**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, customerName, customerContact, customerAddress.**

**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.

**Create Address Class**

**package** com.capgemini.springAssignment.Q1;

**public** **class** Address {

**private** String street;

**private** String city;

**private** String state;

**private** String country;

**private** String zip;

// Constructor

**public** Address(){}

**public** Address(String street, String city, String state, String country, String zip) {

**this**.street = street;

**this**.city = city;

**this**.state = state;

**this**.country = country;

**this**.zip = zip;

}

// Getter and Setter

**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** String getCountry() {

**return** country;

}

**public** **void** setCountry(String country) {

**this**.country = country;

}

**public** String getZip() {

**return** zip;

}

**public** **void** setZip(String zip) {

**this**.zip = zip;

}

}

**Create Customer Class**

**package** com.capgemini.springAssignment.Q1;

**public** **class** Customer {

**private** **int** customerId;

**private** String customerName;

**private** String customerContact;

**private** Address customerAddress;

**public** Customer(){}

**public** Customer(**int** customerId, String customerName, String customerContact, Address customerAddress) {

**this**.customerId = customerId;

**this**.customerName = customerName;

**this**.customerContact = customerContact;

**this**.customerAddress = customerAddress;

}

**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** String getCustomerContact() {

**return** customerContact;

}

**public** **void** setCustomerContact(String customerContact) {

**this**.customerContact = customerContact;

}

**public** Address getCustomerAddress() {

**return** customerAddress;

}

**public** **void** setCustomerAddress(Address customerAddress) {

**this**.customerAddress = customerAddress;

}

**public** **void** customerDetail(){

System.***out***.println("Customer Id: "+getCustomerId()+"\n"+

"Name: "+getCustomerName()+"\n"+

"Customer Contact: "+getCustomerContact()+"\n"+

"Address: "+

"Street: "+getCustomerAddress().getStreet()+" City: "+getCustomerAddress().getCity()+" State: "+getCustomerAddress().getState()+" Country: "+getCustomerAddress().getCountry()+" zip: "+getCustomerAddress().getZip());

}

}

**Create Test Class**

**package** com.capgemini.springAssignment.Q1;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** Test {

**private** **static** ApplicationContext *context*;

**public** **static** **void** main(String[] args) {

*context* = **new** ClassPathXmlApplicationContext("spring1.xml");

// using the setter injection

Customer customer=(Customer) *context*.getBean("customer");

customer.customerDetail();

// using constructor injection

Customer customerCon=(Customer) *context*.getBean("customerCon");

customerCon.customerDetail();

}

}

**XML Configuration File**

<?xml version="1.0" encoding="UTF-8" ?>

<!--

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "<http://www.springframework.org/dtd/spring-beans-2.0.dtd>">

-->

<beans xmlns = "<http://www.springframework.org/schema/beans>"

xmlns:xsi = "<http://www.w3.org/2001/XMLSchema-instance>"

xmlns:context = "<http://www.springframework.org/schema/context>"

xsi:schemaLocation = "<http://www.springframework.org/schema/beans>

<http://www.springframework.org/schema/beans/spring-beans-3.0.xsd>

<http://www.springframework.org/schema/context>

<http://www.springframework.org/schema/context/spring-context-3.0.xsd>">

<!-- Using setter injection -->

<bean id="customer" class="com.capgemini.springAssignment.Q1.Customer">

<property name="customerId" value="111"/>

<property name="customerName" value="Ria"/>

<property name="customerContact" value="8755946301"/>

<property name="customerAddress" ref="custAddress"/>

</bean>

<bean id="custAddress" class="com.capgemini.springAssignment.Q1.Address">

<property name="street" value="Khaperkheda"/>

<property name="city" value="Nagpur"/>

<property name="state" value="Maharashtra"/>

<property name="country" value="India"/>

<property name="zip" value="262531" />

</bean>

<!-- Using constructor injection -->

<bean id="customerCon" class = "com.capgemini.springAssignment.Q1.Customer">

<constructor-arg type ="int" value="112"/>

<constructor-arg value="Hritik"/>

<constructor-arg value="9876543210"/>

<constructor-arg ref="custAddressCon"/>

</bean>

<bean id="custAddressCon" class="com.capgemini.springAssignment.Q1.Address">

<constructor-arg value="Place"/>

<constructor-arg value="Mumbai"/>

<constructor-arg value="Maharashtra"/>

<constructor-arg value="India"/>

<constructor-arg value="262531" />

</bean>

<!-- instead of adding each bean postPorcessor -->

<context:annotation-config/>

</beans>

**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.**

* Write a program where answers is of type List<String> or String []
* Write a program where answers is of type Set<String>
* Write a program where answers is of type Map<Integer, String>

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

* Create a Test class with main() method, get Question bean from ApplicationContext object and print question and its answers.
* Also write the JUnit Test cases for above program.

- Use XML based configuration.

**Create Question Class:**

**package** com.capgemini.springAssignment.Q2;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.Map.Entry;

**import** java.util.Set;

**public** **class** Question {

**private** **int** questionId;

**private** String question;

**private** List<String> answers;

**private** Set<String> answersSet;

**private** Map<Integer,String> answersMap;

**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;

}

**public** Set<String> getAnswersSet() {

**return** answersSet;

}

**public** **void** setAnswersSet(Set<String> answersSet) {

**this**.answersSet = answersSet;

}

**public** Map<Integer, String> getAnswersMap() {

**return** answersMap;

}

**public** **void** setAnswersMap(Map<Integer, String> answersMap) {

**this**.answersMap = answersMap;

}

**public** **void** displayAnswerList(){

System.***out***.println(questionId+". "+question);

**for**(String ans:answers){

System.***out***.println(ans);

}

}

**public** **void** displayAnswerSet(){

System.***out***.println(questionId+". "+question);

**for**(String ans:answersSet){

System.***out***.println(ans);

}

}

**public** **void** displayAnswerMap(){

System.***out***.println(questionId+". "+question);

Set<Entry<Integer, String>> set = answersMap.entrySet();

Iterator<Entry<Integer, String>> itr = set.iterator();

**while** (itr.hasNext()){

Entry<Integer, String> m1 =itr.next();

System.***out***.println(m1.getKey()+". "+m1.getValue());

}

}

}

**Create Test Class:**



**Output**



**3. Example on autowiring**

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

* Create a BankAccount class with following attributes: accountId, accountHolderName, accountType, accountBalance
* Create an interface BankAccountRepository with following methods: public double getBalance(long accountId)

public double updateBalance(long accountId, double newBalance): Note: Above method returns updated balance.

* Create a class BankAccountepositoryImpl that implements BankAccountRepository interface.

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

* 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)

* Create a class BankAccountServiceImpl that implements BankAccountService interface.
* 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)

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

**Create BankAccount class**

**package** com.capgemini.springAssignment.Q3;

**public** **class** BankAccount {

**private** **long** accountId;

**private** String accountHolderName;

**private** String accountType;

**private** **double** accountBalance;

**public** BankAccount(){}

**public** BankAccount(**long** accountId, String accountHolderName, String accountType, **double** accountBalance) {

**this**.accountId = accountId;

**this**.accountHolderName = accountHolderName;

**this**.accountType = accountType;

**this**.accountBalance = accountBalance;

}

**public** **long** getAccountId() {

**return** accountId;

}

**public** **void** setAccountId(**long** 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{" +

"accountId=" + accountId +

", accountHolderName='" + accountHolderName + '\'' +

", accountType='" + accountType + '\'' +

", accountBalance=" + accountBalance +

'}';

}

}

**Create BankAccountRepository Interface**

**package** com.capgemini.springAssignment.Q3;

**public** **interface** BankAccountRepository {

**public** **double** getBalance(**long** accountId);

**public** **double** updateBalance(**long** accountId, **double** newBalance);

}

**Create BankAccountepositoryImpl Class**

**package** com.capgemini.springAssignment.Q3;

**public** **class** BankAccountepositoryImpl **implements** BankAccountRepository {

**private** BankAccount bankAccount1;

**private** BankAccount bankAccount2;

**public** BankAccountepositoryImpl(){}

**public** BankAccountepositoryImpl(BankAccount bankAccount1, BankAccount bankAccount2) {

**this**.bankAccount1 = bankAccount1;

**this**.bankAccount2 = bankAccount2;

}

**public** **double** getBalance(**long** accountId) {

**if**(**this**.bankAccount1.getAccountId()==accountId){

**return** bankAccount1.getAccountBalance();

}

**else**{

**return** bankAccount2.getAccountBalance();

}

}

**public** **double** updateBalance(**long** accountId, **double** newBalance) {

**double** updatedBal=0;

**if**(**this**.bankAccount1.getAccountId()==accountId){

updatedBal = newBalance;

bankAccount1.setAccountBalance(updatedBal);

}

**else**{

updatedBal = newBalance;

bankAccount2.setAccountBalance(updatedBal);

}

**return** updatedBal;

}

}

**Create BankAccountService interface**

**package** com.capgemini.springAssignment.Q3;

**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);

}

**Create BankAccountServiceImpl Class**

**package** com.capgemini.springAssignment.Q3;

**public** **class** BankAccountServiceImpl **implements** BankAccountService {

**public** **double** withdraw(**long** accountId, **double** balance) {

**return** 0;

}

**public** **double** deposit(**long** accountId, **double** balance) {

**return** 0;

}

**public** **double** getBalance(**long** accountId) {

**return** 0;

}

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

**return** **false**;

}

}

**Create BankAccountController Class**

**package** com.capgemini.springAssignment.Q3;

**import** org.springframework.beans.BeansException;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.ApplicationContextAware;

**public** **class** BankAccountController **implements** ApplicationContextAware {

**private** ApplicationContext context;

**public** **double** withdraw(**long** accountId, **double** balanceToBeWithdraw){

**double** newBalance=0;

BankAccountepositoryImpl acc=(BankAccountepositoryImpl) context.getBean("BankRepo");

**if**(acc.getBalance(accountId)>=balanceToBeWithdraw)

newBalance = acc.getBalance(accountId)-balanceToBeWithdraw;

acc.updateBalance(accountId,newBalance);

**return** newBalance;

}

**public** **double** deposit(**long** accountId, **double** balance){

BankAccountepositoryImpl acc=(BankAccountepositoryImpl) context.getBean("BankRepo");

**double** Newbalance = acc.getBalance(accountId)+balance;

**return** acc.updateBalance(accountId,Newbalance);

}

**public** **double** getBalance(**long** accountId){

BankAccountepositoryImpl balance=(BankAccountepositoryImpl) context.getBean("BankRepo");

**return** balance.getBalance(accountId);

}

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

BankAccountepositoryImpl accountepository=(BankAccountepositoryImpl) context.getBean("BankRepo");

**if**(accountepository.getBalance(fromAccount)>=amont){

**double** updatedAmt= amont+accountepository.getBalance(toAccount);

// updating both the accounts

accountepository.updateBalance(fromAccount,accountepository.getBalance(fromAccount)-amont);

accountepository.updateBalance(toAccount,updatedAmt);

**return** **true**;

}

**return** **false**;

}

**public** **void** setApplicationContext(ApplicationContext applicationContext) **throws** BeansException {

**this**.context=applicationContext;

}

}

**Create Test Class**

**package** com.capgemini.springAssignment.Q3;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** Test {

**private** **static** ApplicationContext *context*;

**public** **static** **void** main(String[] args) {

*context* = **new** ClassPathXmlApplicationContext("springQ3.xml");

BankAccountController controller = (BankAccountController)*context*.getBean("controller");

System.***out***.println(controller.getBalance(1171));

System.***out***.println(controller.deposit(1171,5000));

System.***out***.println("+++++++++++++++++++++++++++");

System.***out***.println(controller.withdraw(1171,5000));

System.***out***.println(controller.getBalance(1171));

System.***out***.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.***out***.println(controller.fundTransfer(1171,1172,5000));

System.***out***.println(controller.getBalance(1171));

System.***out***.println(controller.getBalance(1172));

}

}

**XML Configuration File**

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "<http://www.springframework.org/dtd/spring-beans-2.0.dtd>">

<beans>

<bean id="bankAccount1" class="com.capgemini.springAssignment.Q3.BankAccount">

<constructor-arg type = "long" value="1171"/>

<constructor-arg value="neeraj"/>

<constructor-arg value="Saving"/>

<constructor-arg type="double" value="15000"/>

</bean>

<bean id="bankAccount2" class="com.capgemini.springAssignment.Q3.BankAccount">

<constructor-arg type = "long" value="1172"/>

<constructor-arg value="Ramesh"/>

<constructor-arg value="Current"/>

<constructor-arg type="double" value="20000"/>

</bean>

<!-- without autowiring

<bean id="BankRepo" class="com.capgemini.springAssignment.Q3.BankAccountepositoryImpl">

<property name="bankAccount">

<list>

<ref bean="BankAccount1"/>

<ref bean="BankAccount2"/>

</list>

</property>

</bean>

-->

<!-- autowiring by Name

<bean id="BankRepo" class="com.capgemini.springAssignment.Q3.BankAccountepositoryImpl" autowire="byName">

</bean>

-->

<!-- autowiring by type ( it will work if we have only one type )

<bean id="BankRepo" class="com.capgemini.springAssignment.Q3.BankAccountepositoryImpl" autowire="byType"/>

-->

<!-- autowiring by constructor -->

<bean id="BankRepo" class="com.capgemini.springAssignment.Q3.BankAccountepositoryImpl" autowire="constructor">

</bean>

<bean id="controller" class="com.capgemini.springAssignment.Q3.BankAccountController">

</bean>

</beans>

**OutPut:**



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

**Modify the above application, use annotations and java based configuration.**

**Create BankAccount Class**

**package** com.capgemini.springAssignment.Q4;

**public** **class** BankAccount {

**private** **long** accountId;

**private** String accountHolderName;

**private** String accountType;

**private** **double** accountBalance;

**public** BankAccount(){}

**public** BankAccount(**long** accountId, String accountHolderName, String accountType, **double** accountBalance) {

**this**.accountId = accountId;

**this**.accountHolderName = accountHolderName;

**this**.accountType = accountType;

**this**.accountBalance = accountBalance;

}

**public** **long** getAccountId() {

**return** accountId;

}

**public** **void** setAccountId(**long** 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{" +

"accountId=" + accountId +

", accountHolderName='" + accountHolderName + '\'' +

", accountType='" + accountType + '\'' +

", accountBalance=" + accountBalance +

'}';

}

}

**Create BankAccountController**





**Create BankAccountepositoryImpl**

**package** com.capgemini.springAssignment.Q4;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

@Repository("BankRepo")

**public** **class** BankAccountepositoryImpl **implements** BankAccountRepository {

@Autowired

**private** BankAccount bankAccount1;

@Autowired

**private** BankAccount bankAccount2;

**public** BankAccountepositoryImpl(){}

**public** BankAccountepositoryImpl(BankAccount bankAccount1, BankAccount bankAccount2) {

**this**.bankAccount1 = bankAccount1;

**this**.bankAccount2 = bankAccount2;

}

/\*

public BankAccount getBankAccount1() {

return bankAccount1;

}

public void setBankAccount1(BankAccount bankAccount1) {

this.bankAccount1 = bankAccount1;

}

public BankAccount getBankAccount2() {

return bankAccount2;

}

public void setBankAccount2(BankAccount bankAccount2) {

this.bankAccount2 = bankAccount2;

}

\*/

/\*

public List<BankAccount> getBankAccount() {

return bankAccount;

}

public void setBankAccount(List<BankAccount> bankAccount) {

this.bankAccount = bankAccount;

}

\*/

**public** **double** getBalance(**long** accountId) {

/\* using a list

for(BankAccount account:bankAccount){

if(accountId==account.getAccountId()){

return account.getAccountBalance();

}

}

\*/

**if**(**this**.bankAccount1.getAccountId()==accountId){

**return** bankAccount1.getAccountBalance();

}

**else**{

**return** bankAccount2.getAccountBalance();

}

}

**public** **double** updateBalance(**long** accountId, **double** newBalance) {

**double** updatedBal=0;

**if**(**this**.bankAccount1.getAccountId()==accountId){

updatedBal = newBalance;

bankAccount1.setAccountBalance(updatedBal);

}

**else**{

updatedBal = newBalance;

bankAccount2.setAccountBalance(updatedBal);

}

/\*

for(BankAccount account:bankAccount){

if(accountId==account.getAccountId()){

updatedBal = newBalance;

account.setAccountBalance(updatedBal);

}

}

\*/

**return** updatedBal;

}

}

**Create BankAccountRepository**

**package** com.capgemini.springAssignment.Q4;

**public** **interface** BankAccountRepository {

**public** **double** getBalance(**long** accountId);

**public** **double** updateBalance(**long** accountId, **double** newBalance);

}

**Create BankAccountService Interface**

**package** com.capgemini.springAssignment.Q4;

**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);

}

**Create BankAccountServiceImpl Class**

**package** com.capgemini.springAssignment.Q4;

**import** org.springframework.beans.BeansException;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.ApplicationContextAware;

**import** org.springframework.stereotype.Service;

@Service("service")

**public** **class** BankAccountServiceImpl **implements** BankAccountService, ApplicationContextAware {

**private** ApplicationContext context;

**public** **double** withdraw(**long** accountId, **double** balanceToBeWithdraw) {

**double** newBalance=0;

BankAccountepositoryImpl acc=(BankAccountepositoryImpl) context.getBean("BankRepo");

**if**(acc.getBalance(accountId)>=balanceToBeWithdraw)

newBalance = acc.getBalance(accountId)-balanceToBeWithdraw;

acc.updateBalance(accountId,newBalance);

**return** newBalance;

}

**public** **double** deposit(**long** accountId, **double** balance) {

BankAccountepositoryImpl acc=(BankAccountepositoryImpl) context.getBean("BankRepo");

**double** Newbalance = acc.getBalance(accountId)+balance;

**return** acc.updateBalance(accountId,Newbalance);

}

**public** **double** getBalance(**long** accountId) {

BankAccountepositoryImpl balance=(BankAccountepositoryImpl) context.getBean("BankRepo");

**return** balance.getBalance(accountId);

}

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

BankAccountepositoryImpl accountepository=(BankAccountepositoryImpl) context.getBean("BankRepo");

**if**(accountepository.getBalance(fromAccount)>=amont){

**double** updatedAmt= amont+accountepository.getBalance(toAccount);

// updating both the accounts

accountepository.updateBalance(fromAccount,accountepository.getBalance(fromAccount)-amont);

accountepository.updateBalance(toAccount,updatedAmt);

**return** **true**;

}

**return** **false**;

}

**public** **void** setApplicationContext(ApplicationContext applicationContext) **throws** BeansException {

**this**.context=applicationContext;

}

}

**Create Test Class**

**package** com.capgemini.springAssignment.Q4;

**import** org.springframework.context.ApplicationContext;

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

**import** com.capgemini.springAssignment.Q3.BankAccountController;

**public** **class** Test {

**private** **static** ApplicationContext *context*;

**public** **static** **void** main(String[] args) {

*context* = **new** ClassPathXmlApplicationContext("springQ4.xml");

BankAccountController controller = (BankAccountController)*context*.getBean("controller");

System.***out***.println(controller.getBalance(1171));

System.***out***.println(controller.deposit(1171,5000));

System.***out***.println("+++++++++++++++++++++++++++");

System.***out***.println(controller.withdraw(1171,5000));

System.***out***.println(controller.getBalance(1171));

System.***out***.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.***out***.println(controller.fundTransfer(1171,1172,5000));

System.***out***.println(controller.getBalance(1171));

System.***out***.println(controller.getBalance(1172));

}

}

**XML Configuration File**

<?xml version="1.0" encoding="UTF-8" ?>

<beans xmlns = "<http://www.springframework.org/schema/beans>"

xmlns:xsi = "<http://www.w3.org/2001/XMLSchema-instance>"

xmlns:context = "<http://www.springframework.org/schema/context>"

xsi:schemaLocation = "<http://www.springframework.org/schema/beans>

<http://www.springframework.org/schema/beans/spring-beans-3.0.xsd>

<http://www.springframework.org/schema/context>

<http://www.springframework.org/schema/context/spring-context-3.0.xsd>">

<bean id="bankAccount1" class="com.capgemini.springAssignment.Q4.BankAccount">

<constructor-arg type = "long" value="1171"/>

<constructor-arg value="neeraj"/>

<constructor-arg value="Saving"/>

<constructor-arg type="double" value="15000"/>

</bean>

<bean id="bankAccount2" class="com.capgemini.springAssignment.Q3.BankAccount">

<constructor-arg type = "long" value="1172"/>

<constructor-arg value="Ramesh"/>

<constructor-arg value="Current"/>

<constructor-arg type="double" value="20000"/>

</bean>

<context:annotation-config/> <context:component-scan base-package="com.capgemini.springAssignment.Q4"/>

<context:annotation-config/>

</beans>

**Output**



**5. Write a program to demonstrate use of @Resource, @Inject, @Required annotations**

**6. Example of @Component, @Value, @PropertySource & Environment**

* Create a dbConfig.properties file which contains database configuration details like driver class name, dburl, username, password.
* Create a Java class in which you have to read all properties and display on a console. (Use @Component, @Value or Environment and @PropertyResource).

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

**Create SPELDemo Class**

**package** com.capgemini.springAssignment.Q7;

**import** org.springframework.expression.Expression;

**import** org.springframework.expression.ExpressionParser;

**import** org.springframework.expression.spel.standard.SpelExpressionParser;

**public** **class** SPELDemo {

**public** **static** **void** main(String[] args) {

/\*\*

\* SpEL is an exression language supporting the features of querying and manipulating an object graph at runtime.

\*/

ExpressionParser parser = **new** SpelExpressionParser();

Expression exp = parser.parseExpression("'Hello SPEL'");

String msg = (String)exp.getValue();

System.***out***.println(msg);

ExpressionParser parser1 = **new** SpelExpressionParser();

Expression exp1 = parser1.parseExpression("'Welcome SPEL'.concat('!')");

String message = (String) exp1.getValue();

System.***out***.println(message);

}

}

**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)**

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

**Create BeanLifeCycle Class**

**package** com.capgemini.springAssignment.Q9;

**import** org.springframework.beans.factory.DisposableBean;

**import** org.springframework.beans.factory.InitializingBean;

**public** **class** BeanLifeCycle **implements** InitializingBean, DisposableBean {

**private** **int** id;

**private** String name;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** display(){

System.***out***.println("id: "+getId()+"\n"+"Name: "+getName());

}

**public** **void** destroy() **throws** Exception {

System.***out***.println("Destroying the bean");

}

**public** **void** afterPropertiesSet() **throws** Exception {

System.***out***.println("Initization bean");

}

}

**Create Test Class**

**package** com.capgemini.springAssignment.Q9;

**import** org.springframework.context.support.AbstractApplicationContext;

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

**public** **class** Test {

**private** **static** AbstractApplicationContext *context*;

**public** **static** **void** main(String[] args) {

*context* = **new** ClassPathXmlApplicationContext("springQ9.xml");

*context*.registerShutdownHook();

BeanLifeCycle beanLifeCycle=(BeanLifeCycle)*context*.getBean("LifeCycle");

beanLifeCycle.display();

}

}

**XML Configuration file**

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "<http://www.springframework.org/dtd/spring-beans-2.0.dtd>">

<beans>

<bean id="LifeCycle" class="com.capgemini.springAssignment.Q9.BeanLifeCycle">

<property name="id" value="10"/>

<property name="name" value="Captain America"/>

</bean>

<bean id="LifeCycle1" class="com.capgemini.springAssignment.Q9.BeanLifeCycle">

<property name="id" value="11"/>

<property name="name" value="Iron Man"/>

</bean>

</beans>

**Output**



**10. Write a java program to demonstrate ApplicationContextAware interface.**

**Create Demo Class**

**package** com.capgemini.springAssignment.Q10;

**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;

}

}

**Create ContectAwareExample Class**

**package** com.capgemini.springAssignment.Q10;

**import** org.springframework.beans.BeansException;

**import** org.springframework.beans.factory.BeanNameAware;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.ApplicationContextAware;

**public** **class** ContextAwareExample **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;

}

**public** **void** display(){

System.***out***.println("Coordinate of "+getPlace()+" ("+getDemo().getX()+","+getDemo().getY()+")");

}

**public** **void** setApplicationContext(ApplicationContext applicationContext) **throws** BeansException {

System.***out***.println("application name "+applicationContext);

}

**public** **void** setBeanName(String s) {

System.***out***.println("Bean Name is:"+s);

}

}

**Create Test Class**

**package** com.capgemini.springAssignment.Q10;

**import** org.springframework.context.ApplicationContext;

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

**public** **class** Test {

**private** **static** ApplicationContext *context*;

**public** **static** **void** main(String[] args) {

*context* = **new** ClassPathXmlApplicationContext("springQ10.xml");

ContextAwareExample contextAwareExample = (ContextAwareExample) *context*.getBean("contextAware");

contextAwareExample.display();

ContextAwareExample contextAwareExample1 = (ContextAwareExample) *context*.getBean("contextAware1");

contextAwareExample1.display();

}

}

**XML Configuration File**

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "<http://www.springframework.org/dtd/spring-beans-2.0.dtd>">

<beans>

<bean id="contextAware" class="com.capgemini.springAssignment.Q10.ContextAwareExample">

<property name="place" value="Delhi"/>

<property name="demo" ref="Demo"/>

</bean>

<bean id="contextAware1" class="com.capgemini.springAssignment.Q10.ContextAwareExample">

<property name="place" value="Mumbai"/>

<property name="demo" ref="Demo"/>

</bean>

<bean id="Demo" class="com.capgemini.springAssignment.Q10.Demo" scope="prototype">

<property name="x" value="10"/>

<property name="y" value="20"/>

</bean>

</beans>

**Output**

