**Q1. Create a class Database with 2 instance variables port and name. Configure Database class bean in spring container through Spring XML. Initialize instance variables using setter method**

**Solution**

**Database.java**

**package demo.springCore.question1;**

**public class Database {**

**int port;**

**String name;**

**@Override**

**public String toString() {**

**return "Database{" +**

**"port=" + port +**

**", name='" + name + '\'' +**

**'}';**

**}**

**public int getPort() {**

**return port;**

**}**

**public void setPort(int port) {**

**this.port = port;**

**}**

**public String getName() {**

**return name;**

**}**

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

**this.name = name;**

**}**

**}**

**Spring-config.xml**

***<!-- Bean for question1-Database-->***

**<bean id="database" class="demo.springCore.question1.Database">**

**<property name="name" value="MySql"></property>**

**<property name="port" value="3306"></property>**

**</bean>**

**Q2. Get the bean of the class from spring container and print the values of the instance variable.**

**Solution**

**package demo.springCore.question2;**

**import demo.springCore.question1.Database;**

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

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

**public class Main {**

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

**ApplicationContext applicationContext=new ClassPathXmlApplicationContext("spring-config.xml");**

**Database database=applicationContext.getBean("database",Database.class);**

**System.*out*.println("Got database object with following properties: ");**

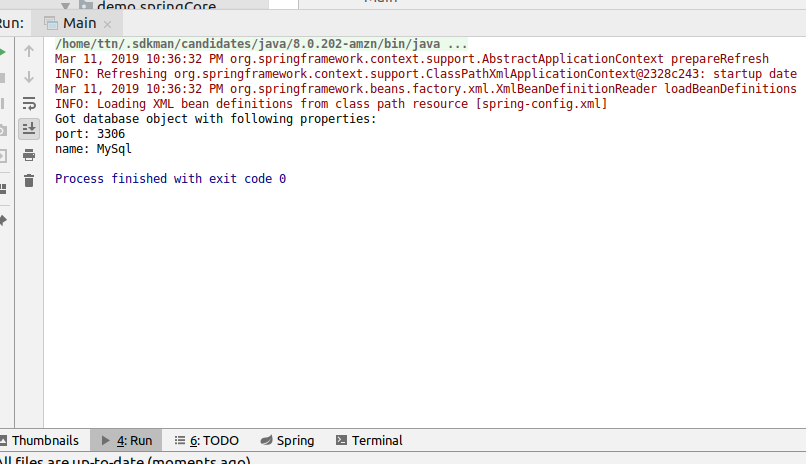
**System.*out*.println("port: "+database.getPort());**

**System.*out*.println("name: "+database.getName());**

**}**

**}**

**Output**



**Q3. Create a class Restaurant. Create an interface HotDrink with an abstract method prepareHotDrink. Create two classes Tea and ExpressTea which implements HotDrink Class. Create an instance variable of type HotDrink in Restaurant class. Configure Tea and ExpressTea classes beans in Spring XML. create a bean with the name teaRestaurant of type Restaurant which inject Tea object as dependency using setter method**

**Solution**

**HotDrink.java**

**package demo.springCore.question3;**

**public interface HotDrink {**

**void prepareHotDrink();**

**}**

**Tea.java**

**package demo.springCore.question3;**

**public class Tea implements HotDrink {**

**@Override**

**public void prepareHotDrink() {**

**System.*out*.println("Preparing tea");**

**}**

**}**

**ExpressTea.java**

**package demo.springCore.question3;**

**public class ExpressTea implements HotDrink {**

**@Override**

**public void prepareHotDrink() {**

**System.*out*.println("Preparing express tea");**

**}**

**}**

**Restaurant.java**

**package demo.springCore.question3;**

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

**public class Restaurant {**

**HotDrink hotDrink;**

**public HotDrink getHotDrink() {**

**return hotDrink;**

**}**

**public void setHotDrink(HotDrink hotDrink) {**

**this.hotDrink = hotDrink;**

**}**

**public Restaurant(){}**

**public Restaurant(HotDrink hotDrink) {**

**this.hotDrink = hotDrink;**

**}**

**}**

**Spring-config.xml**

***<!--Beans for Question3-->***

**<bean id="tea" class="demo.springCore.question3.Tea" primary="true"/>**

**<bean id="expressTea" class="demo.springCore.question3.ExpressTea"/>**

***<!--For question7 added scope=prototype-->***

**<bean name="teaRestaurant" class="demo.springCore.question3.Restaurant" scope="prototype">**

**<property name="hotDrink" ref="tea"/>**

**</bean>**

**Q4. Get both the beans and invoke prepareHotDrink method via hotDrink   
instance variables. Try inner bean with expressTeaRestaurant.**

**Solution**

**Spring-config.xml**

***<!--inner bean for express tea-->***

**<bean name="expressTeaRestaurant" class="demo.springCore.question3.Restaurant">**

**<property name="hotDrink">**

**<bean class="demo.springCore.question3.ExpressTea"/>**

**</property>**

**</bean>**

**Main.java**

**package demo.springCore.question4;**

**import demo.springCore.question3.Restaurant;**

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

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

**public class Main {**

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

**ApplicationContext context=new ClassPathXmlApplicationContext("spring-config.xml");**

**Restaurant teaRestaurant=context.getBean("teaRestaurant",Restaurant.class);**

**teaRestaurant.getHotDrink().prepareHotDrink();**

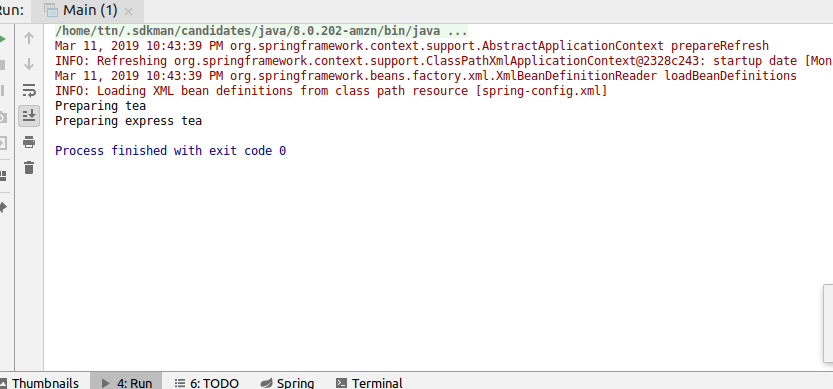
**Restaurant expressTeaRestaurant=context.getBean("expressTeaRestaurant",Restaurant.class);**

**expressTeaRestaurant.getHotDrink().prepareHotDrink();**

**}**

**}**

**Output**



**Q5. Create Class Complex as follows: class complex { List list;**

**Set set;**

**Map map;**

**}**

**Initialize all the instance variables of the complex class using Spring XML. Give bean name as complexBean. Get the bean and display the properties.**

**Solution**

**Complex.java**

**package demo.springCore.question5;**

**import java.util.List;**

**import java.util.Map;**

**import java.util.Set;**

**public class Complex {**

**List list;**

**Set set;**

**Map map;**

**public List getList() {**

**return list;**

**}**

**public void setList(List list) {**

**this.list = list;**

**}**

**public Set getSet() {**

**return set;**

**}**

**public void setSet(Set set) {**

**this.set = set;**

**}**

**public Map getMap() {**

**return map;**

**}**

**public void setMap(Map map) {**

**this.map = map;**

**}**

**@Override**

**public String toString() {**

**return "Complex{" +**

**"list=" + list +**

**", set=" + set +**

**", map=" + map +**

**'}';**

**}**

**}**

**Spring-config.xml**

***<!--bean for question5-->***

**<bean name="complexBean" class="demo.springCore.question5.Complex">**

**<property name="list">**

**<list>**

**<value>1</value>**

**<value>2</value>**

**<value>3</value>**

**</list>**

**</property>**

**<property name="set">**

**<set>**

**<value>Surbhi</value>**

**<value>Sakshi</value>**

**<value>Yukti</value>**

**</set>**

**</property>**

**<property name="map">**

**<map>**

**<entry key="1" value="Surbhi"/>**

**<entry key="2" value="Shivangi"/>**

**<entry key="3" value="Rishabh"/>**

**</map>**

**</property>**

**</bean>**

**Main.java**

**package demo.springCore.question5;**

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

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

**public class Main {**

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

**ApplicationContext applicationContext=new ClassPathXmlApplicationContext("spring-config.xml");**

**Complex complex=applicationContext.getBean("complexBean",Complex.class);**

**System.*out*.println("list: "+complex.getList());**

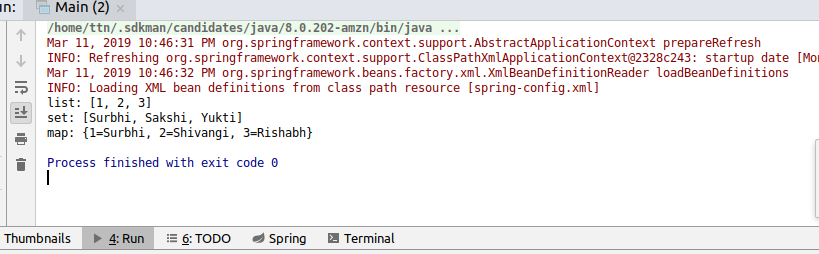
**System.*out*.println("set: "+complex.getSet());**

**System.*out*.println("map: "+complex.getMap());**

**}**

**}**

**Output**



**Q6. Autowire hotDrink in Restaurant with tea bean byName, byType and constructor.**

**Solution**

Primary attribute is used with bean of Tea class so that when we autowire byType and constructor(which internally looks for bean byType), we don’t get NoUniqueBeanDefinitionException. If we don’t use primary attribute, byType will be ambiguous as there are two implementations of HotDrink type.

**Spring-config.xml**

**<bean id="tea" class="demo.springCore.question3.Tea" primary="true"/>**

***<!--bean for question6-->***

**<bean name="restaurantAutowireByName" class="demo.springCore.question3.Restaurant" autowire="byName">**

**</bean>**

**<bean name="hotDrink" class="demo.springCore.question3.Tea"/>**

**<bean name="restaurantAutowireByType" class="demo.springCore.question3.Restaurant" autowire="byType"/>**

**<bean name="restaurantAutowireByConstructor" class="demo.springCore.question3.Restaurant" autowire="constructor"/>**

**Q7. Set the scope of the teaRestaurant bean as proptotype and check the scope type after accessing the bean.**

**Solution**

**Spring-config.xml**

***<!--For question7 added scope=prototype-->***

**<bean name="teaRestaurant" class="demo.springCore.question3.Restaurant" scope="prototype">**

**<property name="hotDrink" ref="tea"/>**

**</bean>**

**Main.java**

**package demo.springCore.question7;**

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

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

**public class Main {**

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

**ApplicationContext applicationContext=new ClassPathXmlApplicationContext("spring-config.xml");**

**System.*out*.println(applicationContext.getBean("teaRestaurant"));**

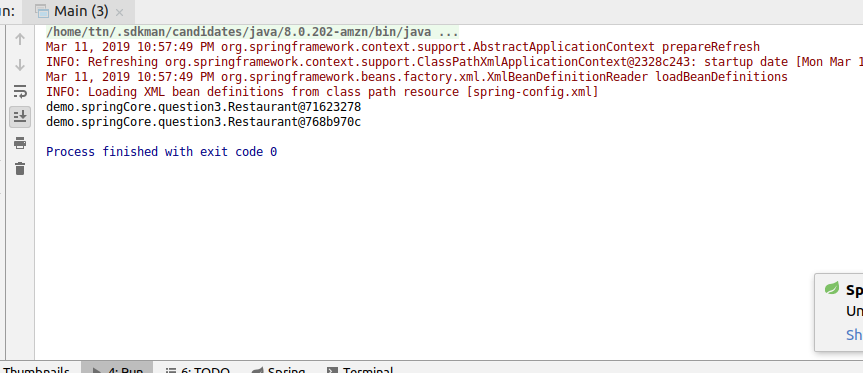
**System.*out*.println(applicationContext.getBean("teaRestaurant"));**

**}**

**}**

**Output**

Both instances result in different hash codes because scope is set to prototype



**Q8. Use @Required annotation for hotDrink setter method in Restaurant class.**

**Solution**

**Restaurant.java**

**package** demo.springCore.question8Andquestion9;

**import** demo.springCore.question3.HotDrink;

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

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

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

**public class** Restaurant {

HotDrink **hotDrink**;

**public** HotDrink getHotDrink() {

**return hotDrink**;

}

*//question-8 --added @Required--uncomment to see effect*

*@Required*

**public void** setHotDrink(HotDrink hotDrink) {

**this**.**hotDrink** = hotDrink;

}

**public** Restaurant(){}

**public** Restaurant(@Qualifier(**"tea"**) HotDrink hotDrink) {

**this**.**hotDrink** = hotDrink;

}

}

By using @Required on setter, setter injection of HotDrink becomes mandatory for all beans of Restaurant type.

**Q9. Use @Autowired annotation to wire Tea with Restaurant class Using setter method, field and constructor.**

**Solution**

@Qualifier -- When there are multiple implementations and we want to access a specific bean, say Tea here, we can use @Qualifier

**Restaurant.java (Using setter method)**

**package demo.springCore.question8Andquestion9;**

**import demo.springCore.question3.HotDrink;**

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

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

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

**public class Restaurant {**

**HotDrink hotDrink;**

**public HotDrink getHotDrink() {**

**return hotDrink;**

**}**

***//question-8 --added @Required--uncomment to see effect***

***// @Required***

***//autowiring using setter--uncomment these two lines to run this***

***@Autowired***

***@Qualifier("tea")***

**public void setHotDrink(HotDrink hotDrink) {**

**this.hotDrink = hotDrink;**

**}**

**public Restaurant(){}**

**public Restaurant(@Qualifier("tea") HotDrink hotDrink) {**

**this.hotDrink = hotDrink;**

**}**

**}**

**Restaurant.java(Using field)**

**package demo.springCore.question8Andquestion9;**

**import demo.springCore.question3.HotDrink;**

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

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

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

**public class Restaurant {**

***//autowiring using field--uncomment to see effect***

***@Autowired***

***@Qualifier("tea")***

**HotDrink hotDrink;**

**public HotDrink getHotDrink() {**

**return hotDrink;**

**}**

***//question-8 --added @Required--uncomment to see effect***

***// @Required***

**public void setHotDrink(HotDrink hotDrink) {**

**this.hotDrink = hotDrink;**

**}**

**public Restaurant(){}**

**public Restaurant( HotDrink hotDrink) {**

**this.hotDrink = hotDrink;**

**}**

**}**

**Restaurant.java(Using Constructor)**

**package demo.springCore.question8Andquestion9;**

**import demo.springCore.question3.HotDrink;**

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

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

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

**public class Restaurant {**

***//autowiring using field--uncomment to see effect***

***// @Autowired***

***// @Qualifier("tea")***

**HotDrink hotDrink;**

**public HotDrink getHotDrink() {**

**return hotDrink;**

**}**

***//question-8 --added @Required--uncomment to see effect***

***// @Required***

***//autowiring using setter--uncomment these two lines to run this***

***// @Autowired***

***// @Qualifier("tea")***

**public void setHotDrink(HotDrink hotDrink) {**

**this.hotDrink = hotDrink;**

**}**

**public Restaurant(){}**

***//using contructor***

**@Autowired**

**public Restaurant(@Qualifier("tea") HotDrink hotDrink) {**

**this.hotDrink = hotDrink;**

**}**

**}**

**Spring-config.xml**

***<!--bean for Question9-->***

**<bean name="restaurant1" class="demo.springCore.question8Andquestion9.Restaurant"/>**

**<context:annotation-config></context:annotation-config>**

**Output (in all 3 cases)**



**Q10. Use @Component, @Controller, @Repository etc annotation to configure Tea and Restaurant in Spring Container.**

**Solution**

**HotDrink.java**

**package demo.springCore.question10;**

**public interface HotDrink {**

**void prepareHotDrink();**

**}**

**Tea.java**

**package demo.springCore.question10;**

**import org.springframework.stereotype.Component;**

**@Component("teaDrink")**

**public class Tea implements HotDrink {**

**@Override**

**public void prepareHotDrink() {**

**System.*out*.println("Preparing tea");**

**}**

**}**

**ExpressTea.java**

**package demo.springCore.question10;**

**import org.springframework.stereotype.Controller;**

**@Controller("expressTeaDrink")**

**public class ExpressTea implements HotDrink {**

**@Override**

**public void prepareHotDrink() {**

**System.*out*.println("Preparing express tea");**

**}**

**}**

**Restaurant.java**

**package demo.springCore.question10;**

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

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

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

**@Repository("restaurant2")**

**public class Restaurant {**

**@Autowired**

**@Qualifier("teaDrink")**

**HotDrink hotDrink;**

**public HotDrink getHotDrink() {**

**return hotDrink;**

**}**

**public void setHotDrink(HotDrink hotDrink) {**

**this.hotDrink = hotDrink;**

**}**

**public Restaurant(){}**

**public Restaurant(HotDrink hotDrink) {**

**this.hotDrink = hotDrink;**

**}**

**}**

**Spring-config.xml**

***<!--For question10-->***

**<context:component-scan base-package="demo.springCore.question10"/>**

**Main.java**

**package demo.springCore.question10;**

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

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

**public class Main {**

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

**ApplicationContext applicationContext=new ClassPathXmlApplicationContext("spring-config.xml");**

**Tea teaDrink=applicationContext.getBean("teaDrink",Tea.class);**

**ExpressTea expressTea=applicationContext.getBean("expressTeaDrink",ExpressTea.class);**

**Restaurant restaurant=applicationContext.getBean("restaurant2",Restaurant.class);**

**teaDrink.prepareHotDrink();**

**expressTea.prepareHotDrink();**

**restaurant.getHotDrink().prepareHotDrink();**

**}**

**}**

**Output**

