Lexa Kiwi Showroom

**Grade settings**: Maximum grade: 100  
**Disable external file upload, paste and drop external content**: Yes  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes **Maximum execution time**: 120 s **Maximum memory used**: 1.50 GiB **Maximum execution file size**: 128 MiB

**Lexa Kiwi Showroom – OnRoad Price Calculation**

[**Click here to download the Code Skeleton**](https://cognizant.tekstac.com/pluginfile.php/19294/mod_vpl/intro/LexaKiwiShowroom.zip)

Lexa-Kiwi is one of the leading automobile showrooms in the world. They are in need of an application to calculate the on-road price of a car based on the car model preferred by the customer. You being their software consultant have been approached by them to develop an application which can be used for managing their business.

**Service 1:** Calculating on-road price of a car

**Automobile class**with the below **private** attributes is provided as a part of code skeleton

|  |  |
| --- | --- |
| brandName | String |
| brandModel | String |
| showroomDetails | Showroom |
| carColour | String |
| carFuelType | String |

**Getter and setter** methods for all the above attributes are provided as a part of code skeleton. Use appropriate spring annotation above the class to denote the class as component.  One argument constructor is provided as the part the code skeleton. The **Showroom** object should be autowired above the constructor in **Automobile** class via annotations.

**Showroom** class with the below **private attributes**is provided as a part of code skeleton

|  |  |
| --- | --- |
| showroomName | String |
| showroomLocation | String |
| carPriceDetails | Map<String,Double> |

**Getter and setter** methods for all the above attributes are provided as a part of code skeleton. Use appropriate spring annotation above the class to denote the class as component.

The attribute should be configured using @Value annotation with the below **values**

|  |  |
| --- | --- |
| showroomName | Lexa Kiwi |
| showroomLocation | Melbourne |

The Map should be configured using **@Value** annotation with the below **key-values**

|  |  |
| --- | --- |
| **Key carName (String)** | **Value registrationFees (Double)** |
| Climber | 429000.00 |
| Dusture | 857000.00 |
| Triber | 1503000.00 |
| Elantra | 910000.00 |
| Sonata | 3750000.00 |

Create a class called **ApplicationConfig** that has the required annotations for **scanning** and **registering** the bean definitions.

**Overview of Service 1:**

Write a method called **public double calculateOnRoadPrice (Automobile carObj, String carName)** in**AutomobileBO class**which will return the on road price; this method should get the car price based on the carName which is available in the map.

With the car price, Registration fees and Insurance Amount should be added to calculate the on road priceand return the same**.**

Assume the Registration fees for any car will be Rs. 38,200 (Thirty Eight Thousand Two Hundred) and the Insurance amount will be 3% of the car price.

**For Example:**

If the **car name** is **Dusture** then,

on road price = carPrice + Registration Fees + Insurance Amount;

**on road price = 857000.00 + 38200 + (857000.00 \*3) / 100** **==> 920910.0**

**Assumption:** Car name should be only the key values which are available in the map.

Create a class called **Driver** with the main method and get the inputs like **brandName**, **brandModel**, **carColour, carFuelType**and**carName** from the user. Get the **AutomobileService** class object by loading **ApplicationConfig** class and invoke themethod**calculateOnRoadPrice (brandName, brandModel**, **carColour, carFuelType, carName)** which is in the **AutomobileService** class to perform the implementation. Display the **on road price** which is returned from **calculateOnRoadPrice (Automobile carObj, String carName)**method in**AutomobileBO class.**

**Technical Specifications:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Method Name** | **Input** | **Output** | **Exception** |
| AutomobileService | double calculateOnRoadPrice | String brandName, String brandModel, String carColour, String carFuelType  String carName | double –  onRoadPrice | InvalidFuelTypeException  This Exception to be caught and thrown back to Main class |
| AutomobileBO | double calculateOnRoadPrice | Automobile carObj,  String carName | double - onRoadPrice |  |
| ApplicationConfig | Contains all the configurations related to Service |  |  |  |

**Business Rules & Validations:**

In **AutomobileService**class included the following private attribute. Use appropriate spring annotation above the class to denote the class as component.

**private AutomobileBO automobileBOObj;**

**Getter and setter** method for the above attribute is provided as a part of code skeleton.  One argument constructor is provided as the part of code skeleton. The **AutomobileBO**object should be **autowired**above the **constructor**via **annotations**.

In this**AutomobileService class**, the method **public double calculateOnRoadPrice (brandName, brandModel, carColour, carFuelType, carName)** accepts brandName, brandModel, carColour, carFuelType and carName as the arguments.

Validate the **carFuelType**,**[ carFuelType should be either Petrol or Diesel, Case-Insensitive]** if the carFuelType is **valid**get the Car object and set the brandName, brandModel, carColour and carFuelType in that object. In case the **carFuelType**is**NOT valid**, a user-defined Exception **InvalidFuelTypeException**should be thrown with message **“Fuel Type is Not Valid”**.

If the validation is done call the method **calculateOnRoadPrice (carObj, carName)** in **AutomobileBO**class and perform the implementation.

**Limitations and Constraints:**

**1.       Automobile**and **Showroom** class should be in **com.spring.model package**.

2.       The price for each car should be declared as a MAP with the given value and should be injected using **@Value** annotation above the attribute.

3.       **ApplicationConfig** class should be in **com.spring.config package**.

4.       **InvalidFuelTypeException**class should be in **com.spring.exception package**.

5.       **AutomobileService**class should be in **com.spring.service package**.

6.       **AutomobileBO** class should be in **com.spring.bo package**.

7.       **Driver**class should be in **com.spring.main package**.

8.       All of the above mentioned java classes to be **configured as component** class using appropriate spring annotation.

9.       **Showroom**should be **injected**into **Automobile**class via **constructor based Injection** using appropriate annotation.

10.   **AutomobileBO**should be injected into **AutomobileService**class via **constructor based Injection** using annotation.

**Sample Input Output 1:**

Enter the brand name:

**lexus**

Enter the brand Model:

**LEX9215**

Enter the preferred car colour:

**Silky Red**

Enter the preferred fuel type:

**petrol  // fuel type is case insensitive**

Enter the car Name:

**Elantra // Car name available in Map which is case sensitive**

Estimated on road price for the preferred car is: 975500.0

**Sample Input Output 2:**

Enter the brand name:

**lexus**

Enter the brand Model:

**LEX9215**

Enter the preferred car colour:

**Silky Red**

Enter the preferred fuel type:

**Gas**

Enter the car Name:

**Climber // Car name available in Map which is case sensitive**

Fuel Type is Not Valid

### **Automatic evaluation**[**[-]**](javascript:void(0);)

**Proposed grade: 100.0 / 100**  
**Result Description**  
[[+]](javascript:void(0);)**Grading and Feedback**

#### LexaKiwiShowroom/src/main/java/com/spring/model/Automobile.java

1 *package* com.spring.model;

2

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

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

5

6 @Component

7 *public* *class* Automobile {

8

9 *private* String brandName;

10 *private* String brandModel;

11 *private* Showroom showroomDetails;

12 *private* String carColour;

13 *private* String carFuelType;

14

15 @Autowired

16 *public* Automobile(Showroom showroomDetails) {

17 *super*();

18 *this*.showroomDetails = showroomDetails;

19 }

20 *public* String getBrandName() {

21 *return* brandName;

22 }

23 *public* *void* setBrandName(String brandName) {

24 *this*.brandName = brandName;

25 }

26 *public* String getBrandModel() {

27 *return* brandModel;

28 }

29 *public* *void* setBrandModel(String brandModel) {

30 *this*.brandModel = brandModel;

31 }

32 *public* Showroom getShowroomDetails() {

33 *return* showroomDetails;

34 }

35 *public* *void* setShowroomDetails(Showroom showroomDetails) {

36 *this*.showroomDetails = showroomDetails;

37 }

38 *public* String getCarColour() {

39 *return* carColour;

40 }

41 *public* *void* setCarColour(String carColour) {

42 *this*.carColour = carColour;

43 }

44 *public* String getCarFuelType() {

45 *return* carFuelType;

46 }

47 *public* *void* setCarFuelType(String carFuelType) {

48 *this*.carFuelType = carFuelType;

49 }

50

51 }

52

#### LexaKiwiShowroom/src/main/java/com/spring/model/Showroom.java

1 *package* com.spring.model;

2

3 *import* java.util.Map;

4

5 *import* org.springframework.beans.factory.annotation.Value;

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

7

8

9 @Component

10 *public* *class* Showroom {

11 @Value("Lexa Kiwi")

12 *private* String showroomName;

13

14 @Value("Melbourne")

15 *private* String showroomLocation;

16

17 @Value("#{${carPriceDetails:{Climber:'429000.00',Dusture:'857000.00',Triber:'1503000.00',Elantra:'910000.00',Sonata:'3750000.00'}}}")

18 *private* Map<String,Double> carPriceDetails;

19

20 *public* String getShowroomName() {

21 *return* showroomName;

22 }

23 *public* *void* setShowroomName(String showroomName) {

24 *this*.showroomName = showroomName;

25 }

26 *public* String getShowroomLocation() {

27 *return* showroomLocation;

28 }

29 *public* *void* setShowroomLocation(String showroomLocation) {

30 *this*.showroomLocation = showroomLocation;

31 }

32 *public* Map<String, Double> getCarPriceDetails() {

33 *return* carPriceDetails;

34 }

35 *public* *void* setCarPriceDetails(Map<String, Double> carPriceDetails) {

36 *this*.carPriceDetails = carPriceDetails;

37 }

38

39 }

40

#### LexaKiwiShowroom/src/main/java/com/spring/service/AutomobileService.java

1 *package* com.spring.service;

2

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

4 *import* org.springframework.context.ConfigurableApplicationContext;

5 *import* org.springframework.context.annotation.AnnotationConfigApplicationContext;

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

7

8 *import* com.spring.bo.AutomobileBO;

9 *import* com.spring.config.ApplicationConfig;

10 *import* com.spring.exception.InvalidFuelTypeException;

11 *import* com.spring.model.Automobile;

12

13 @Component

14 *public* *class* AutomobileService {

15

16 *private* AutomobileBO automobileBOObj;

17

18 @Autowired

19 *public* AutomobileService(AutomobileBO automobileBOObj) {

20 *super*();

21 *this*.automobileBOObj = automobileBOObj;

22 }

23

24 *public* AutomobileBO getAutomobileBOObj() {

25 *return* automobileBOObj;

26 }

27

28 *public* *void* setAutomobileBOObj(AutomobileBO automobileBOObj) {

29 *this*.automobileBOObj = automobileBOObj;

30 }

31

32

33 *public* *double* calculateOnRoadPrice(String brandName,String brandModel,String carColour,String carFuelType,String carName) throws InvalidFuelTypeException {

34 *double* amount=0;

35 // fill the code

36 *if*(carFuelType.equalsIgnoreCase("petrol")||carFuelType.equalsIgnoreCase("diesel"))

37 {

38 ConfigurableApplicationContext context=*new* AnnotationConfigApplicationContext(ApplicationConfig.*class*);

39 //ApplicationConfig config=new ApplicationConfig();

40 Automobile automobile=(Automobile)context.getBean(Automobile.*class*);

41 automobile.setBrandName(brandName);

42 automobile.setBrandModel(brandModel);

43 automobile.setCarColour(carColour);

44 automobile.setCarFuelType(carFuelType);

45 amount=automobileBOObj.calculateOnRoadPrice(automobile, carName);

46 }

47 *else*

48 {

49 *throw* *new* InvalidFuelTypeException("Fuel Type is Not Valid");

50 }

51

52 *return* amount;

53 }

54

55 }

56

#### LexaKiwiShowroom/pom.xml

1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

2 <modelVersion>4.0.0</modelVersion>

3 <groupId>LexaKiwiShowroom</groupId>

4 <artifactId>LexaKiwiShowroom</artifactId>

5 <version>0.0.1-SNAPSHOT</version>

6

7 <dependencies>

8

9 <dependency>

10 <groupId>org.springframework</groupId>

11 <artifactId>spring-context</artifactId>

12 <version>4.3.10.RELEASE</version>

13 </dependency>

14

15 </dependencies>

16

17 </project>

#### LexaKiwiShowroom/src/main/java/com/spring/bo/AutomobileBO.java

1 *package* com.spring.bo;

2

3 *import* java.util.Map;

4

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

6

7 *import* com.spring.model.Automobile;

8

9 @Component

10 *public* *class* AutomobileBO {

11

12 *public* *double* calculateOnRoadPrice(Automobile carObj,String carName) {

13

14 *double* amount=0;

15 // fill the code

16 Map<String,Double> map=carObj.getShowroomDetails().getCarPriceDetails();

17 *if*(map.containsKey(carName))

18 {

19 *double* price=map.get(carName);

20 amount=price+38200+((price\*3)/100);

21 }

22

23 *return* amount;

24 }

25

26 }

27

#### LexaKiwiShowroom/src/main/java/com/spring/config/ApplicationConfig.java

1 *package* com.spring.config;

2

3

4 *import* org.springframework.context.annotation.Bean;

5 *import* org.springframework.context.annotation.ComponentScan;

6 *import* org.springframework.context.annotation.Configuration;

7

8 *import* com.spring.bo.AutomobileBO;

9 *import* com.spring.model.Automobile;

10 *import* com.spring.model.Showroom;

11 *import* com.spring.service.AutomobileService;

12

13 // fill the code

14 @Configuration

15 @ComponentScan

16 *public* *class* ApplicationConfig {

17 @Bean

18 *public* Automobile automobile() {

19 *return* *new* Automobile(showroom());

20 }

21 @Bean

22 *public* Showroom showroom() {

23 *return* *new* Showroom();

24 }

25 @Bean

26 *public* AutomobileBO automobileBo() {

27 *return* *new* AutomobileBO();

28 }

29 @Bean

30 *public* AutomobileService autoMobileService() {

31 *return* *new* AutomobileService(automobileBo());

32 }

33

34

35 }

36

#### LexaKiwiShowroom/src/main/java/com/spring/exception/InvalidFuelTypeException.java

1 *package* com.spring.exception;

2

3 *public* *class* InvalidFuelTypeException *extends* Exception {

4

5 *public* InvalidFuelTypeException(String msg) {

6

7 // fill the code

8 *super*(msg);

9 }

10

11 }

12

#### LexaKiwiShowroom/src/main/java/com/spring/main/Driver.java

1 *package* com.spring.main;

2

3 *import* java.util.Scanner;

4

5 *import* org.springframework.context.ConfigurableApplicationContext;

6 *import* org.springframework.context.annotation.AnnotationConfigApplicationContext;

7

8 *import* com.spring.config.ApplicationConfig;

9 *import* com.spring.exception.InvalidFuelTypeException;

10 *import* com.spring.service.AutomobileService;

11

12 *public* *class* Driver {

13

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

15

16 Scanner in=*new* Scanner(System.in);

17

18 // fill the code

19 System.out.println("Enter the brand name:");

20 String a=in.nextLine();

21 System.out.println("Enter the brand Model:");

22 String b=in.nextLine();

23 System.out.println("Enter the preferred car colour:");

24 String c=in.nextLine();

25 System.out.println("Enter the preferred fuel type:");

26 String d=in.nextLine();

27 System.out.println("Enter the car Name:");

28 String e=in.nextLine();

29 ConfigurableApplicationContext context=*new* AnnotationConfigApplicationContext(ApplicationConfig.*class*);

30 AutomobileService ams=(AutomobileService)context.getBean(AutomobileService.*class*);

31 *double* onRoadPrice=ams.calculateOnRoadPrice(a, b, c, d, e);

32 System.out.println("Estimated on road price for the preferred car is: "+onRoadPrice);

33 }

34

35 }

36

## Grade

Reviewed on Thursday, 22 April 2021, 6:48 PM by Automatic grade  
**Grade** 100 / 100  
**Assessment report**  
[[+]](javascript:void(0);)**Grading and Feedback**