:**

package org.example.AssignmentQ3;  
  
public interface *BankAccountRepository* {  
 public double getBalance(long accountId);  
 public double updateBalance(long accountId, double newBalance);  
  
}

Note: Above method returns updated balance.

1. Create a class BankAccountepositoryImpl that implements BankAccountRepository interface.

You can use database or any collection object as persistence store.

**BankAccountRepositoryImpl:**

package org.example.AssignmentQ3;  
  
public class BankAccountRepositoryImpl implements *BankAccountRepository*{  
  
 private BankAccount bankAccount1;  
 private BankAccount bankAccount2;  
  
 public BankAccountRepositoryImpl() {  
 }  
  
 public BankAccountRepositoryImpl(BankAccount bankAccount1, BankAccount bankAccount2) {  
 this.bankAccount1 = bankAccount1;  
 this.bankAccount2 = bankAccount2;  
 }  
  
 @Override  
 public double getBalance(long accountId) {  
 if(this.bankAccount1.getAccountId()==accountId){  
 return bankAccount1.getAccountBalance();  
 }  
 else {  
 return bankAccount2.getAccountBalance();  
 }  
 }  
  
 @Override  
 public double updateBalance(long accountId, double newBalance) {  
 double updateBal = 0;  
 if(this.bankAccount1.getAccountId()==accountId){  
 updateBal = newBalance;  
 bankAccount1.setAccountBalance(updateBal);  
 }  
 else {  
 updateBal = newBalance;  
 bankAccount2.setAccountBalance(updateBal);  
 }  
 return updateBal;  
 }  
}

1. Create an interface BankAccountService with following methods:

public double withdraw(long accountId, double balance)

public double deposit(long accountId, double balance)

public double getBalance(long accountId)

public boolean fundTransfer(long fromAccount, long toAccount, double amont)

**Interface BankAccountService:**

package org.example.AssignmentQ3;  
  
public interface *BankAccountService* {  
 public double withdraw(long accountId, double balance);  
 public double deposit(long accountId, double balance);  
 public double getBalance(long accountId);  
 public boolean fundTransfer(long fromAccount, long toAccount, double amont);  
  
}

1. Create a class BankAccountServiceImpl that implements BankAccountService interface.

**Class BankAccountServiceImpl:**

1. package org.example.AssignmentQ3;  
     
   public class BankAccountServiceImp implements *BankAccountService*{  
     
    private BankAccount bankAccount;  
    @Override  
    public double withdraw(long accountId, double balance) {  
    return 0;  
    }  
     
    @Override  
    public double deposit(long accountId, double balance) {  
    return 0;  
    }  
     
    @Override  
    public double getBalance(long accountId) {  
    return 0;  
    }  
     
    @Override  
    public boolean fundTransfer(long fromAccount, long toAccount, double amont) {  
    return false;  
    }  
   }
2. Create a class BankAccount controller with following operations:

public double withdraw(long accountId, double balance)

public double deposit(long accountId, double balance)

public double getBalance(long accountId)

public boolean fundTransfer(long fromAccount, long toAccount, double amont)

**Class BankAccount Controller:**

package org.example.AssignmentQ3;  
  
import org.springframework.beans.BeansException;  
import org.springframework.context.*ApplicationContext*;  
import org.springframework.context.*ApplicationContextAware*;  
import org.springframework.context.support.AbstractApplicationContext;  
  
public class BankAccountController implements *ApplicationContextAware* {  
 private *ApplicationContext* context;  
  
 public double withdraw(long accountId, double balance){  
 double newBalance = 0;  
 BankAccountRepositoryImpl accountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");  
   
 if(accountRepository.getBalance(accountId)>=balance){  
 newBalance = accountRepository.getBalance(accountId)-balance;  
 accountRepository.updateBalance(accountId,newBalance);  
 }  
 return newBalance;  
 }  
 public double deposit(long accountId, double balance){  
 BankAccountRepositoryImpl bankAccountRepository = context.getBean("BankRepo",BankAccountRepositoryImpl.class);  
 double newBalance = bankAccountRepository.getBalance(accountId)+balance;  
 return bankAccountRepository.updateBalance(accountId,newBalance);  
 }  
 public double getBalance(long accountId){  
 BankAccountRepositoryImpl bankAccountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");  
 return bankAccountRepository.getBalance(accountId);  
 }  
 public boolean fundTransfer(long fromAccount, long toAccount, double amont){  
 BankAccountRepositoryImpl accountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");  
 if(accountRepository.getBalance(fromAccount)>=amont){  
 double updateAmt = amont+accountRepository.getBalance(toAccount);  
  
 accountRepository.updateBalance(fromAccount,accountRepository.getBalance(fromAccount)-amont);  
 accountRepository.updateBalance(toAccount,accountRepository.getBalance(toAccount)+amont);  
 return true;  
 }  
 return false;  
 }  
 @Override  
 public void setApplicationContext(*ApplicationContext* applicationContext) throws BeansException {  
 this.context = applicationContext;  
 }  
}

1. Create a Test class with main() method, get BankAccountController bean object from ApplicationContext and perform all the operations.

**Test.java**

1. package org.example.AssignmentQ3;  
     
   import org.springframework.context.*ApplicationContext*;  
   import org.springframework.context.support.ClassPathXmlApplicationContext;  
     
   public class Test {  
    public static void main(String[] args) {  
     
    *ApplicationContext* context = new ClassPathXmlApplicationContext("Assignment3.xml");  
     
    BankAccountController controller = (BankAccountController) context.getBean("controller");  
     
    System.out.println(controller.getBalance(5498));  
    System.out.println(controller.deposit(5498,5000));  
    System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
     
    System.out.println(controller.withdraw(5498,10000));  
    System.out.println(controller.getBalance(5498));  
    System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
     
    System.out.println(controller.fundTransfer(5498,5496,10000));  
    System.out.println(controller.getBalance(5498));  
    System.out.println(controller.getBalance(5496));  
     
     
    }  
   }
2. Also write the JUnit Test cases for above program.

- Use XML based configuration and perform autowiring with different types. (byName, byType and constructor). Use one autowiring type at a time.

**Using byName:**

<bean *class*="org.example.AssignmentQ3.BankAccountRepositoryImpl" *id*="BankRepo" *autowire*="byName"/>

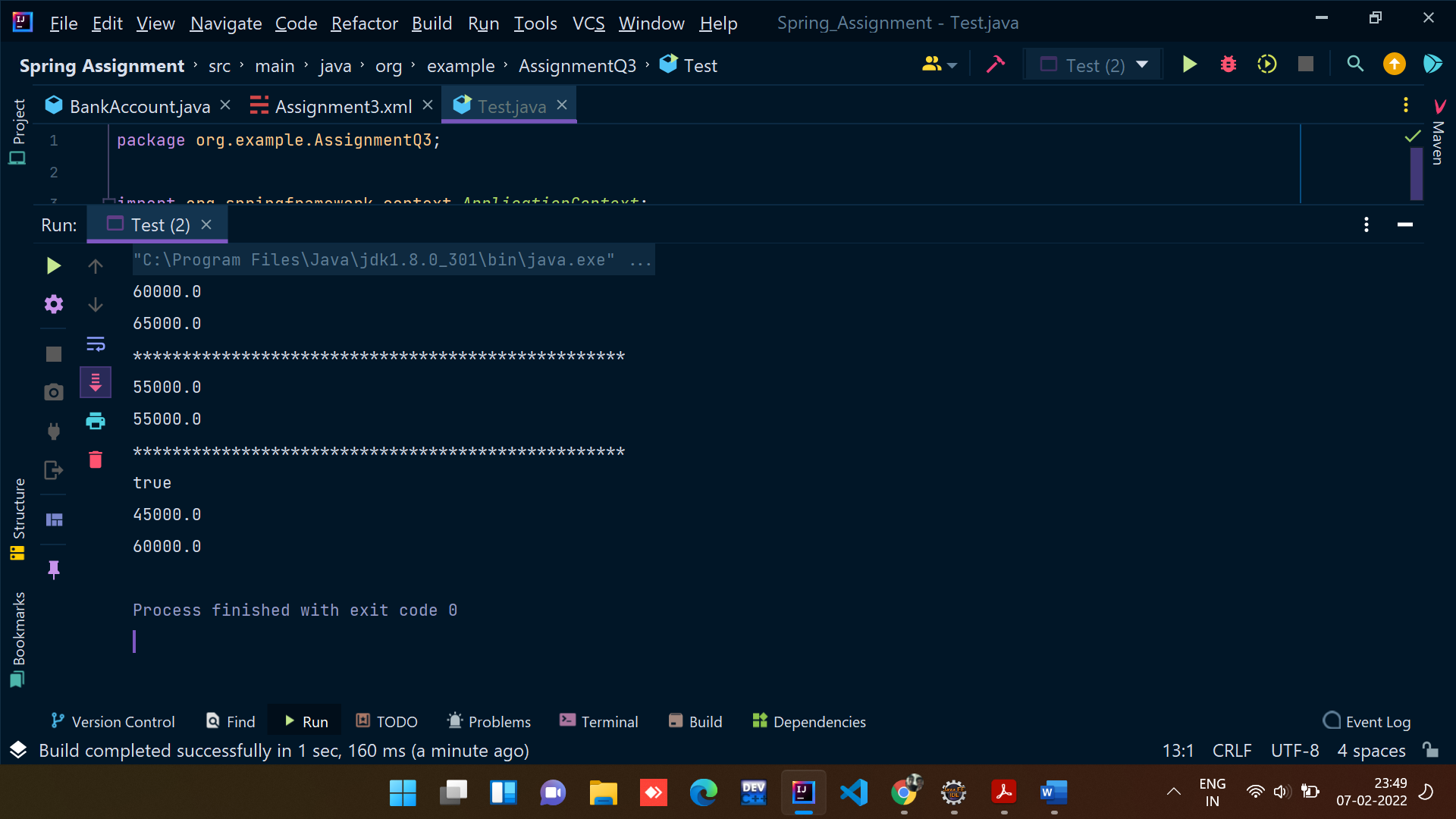
**Using byType:**

<bean *class*="org.example.AssignmentQ3.BankAccountRepositoryImpl" *id*="BankRepo" *autowire*="byType"/>

**Using constructor:**

<?*xml version*="1.0" *encoding*="UTF-8"?>  
<beans *xmlns:xsi*="http://www.w3.org/2001/XMLSchema-instance"  
 *xmlns:context*="http://www.springframework.org/schema/context"  
 *xmlns*="http://www.springframework.org/schema/beans"  
 *xmlns:util*="http://www.springframework.org/schema/util"  
 *xsi:schemaLocation*="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd  
 http://www.springframework.org/schema/util  
 http://www.springframework.org/schema/util/spring-util.xsd">  
  
 <bean *class*="org.example.AssignmentQ3.BankAccount" *id*="bankAccount1">  
 <constructor-arg *type*="int" *value*="5496"/>  
 <constructor-arg *value*="Ayush Agrawal"/>  
 <constructor-arg *value*="Saving"/>  
 <constructor-arg *type*="double" *value*="50000"/>  
 </bean>  
  
 <bean *class*="org.example.AssignmentQ3.BankAccount" *id*="bankAccount2">  
 <constructor-arg *type*="int" *value*="5498"/>  
 <constructor-arg *value*="Ritik Dixit"/>  
 <constructor-arg *value*="Current"/>  
 <constructor-arg *type*="double" *value*="60000"/>  
 </bean>  
  
 <bean *class*="org.example.AssignmentQ3.BankAccountRepositoryImpl" *id*="BankRepo" *autowire*="constructor"/>  
 <bean *class*="org.example.AssignmentQ3.BankAccountController" *id*="controller"/>  
</beans>

**Output:**



4) Example on @Controller, @Service, @Repository, @Autowired, @Configuration and @Bean

**BankAccount.java**

package org.example.AssignmentQ4;  
  
public class BankAccount {  
 private int accountId;  
 private String accountHolderName;  
 private String accountType;  
 private double accountBalance;  
  
 public BankAccount() {  
 super();  
 }  
  
 public BankAccount(int accountId, String accountHolderName, String accountType, double accountBalance) {  
 this.accountId = accountId;  
 this.accountHolderName = accountHolderName;  
 this.accountType = accountType;  
 this.accountBalance = accountBalance;  
 }  
  
 public int getAccountId() {  
 return accountId;  
 }  
  
 public void setAccountId(int accountId) {  
 this.accountId = accountId;  
 }  
  
 public String getAccountHolderName() {  
 return accountHolderName;  
 }  
  
 public void setAccountHolderName(String accountHolderName) {  
 this.accountHolderName = accountHolderName;  
 }  
  
 public String getAccountType() {  
 return accountType;  
 }  
  
 public void setAccountType(String accountType) {  
 this.accountType = accountType;  
 }  
  
 public double getAccountBalance() {  
 return accountBalance;  
 }  
  
 public void setAccountBalance(double accountBalance) {  
 this.accountBalance = accountBalance;  
 }  
  
 @Override  
 public String toString() {  
 return "BankAccount: [" + '\n' +  
 "accountId=" + accountId + '\n' +  
 ", accountHolderName='" + accountHolderName + '\n' +  
 ", accountType='" + accountType + '\n' +  
 ", accountBalance=" + accountBalance + '\n' +  
 ']';  
 }  
}

**Interface BankAccountRepository:**

package org.example.AssignmentQ4;  
  
public interface *BankAccountRepository* {  
 public double getBalance(long accountId);  
 public double updateBalance(long accountId, double newBalance);  
  
}

**BankAccountRepositoryImpl Class:**

package org.example.AssignmentQ4;  
  
import org.example.AssignmentQ3.BankAccount;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Repository;  
  
@Repository("BankRepo")  
public class BankAccountRepositoryImpl implements *BankAccountRepository* {  
  
 @Autowired  
 private BankAccount bankAccount1;  
 @Autowired  
 private BankAccount bankAccount2;  
  
 public BankAccountRepositoryImpl() {  
 }  
  
 public BankAccountRepositoryImpl(BankAccount bankAccount1, BankAccount bankAccount2) {  
 this.bankAccount1 = bankAccount1;  
 this.bankAccount2 = bankAccount2;  
 }  
  
 @Override  
 public double getBalance(long accountId) {  
 if(this.bankAccount1.getAccountId()==accountId){  
 return bankAccount1.getAccountBalance();  
 }  
 else {  
 return bankAccount2.getAccountBalance();  
 }  
 }  
  
 @Override  
 public double updateBalance(long accountId, double newBalance) {  
 double updateBal = 0;  
 if(this.bankAccount1.getAccountId()==accountId){  
 updateBal = newBalance;  
 bankAccount1.setAccountBalance(updateBal);  
 }  
 else {  
 updateBal = newBalance;  
 bankAccount2.setAccountBalance(updateBal);  
 }  
 return updateBal;  
 }  
}

**Interface BankAccountService:**

package org.example.AssignmentQ4;  
  
public interface *BankAccountService* {  
 public double withdraw(long accountId, double balance);  
 public double deposit(long accountId, double balance);  
 public double getBalance(long accountId);  
 public boolean fundTransfer(long fromAccount, long toAccount, double amont);  
  
}

**Class** BankAccountServiceImp:

package org.example.AssignmentQ4;  
  
  
import org.springframework.beans.factory.annotation.Autowired;  
  
public class BankAccountServiceImp implements *BankAccountService* {  
  
 @Autowired  
 private BankAccount bankAccount1;  
  
 @Autowired  
 private BankAccount bankAccount2;  
  
 public BankAccountServiceImp(){  
 }  
  
 public BankAccountServiceImp(BankAccount bankAccount1, BankAccount bankAccount2) {  
 this.bankAccount1 = bankAccount1;  
 this.bankAccount2 = bankAccount2;  
 }  
  
 @Override  
 public double withdraw(long accountId, double balance) {  
 return 0;  
 }  
  
 @Override  
 public double deposit(long accountId, double balance) {  
 return 0;  
 }  
  
 @Override  
 public double getBalance(long accountId) {  
 if(this.bankAccount1.getAccountId()==accountId){  
 return bankAccount1.getAccountBalance();  
 }  
 else {  
 return bankAccount2.getAccountBalance();  
 }  
 }  
  
 @Override  
 public boolean fundTransfer(long fromAccount, long toAccount, double amont) {  
 return false;  
 }  
}

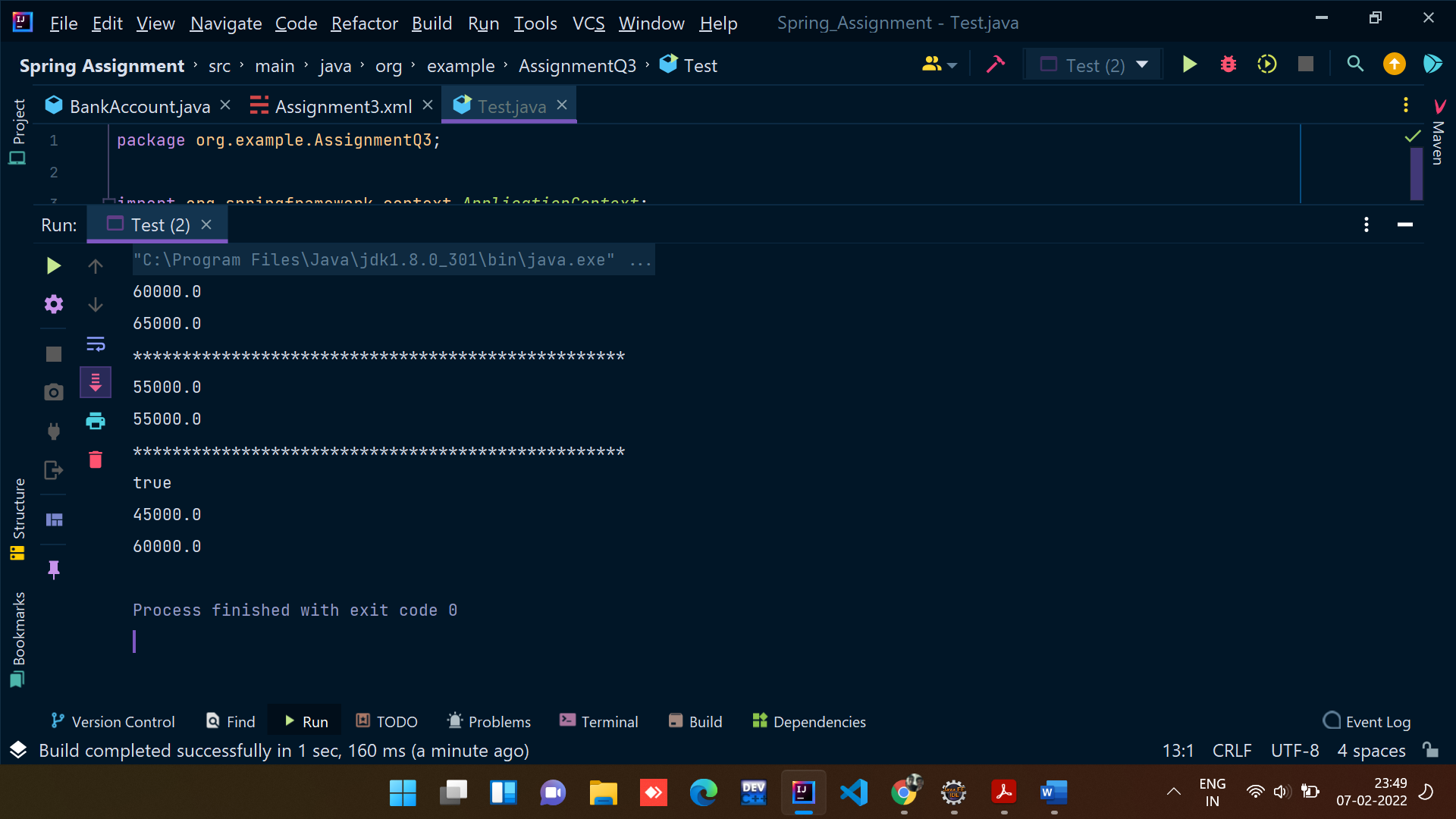
**BankAccountController Class:**

package org.example.AssignmentQ4;  
  
import org.springframework.beans.BeansException;  
import org.springframework.context.*ApplicationContext*;  
import org.springframework.context.*ApplicationContextAware*;  
import org.springframework.stereotype.Service;  
  
@Service("service")  
public class BankAccountController implements *ApplicationContextAware*,*BankAccountService* {  
 private *ApplicationContext* context;  
  
 @Override  
 public double withdraw(long accountId, double balance){  
 double newBalance = 0;  
 BankAccountRepositoryImpl accountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");  
   
 if(accountRepository.getBalance(accountId)>=balance){  
 newBalance = accountRepository.getBalance(accountId)-balance;  
 accountRepository.updateBalance(accountId,newBalance);  
 }  
 return newBalance;  
 }  
  
 @Override  
 public double deposit(long accountId, double balance){  
 BankAccountRepositoryImpl bankAccountRepository = context.getBean("BankRepo", BankAccountRepositoryImpl.class);  
 double newBalance = bankAccountRepository.getBalance(accountId)+balance;  
 return bankAccountRepository.updateBalance(accountId,newBalance);  
 }  
  
 @Override  
 public double getBalance(long accountId){  
 BankAccountRepositoryImpl bankAccountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");  
 return bankAccountRepository.getBalance(accountId);  
 }  
  
 @Override  
 public boolean fundTransfer(long fromAccount, long toAccount, double amont){  
 BankAccountRepositoryImpl accountRepository = (BankAccountRepositoryImpl) context.getBean("BankRepo");  
 if(accountRepository.getBalance(fromAccount)>=amont){  
 double updateAmt = amont+accountRepository.getBalance(toAccount);  
  
 accountRepository.updateBalance(fromAccount,accountRepository.getBalance(fromAccount)-amont);  
 accountRepository.updateBalance(toAccount,accountRepository.getBalance(toAccount)+amont);  
 return true;  
 }  
 return false;  
 }  
 @Override  
 public void setApplicationContext(*ApplicationContext* applicationContext) throws BeansException {  
 this.context = applicationContext;  
 }  
}

**Config.java**

package org.example.AssignmentQ4;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
@ComponentScan(basePackages = "org.example.AssignmentQ4")  
public class Config {  
 @Bean  
 public BankAccount getBankAccount(){  
 BankAccount bankAccount1 = new BankAccount(5498,"Ayush Agrawal","Saving",50000);  
 return bankAccount1;  
 }  
  
 @Bean  
 public BankAccount getBankAccount1(){  
 BankAccount bankAccount1 = new BankAccount(5496,"Ritik Dixit","Current",60000);  
 return bankAccount1;  
 }  
  
}

**Output:**

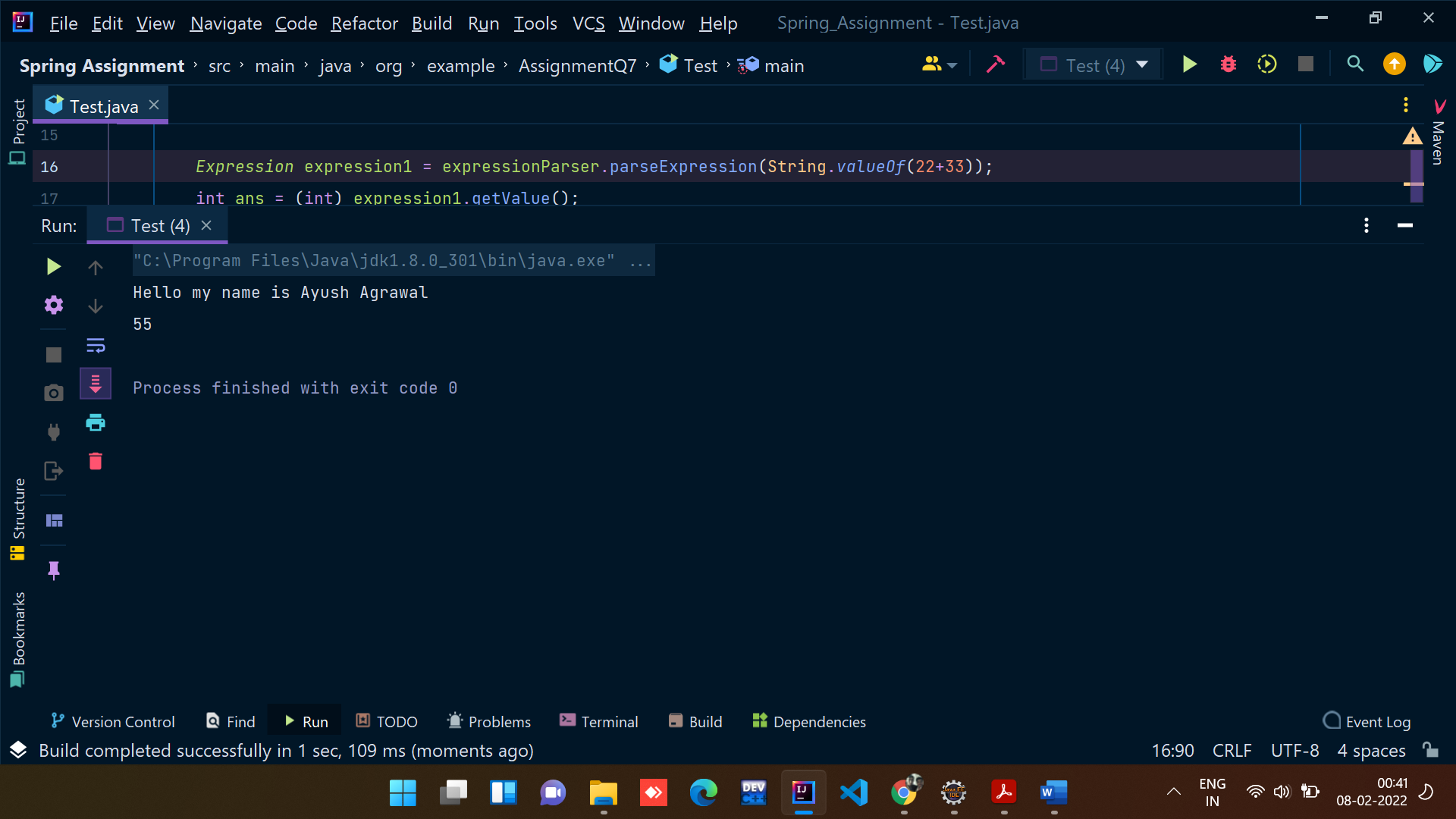


7) Write a Java program to demonstrate SPEL (Spring Expression language).

**Code:**

package org.example.AssignmentQ7;  
  
import org.springframework.expression.*Expression*;  
import org.springframework.expression.*ExpressionParser*;  
import org.springframework.expression.spel.standard.SpelExpressionParser;  
  
public class Test {  
 public static void main(String[] args) {  
  
 *ExpressionParser* expressionParser = new SpelExpressionParser();  
  
 *Expression* expression = expressionParser.parseExpression("'Hello my name is Ayush Agrawal'");  
 String message = (String) expression.getValue();  
 System.out.println(message);  
  
 *Expression* expression1 = expressionParser.parseExpression(String.*valueOf*(22+33));  
 int ans = (int) expression1.getValue();  
 System.out.println(ans);  
 }  
}

Output:



8) Write a Java program to demonstrate InitializingBean and DisposableBean.

Try Different ways:

(Use init-method and destroy-method in xml config file)

(Use @PostConstruct and @PreDestroy)

**Class XmlBeanLifeCycle:**

package org.example.AssignmentQ8;  
  
public class XmlBeanLifeCycle {  
 private double price;  
  
 public XmlBeanLifeCycle() {  
 super();  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 System.out.println("Setting the value of data members");  
 this.price = price;  
 }  
  
 public void init(){  
 System.out.println("Calling the initialization method of bean");  
 }  
 public void destroy(){  
 System.out.println("Calling the destroy method of bean");  
 }  
 @Override  
 public String toString() {  
 return "ORDER PRICE: {" +  
 "price=" + price +  
 '}';  
 }  
}

**Class InterfaceBeanLifeCycle:**

package org.example.AssignmentQ8;  
  
import org.springframework.beans.factory.*DisposableBean*;  
import org.springframework.beans.factory.*InitializingBean*;  
  
public class InterfaceBeanLifeCycle implements *InitializingBean*, *DisposableBean* {  
 private double price;  
  
 public InterfaceBeanLifeCycle() {  
 super();  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 @Override  
 public String toString() {  
 return "Interface Price: {" +  
 "price=" + price +  
 '}';  
 }  
  
 @Override  
 public void afterPropertiesSet() throws Exception {  
 System.out.println("Calling init method using interface");  
 }  
  
 @Override  
 public void destroy() throws Exception {  
 System.out.println("Calling destroy method using interface");  
 }  
}

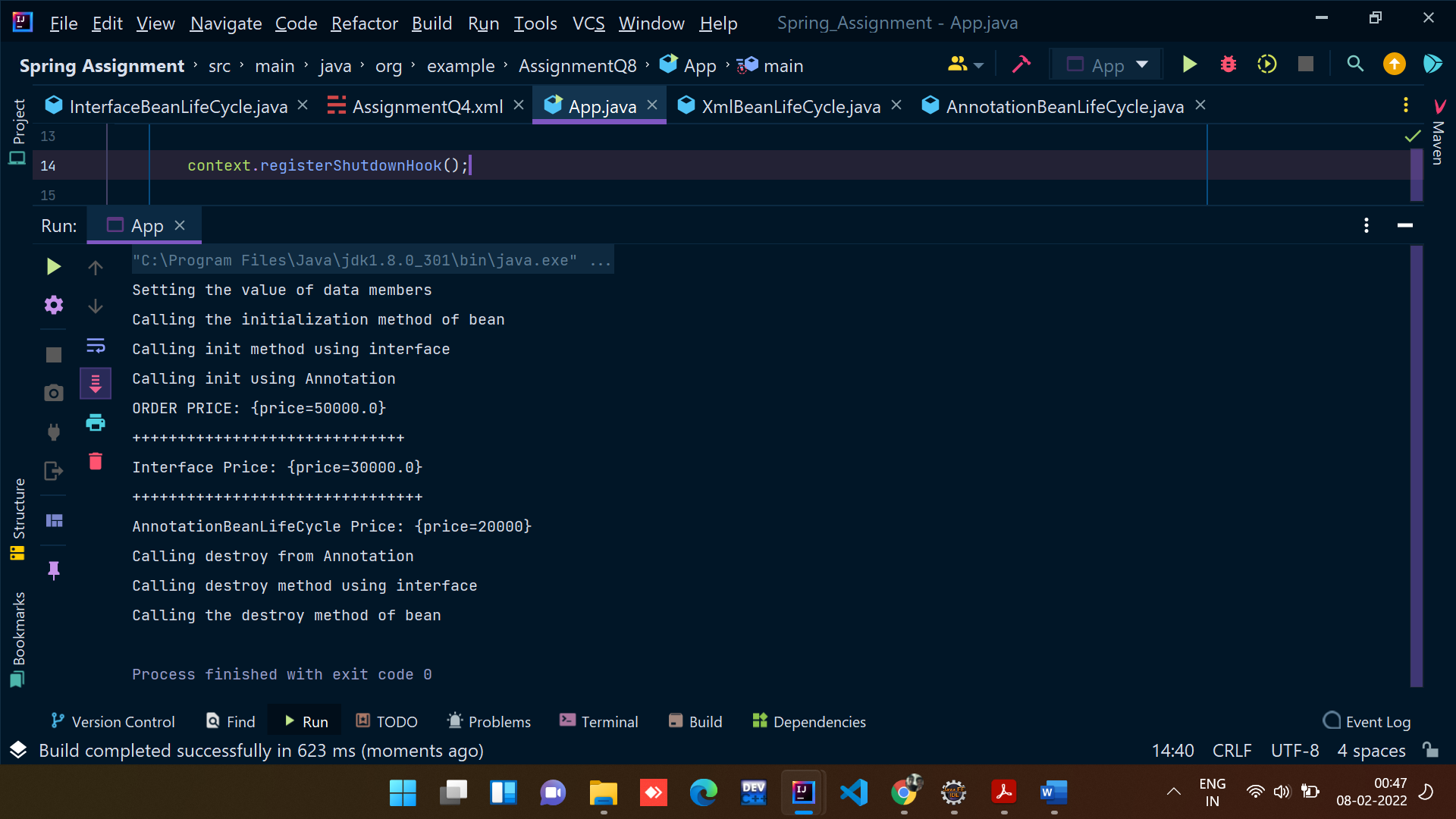
**Class AnnotationBeanLifeCycle:**

package org.example.AssignmentQ8;  
  
import javax.annotation.PostConstruct;  
import javax.annotation.PreDestroy;  
  
public class AnnotationBeanLifeCycle {  
 int price;  
  
 public AnnotationBeanLifeCycle() {  
 super();  
 }  
  
 @Override  
 public String toString() {  
 return "AnnotationBeanLifeCycle Price: {" +  
 "price=" + price +  
 '}';  
 }  
  
 public int getPrice() {  
 return price;  
 }  
  
 public void setPrice(int price) {  
 this.price = price;  
 }  
  
 @PostConstruct  
 public void init(){  
 System.out.println("Calling init using Annotation");  
 }  
 @PreDestroy  
 public void destroy(){  
 System.out.println("Calling destroy from Annotation");  
 }  
}

**App.java**

package org.example.AssignmentQ8;  
  
import org.springframework.context.support.AbstractApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class App {  
 public static void main(String[] args) {  
 *//This is the code segment for implementing bean lifecycle for XML.* AbstractApplicationContext context = new ClassPathXmlApplicationContext("AssignmentQ4.xml");  
 XmlBeanLifeCycle obj = (XmlBeanLifeCycle) context.getBean("XML1");  
  
 System.out.println(obj);  
  
 context.registerShutdownHook();  
  
 System.out.println("++++++++++++++++++++++++++++++");  
 *//This is the code segment for implementing bean lifecycle for interfaces.* InterfaceBeanLifeCycle obj1 = (InterfaceBeanLifeCycle) context.getBean("Interface1");  
 System.out.println(obj1);  
  
 System.out.println("++++++++++++++++++++++++++++++++");  
 *//This is the code segment for implementing bean lifecycle for Annotations.* AnnotationBeanLifeCycle obj2 = (AnnotationBeanLifeCycle)context.getBean("Annotation1");  
 System.out.println(obj2);  
 }  
  
}

**Output:**



9) Write a Java program to demonstrate Complete Bean Life cycle.

**Class XmlBeanLifeCycle:**

package org.example.AssignmentQ8;  
  
public class XmlBeanLifeCycle {  
 private double price;  
  
 public XmlBeanLifeCycle() {  
 super();  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 System.out.println("Setting the value of data members");  
 this.price = price;  
 }  
  
 public void init(){  
 System.out.println("Calling the initialization method of bean");  
 }  
 public void destroy(){  
 System.out.println("Calling the destroy method of bean");  
 }  
 @Override  
 public String toString() {  
 return "ORDER PRICE: {" +  
 "price=" + price +  
 '}';  
 }  
}

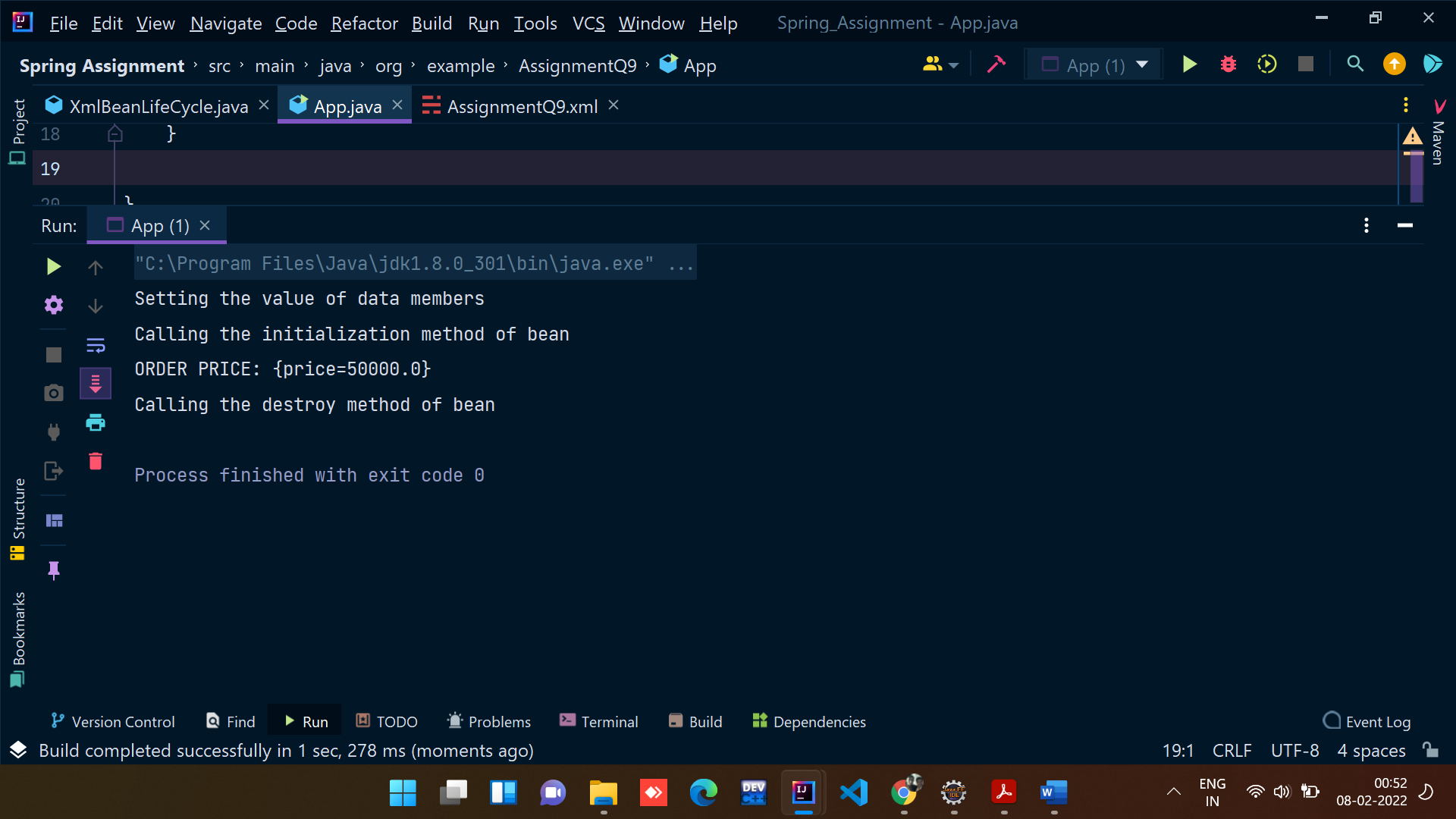
**App.java**

package org.example.AssignmentQ9;  
  
import org.example.AssignmentQ8.AnnotationBeanLifeCycle;  
import org.example.AssignmentQ8.InterfaceBeanLifeCycle;  
import org.example.AssignmentQ8.XmlBeanLifeCycle;  
import org.springframework.context.support.AbstractApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class App {  
 public static void main(String[] args) {  
 *//This is the code segment for implementing bean lifecycle for XML.* AbstractApplicationContext context = new ClassPathXmlApplicationContext("AssignmentQ9.xml");  
 org.example.AssignmentQ8.XmlBeanLifeCycle obj = (XmlBeanLifeCycle) context.getBean("XML1");  
  
 System.out.println(obj);  
  
 context.registerShutdownHook();  
 }  
  
}

**AssignmentQ9.xml:**

<?*xml version*="1.0" *encoding*="UTF-8"?>  
<beans *xmlns:xsi*="http://www.w3.org/2001/XMLSchema-instance"  
 *xmlns:context*="http://www.springframework.org/schema/context"  
 *xmlns*="http://www.springframework.org/schema/beans"  
 *xsi:schemaLocation*="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd">  
  
 <context:annotation-config/>  
  
 <bean *class*="org.example.AssignmentQ8.XmlBeanLifeCycle" *name*="XML1" *init-method*="init" *destroy-method*="destroy">  
 <property *name*="price" *value*="50000"/>  
 </bean>  
  
</beans>

**Output:**



10) Write a java program to demonstrate ApplicationContextAware interface.

**Example.java**

package org.example.AssignmentQ10;  
  
import org.springframework.beans.BeansException;  
import org.springframework.beans.factory.*BeanNameAware*;  
import org.springframework.context.*ApplicationContext*;  
import org.springframework.context.*ApplicationContextAware*;  
  
public class Example implements *ApplicationContextAware*, *BeanNameAware* {  
 private String place;  
 Demo demo;  
  
 public String getPlace() {  
 return place;  
 }  
  
 public void setPlace(String place) {  
 this.place = place;  
 }  
  
 public Demo getDemo() {  
 return demo;  
 }  
  
 public void setDemo(Demo demo) {  
 this.demo = demo;  
 }  
  
 @Override  
 public String toString() {  
 return "Example: [" +  
 "place='" + place + '\'' +  
 ", demo=" + demo +  
 ']';  
 }  
  
 @Override  
 public void setBeanName(String name) {  
 System.out.println("Bean name is: "+name);  
 }  
  
 @Override  
 public void setApplicationContext(*ApplicationContext* applicationContext) throws BeansException {  
 System.out.println("Application name: "+ applicationContext);  
 }  
}

**Demo.java**

package org.example.AssignmentQ10;  
  
public class Demo {  
 private int x;  
 private int y;  
  
 public int getX() {  
 return x;  
 }  
  
 public void setX(int x) {  
 this.x = x;  
 }  
  
 public int getY() {  
 return y;  
 }  
  
 public void setY(int y) {  
 this.y = y;  
 }  
}

**Test.java**

package org.example.AssignmentQ10;  
  
import org.springframework.context.*ApplicationContext*;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Test {  
 public static void main(String[] args) {  
 *ApplicationContext* context = new ClassPathXmlApplicationContext("AssignmentQ10.xml");  
 Example example1 = context.getBean("example1", Example.class);  
 System.out.println(example1);  
 }  
}

**XML file:**

<?*xml version*="1.0" *encoding*="UTF-8"?>  
<beans *xmlns:xsi*="http://www.w3.org/2001/XMLSchema-instance"  
 *xmlns:context*="http://www.springframework.org/schema/context"  
 *xmlns*="http://www.springframework.org/schema/beans"  
 *xsi:schemaLocation*="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd">  
  
 <context:annotation-config/>  
 <bean *class*="org.example.AssignmentQ10.Demo" *id*="demo1">  
 <property *name*="x" *value*="20"/>  
 <property *name*="y" *value*="40"/>  
 </bean>  
  
 <bean *class*="org.example.AssignmentQ10.Example" *name*="example1" *autowire*="byType">  
 <property *name*="place" *value*="Korba"/>  
 </bean>  
</beans>

**Output:**

