BillController

```
package com.controller;

import java.util.ArrayList;
import java.util.List;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.model.Ticket;
import com.service.BillService;

//use appropriate annotation to configure BillController as Controller
@Controller
public class BillController {
	@Autowired
	private BillService billService;

	@ModelAttribute("ticketList")
	  public List<String> populateTicketType(){
	  List<String> servicList = new ArrayList<String>();
	  servicList.add("Balcony");
	  servicList.add("Floor");
	  servicList.add("KingCircle");
	  servicList.add("QueenCircle");
	  return servicList;

	 }

	@ModelAttribute("addOnList")
	  public List<String> populateRefreshmentOption(){
		List<String> serviceList = new ArrayList<String>();
		serviceList.add("YES");
		serviceList.add("NO");
```

```
            return serviceList;
        }

        @RequestMapping(value="/showPage" , method=RequestMethod.GET)
         public String ShowPage(@ModelAttribute("ticket") Ticket ticket) {
          return "showpage";
         }

        @RequestMapping(value="/billDesk",method=RequestMethod.POST)
        public String calculateTotalCost(@Valid @ModelAttribute("ticket") Ticket ticket,
                        BindingResult result,ModelMap model) {

                if(result.hasErrors()) {
                        return "showpage";
                }
                double totalCost = billService.calculateTotalCost(ticket);
                model.put("totalCost",totalCost);
                return "billdesk";

        }

}


}
```

InternationalizationConifg

```
package com.controller;

import java.util.Locale;

import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ReloadableResourceBundleMessageSource;
import org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;
import org.springframework.web.servlet.LocaleResolver;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;
```

```
@Configuration
public class InternationalizationConfig extends WebMvcConfigurerAdapter {

//Create a SessionLocaleResolver object and set the default locale to English  return the
SessionLocaleResolver object
        @Bean
        public LocaleResolver localeResolver() {
                SessionLocaleResolver localeResolver = new SessionLocaleResolver();
                localeResolver.setDefaultLocale(Locale.US);
                return localeResolver;
        }

        // Create LocaleChangeInterceptor object and set the parameter name as language
        // and return the localeChangeInterceptor
        @Bean
        public LocaleChangeInterceptor localeChangeInterceptor() {
                LocaleChangeInterceptor interceptor = new LocaleChangeInterceptor();
                interceptor.setParamName("language");
                return interceptor;

        }
        @Bean
        public LocalValidatorFactoryBean getValidator() {
           LocalValidatorFactoryBean bean = new LocalValidatorFactoryBean();
           bean.setValidationMessageSource(messageSource());
           return bean;
        }
        @Bean
        public MessageSource messageSource() {
                ReloadableResourceBundleMessageSource messageSource = new
ReloadableResourceBundleMessageSource();
                messageSource.setBasename("classpath:messages");
                messageSource.setDefaultEncoding("UTF-8");
                return messageSource;
        }


  //register the LocaleChangeInterceptor
  @Override
  public void addInterceptors(InterceptorRegistry registry) {

          registry.addInterceptor(localeChangeInterceptor());
```

```
  }
}
```

MovieTicketBookingApplication

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan({"com.controller","com.model","com.service"})
public class MovieTicketBookingApplication {

	public static void main(String[] args) {
		SpringApplication.run(MovieTicketBookingApplication.class, args);
	}

}
```

Ticket

```
package com.model;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Positive;

import org.springframework.stereotype.Component;

//pojo class with required attributes,getters and setters
@Component
public class Ticket {

	@NotBlank(message="{error.customerName.blank}")
	private String customerName;
	@NotBlank(message="{error.contactNumber.blank}")
	private String contactNumber;
	@Positive(message="{error.noOfTickets.negative}")
```

```
	private int noOfTickets;
	private String ticketType;
	private String refreshment;

	public String getCustomerName() {
		return customerName;
	}
	public void setCustomerName(String customerName) {
		this.customerName = customerName;
	}
	public String getContactNumber() {
		return contactNumber;
	}
	public void setContactNumber(String contactNumber) {
		this.contactNumber = contactNumber;
	}
	public int getNoOfTickets() {
		return noOfTickets;
	}
	public void setNoOfTickets(int noOfTickets) {
		this.noOfTickets = noOfTickets;
	}
	public String getTicketType() {
		return ticketType;
	}
	public void setTicketType(String ticketType) {
		this.ticketType = ticketType;
	}
	public String getRefreshment() {
		return refreshment;
	}
	public void setRefreshment(String refreshment) {
		this.refreshment = refreshment;
	}


}
```

BillService

```
package com.service;
```

```
import org.springframework.stereotype.Service;

import com.model.Ticket;

//use appropriate annotation to configure BillService as a Service
@Service
public class BillService {

	//calculate the totalCost and return the cost
	public double calculateTotalCost(Ticket ticket) {
		double cost=0.0;
		//fill the code
		double ref=0.0;
		double totalcost=0.0;
		double perProductPrice = 0;
		if (ticket.getTicketType().equalsIgnoreCase("Balcony")) {
			perProductPrice = 280.0;
		}
		else if (ticket.getTicketType().equalsIgnoreCase("Floor")) {
			perProductPrice = 100.0;
		}
		else if (ticket.getTicketType().equalsIgnoreCase("KingCircle")) {
			perProductPrice = 180.0;
		}
		else if (ticket.getTicketType().equalsIgnoreCase("QueenCircle")) {
			perProductPrice =200.0;
		}

		cost = ticket.getNoOfTickets() * perProductPrice ;

		if(ticket.getRefreshment().equalsIgnoreCase("YES")) {
			 ref=ticket.getNoOfTickets()*100.0;
		}
		totalcost=cost+ref;

		if(ticket.getNoOfTickets()>=10) {
			totalcost=totalcost-(totalcost*5.0)/100;

		}
			return totalcost;
	}
}
```

---

showPage

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<style>
.center{
margin-left:auto;
margin-right:auto;

}
</style>

</head>
<body style="background-color:lavender">
<h1><center> Welcome to Online Ticket Booking </center></h1>
<!--  Do not change the below line -->
<a href="/showPage?language=en">English</a>|<a
href="/showPage?language=de">German</a>|<a
href="/showPage?language=fr">French</a></align>
<!--  Design the page as per the requirements-->


<form:form modelAttribute="ticket" method="post" action="/billDesk">
        <table  class="center">
                <tr>
                        <td id="id1"><spring:message code="label.customerName" /></td>
                        <td><form:input path="customerName" id="customerName"
                                        name="customerName" /></td>
                        <td><form:errors path="customerName"></form:errors></td>
                </tr>
                <tr>
                        <td id="id2"><spring:message code="label.contactNumber" /></td>
                        <td><form:input path="contactNumber" id="contactNumber"
name="contactNumber" /></td>
```

```
                <td><form:errors path="contactNumber"></form:errors></td>
            </tr>
            <tr>
                <td id="id3"><spring:message code="label.noOfTickets" /></td>
                <td><form:input path="noOfTickets"
                                name="noOfTicket" id="noOfTickets" /></td>
                <td><form:errors path="noOfTickets"></form:errors></td>
            </tr>
            <tr>
                <td id="id4"><spring:message code="label.ticketType" /></td>
                <td><form:select path="ticketType" id="ticketType" name="ticketType"
                                items="${ticketList}" /></td>
            </tr>
            <tr>
                <td id="id5"><spring:message code="label.refreshment" /></td>
                <td><form:select path="refreshment" id="refreshment"
name="refreshment" items="${addOnList}" /></td>
            </tr>
            <tr>
                <td><input type="submit" id="submit" name="submit"
                            value="BookTicket" /></td>
                <td><input type="reset" id="reset" name="reset"
                            value="Cancel" /></td>
            </tr>
        </table>
        </form:form>
</body>
</html>
```

---

billDesk

```
<%@page isELIgnored="false" %>
<html>
<body bgcolor="lavender">

<h2>Total Ticket booking cost is Rs.${totalCost} </h2>


</body>
</html>
```

Buffet Restaurant V1

Grade settings: Maximum grade: 100
Based on: Buffet Restaurant V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 2 GiB **Maximum execution file size**: 1.50 GiB **Maximum number of processes**: 100000

Buffet Restaurant

Result Description

Submitted by sathya .

Automatic evaluation[-]

Proposed grade: 98.75 / 100
Result Description
[-]SOURCE CODE ANALYZER REPORT

```
Error Msg: A method should have only one exit point, and that should be the last statement
in the method
Variable Name:
Class Name: BookController
```

Assessment Completed Successfully
[-]Grading and Feedback

```
Feature Test  - 30.00 / 30.00(Success)
         * check whether the controller annotation is present above the Controller - 6.00
/ 6.00
         * check whether the ModelAttribute annotation is present above the Method - 6.00
/ 6.00
         * check whether the Service is autowired inside the Controller - 6.00 / 6.00
         * check whether Service annotation is present above the Service - 6.00 / 6.00
         * check whether the ModelAttribute annotation has the appropriate value above the
Method - 6.00 / 6.00

Functional testing - 20.00 / 20.00(Success)
         * check whether the Total cost is calculated correctly for the Hall Type AC -
10.00 / 10.00
         * check whether the Total cost is calculated correctly for the Hall Type NonAC -
10.00 / 10.00

Spring Testing in the Controller - 20.00 / 20.00(Success)
         * check whether the request is mapped for showPage and redirected to showpage -
5.00 / 5.00
```

```
        * check whether layering is followed that is whether total cost is calculated by
invoking the service method inside the Controller - 10.00 / 10.00
        * check whether the request is mapped for calculateTotalCost and redirected to
result page - 5.00 / 5.00

UI Testing - 25.00 / 25.00(Success)
        * check whether the showpage jsp is rendered for AC Hall - 1.00 / 1.00
        * check whether the showpage jsp is rendered for Non AC Hall - 1.00 / 1.00
        * check whether the showpage jsp is rendered with proper constraints for
customerName - 0.50 / 0.50
        * check whether the showpage jsp is rendered with proper constraints for
contactNumber - 0.50 / 0.50
        * check whether the showpage jsp is rendered with proper constraints for
adultVegCount - 0.50 / 0.50
        * check whether the showpage jsp is rendered with proper constraints for
kidsVegCount - 0.50 / 0.50
        * check whether the showpage jsp is rendered with proper constraints for
kidsNonVegCount - 0.50 / 0.50
        * check whether the showpage jsp is rendered with proper constraints for
adultNonVegCount - 0.50 / 0.50
        * check whether UI from showpage  to result page is redirected after calculating
the total cost for AC hall - 1.00 / 1.00
        * check whether UI from showpage  to result page is redirected after calculating
the course fees for JEE for FastTrack - 1.00 / 1.00
        * check whether label names from the property file is correctly displayed for
English - 2.00 / 2.00
        * check whether label names from the property file is correctly displayed for
German - 2.00 / 2.00
        * check whether label names from the property file is correctly displayed for
French - 2.00 / 2.00
        * check whether error message for Customer Name in blank from the property file
is correctly displayed for German - 2.00 / 2.00
        * check whether error message for Contact Number in blank from the property file
is correctly displayed for German - 2.00 / 2.00
        * check whether error message for Adult Count For Veg in negative from the
property file is correctly displayed for French - 2.00 / 2.00
        * check whether error message for Kids count for Veg in negative from the
property file is correctly displayed for English - 2.00 / 2.00
        * check whether error message for Adult Count For NonVeg in negative from the
property file is correctly displayed for French - 2.00 / 2.00
        * check whether error message for Kids Count For NonVeg in negative from the
property file is correctly displayed for English - 2.00 / 2.00

Comments and best practices/standards - 3.75 / 5.0(Partial)
```

BuffetRestaurant/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.1.8.RELEASE</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>
        <groupId>com</groupId>
        <artifactId>EBBill</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>EBBill</name>
        <description>Demo project for Spring Boot</description>
```

```
16
17          <properties>
18                      <java.version>1.8</java.version>
19          </properties>
20
21          <dependencies>
22                      <dependency>
23                                  <groupId>org.springframework.boot</groupId>
24                                  <artifactId>spring-boot-starter</artifactId>
25                      </dependency>
26
27
28          <dependency>
29                                  <groupId>org.springframework.boot</groupId>
30                                  <artifactId>spring-boot-starter-web</artifactId>
31                      </dependency>
32
33                       <dependency>
34     <groupId>org.apache.tomcat.embed</groupId>
35     <artifactId>tomcat-embed-jasper</artifactId>
36     <scope>provided</scope>
37   </dependency>
38   <dependency>
39   <groupId>javax.servlet</groupId>
40   <artifactId>servlet-api</artifactId>
41   <version>2.5</version>
42   <scope>provided</scope>
43 </dependency>
44 <dependency>
45   <groupId>javax.servlet.jsp</groupId>
46   <artifactId>jsp-api</artifactId>
47   <version>2.1</version>
48   <scope>provided</scope>
49 </dependency>
50
51 <dependency>
52   <groupId>taglibs</groupId>
53   <artifactId>standard</artifactId>
54   <version>1.1.2</version>
55 </dependency>
56 <dependency>
57   <groupId>javax.servlet</groupId>
58   <artifactId>jstl</artifactId>
59   <version>1.2</version>
60 </dependency>
61                      <dependency>
62                                  <groupId>org.springframework.boot</groupId>
63                                  <artifactId>spring-boot-starter-test</artifactId>
64                                  <scope>test</scope>
65                                  <exclusions>
66                                              <exclusion>
67                                                          <groupId>org.junit.vintage</groupId>
68                                                          <artifactId>junit-vintage-engine</artifactId>
69                                              </exclusion>
70                                  </exclusions>
71                      </dependency>
72          </dependencies>
73
74          <build>
75                      <plugins>
76                                  <plugin>
77                                              <groupId>org.springframework.boot</groupId>
78                                              <artifactId>spring-boot-maven-plugin</artifactId>
79                                  </plugin>
80                      </plugins>
81          </build>
82
```

```
</project>


```

BuffetRestaurant/src/main/java/com/controller/BookController.java

```
package com.controller;
import java.util.HashMap;
import java.util.Map;
import javax.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.model.BookRestaurant;
import com.service.BookService;

@Controller
public class BookController {

	@Autowired
	private BookService bookService;

	@RequestMapping(value = "/showPage", method = RequestMethod.GET)
	public String showPage(@ModelAttribute("book") BookRestaurant book) {

		return "showpage";
	}

	@ModelAttribute("hallList")
	public  Map<String, String> populateHallType(){

		Map<String, String> serviceMap = new HashMap<String, String>();
		serviceMap.put("AC", "AC");
		serviceMap.put("NonAC", "NonAC");
		return serviceMap;

	}

	@RequestMapping(value = "/billDesk", method = RequestMethod.POST)
	public String calculateTotalCost(@Valid@ModelAttribute("book") BookRestaurant book,
				BindingResult result,ModelMap model) {

		if(result.hasErrors())
		{
			return "showpage";
		}
		else
		{
		double cost=bookService.calculateTotalCost(book);
		model.addAttribute("cost", cost);
		return "billdesk";
		}
	}

}

```

BuffetRestaurant/src/main/java/com/controller/InternationalizationConfig.java

```
package com.controller;

```

```
3 import java.util.Locale;
4
5 import org.springframework.context.annotation.Bean;
6 import org.springframework.context.annotation.Configuration;
7 import org.springframework.context.support.ResourceBundleMessageSource;
8 import org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;
9 import org.springframework.web.servlet.LocaleResolver;
10 import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
11 import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
12 import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
13 import org.springframework.web.servlet.i18n.SessionLocaleResolver;
14
15 @Configuration
16 public class InternationalizationConfig extends WebMvcConfigurerAdapter {
17
18 //Create a SessionLocaleResolver object and set the default locale to English  return the
SessionLocaleResolver object
19         @Bean
20   public LocaleResolver localeResolver() {
21
22                         SessionLocaleResolver slr=new SessionLocaleResolver();
23                         slr.setDefaultLocale(Locale.US);
24                         return slr;
25
26
27   }
28
29   //Create LocaleChangeInterceptor object and set the parameter name as language and return the
localeChangeInterceptor
30   @Bean
31   public LocaleChangeInterceptor localeChangeInterceptor() {
32           LocaleChangeInterceptor lci=new LocaleChangeInterceptor();
33                         lci.setParamName("language");
34                         return lci;
35
36
37   }
38
39
40   //register the LocaleChangeInterceptor
41   @Override
42   public void addInterceptors(InterceptorRegistry registry) {
43           registry.addInterceptor(localeChangeInterceptor());
44
45   }
46
47   @Bean
48         public ResourceBundleMessageSource messageSource()
49         {
50                         ResourceBundleMessageSource rms=new ResourceBundleMessageSource();
51                         rms.setBasenames("messages","messages_fr","messages_de");
52                         return rms;
53
54
55         }
56
57         @Bean
58   public LocalValidatorFactoryBean getValidator() {
59      LocalValidatorFactoryBean bean = new LocalValidatorFactoryBean();
60      bean.setValidationMessageSource(messageSource());
61      return bean;
62
63   }
64 }
65
```

BuffetRestaurant/src/main/java/com/example/demo/BuffetRestaurantApplication.java

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.context.annotation.ComponentScan;
6
7 @SpringBootApplication
8 @ComponentScan({"com.controller","com.model","com.service"})
9 public class BuffetRestaurantApplication {
10
11         public static void main(String[] args) {
12                 SpringApplication.run(BuffetRestaurantApplication.class, args);
13         }
14
15 }
16
```

BuffetRestaurant/src/main/java/com/model/BookRestaurant.java

```
1 package com.model;
2
3 import javax.validation.constraints.NotEmpty;
4 import javax.validation.constraints.PositiveOrZero;
5
6 import org.springframework.stereotype.Component;
7
8 @Component
9 public class BookRestaurant {
10
11         @NotEmpty(message = "{error.customerName.blank}")
12         private String customerName;
13         @NotEmpty(message = "{error.contactNumber.blank}")
14         private String contactNumber;
15         private String hallType;
16         @PositiveOrZero(message="{error.adultCount.forVeg.positive}")
17         private int adultVegCount;
18         @PositiveOrZero(message="{error.kidsCount.forVeg.positive}")
19         private int kidsVegCount;
20         @PositiveOrZero(message="{error.adultCount.forNonVeg.positive}")
21         private int adultNonVegCount;
22         @PositiveOrZero(message="{error.kidsCount.forNonVeg.positive}")
23         private int kidsNonVegCount;
24
25         public String getCustomerName() {
26                 return customerName;
27         }
28         public void setCustomerName(String customerName) {
29                 this.customerName = customerName;
30         }
31         public String getContactNumber() {
32                 return contactNumber;
33         }
34         public void setContactNumber(String contactNumber) {
35                 this.contactNumber = contactNumber;
36         }
37         public String getHallType() {
38                 return hallType;
39         }
40         public void setHallType(String hallType) {
41                 this.hallType = hallType;
42         }
43         public int getAdultVegCount() {
44                 return adultVegCount;
```

```
	}
	public void setAdultVegCount(int adultVegCount) {
		this.adultVegCount = adultVegCount;
	}
	public int getKidsVegCount() {
		return kidsVegCount;
	}
	public void setKidsVegCount(int kidsVegCount) {
		this.kidsVegCount = kidsVegCount;
	}
	public int getAdultNonVegCount() {
		return adultNonVegCount;
	}
	public void setAdultNonVegCount(int adultNonVegCount) {
		this.adultNonVegCount = adultNonVegCount;
	}
	public int getKidsNonVegCount() {
		return kidsNonVegCount;
	}
	public void setKidsNonVegCount(int kidsNonVegCount) {
		this.kidsNonVegCount = kidsNonVegCount;
	}




}
```

BuffetRestaurant/src/main/java/com/service/BookService.java

```
package com.service;

import org.springframework.stereotype.Service;

import com.model.BookRestaurant;

@Service
public class BookService {

	public double calculateTotalCost(BookRestaurant book) {

		double cost=0.0;
		if(book.getHallType().equalsIgnoreCase("AC"))
		{
			cost = ((book.getAdultVegCount() * 599.0)+(book.getKidsVegCount() * 249.0)+(book.getAdultNonVegCount() * 699.0)+(book.getKidsNonVegCount() * 349.0))+500.0;
		}
		else
		{
			cost = ((book.getAdultVegCount() * 599.0)+(book.getKidsVegCount() * 249.0)+(book.getAdultNonVegCount() * 699.0)+(book.getKidsNonVegCount() * 349.0));
		}
		return cost;

	}

}
```

BuffetRestaurant/src/main/resources/application.properties

```
#Do not edit the key server.port.Port number can be changed
#include other needed configurations
server.port=9095
```

```
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**


```

BuffetRestaurant/src/main/resources/messages.properties

```
label.customerName=Customer Name in English
label.contactNumber=Contact Number in English
label.hallType=Hall Type in English
label.adultVegCount=Adult Count For Veg in English
label.kidsVegCount=Kids Count For Veg in English
label.adultNonVegCount=Adult Count For NonVeg in English
label.kidsNonVegCount=Kids Count For NonVeg in English

error.customerName.blank=Customer Name cannot be empty in English
error.contactNumber.blank=Contact Number cannot be empty in English
error.adultCount.forVeg.positive=Adult count for Veg should not be a negative number in English
error.kidsCount.forVeg.positive=Kids count for Veg should not be a negative number in English
error.adultCount.forNonVeg.positive=Adult count for NonVeg should not be a negative number in English
error.kidsCount.forNonVeg.positive=Kids count for NonVeg should not be a negative number in English

```

BuffetRestaurant/src/main/resources/messages_de.properties

```
label.customerName=Customer Name in German
label.contactNumber=Contact Number in German
label.hallType=Hall Type in German
label.adultVegCount=Adult Count For Veg in German
label.kidsVegCount=Kids Count For Veg in German
label.adultNonVegCount=Adult Count For NonVeg in German
label.kidsNonVegCount=Kids Count For NonVeg in German

error.customerName.blank=Customer Name cannot be empty in German
error.contactNumber.blank=Contact Number cannot be empty in German
error.adultCount.forVeg.positive=Adult count for Veg should not be a negative number in German
error.kidsCount.forVeg.positive=Kids count for Veg should not be a negative number in German
error.adultCount.forNonVeg.positive=Adult count for NonVeg should not be a negative number in German
error.kidsCount.forNonVeg.positive=Kids count for NonVeg should not be a negative number in German

```

BuffetRestaurant/src/main/resources/messages_fr.properties

```
label.customerName=Customer Name in French
label.contactNumber=Contact Number in French
label.hallType=Hall Type in French
label.adultVegCount=Adult Count For Veg in French
label.kidsVegCount=Kids Count For Veg in French
label.adultNonVegCount=Adult Count For NonVeg in French
label.kidsNonVegCount=Kids Count For NonVeg in French

error.customerName.blank=Customer Name cannot be empty in French
error.contactNumber.blank=Contact Number cannot be empty in French
error.adultCount.forVeg.positive=Adult count for Veg should not be a negative number in French
error.kidsCount.forVeg.positive=Kids count for Veg should not be a negative number in French
error.adultCount.forNonVeg.positive=Adult count for NonVeg should not be a negative number in French
error.kidsCount.forNonVeg.positive=Kids count for NonVeg should not be a negative number in French

```

BuffetRestaurant/src/main/webapp/WEB-INF/views/billdesk.jsp

```
<%@page isELIgnored="false" %>
<html>
<body bgcolor="lavender">
<h2>Thanks for Booking!!! Total Amount to be paid is Rs.${cost}</h2>
</body>
</html>
```

BuffetRestaurant/src/main/webapp/WEB-INF/views/showpage.jsp

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1><center> Delicious Buffet Restaurant </center></h1>
<a href="/showPage?language=en">English</a>|<a href="/showPage?language=de">German</a>|<a href="/showPage?language=fr">French</a>

<form:form method="post" action="billDesk" modelAttribute="book">

<table style="margin: 0px auto; margin-left: auto; margin-right:auto">

        <tr>
        <td><spring:message code="label.customerName"/></td>
        <td><form:input path="customerName" id="customerName"/></td>
        <td><form:errors path="customerName"/></td>
        </tr>
                        <tr>
                        <td><spring:message code="label.contactNumber"/></td>
        <td><form:input path="contactNumber" id="contactNumber"/></td>
        <td><form:errors path="contactNumber"/></td>
                        </tr>
                        <tr>
                        <td><spring:message code="label.hallType"/></td>
                        <td><form:select path="hallType" id="hallType">
                            <form:options items="${hallList}"/></form:select>
                        </td>
                        </tr>
                       <tr>
                       <td><spring:message code="label.adultVegCount"/></td>
                       <td><form:input path="adultVegCount" id="adultVegCount"/></td>
                       <td><form:errors path="adultVegCount"/></td>
                       </tr>
                       <tr>
                       <td><spring:message code="label.kidsVegCount"/></td>
                       <td><form:input path="kidsVegCount" id="kidsVegCount"/></td>
                       <td><form:errors path="kidsVegCount"/></td>
                       </tr>
                       <tr>
                       <td><spring:message code="label.adultNonVegCount"/></td>
                       <td><form:input path="adultNonVegCount" id="adultNonVegCount"/></td>
                       <td><form:errors path="adultNonVegCount"/></td>
                       </tr>
                       <tr>
                       <td><spring:message code="label.kidsNonVegCount"/></td>
                       <td><form:input path="kidsNonVegCount" id="kidsNonVegCount"/></td>
                       <td><form:errors path="kidsNonVegCount"/></td>
                       </tr>
                       <tr>
                       <td><input type="submit" value="CalculateCost" id="submit" /></td>
                       <td><input type="reset" value="Cancel"/></td>
                       </tr>

                       </table>
</form:form>


```

```
</body>
</html>

```

Grade

Reviewed on Saturday, 26 February 2022, 2:22 PM by Automatic grade
Grade 98.75 / 100
Assessment report
[-]SOURCE CODE ANALYZER REPORT

```
Error Msg: A method should have only one exit point, and that should be the last statement
in the method
Variable Name:
Class Name: BookController
```

Assessment Completed Successfully
[-]Grading and Feedback

```
Feature Test  - 30.00 / 30.00(Success)
         * check whether the controller annotation is present above the Controller - 6.00
/ 6.00
         * check whether the ModelAttribute annotation is present above the Method - 6.00
/ 6.00
         * check whether the Service is autowired inside the Controller - 6.00 / 6.00
         * check whether Service annotation is present above the Service - 6.00 / 6.00
         * check whether the ModelAttribute annotation has the appropriate value above the
Method - 6.00 / 6.00

Functional testing - 20.00 / 20.00(Success)
         * check whether the Total cost is calculated correctly for the Hall Type AC -
10.00 / 10.00
         * check whether the Total cost is calculated correctly for the Hall Type NonAC -
10.00 / 10.00

Spring Testing in the Controller - 20.00 / 20.00(Success)
         * check whether the request is mapped for showPage and redirected to showpage -
5.00 / 5.00
         * check whether layering is followed that is whether total cost is calculated by
invoking the service method inside the Controller - 10.00 / 10.00
         * check whether the request is mapped for calculateTotalCost and redirected to
result page - 5.00 / 5.00

UI Testing - 25.00 / 25.00(Success)
         * check whether the showpage jsp is rendered for AC Hall - 1.00 / 1.00
         * check whether the showpage jsp is rendered for Non AC Hall - 1.00 / 1.00
         * check whether the showpage jsp is rendered with proper constraints for
customerName - 0.50 / 0.50
         * check whether the showpage jsp is rendered with proper constraints for
contactNumber - 0.50 / 0.50
         * check whether the showpage jsp is rendered with proper constraints for
adultVegCount - 0.50 / 0.50
         * check whether the showpage jsp is rendered with proper constraints for
kidsVegCount - 0.50 / 0.50
         * check whether the showpage jsp is rendered with proper constraints for
kidsNonVegCount - 0.50 / 0.50
         * check whether the showpage jsp is rendered with proper constraints for
adultNonVegCount - 0.50 / 0.50
         * check whether UI from showpage  to result page is redirected after calculating
the total cost for AC hall - 1.00 / 1.00
         * check whether UI from showpage  to result page is redirected after calculating
the course fees for JEE for FastTrack - 1.00 / 1.00
         * check whether label names from the property file is correctly displayed for
English - 2.00 / 2.00
         * check whether label names from the property file is correctly displayed for
German - 2.00 / 2.00
```

** check whether label names from the property file is correctly displayed for French - 2.00 / 2.00*
** check whether error message for Customer Name in blank from the property file is correctly displayed for German - 2.00 / 2.00*
** check whether error message for Contact Number in blank from the property file is correctly displayed for German - 2.00 / 2.00*
** check whether error message for Adult Count For Veg in negative from the property file is correctly displayed for French - 2.00 / 2.00*
** check whether error message for Kids count for Veg in negative from the property file is correctly displayed for English - 2.00 / 2.00*
** check whether error message for Adult Count For NonVeg in negative from the property file is correctly displayed for French - 2.00 / 2.00*
** check whether error message for Kids Count For NonVeg in negative from the property file is correctly displayed for English - 2.00 / 2.00*

Comments and best practices/standards - 3.75 / 5.0(Partial)

LOAN ELIGIBILTY

```
------------- loan eligibility .jsp
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" isELIgnored="false"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Loan Bazaar : Loan Eligibility Check</title>
</head>
<body style="background-color: rgb(200, 250, 300)">
<!-- ADD YOUR CODE HERE -->
<center>
<h3 style="color: blue">Loan Bazaar - Find the Best Loan For
You</h3>
<form:form modelAttribute="customerDetails"
action="eligibilityCheck"
method="post" id="loanEligibilityForm">
<table>
<tr>
<td>Name</td>
<td><form:input type="text" path="name" id="name"
/></td>
<td><form:errors path="name"
cssStyle="color:red"></form:errors></td>
</tr>
<tr>
<td>Gender</td>
<td><form:radiobuttons path="gender"
items="${genderList}" /></td>
<td><form:errors path="gender"
cssStyle="color:red"></form:errors>
</td>
</tr>
<tr>
<td>Email</td>
<td><form:input type="text" path="email"
id="email" /></td>
<td><form:errors path="email"
cssStyle="color:red" ></form:errors></td>
</tr>
<tr>
<td>Customer City</td>
<td><form:select path="customerCity"
items="${cityList}"
id="customerCity"/></td>
<td><form:errors path="customerCity"
cssStyle="color:red"></form:errors></td>
</tr>
<tr>
<td>Employment Type</td>
<td><form:select path="employmentType"
items="${employeeList}"
id="employmentType" /></td>
<td><form:errors path="employmentType"
cssStyle="color:red"></form:errors></td>
</tr>
```

```
<tr>
<td>Monthly Income in INR</td>
<td><form:input type="text" path="monthlyIncome"
id="monthlyIncome" /></td>
<td><form:errors path="monthlyIncome"
cssStyle="color:red"></form:errors></td>
</tr>
<tr>
<td>Desired Loan Amount in INR</td>
<td><form:input type="text"
path="desiredLoanAmount"
id="desiredLoanAmount" /></td>
<td><form:errors path="desiredLoanAmount"
cssStyle="color:red"></form:errors></td>
</tr>
</table>
<button type="submit" id="submit">Submit</button>
</form:form>
</center>
</body>
</html>
------------------ Results.jsp
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" isELIgnored="false"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Loan Bazaar : Loan Eligibility Check Results</title>
</head>
<body style="background-color: rgb(200, 250, 300)">
<!-- ADD YOUR CODE HERE -->
<h3>${msg}</h3>
<form:form id="results">
<c:if test="${not empty data}">
<table id="resultsTable" border="1">
<tr>
<td>Bank Name</td>
<td>Loan Product Name</td>
<td>Max Eligible Loan Amount in Rupees</td>
<td>Tenure</td>
<td>Interest</td>
<td>EMI in Rupees</td>
</tr>
<c:forEach items="${data}" var="a">
<tr>
<td>${a.bankName}</td>
<td>${a.loanProductName}</td>
<td>${a.maxLoanAmount}</td>
<td>${a.tenure}</td>
<td>${a.interest}</td>
<td>${a.monthlyInstallment}</td>
</tr>
</c:forEach>
</table>
</c:if>
</form:form>
</html>
-------------------- Error.jsp
```

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" isELIgnored="false"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Loan Bazaar : Loan Eligibility Errors</title>
</head>
<body style="background-color: rgb(200, 250, 300)">
<!-- ADD YOUR CODE HERE -->
<form:form id="error">
<h3>Unable to retrieve loan information. Below are the error details:</h3>
<h3>Response Code: ${code }</h3>
<h3>Error Message: ${message }</h3>
<h3>Error Occurred on: ${curtime }</h3>
</form:form>
</body>
</html>
------------------- Loan EligibiltyController.java
/*******************************************************************************
**************************
* This class LoanEligibilityController is control the views and model objects
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION
CLAUSES, RETURN TYPES
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
* CHANGE THE RETURN TYPE FROM NULL OF THE METHODS ONCE YOU BUILT THE LOGIC
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED
BE,
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
EXCEPTION CLAUSE
* ADD REQUEST MAPPING URI AND RETURN TYPE AS PER DESIGN DOCUMENT
*
*******************************************************************************
**************************/
package com.cts.loanbazaar.loaneligibility.controller;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.cts.loanbazaar.loaneligibility.exception.ApplicationException;
import com.cts.loanbazaar.loaneligibility.model.CustomerDetails;
import com.cts.loanbazaar.loaneligibility.model.LoanProduct;
import com.cts.loanbazaar.loaneligibility.service.LoanEligibilityService;
/**
* Loan Eligibility Controller
```

```
*
*/
@Controller
public class LoanEligibilityController {
@Autowired
LoanEligibilityService loanEligibilityService;
/**
* @param model
* @return String
*/
@RequestMapping(value = "/home", method = RequestMethod.GET)
public String showHomePage(Model model) {
CustomerDetails customerDetails = new CustomerDetails();
model.addAttribute("customerDetails", customerDetails);
return "loanEligibility";
}
/**
* @param model
* @param request
* @param response
* @param customerDetails
* @param result
* @return String
* @throws ApplicationException
*/
@RequestMapping(value = "/eligibilityCheck", method = RequestMethod.POST)
public String getLoanProducts(Model model, HttpServletRequest request,
HttpServletResponse response,
@Valid CustomerDetails customerDetails, BindingResult result) throws
ApplicationException {
if (!result.hasErrors()) {
List<LoanProduct> data = new ArrayList<LoanProduct>();
data = loanEligibilityService.checkEligibleLoanProducts(customerDetails);
if (data.isEmpty()) {
model.addAttribute("msg", "Sorry, no loan products matching your
profile.");
} else {
model.addAttribute("data", data);
model.addAttribute("msg", "Congratulations. You are Eligible for the
below Loan Products:");
}
}
return (result.hasErrors()?"loanEligibility":"results");
}
/**
* @return List<String>
*/
@ModelAttribute("cityList")
public List<String> getCities() {
List<String> cityList = new ArrayList<String>();
cityList.add("");
cityList.add("Chennai");
cityList.add("Mumbai");
cityList.add("Bangalore");
cityList.add("Delhi");
// cityList.add("Pune");
cityList.add("Kolkatta");
```

```
// cityList.add("Delhi");
// cityList.add("Bangalore");
return cityList;
}
/**
* @return List<String>
*/
@ModelAttribute("employeeList")
public List<String> getEmploymentTypes() {
List<String> employeeList = new ArrayList<String>();
employeeList.add("");
employeeList.add("Salaried");
employeeList.add("Self-Employed");
employeeList.add("Contractual Employment");
employeeList.add("Student");
employeeList.add("Pensioner");
return employeeList;
}
/**
* @return List<String>
*/
@ModelAttribute("genderList")
public List<String> getGenderOptions() {
List<String> genderList = new ArrayList<String>();
genderList.add("Male");
genderList.add("Female");
return genderList;
}
}
```

----------customer details.java

```
/*****************************************************************************
*************************
* This class CustomerDetails is the value object that must be binded to loanEligibility view
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION
CLAUSES, RETURN TYPES
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
* CHANGE THE RETURN TYPE FROM NULL OF THE METHODS ONCE YOU BUILT THE LOGIC
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED
BE,
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
EXCEPTION CLAUSE
*
*****************************************************************************
*************************/
package com.cts.loanbazaar.loaneligibility.model;
import javax.validation.constraints.Size;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
public class CustomerDetails {
/// DO NOT CHANGE THE VARIABLE NAMES OR DATA TYPES OR ACCESS SPECIFIERS
@NotNull(message = "must not be null")
@Size(min = 4, max = 30, message = "size must be between 4 and 30")
public String name;
@NotNull(message = "must not be null")
```

```
public String gender;
@NotEmpty(message = "must not be empty")
@Email(message = "must be a well-formed email address")
public String email;
@NotNull(message = "must not be null")
public Double monthlyIncome;
@NotEmpty(message = "must not be empty")
public String customerCity;
@NotEmpty(message = "must not be empty")
public String employmentType;
@NotNull(message = "must not be null")
public Double desiredLoanAmount;
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
public String getGender() {
return gender;
}
public void setGender(String gender) {
this.gender = gender;
}
public String getEmail() {
return email;
}
public void setEmail(String email) {
this.email = email;
}
public Double getMonthlyIncome() {
return monthlyIncome;
}
public void setMonthlyIncome(Double monthlyIncome) {
this.monthlyIncome = monthlyIncome;
}
public String getCustomerCity() {
return customerCity;
}
public void setCustomerCity(String customerCity) {
this.customerCity = customerCity;
}
public String getEmploymentType() {
return employmentType;
}
public void setEmploymentType(String employmentType) {
this.employmentType = employmentType;
}
public Double getDesiredLoanAmount() {
return desiredLoanAmount;
}
public void setDesiredLoanAmount(Double desiredLoanAmount) {
this.desiredLoanAmount = desiredLoanAmount;
}
}
----------------- Error response .java
/*****************************************************************************
```

```
*************************
* This class ErrorResponse is the value object that must be binded to error view
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION
CLAUSES, RETURN TYPES
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
* CHANGE THE RETURN TYPE FROM NULL OF THE METHODS ONCE YOU BUILT THE LOGIC
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED
BE,
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
EXCEPTION CLAUSE
*
*******************************************************************************
*************************/
package com.cts.loanbazaar.loaneligibility.model;
/**
*
*/
public class ErrorResponse {
/// DO NOT CHANGE THE VARIABLE NAMES OR DATA TYPES OR ACCESS SPECIFIERS
private String errorMessage;
private String requestedURI;
public String getErrorMessage() {
return errorMessage;
}
public void setErrorMessage(String errorMessage) {
this.errorMessage = errorMessage;
}
public String getRequestedURI() {
return requestedURI;
}
public void setRequestedURI(String requestedURI) {
this.requestedURI = requestedURI;
}
}
----------- Loan product .java
/*******************************************************************************
*************************
* This class ErrorResponse is the value object that must be binded to results view
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION
CLAUSES, RETURN TYPES
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
* CHANGE THE RETURN TYPE FROM NULL OF THE METHODS ONCE YOU BUILT THE LOGIC
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED
BE,
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
EXCEPTION CLAUSE
*
*******************************************************************************
*************************/
package com.cts.loanbazaar.loaneligibility.model;
public class LoanProduct {
///DO NOT CHANGE THE VARIABLE NAMES OR DATA TYPES OR ACCESS SPECIFIERS
public String bankName;
```

```
public String loanProductName;
public Double maxLoanAmount;
public Integer tenure;
public Double interest;
public Double monthlyInstallment;
/**
* @return the bankName
*/
public String getBankName() {
return bankName;
}
/**
* @param bankName the bankName to set
*/
public void setBankName(String bankName) {
this.bankName = bankName;
}
/**
* @return the loanProductName
*/
public String getLoanProductName() {
return loanProductName;
}
/**
* @param loanProductName the loanProductName to set
*/
public void setLoanProductName(String loanProductName) {
this.loanProductName = loanProductName;
}
/**
* @return the maxLoanAmount
*/
public Double getMaxLoanAmount() {
return maxLoanAmount;
}
/**
* @param maxLoanAmount the maxLoanAmount to set
*/
public void setMaxLoanAmount(Double maxLoanAmount) {
this.maxLoanAmount = maxLoanAmount;
}
/**
* @return the tenure
*/
public Integer getTenure() {
return tenure;
}
/**
* @param tenure the tenure to set
*/
public void setTenure(Integer tenure) {
this.tenure = tenure;
}
/**
* @return the interest
*/
public Double getInterest() {
```

```
return interest;
}
/**
* @param interest the interest to set
*/
public void setInterest(Double interest) {
this.interest = interest;
}
/**
* @return the monthlyInstallment
*/
public Double getMonthlyInstallment() {
return monthlyInstallment;
}
/**
* @param monthlyInstallment the monthlyInstallment to set
*/
public void setMonthlyInstallment(Double monthlyInstallment) {
this.monthlyInstallment = monthlyInstallment;
}
}
```

---------- service

```
/*******************************************************************************
*************************
* This class LoanEligibilityService is to build the eligible loan products for the customer employment
type
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION
CLAUSES, RETURN TYPES
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
* CHANGE THE RETURN TYPE FROM NULL OF THE METHODS ONCE YOU BUILT THE LOGIC
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED
BE,
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
EXCEPTION CLAUSE
*
*******************************************************************************
*************************/
package com.cts.loanbazaar.loaneligibility.service;
import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Service;
import com.cts.loanbazaar.loaneligibility.exception.ApplicationException;
import com.cts.loanbazaar.loaneligibility.model.CustomerDetails;
import com.cts.loanbazaar.loaneligibility.model.LoanProduct;
@Service
public class LoanEligibilityService {
/**
* @param details
* @return List<LoanProduct>
* @throws ApplicationException
*/
public List<LoanProduct> checkEligibleLoanProducts(CustomerDetails details) throws
ApplicationException {
//TODO add your code here
if(details.getDesiredLoanAmount()> (12*details.getMonthlyIncome())) {
```

```
throw new ApplicationException("Customer Not Eligible for the Loan");
}
List<LoanProduct> data = new ArrayList<LoanProduct>();
if(details.getEmploymentType().length()>=4
&&details.getEmploymentType().length()<=30) {
String employmentType = details.getEmploymentType();
List<LoanProduct> product = new ArrayList<LoanProduct>();
product = buildLoanProducts();
for(int i=0;i<product.size();i++) {
if(product.get(i).getLoanProductName().contains(employmentType)) {
data.add(product.get(i));
}
}
}
return data; //TODO CHANGE THIS RETURN TYPE
}
/**
* Use the below method to test the appropriate loan products based on employment type
of the customer
* DO NOT CHANGE THE VALUES OF THE LOAN PRODUCT DETAILS
* @return List<LoanProduct>
*/
private List<LoanProduct> buildLoanProducts() {
List<LoanProduct> products = new ArrayList<LoanProduct>();
LoanProduct product1 = new LoanProduct();
product1.setBankName("MNQ Bank");
product1.setLoanProductName("Pensioner Pre-Approved Personal Loan");
product1.setMaxLoanAmount(500000.00);
product1.setMonthlyInstallment(7000.00);
product1.setTenure(24);
product1.setInterest(16.40);
products.add(product1);
LoanProduct product2 = new LoanProduct();
product2.setBankName("PMT Bank");
product2.setLoanProductName("Student Pre-Approved Education Loan");
product2.setMaxLoanAmount(1200000.00);
product2.setMonthlyInstallment(11000.00);
product2.setTenure(48);
product2.setInterest(12.40);
products.add(product2);
LoanProduct product3 = new LoanProduct();
product3.setBankName("MNQ Bank");
product3.setLoanProductName("Pre-Approved Personal Loan for Salaried");
product3.setMaxLoanAmount(1000000.00);
product3.setMonthlyInstallment(9000.00);
product3.setTenure(36);
product3.setInterest(15.40);
products.add(product3);
LoanProduct product4 = new LoanProduct();
product4.setBankName("MNQ Bank");
product4.setLoanProductName("Pre-Approved Personal Loan for Salaried");
product4.setMaxLoanAmount(700000.00);
product4.setMonthlyInstallment(8000.00);
product4.setTenure(24);
product4.setInterest(15.20);
products.add(product4);
LoanProduct product5 = new LoanProduct();
```

```
product5.setBankName("MNQ Bank");
product5.setLoanProductName("Pre-Approved Personal Loan for Self-Employed");
product5.setMaxLoanAmount(2500000.00);
product5.setMonthlyInstallment(34000.00);
product5.setTenure(720);
product5.setInterest(11.40);
products.add(product5);
return products;
}
}
```

---------- ApplicationException

```
/*****************************************************************************
*************************
* This class ApplicationException is a user defined exception for the proposed system
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION
CLAUSES, RETURN TYPES
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
* CHANGE THE RETURN TYPE FROM NULL OF THE METHODS ONCE YOU BUILT THE LOGIC
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED
BE,
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
EXCEPTION CLAUSE
* ADD CONSTRUCTORS AS NEEDED
*
*****************************************************************************
*************************/
package com.cts.loanbazaar.loaneligibility.exception;
/**
*
*/
public class ApplicationException extends Exception {
private static final long serialVersionUID = -9079454849611061074L;
public ApplicationException(String message) {
super(message);
}
}
```

---------- ExceptionHandlerControllerAdvice

```
/*****************************************************************************
*************************
* This class ExceptionHandlerControllerAdvice is used to handle different exceptions raised by
Controller
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION
CLAUSES, RETURN TYPES
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
* CHANGE THE RETURN TYPE FROM NULL OF THE METHODS ONCE YOU BUILT THE LOGIC
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED
BE,
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
EXCEPTION CLAUSE
*
*****************************************************************************
*************************/
package com.cts.loanbazaar.loaneligibility.exception;
```

```
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Date;
import javax.servlet.http.HttpServletRequest;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.http.HttpStatus;
import com.cts.loanbazaar.loaneligibility.model.ErrorResponse;
@ControllerAdvice
public class ExceptionHandlerControllerAdvice {
@ExceptionHandler(ApplicationException.class)
@ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
@ResponseBody
public ModelAndView handleResourceNotFound(final ApplicationException exception, final
HttpServletRequest request,
final Model model) {
// TODO add your code here
int statusCode = 0;
ErrorResponse err = new ErrorResponse();
ModelAndView mav = new ModelAndView();
err.setErrorMessage(exception.getMessage());
err.setRequestedURI("http://localhost:8085/" + request.getRequestURI());
try {
URL url = new URL(err.getRequestedURI());
HttpURLConnection http = (HttpURLConnection) url.openConnection();
statusCode = http.getResponseCode();
mav.addObject("code", statusCode);
mav.addObject("curtime", new Date());
mav.addObject("message", err.getErrorMessage());
mav.setViewName("error");
} catch (IOException e) {
System.out.println(e);
}
return mav; // TODO change the return type here
}
}
-------------------- application
/*****************************************************************************
*************************
* This class LoanEligibilityServiceApplication is the starter class for Spring Boot
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION
CLAUSES, RETURN TYPES
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
* CHANGE THE RETURN TYPE FROM NULL OF THE METHODS ONCE YOU BUILT THE LOGIC
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED
BE,
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
EXCEPTION CLAUSE
*
*****************************************************************************
```

```
*************************/
package com.cts.loanbazaar.loaneligibility.main;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
@SpringBootApplication
@ComponentScan(basePackages =
{"com.cts.loanbazaar.loaneligibility.controller","com.cts.loanbazaar.loaneligibility.model","com.cts.l
oanbazaar.loaneligibility.service","com.cts.loanbazaar.loaneligibility.exception"})
public class LoanEligibilityServiceApplication {
public static void main(String[] args) {
SpringApplication.run(LoanEligibilityServiceApplication.class, args);
}
}
------------ app.pro
#DO NOT CHANGE THE BELOW VALUES
server.port=8085
logging.level.org.springframework.web=DEBUG
#ADD YOUR CODE BELOW
spring.mvc.view.prefix = /WEB-INF/jsp/
spring.mvc.view.suffix = .jsp
```

CAR

```
package com.cts.carstore;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import com.cts.carstore.skeletonvalidator.SkeletonValidator;
@SpringBootApplication
@ComponentScan("com.cts.*")
public class CarStoreApplication {
public static void main(String[] args) {
SpringApplication.run(CarStoreApplication.class, args);
new SkeletonValidator();
}
}
package com.cts.carstore.controller;
import java.util.ArrayList;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.validation.Validator;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.cts.carstore.exception.ApplicationException;
import com.cts.carstore.model.CarSearch;
import com.cts.carstore.service.CarStoreService;
@Controller
public class CarStoreController {
@Autowired
private CarStoreService service;
@Autowired
```

```
private Validator validator;
public CarStoreController() {
System.out.println("in default constr of controller");
}
@Autowired
public CarStoreController(CarStoreService service, Validator validator) {
this.service=service;
this.validator=validator;
}
@RequestMapping(value = "/", method = RequestMethod.GET)
public String showCarSearchForm(@ModelAttribute("carSearch") CarSearch carSearch) {
//Add code here..
return "carSearch"; //TODO, modify return value
}
@RequestMapping(value = "/getCarSearchResultPage", method = RequestMethod.POST)
public String getCarSearchResultForm(@ModelAttribute("carSearch") CarSearch carSearch,
ModelMap map,
BindingResult result)throws ApplicationException {
//Add code here..
// Redirecting to the same form page if invalid
if (result.hasErrors()) {
return "carSearch";
}
else {
return "carSearchResult"; //TODO, modify return value
}
}
@ModelAttribute("cityList")
public List<String> populateCities() {
List<String> cities = new ArrayList<String>();
cities.add("Chennai");
cities.add("Mumbai");
cities.add("Delhi");
cities.add("Bangalore");
cities.add("Pune");
cities.add("Kolkatta");
return cities;
}
@ModelAttribute("brandList")
public List<String> populateBrands() {
List<String> brands = new ArrayList<String>();
brands.add("Maruti Suzuki");
brands.add("Honda");
brands.add("Mahindra");
brands.add("Toyota");
brands.add("Hundai");
return brands;
}
@ModelAttribute("budgetList")
public List<String> populateBudget() {
List<String> budgetUpto = new ArrayList<String>();
budgetUpto.add("Below 3 Lakh");
budgetUpto.add("3 Lakh");
budgetUpto.add("5 Lakh");
budgetUpto.add("7 Lakh");
budgetUpto.add("10 Lakh");
return budgetUpto;
```

```
}
@ModelAttribute("fuelTypeList")
public List<String> populateFuelTypes() {
List<String> fuelTypes = new ArrayList<String>();
fuelTypes.add("Petrol");
fuelTypes.add("Diesel");
fuelTypes.add("CNG");
return fuelTypes;
}
}
package com.cts.carstore.exception;
public class ApplicationException extends Exception {
private static final long serialVersionUID = -9079454849611061074L;
public String errorMessage;
}
package com.cts.carstore.exception;
import java.util.Date;
import javax.servlet.http.HttpServletRequest;
import org.springframework.http.HttpStatus;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.servlet.ModelAndView;
@ControllerAdvice
public class ExceptionHandlerControllerAdvice {
@ResponseStatus(value = HttpStatus.INTERNAL_SERVER_ERROR)
public ModelAndView handleResourceNotFound(final ApplicationException exception,
final HttpServletRequest request, final Model model) {
//Add code here..
String message="Low Budget- No car available below 3 lakh price";
Date date= new Date();
//add attribute error and timestamp
return null; //TODO, change return value
}
}
package com.cts.carstore.model;
public class Car {
private String brandName;
private String modelName;
private double price; //price in lakh
private String fuelType;
private double mileage;
private int seatingCapacity;
public Car() {
// TODO Auto-generated constructor stub
}
public Car(String brandName, String modelName, double price, String
fuelType,double mileage, int seatingCapacity) {
super();
this.brandName = brandName;
this.modelName = modelName;
this.price = price;
this.fuelType = fuelType;
this.mileage = mileage;
this.seatingCapacity = seatingCapacity;
}
public String getBrandName() {
return brandName;
}
public void setBrandName(String brandName) {
```

```
this.brandName = brandName;
}
public String getModelName() {
return modelName;
}
public void setModelName(String modelName) {
this.modelName = modelName;
}
public double getPrice() {
return price;
}
public void setPrice(double price) {
this.price = price;
}
public String getFuelType() {
return fuelType;
}
public void setFuelType(String fuelType) {
this.fuelType = fuelType;
}
public int getSeatingCapacity() {
return seatingCapacity;
}
public void setSeatingCapacity(int seatingCapacity) {
this.seatingCapacity = seatingCapacity;
}
public double getMileage() {
return mileage;
}
public void setMileage(double mileage) {
this.mileage = mileage;
}
}
package com.cts.carstore.model;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
public class CarSearch {
// Use validation annotations as per the requirement
@NotNull(message = "Customer name is required")
private String customerName;
@NotBlank(message = "Gender is required")
private String gender;
@NotBlank(message = "Mobile number is required")
@Pattern(regexp="([7,8,9]{1}[0-9]{9})",message="Mobile number should be 10
digit starting with 7,8,9")
private String mobileNumber;
private String customerCity;
private String brand;
private String budgetUpto;
private String fuelType;
public CarSearch() {
// TODO Auto-generated constructor stub
}
public String getCustomerName() {
return customerName;
}
public void setCustomerName(String customerName) {
this.customerName = customerName;
}
public String getGender() {
return gender;
}
```

```
public void setGender(String gender) {
this.gender = gender;
}
public String getMobileNumber() {
return mobileNumber;
}
public void setMobileNumber(String mobileNumber) {
this.mobileNumber = mobileNumber;
}
public String getCustomerCity() {
return customerCity;
}
public void setCustomerCity(String customerCity) {
this.customerCity = customerCity;
}
public String getBrand() {
return brand;
}
public void setBrand(String brand) {
this.brand = brand;
}
public String getBudgetUpto() {
return budgetUpto;
}
public void setBudgetUpto(String budgetUpto) {
this.budgetUpto = budgetUpto;
}
public String getFuelType() {
return fuelType;
}
public void setFuelType(String fuelType) {
this.fuelType = fuelType;
}
}
package com.cts.carstore.model;
public class ErrorResponse {
private String errorMessage;
private String requestedURI;
//add code here
}
package com.cts.carstore.service;
import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Service;
import com.cts.carstore.exception.ApplicationException;
import com.cts.carstore.model.Car;
import com.cts.carstore.model.CarSearch;
@Service
public class CarStoreService {
public List<Car> getCarSearchResult(CarSearch carSearch) throws ApplicationException {
//Add code here..
List<Car> l=buildCars();
List<Car> r=new ArrayList<>();
for(Car i:l) {
if(i.getBrandName().equals(carSearch.getBrand()) &&
i.getFuelType().equals(carSearch.getFuelType()) &&
i.getPrice()<=Double.parseDouble(carSearch.getBudgetUpto().substring(0, 1))) {
r.add(i);
}
}
```

```
return r; //TODO, modify this return value
}
// DO NOT CHANGE THIS METHOD
//DO NOT CHANGE CODE WITHIN METHOD
private List<Car> buildCars() {
List<Car> cars = new ArrayList<Car>();
// brand,modelName,price,fuelType,seatingCapacity
Car car1 = new Car("Maruti Suzuki", "Swift", 5.20, "Petrol", 21.21, 5);
Car car2 = new Car("Maruti Suzuki", "Alto", 3, "Petrol", 22.05, 5);
Car car3 = new Car("Maruti Suzuki", "Ertiga LXi", 7.5, "Petrol", 19.01, 7);
Car car4 = new Car("Maruti Suzuki", "Ertiga VXi", 8.9, "CNG", 26.2, 7);
Car car5 = new Car("Maruti Suzuki", "WagonR", 4.5, "Petrol", 20.52, 5);
Car car6 = new Car("Maruti Suzuki", "WagonR LXi", 5.20, "CNG", 32.52, 5);
Car car7 = new Car("Honda", "Amaze", 6.20, "Petrol", 18.6, 5);
Car car8 = new Car("Honda", "Amaze1.2E MT", 7.6, "Diesel", 24.7, 5);
Car car9 = new Car("Honda", "Amaze1.5E MT", 8, "Diesel", 25.2, 5);
Car car10 = new Car("Honda", "City", 9.30, "Petrol", 17.4, 5);
Car car11 = new Car("Mahindra", "KUV100", 5.6, "Petrol", 15, 6);
Car car12 = new Car("Toyota", "Yaris", 8.8, "Petrol", 18.1, 5);
Car car13 = new Car("Hundai", "Aura", 7.30, "CNG", 20.5, 5);
Car car14 = new Car("Hundai", "Creta", 9.9, "Petrol", 21.5, 5);
cars.add(car1);
cars.add(car2);
cars.add(car3);
cars.add(car4);
cars.add(car5);
cars.add(car6);
cars.add(car7);
cars.add(car8);
cars.add(car9);
cars.add(car10);
cars.add(car11);
cars.add(car12);
cars.add(car13);
cars.add(car14);
return cars;
}
}
package com.cts.carstore.skeletonvalidator;
import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;
public class SkeletonValidator {
public SkeletonValidator() {
validateClassName("com.cts.carstore.model.Car");
validateClassName("com.cts.carstore.model.CarSearch");
validateClassName("com.cts.carstore.model.ErrorResponse");
validateClassName("com.cts.carstore.service.CarStoreService");
validateClassName("com.cts.carstore.controller.CarStoreController");
validateClassName("com.cts.carstore.exception.ApplicationException");
validateClassName("com.cts.carstore.exception.ExceptionHandlerControllerAdvice");
validateMethodSignature("getCarSearchResult:java.util.List",
"com.cts.carstore.service.CarStoreService");
validateMethodSignature("showCarSearchForm:java.lang.String,populateCities:java.util.List,
populateBrands:java.util.List,populateBudget:java.util.List,populateFuelTypes:java.util.List",
"com.cts.carstore.controller.CarStoreController");
}
```

```
private static final Logger LOG = Logger.getLogger("SkeletonValidator");
protected final boolean validateClassName(String className) {
boolean iscorrect = false;
try {
Class.forName(className);
iscorrect = true;
LOG.info("Class Name " + className + " is correct");
} catch (ClassNotFoundException e) {
LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
+ "and class name as provided in the skeleton");
} catch (Exception e) {
LOG.log(Level.SEVERE,
"There is an error in validating the " + "Class Name. Please
manually verify that the "
+ "Class name is same as skeleton before
uploading");
}
return iscorrect;
}
protected final void validateMethodSignature(String methodWithExcptn, String className) {
Class cls = null;
try {
String[] actualmethods = methodWithExcptn.split(",");
boolean errorFlag = false;
String[] methodSignature;
String methodName = null;
String returnType = null;
for (String singleMethod : actualmethods) {
boolean foundMethod = false;
methodSignature = singleMethod.split(":");
methodName = methodSignature[0];
returnType = methodSignature[1];
cls = Class.forName(className);
Method[] methods = cls.getMethods();
for (Method findMethod : methods) {
if (methodName.equals(findMethod.getName())) {
foundMethod = true;
if
(!(findMethod.getReturnType().getName().equals(returnType))) {
errorFlag = true;
LOG.log(Level.SEVERE, " You have changed
the " + "return type in '" + methodName
+ "' method. Please stick to
the " + "skeleton provided");
} else {
LOG.info("Method signature of " +
methodName + " is valid");
}
}
}
if (!foundMethod) {
errorFlag = true;
LOG.log(Level.SEVERE, " Unable to find the given public
method " + methodName
+ ". Do not change the " + "given public
method name. " + "Verify it with the skeleton");
```

```
}
}
if (!errorFlag) {
LOG.info("Method signature is valid");
}
} catch (Exception e) {
LOG.log(Level.SEVERE,
" There is an error in validating the " + "method structure.
Please manually verify that the "
+ "Method signature is same as the skeleton
before uploading");
}
}
}
server.port=3030
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
spring.mvc.static-class-path=/resources/**
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<br>
<br>
<!-- Add code here.. -->
<h1 id="heading">Windsor Car Showroom</h1>
<table>
<center></center><caption style="font-weight: bolder;">Search
Cars</caption></center>
<sf:form action="getCarSearchResultPage" modelAttribute="carSearch"
name="form">
<tr>
<td>Customer Name:</td>
<td><sf:input path="customerName" id="customerName"
name="customerName"/></td>
<td><sf:errors path="customerName" /></td>
</tr>
<tr>
<td>Mobile Number:</td>
<td><sf:input path="mobileNumber" id="mobileNumber"
/></td>
<td><sf:errors path="mobileNumber" /></td>
</tr>
<tr>
<td>Gender:</td>
<td>
<sf:radiobutton path="gender" value="male"/>
<label class='radiolabel'>Male</label>
<sf:radiobutton path="gender" value="female" />
<label class='radiolabel'>Female</label> </td>
</tr>
<tr>
<tr>
<td>Customer City:</td>
<td><sf:select path="gender" id="gender"
```

```
items="${cityList}" /></td>
</tr>
<tr>
<tr>
<td>Brand:</td>
<td><sf:select path="gender" id="gender"
items="${brandList}" /></td>
</tr>
<tr>
<tr>
<td>Fuel/Transmission:</td>
<td><sf:select path="gender" id="gender"
items="${fuelTypeList}" /></td>
</tr>
<tr>
<tr>
<tr>
<td>Budget Upto:</td>
<td><sf:select path="gender" id="gender"
items="${budgetList}" /></td>
</tr>
<td><input type="submit" value="CarSearch"
name="CarSearch" /></td>
<td><input type="reset" value="Clear" /></td>
</tr>
</sf:form>
</table>
</body>
</html>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<br>
<br>
<!--Add code here -->
<br>
<br>
<h3>Here are cars matching your search criteria:</h3>
<table border="1">
<tr><td>Brand Name</td><td>Model Name</td><td>Price in
lakh</td><td>Fuel/Transmission</td><td>Mileage</td><td>Seating Capacity</td></tr>
<c:forEach var="i" items="${list}">
<tr><td><c:out value="${i.getBrandName()}"/></td>
<td><c:out value="${i.getModelName()}"/></td>
<td><c:out value="${i.getPrice()}"/></td>
<td><c:out value="${i.getFuelType()}"/></td>
<td><c:out value="${i.getMileage()}"/></td>
<td><c:out value="${i.getSeatingCapacity()}"/></td>
</tr>
</c:forEach>
</table>
</body>
</html>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
```

```
pageEncoding="ISO-8859-1" isELIgnored="false" isErrorPage="true"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE HTML>
<html>
<head>
<title>Car Search Errors</title>
</head>
<body>
<!--Add code here.. -->
<form:form id="error">
<h3>Unable to retrieve loan information. Below are the error details:</h3>
<h3>Response Code: ${code }</h3>
<h3>Error Message: ${message }</h3>
<h3>Error Occurred on: ${curtime }</h3>
</form:form>
</body>
</html>
```

TMS

```
package com;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
@SpringBootApplication
@ComponentScan("com.*")
public class TmsApplication {
/**
* Starting point of the application
*
* @param args Arguments passed to the application
*/
public static void main(String[] args) {
SpringApplication.run(TmsApplication.class, args);
}
}
package com.controller;
import java.util.Locale;
import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ReloadableResourceBundleMessageSource;
import org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;
import org.springframework.web.servlet.LocaleResolver;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;
@Configuration
public class InternationalizationConfig extends WebMvcConfigurerAdapter {
/**
* Set default Locale
*
* @return A bean of LocalResolver
```

```
*/
@Bean
public LocaleResolver localeResolver() {
SessionLocaleResolver slr = new SessionLocaleResolver();
slr.setDefaultLocale(Locale.US);
return slr;
}
/**
* Set path variable name for changing language
*
* @return A bean of LocaleChangeInterceptor
*/
@Bean
public LocaleChangeInterceptor localeChangeInterceptor() {
LocaleChangeInterceptor lci = new LocaleChangeInterceptor();
lci.setParamName("language");
return lci;
}
/**
* Add interceptor into the registry
*/
@Override
public void addInterceptors(InterceptorRegistry registry) {
registry.addInterceptor(localeChangeInterceptor());
}
/**
* Set base name for messages.properties files Set default encoding to UTF-8
*
* @return A bean of MessageSource
*/
@Bean
public MessageSource messageSource() {
ReloadableResourceBundleMessageSource rrbms = new
ReloadableResourceBundleMessageSource();
rrbms.setBasename("classpath:messages");
rrbms.setDefaultEncoding("UTF-8");
return rrbms;
}
/**
* Set validation message source
*
* @return A bean of LocalValidatorFactoryBean
*/
@Bean
public LocalValidatorFactoryBean localValidatorFactoryBean() {
LocalValidatorFactoryBean lvfb = new LocalValidatorFactoryBean();
lvfb.setValidationMessageSource(messageSource());
return lvfb;
}
}
package com.controller;
import java.util.Arrays;
import java.util.List;
import javax.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
```

```
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.model.UserClaim;
import com.service.TaxService;
@Controller
public class TaxController {
@Autowired
public TaxService taxService;
/**
* Display taxclaim.jsp page when a get request is pushed on url
* /getTaxClaimFormPage
*
* @param userClaim Is the UserClaim component
* @return taxclaim as a jsp page
* @see UserClaim
*/
@RequestMapping(value = "/getTaxClaimFormPage", method = RequestMethod.GET)
public String claimPage(@ModelAttribute("userClaim") UserClaim userClaim) {
return "taxclaim";
}
/**
* Return result.jsp age when validation is successful Otherwise return back to
* taxclaim page with error message
*
* @param userClaim UserClaim component
* @param result BindingResult which validate the user input
* @param map ModelMap to put attribute which will be forwarded to next
* page
* @return "result.jsp" page if the validation is successful otherwise
* "taxclaim.jsp" with error included
*/
@RequestMapping(value = "/calculateTax", method = RequestMethod.GET)
public String calculateTax(@Valid @ModelAttribute("userClaim") UserClaim userClaim,
BindingResult result,
ModelMap map) {
if (result.hasErrors()) {
return "taxclaim";
}
double amount = taxService.calculateTax(userClaim);
map.addAttribute("amount", amount);
return "result";
}
/**
* Populate <form:select /> tag in the taxclaim.jsp page
*
* @return List of expenses
*/
@ModelAttribute("expenseList")
public List<String> populateExpense() {
return Arrays.asList("MedicalExpense", "TravelExpense", "FoodExpense");
}
}
package com.model;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.PositiveOrZero;
```

```
import javax.validation.constraints.Size;
import org.springframework.stereotype.Component;
@Component
public class UserClaim {
private String expenseType;
@PositiveOrZero(message = "{error.expenseAmount.negative}")
private double expenseAmt;
@NotBlank(message = "{error.employeeId}")
@Size(min = 5, message = "{error.employeeId.size}")
private String employeeId;
public String getExpenseType() {
return expenseType;
}
public void setExpenseType(String expenseType) {
this.expenseType = expenseType;
}
public double getExpenseAmt() {
return expenseAmt;
}
public void setExpenseAmt(double expenseAmt) {
this.expenseAmt = expenseAmt;
}
public String getEmployeeId() {
return employeeId;
}
public void setEmployeeId(String employeeId) {
this.employeeId = employeeId;
}
}
package com.service;
import org.springframework.stereotype.Service;
import com.model.UserClaim;
@Service
public interface TaxService {
/**
* Calculate Tax
*
* @param userClaim UserClaim bean
* @return Calculated tax
*/
public double calculateTax(UserClaim userClaim);
}
package com.service;
import org.springframework.stereotype.Service;
import com.model.UserClaim;
@Service
public class TaxServiceImpl implements TaxService {
/**
* Calculate the tax according to the srs
*
* @param userClaim UserClaim component to get the values
* @return Calculated tax
*/
@Override
public double calculateTax(UserClaim userClaim) {
String e = userClaim.getExpenseType();
double a = userClaim.getExpenseAmt();
```

```
double t = 0.0;
if (e.startsWith("M")) {
if (a <= 1000) {
t = 15.0;
} else if (a > 1000 && a <= 10000) {
t = 20.0;
} else if (a > 10000) {
t = 25.0;
}
} else if (e.startsWith("T")) {
if (a <= 1000) {
t = 10.0;
} else if (a > 1000 && a <= 10000) {
t = 15.0;
} else if (a > 10000) {
t = 20.0;
}
} else if (e.startsWith("F")) {
if (a <= 1000) {
t = 5.0;
} else if (a > 1000 && a <= 10000) {
t = 10.0;
} else if (a > 10000) {
t = 15.0;
}
}
return a * (t / 100.0);
}
}
server.port=9095
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
spring.mvc.static-class-path=/resources/**
---------------de.pro
label.employeeId=Employee ID in German
label.expenseType=Expense Type in German
label.expenseAmount=Expense Amount in German
error.employeeId=Employee ID cannot be empty in German
error.employeeId.size=Employee ID should be at least 5 characters in German
error.expenseAmount=Expense Amount cannot be empty in German
error.expenseAmount.numeric=Expense amount should be numeric only in German
error.expenseAmount.negative=Expense amount should not be a negative number in
German
------ fr.pro
label.employeeId=Employee ID in French
label.expenseType=Expense Type in French
label.expenseAmount=Expense Amount in French
error.employeeId=Employee ID cannot be empty in French
error.employeeId.size=Employee ID should be at least 5 characters in French
error.expenseAmount=Expense Amount cannot be empty in French
error.expenseAmount.numeric=Expense amount should be numeric only in French
error.expenseAmount.negative=Expense amount should not be a negative number in
French
------------- msg.pro
label.employeeId=Employee ID in English
label.expenseType=Expense Type in English
label.expenseAmount=Expense Amount in English
error.employeeId=Employee ID cannot be empty in English
error.employeeId.size=Employee ID should be at least 5 characters in English
error.expenseAmount=Expense Amount cannot be empty in English
```

```
error.expenseAmount.numeric=Expense amount should be numeric only in English
error.expenseAmount.negative=Expense amount should not be a negative number in
English
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>The tax claim for ${ userClaim.expenseType } with expense amount
${ userClaim.expenseAmt } is ${ amount }</h2>
</body>
</html>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" isELIgnored="false"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color: lavender">
<h1>
<center>Tax: Tax Claim</center>
</h1>
<a href="/getTaxClaimFormPage?language=en">English</a>|
<a href="/getTaxClaimFormPage?language=de">German</a>|
<a href="/getTaxClaimFormPage?language=fr">French</a>
</align>
<form:form action="/calculateTax" method="get" modelAttribute="userClaim">
<table>
<tr>
<td id="id1">
<spring:message code="label.employeeId" />
</td>
<td id="id2">
<form:input path="employeeId" id="employeeId" />
</td>
<td id="id3">
<form:errors path="employeeId" />
</td>
</tr>
<tr>
<td id="id4">
<spring:message code="label.expenseType" />
</td>
<td id="id5">
<form:select path="expenseType" items="${
expenseList }" id="expenseType" />
</td>
<td id="id6"></td>
</tr>
<tr>
<td id="id7">
<spring:message code="label.expenseAmount" />
</td>
<td id="id8">
<form:input path="expenseAmt" id="expenseAmount"
```

```
/>
</td>
<td id=id9>
<form:errors path="expenseAmt" />
</td>
</tr>
<tr>
<td><input type="Submit" name="submit" value="Calculate
Claim" /></td>
<td></td>
</tr>
<tr>
<td><input type="reset" name="reset" value="Clear"
/></td>
<td></td>
</tr>
</table>
</form:form>
</body>
</html>
```

PINK FITNESS

FitnessBO:

```
package com.spring.bo;


import com.spring.model.FitnessFranchise;


public class FitnessBO {


	public double calculateNetProfit(FitnessFranchise franchise) {


		//// Calculating netProfit based on factors


		double expense = franchise.getTotalIncome() - (franchise.getTrainerSalary()

				+ franchise.getCommonExpenses().get("Rent") +
franchise.getCommonExpenses().get("HouseKeeping")

				+ franchise.getCommonExpenses().get("ElectricityCharges")

				+ franchise.getCommonExpenses().get("Maintenence"));


		return expense;
```

```
	}
}
```

No ProfitException

```
package com.spring.exception;


public class NoProfitException extends Exception {


	public NoProfitException(String msg) {
		//fill the code
		super(msg);
	}
}
```

Driver:

```
package com.spring.main;


import java.util.*;


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


import com.spring.exception.NoProfitException;
import com.spring.model.FitnessFranchise;
import com.spring.service.FitnessService;


public class Driver {


	public static void main(String[] args) {
		// TODO Auto-generated method stub
		// fill the code
		Scanner sc = new Scanner(System.in);
		ApplicationContext applicationContext = new
ClassPathXmlApplicationContext("beans.xml");
```

```
System.out.println("Pink Fitness Franchise 1 details");
System.out.println("Enter the total location:");
String location = sc.next();
System.out.println("Enter the total income:");
double totalIncome = sc.nextDouble();
System.out.println("Enter total expense amount for trainer salary:");
double trainerSalary = sc.nextDouble();
FitnessFranchise fitnessFranchise1 = (FitnessFranchise)
applicationContext.getBean("fitnessFranchise");
fitnessFranchise1.setLocation(location);
fitnessFranchise1.setTotalIncome(totalIncome);
fitnessFranchise1.setTrainerSalary(trainerSalary);
System.out.println("Pink Fitness Franchise 2 details");
System.out.println("Enter the total location:");
String location2 = sc.next();
System.out.println("Enter the total income:");
double totalIncome2 = sc.nextDouble();
System.out.println("Enter total expense amount for trainer salary:");
double trainerSalary2 = sc.nextDouble();
FitnessFranchise fitnessFranchise2 = (FitnessFranchise)
applicationContext.getBean("fitnessFranchise");
fitnessFranchise2.setLocation(location2);
fitnessFranchise2.setTotalIncome(totalIncome2);
fitnessFranchise2.setTrainerSalary(trainerSalary2);

FitnessService fitnessService = (FitnessService)
applicationContext.getBean("fitnessService");
try {

fitnessService.calculateNetProfit(fitnessFranchise1);
System.out.println("Pink Fitness at " + fitnessFranchise1.getLocation() + " franchise
Amount is RS:"
+ fitnessFranchise1.getFranchiseAmount());
```

```
		} catch (NoProfitException e) {
			System.out.println(e.getMessage());
		}
		try {

			fitnessService.calculateNetProfit(fitnessFranchise2);
			System.out.println("Pink Fitness at " + fitnessFranchise2.getLocation() + " franchise
Amount is RS:"
						+ fitnessFranchise2.getFranchiseAmount());
		} catch (NoProfitException e) {
			System.out.println(e.getMessage());
		}

	}

}
```

FitnessFranchise:

```
package com.spring.model;

import java.util.Map;
// implement the FitnessFranchisor interface
public class FitnessFranchise implements FitnessFranchisor {
	private String location;
	private double totalIncome;
	private double trainerSalary;
	private double franchiseAmount;
	private double percentage;
	private Map<String, Double> commonExpenses;

	public FitnessFranchise(double percentage, Map<String, Double> commonExpenses) {
		super();
		this.percentage = percentage;
```

```
        this.commonExpenses = commonExpenses;
}

public void setPercentage(double percentage) {
        this.percentage = percentage;
}

public double getPercentage() {
        return percentage;
}

public double getFranchiseAmount() {
        return franchiseAmount;
}

public Map<String, Double> getCommonExpenses() {
        return commonExpenses;
}

public void setCommonExpenses(Map<String, Double> commonExpenses) {
        this.commonExpenses = commonExpenses;
}

public void setFranchiseAmount(double franchiseAmount) {
        this.franchiseAmount = franchiseAmount;
}


public String getLocation() {
        return location;
}
```

```
	public void setLocation(String location) {
		this.location = location;
	}



	public double getTotalIncome() {
		return totalIncome;
	}

	public void setTotalIncome(double totalIncome) {
		this.totalIncome = totalIncome;
	}



	public double getTrainerSalary() {
		return trainerSalary;
	}

	public void setTrainerSalary(double trainerSalary) {
		this.trainerSalary = trainerSalary;
	}

	public void calculateFranchiseAmount(double netProfit) {

	//Calculating Franchise Amount
		this.franchiseAmount=netProfit*(percentage)/100;

	}
}
```

FitnessFranchiser:

```
package com.spring.model;
```

```
public interface FitnessFranchisor {

 public void calculateFranchiseAmount(double netProfit);

}
```

FitnessService:

```
package com.spring.service;

import com.spring.bo.FitnessBO;
import com.spring.exception.NoProfitException;
import com.spring.model.FitnessFranchise;

public class FitnessService {

	private FitnessBO fitnessBO;



	public FitnessBO getFitnessBO() {
		return fitnessBO;
	}

	public void setFitnessBO(FitnessBO fitnessBO) {
		this.fitnessBO = fitnessBO;
	}
	public void calculateNetProfit(FitnessFranchise franchise) throws NoProfitException {
		//fill the code

		double netProfit=fitnessBO.calculateNetProfit(franchise);

		if(netProfit<0)
		{
			throw new NoProfitException("No profit to calculate franchise amount");
```

```
            }
            else
            {
                franchise.calculateFranchiseAmount(netProfit);
            }
        }

        public FitnessService(FitnessBO fitnessBO) {
            super();
            this.fitnessBO = fitnessBO;
        }
}
```

Beans.xml:

```
<?xml version="1.0"?>
<beans xsi:schemaLocation="http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.2.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-
3.2.xsd http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-
task-3.2.xsd" xmlns:task="http://www.springframework.org/schema/task"
xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:jee="http://www.springframework.org/schema/jee"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.springframework.org/schema/beans"><context:property-placeholder
location="classpath:FranchisePercent.properties"/>
<!-- fill the code -->
<bean scope="prototype" class="com.spring.model.FitnessFranchise"
name="fitnessFranchise"><constructor-arg name="percentage" value="${percentage}"/><constructor-arg
name="commonExpenses"><map><entry value="12000.0" key="Rent"/><entry value="4000.0"
key="HouseKeeping"/><entry value="2000.0" key="ElectricityCharges"/><entry value="3000.0"
key="Maintenence"/></map></constructor-arg></bean><bean scope="prototype"
class="com.spring.service.FitnessService" name="fitnessService"><constructor-arg
ref="fitnessBO"/></bean><bean scope="prototype" class="com.spring.bo.FitnessBO"
name="fitnessBO"/></beans>
```

POM.xml

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.0"><modelVersion>4.0.0</modelVersion><groupId>PinkFitness</groupId><artifactId>PinkFitness</artifactId><version>0.0.1-SNAPSHOT</version><dependencies><dependency><groupId>org.springframework</groupId><artifactId>spring-context</artifactId><version>4.3.10.RELEASE</version></dependency><dependency><groupId>junit</groupId><artifactId>junit</artifactId><version>4.12</version><scope>test</scope></dependency><dependency><groupId>xmlunit</groupId><artifactId>xmlunit</artifactId><version>1.6</version><scope>test</scope></dependency>
<!-- https://mvnrepository.com/artifact/com.github.stefanbirkner/system-rules -->
<dependency><groupId>com.github.stefanbirkner</groupId><artifactId>system-rules</artifactId><version>1.16.0</version><scope>test</scope></dependency><dependency><groupId>com.sun.xml.bind</groupId><artifactId>jaxb-impl</artifactId><version>2.1.13</version></dependency>
<!-- Thanks for using https://jar-download.com -->
<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency><groupId>javax.xml.bind</groupId><artifactId>jaxb-api</artifactId><version>2.1</version></dependency></dependencies><build><plugins><plugin><groupId>org.apache.maven.plugins</groupId><artifactId>maven-surefire-plugin</artifactId><version>3.0.0-M1</version><configuration><testFailureIgnore>true</testFailureIgnore></configuration></plugin></plugins></build><properties><maven.compiler.source>1.8</maven.compiler.source><maven.compiler.target>1.8</maven.compiler.target></properties></project>
```


ADVERTISEMENT BOOKING

CostPerSecond.properties

```
costPerSecond=200
```

AdvertisementBO.java

```
package com.spring.bo;


import org.springframework.context.annotation.ComponentScan;


import com.spring.model.Advertisement;


//use appropriate annotation to make this class as component class

@ComponentScan("com.spring")

public class AdvertisementBO {


	public double bookAdvertisement(Advertisement advertisement) {


		double result=0.0;
```

```
                // fill the code
                return result;
        }
}
```

ApplicationConfig.java

```
package com.spring.config;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

// Use appropriate annotation
@Configuration
@ComponentScan("com.spring")

public class ApplicationConfig {



}
```

InvalidDurationException.java

```
package com.spring.exception;

public class InvalidDurationException extends Exception {

public InvalidDurationException(String msg){
        // fill the code
        System.out.println("Invalid Duration");
}
}
```

Driver.java

```
package com.spring.main;

import java.util.Scanner;

import org.springframework.context.ApplicationContext;
```

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

import com.spring.exception.InvalidDurationException;
import com.spring.service.AdvertisementService;

public class Driver {

public static void main(String[] args){
	// fill the code

	ApplicationContext ctx = new ClassPathXmlApplicationContext("ApplicationConfig.java");
	AdvertisementService advertisementService = ctx.getBean(AdvertisementService.class);



	Scanner sc = new Scanner("System.in");

	System.out.println("Enter the Advertisement Id:");
	int advertisementId= sc.nextInt();

	System.out.println("Enter the Advertisement Type:");
	String advertisementType= sc.next();

	System.out.println("Enter the Advertisement Duration(in seconds):");
	int duration = sc.nextInt();

	System.out.println("Enter number of Days/Week to telecast:");
	int daysPerWeek = sc.nextInt();

	System.out.println("Enter preferred time slot");
	String timeSlot = sc.next();

	System.out.println("Enter the television name:");
	String channelName = sc.next();
```

```
	try {
		System.out.println("Total cost to telecast the advertisement is RS:"
+advertisementService.bookAdvertisement(advertisementId, advertisementType, duration, daysPerWeek, timeSlot, channelName));
	} catch (InvalidDurationException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
}

}
```

Advertisement.java

```
package com.spring.model;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

//pojo class with required attributes,getters and setters

//use appropriate annotation to make this class as component class
 @Component
public class Advertisement {
	private int advertisementId;
	private String advertisementType;
	private int duration;
	private int daysPerWeek;
	private String timeSlot;
	private TelevisionChannel televisionChannel;

// fill the code
	@Autowired
	public Advertisement(TelevisionChannel		televisionChannel) {
		super();
		this.televisionChannel = televisionChannel;
	}
```

```
public int getAdvertisementId() {
        return advertisementId;
}

public void setAdvertisementId(int advertisementId) {
        this.advertisementId = advertisementId;
}
public String getTimeSlot() {
        return timeSlot;
}
public void setTimeSlot(String timeSlot) {
        this.timeSlot = timeSlot;
}

public String getAdvertisementType() {
        return advertisementType;
}
public void setAdvertisementType(String advertisementType) {
        this.advertisementType = advertisementType;
}
public int getDuration() {
        return duration;
}
public void setDuration(int duration) {
        this.duration = duration;
}
public int getDaysPerWeek() {
        return daysPerWeek;
}
public void setDaysPerWeek(int daysPerWeek) {
        this.daysPerWeek = daysPerWeek;
}
public TelevisionChannel getTelevisionChannel() {
        return televisionChannel;
```

```
	}
	public void setTelevisionChannel(TelevisionChannel televisionChannel) {
		this.televisionChannel = televisionChannel;
	}



}
```

TelevisionChannel.java

```
package com.spring.model;

import java.util.Map;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;



//pojo class with required attributes,getters and setters

//use appropriate annotation to make this class as component class
@Component()
public class TelevisionChannel {
	private String channelName;
	public String getChannelName() {
		return channelName;
	}

	public void setChannelName(String channelName) {
		this.channelName = channelName;
	}

	// fill the code
	@Value("${costPerSecond.properties}")
	private double costPerSecond;
```

```
	// fill the code
	@Value("#{${maps: {6pm-7pm : '250' , 7pm-8pm :'500' , 8pm-9pm :'750' , 9pm-10pm :'1000'}}}")
	private Map<String,Double> primeRate;

	public double getCostPerSecond() {
		return costPerSecond;
	}

	public void setCostPerSecond(double costPerSecond) {
		this.costPerSecond = costPerSecond;
	}

	public Map<String, Double> getPrimeRate() {
		return primeRate;
	}

	public void setPrimeRate(Map<String, Double> primeRate) {
		this.primeRate = primeRate;
	}

}
```

AdvertisementService.java

```
package com.spring.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.ComponentScan;

import com.spring.bo.AdvertisementBO;
import com.spring.exception.InvalidDurationException;
//Use appropriate annotation
@ComponentScan("com.spring")
public class AdvertisementService {

	private AdvertisementBO advertisementBO;
```

```
	//Use appropriate annotation
	@Autowired
	public AdvertisementService(AdvertisementBO advertisementBO) {
		super();
		this.advertisementBO = advertisementBO;
	}

	public AdvertisementBO getAdvertisementBO() {
		return advertisementBO;
	}

	public void setAdvertisementBO(AdvertisementBO advertisementBO) {
		this.advertisementBO = advertisementBO;
	}

	public double bookAdvertisement(int advertisementId, String advertisementType, int duration, int
daysPerWeek, String timeSlot,String channelName) throws InvalidDurationException {

		//double result=0.0;
		//fill the code

		if(duration<5 || duration>90) {
			throw new InvalidDurationException("Invalid Duration");
		} else {
		double TotalCost = (duration * 200 * daysPerWeek) ;
		return TotalCost;
		}
	}

}
```

POM.XML

```
<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">
  <modelVersion>4.0.0</modelVersion>
```

```
  <groupId>AdvertisementBooking</groupId>
  <artifactId>AdvertisementBooking</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>AdvertisementBooking</name>
  <url>http://maven.apache.org</url>

  <dependencies>
          <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>4.3.10.RELEASE</version>
          </dependency>

          <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
          <dependency>
    <groupId>xmlunit</groupId>
    <artifactId>xmlunit</artifactId>
    <version>1.6</version>
    <scope>test</scope>
</dependency>


          <!-- https://mvnrepository.com/artifact/com.github.stefanbirkner/system-rules -->
<dependency>
    <groupId>com.github.stefanbirkner</groupId>
    <artifactId>system-rules</artifactId>
```

```
   <version>1.16.0</version>
   <scope>test</scope>
</dependency>

<dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-impl</artifactId>
            <version>2.1.13</version>
</dependency>
<!-- Thanks for using https://jar-download.com -->

<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency>
   <groupId>javax.xml.bind</groupId>
   <artifactId>jaxb-api</artifactId>
   <version>2.1</version>
</dependency>

            </dependencies>

             <build>
        <plugins>
    <plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-surefire-plugin</artifactId>
   <version>3.0.0-M1</version>
   <configuration>
      <testFailureIgnore>true</testFailureIgnore>
   </configuration>

  </plugin>

</plugins>
</build>
```

```
  <properties>
   <maven.compiler.source>1.8</maven.compiler.source>
   <maven.compiler.target>1.8</maven.compiler.target>
</properties>
</project>
```

FURNTURN OUTLET

ProductBO.java

```
package com.spring.bo;
import java.util.Map;
import org.springframework.stereotype.Component;
import com.spring.model.Product;

@Component
public class ProductBO {
	public double calculateBill(Product furObj,String woodType) {
		double amount=0;
	  Map<String,Integer> map= furObj.getShopDetails().getDiscountDetails();
	  if(map.containsKey(woodType)) {
		int d=map.get(woodType);
		amount= furObj.getMrpValue()-(furObj.getMrpValue()*d)/100.0;
	  }
		return amount;
	}
}
```

ApplicationConfig.java

```
package com.spring.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.stereotype.Component;
import org.springframework.context.annotation.ComponentScan;
import com.spring.bo.ProductBO;
```

```
import com.spring.model.Product;
import com.spring.model.Shop;
import com.spring.service.ProductService;

@Configuration
@Component
@ComponentScan
public class ApplicationConfig {

        @Bean
        public Shop shop() {
                return new Shop();
        }

    @Bean
    public Product product() {
        return new Product(shop());
    }

    @Bean
    public ProductBO productBO() {
        return new ProductBO();
    }

    @Bean
    public ProductService productService() {
        return new ProductService(productBO());
    }
}
```

InvalidProductIdException.java

```
package com.spring.exception;
import org.springframework.stereotype.Component;

@Component
```

```
public class InvalidProductIdException extends Exception {

	public InvalidProductIdException(String msg) {

		// fill the code
		super(msg);
	}

}
```

Driver.java

```
package com.spring.main;
import java.util.Scanner;
import org.springframework.stereotype.Component;
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import com.spring.config.ApplicationConfig;
import com.spring.exception.InvalidProductIdException;
import com.spring.model.Product;
import com.spring.service.ProductService;

@Component
public class Driver {

	public static void main(String[] args) {

		// fill the code
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the product id:");
		String productId= sc.nextLine();
		System.out.println("Enter the product name:");
		String productName=sc.nextLine();
		System.out.println("Enter the mrp value:");
		double mrp= sc.nextDouble();
		sc.nextLine();
```

```
System.out.println("Enter the dimension details:");
String dimension= sc.nextLine();
System.out.println("Enter the wood type:");
String woodType= sc.nextLine();
ConfigurableApplicationContext context=new
AnnotationConfigApplicationContext(ApplicationConfig.class);
ProductService product=(ProductService)context.getBean(ProductService.class);
try {
double d =product.calculateBill(productId, productName, mrp, dimension, woodType);
System.out.println("Amount to be paid is :"+ d);
} catch (InvalidProductIdException e) {

System.out.println(e.getMessage());
}

}
}
```

Product.java

```
package com.spring.model;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class Product {
private String productId;
private String productName;
private double mrpValue;
private Shop shopDetails;
private String dimension;

@Autowired
public Product(Shop shopDetails) {
```

```
        super();
        this.shopDetails = shopDetails;
    }
    public String getDimension() {
        return dimension;
    }
    public void setDimension(String dimension) {
        this.dimension = dimension;
    }
    public String getProductId() {
        return productId;
    }
    public void setProductId(String productId) {
        this.productId = productId;
    }
    public String getProductName() {
        return productName;
    }
    public void setProductName(String productName) {
        this.productName = productName;
    }
    public double getMrpValue() {
        return mrpValue;
    }
    public void setMrpValue(double mrpValue) {
        this.mrpValue = mrpValue;
    }
    public Shop getShopDetails() {
        return shopDetails;
    }
    public void setShopDetails(Shop shopDetails) {
        this.shopDetails = shopDetails;
    }
```

```
}
```

Shop.java

```
package com.spring.model;
import java.util.Map;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class Shop {

	@Value("FurnTurn Bang")
	private String shopName;

	@Value("Bangalore")
	private String shopLocation;

	@Value("#{${discountDetails : {Teak: '3',Sheesham: '5',RubberWood: '2',Wenge: '8',Pine: '4'}}}")
	private Map<String,Integer> discountDetails;

	public String getShopName() {
		return shopName;
	}
	public void setShopName(String shopName) {
		this.shopName = shopName;
	}
	public String getShopLocation() {
		return shopLocation;
	}
	public void setShopLocation(String shopLocation) {
		this.shopLocation = shopLocation;
	}
	public Map<String, Integer> getDiscountDetails() {
		return discountDetails;
	}
```

```
	public void setDiscountDetails(Map<String, Integer> discountDetails) {
		this.discountDetails = discountDetails;
	}

}
```

ProductService.java

```
package com.spring.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import com.spring.bo.ProductBO;
import com.spring.config.ApplicationConfig;
import com.spring.exception.InvalidProductIdException;
import org.springframework.stereotype.Component;
import com.spring.model.Product;

@Component
public class ProductService {

	private ProductBO productBOObj;
   @Autowired
	public ProductService(ProductBO productBOObj) {
		super();
		this.productBOObj = productBOObj;
	}

	public ProductBO getProductBOObj() {
		return productBOObj;
	}

	public void setProductBOObj(ProductBO productBOObj) {
		this.productBOObj = productBOObj;
	}
```

```
	public double calculateBill(String productId,String productName,double mrpValue,String dimension, String woodType) throws InvalidProductIdException {


		double amount=0;

		// fill the code

		if(productId.matches("[0-9]+")) {

			ConfigurableApplicationContext context=new AnnotationConfigApplicationContext(ApplicationConfig.class);

			Product product=(Product)context.getBean(Product.class);

			product.setDimension(dimension);

			product.setMrpValue(mrpValue);

			product.setProductId(productId);

			product.setProductName(productName);

		amount = productBOObj.calculateBill(product, woodType);

		}

		else

		{

			throw new InvalidProductIdException("Invalid Product ID");

		}


		return amount;


	}


}
```

POM.XML

```
<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">

  <modelVersion>4.0.0</modelVersion>

  <groupId>FurnTurnOutlet</groupId>

  <artifactId>FurnTurnOutlet</artifactId>

  <version>0.0.1-SNAPSHOT</version>


   <dependencies>
```

```
        <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>4.3.10.RELEASE</version>
        </dependency>

  </dependencies>

</project>
```

GO-GO PARCEL

CourierBO.java

```
package com.spring.bo;

import com.spring.model.Courier;            //importing the packages and subpackages required
import java.util.Map;
public class CourierBO  {

        public double calculateCourierCharge(Courier cObj,String city) {

                double courierCharge=0.0;
                //fill the code
                courierCharge = cObj.getWeight() * cObj.getChargePerKg();
                Map<String,Float> data = cObj.getServiceCharge().getLocationServiceCharge();
                if(data.containsKey(city)){
                   courierCharge += data.get(city);     //setting as counter
                }

           return courierCharge;
        }

}
```

InvalidParcelWeightException.java

```
package com.spring.exception;


public class InvalidParcelWeightException extends Exception {


	public InvalidParcelWeightException(String msg) {
		super(msg);		//argumentizing the string to superclass
		//fill the code
	}


}
```

Driver.java

```
package com.spring.main;


import com.spring.exception.InvalidParcelWeightException;	//importing packages and subpackages required
import com.spring.service.CourierService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.util.Scanner;
public class Driver {


	public static void main(String[] args) {


		//fill the code


		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the courier ID:");
		int courierId = Integer.parseInt(sc.nextLine());
		System.out.println("Enter the total weight of parcel:");
		int weight = Integer.parseInt(sc.nextLine());
		System.out.println("Enter the city:");
		String city = sc.nextLine();


		ApplicationContext context=new ClassPathXmlApplicationContext("beans.xml");
```

```
		CourierService courierService=(CourierService)context.getBean("courierService");
		//catching the unexpected exceptions using try catch block
		try{
		System.out.println("Total Courier Charge: " +
courierService.calculateCourierCharge(courierId,weight,city));
		}
		catch(InvalidParcelWeightException e){
		   System.out.println(e.getMessage());
		}

	}

}
```

Courier.java

```
package com.spring.model;

public class Courier {

	private int courierId;
	private int weight;
	private float chargePerKg;
	private ServiceChargeInfo serviceCharge;

	public ServiceChargeInfo getServiceCharge() {
		return serviceCharge;
	}
	public void setServiceCharge(ServiceChargeInfo serviceCharge) {
		this.serviceCharge = serviceCharge;
	}
	public int getCourierId() {
		return courierId;
	}
	public void setCourierId(int courierId) {
		this.courierId = courierId;
```

```
	}
	public int getWeight() {
		return weight;
	}
	public void setWeight(int weight) {
		this.weight = weight;
	}

	public float getChargePerKg() {
		return chargePerKg;
	}
	public void setChargePerKg(float chargePerKg) {
		this.chargePerKg = chargePerKg;
	}

}
```

ServiceChargeInfo.java

```
package com.spring.model;

import java.util.Map;

public class ServiceChargeInfo {

	private Map<String,Float> locationServiceCharge;

	public Map<String, Float> getLocationServiceCharge() {
		return locationServiceCharge;
	}

	public void setLocationServiceCharge(Map<String, Float> locationServiceCharge) {
		this.locationServiceCharge = locationServiceCharge;
	}
}
```

CourierService.java

```
package com.spring.service;

import com.spring.bo.CourierBO;     //importing packages and subpackages required
import com.spring.exception.InvalidParcelWeightException;
import com.spring.model.Courier;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class CourierService {

        private CourierBO cBoObj;

        public CourierBO getcBoObj() {
                return cBoObj;
        }

        public void setcBoObj(CourierBO cBoObj) {
                this.cBoObj = cBoObj;
        }

        public double calculateCourierCharge(int courierId,int weight,String city)throws
InvalidParcelWeightException  {

                double courierCharge=0.0;
                //fill your code
                if(weight>0 && weight<1000)
                {
                   ApplicationContext context=new ClassPathXmlApplicationContext("beans.xml");
                   Courier cObj=(Courier)context.getBean("courier");
                   cObj.setCourierId(courierId);
                   cObj.setWeight(weight);
                   courierCharge = cBoObj.calculateCourierCharge(cObj, city);
                }
```

```
                else
                {
                    throw new InvalidParcelWeightException("Invalid Parcel Weight");
                }
                return courierCharge;
        }

}
```

Beans.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
        xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:context="http://www.springframework.org/schema/context"
        xmlns:jee="http://www.springframework.org/schema/jee"
xmlns:tx="http://www.springframework.org/schema/tx"
        xmlns:task="http://www.springframework.org/schema/task"
        xsi:schemaLocation="http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.2.xsd http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-
context-3.2.xsd http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd http://www.springframework.org/schema/task
http://www.springframework.org/schema/task/spring-task-3.2.xsd">
   <context:property-placeholder location="classpath:charges.properties" />



   <bean id="courierBoObj" class="com.spring.bo.CourierBO" />
   <bean id="courierService" class="com.spring.service.CourierService">
      <property name="cBoObj" ref="courierBoObj" />

   </bean>



   <bean id="courier" class="com.spring.model.Courier">
      <property name="courierId" value="123"/>
```

```
    <property name="weight" value="45"/>
    <property name="chargePerKg" value="${chargePerKg}"></property>
    <property name="serviceCharge">
      <bean class="com.spring.model.ServiceChargeInfo">
    <property name="locationServiceCharge">
      <map>
        <entry key="Coimbatore" value="200.0"/>
        <entry key="Chennai" value="300.0"/>
        <entry key="Madurai" value="150.0"/>
      </map>
    </property>
  </bean>
    </property>

  </bean>
</beans>
```

Charges.properties

```
chargePerKg=20.0
```

POM.XML

```
<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">
 <modelVersion>4.0.0</modelVersion>
 <groupId>GoGoParcelService</groupId>
 <artifactId>GoGoParcelService</artifactId>
 <version>0.0.1-SNAPSHOT</version>

  <dependencies>

        <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>4.3.10.RELEASE</version>
        </dependency>
```

```
  </dependencies>

</project>
```

LEXA KIWI SHOWROOM

Automobile.java

```
package com.spring.model;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class Automobile {

	private String brandName;
	private String brandModel;
	private Showroom showroomDetails;
	private String carColour;
	private String carFuelType;

	@Autowired
	public Automobile(Showroom showroomDetails) {
		super();
		this.showroomDetails = showroomDetails;
	}
	public String getBrandName() {
		return brandName;
	}
	public void setBrandName(String brandName) {
		this.brandName = brandName;
	}
	public String getBrandModel() {
		return brandModel;
	}
```

```
	public void setBrandModel(String brandModel) {
		this.brandModel = brandModel;
	}
	public Showroom getShowroomDetails() {
		return showroomDetails;
	}
	public void setShowroomDetails(Showroom showroomDetails) {
		this.showroomDetails = showroomDetails;
	}
	public String getCarColour() {
		return carColour;
	}
	public void setCarColour(String carColour) {
		this.carColour = carColour;
	}
	public String getCarFuelType() {
		return carFuelType;
	}
	public void setCarFuelType(String carFuelType) {
		this.carFuelType = carFuelType;
	}

}
```

Showroom.java

```
package com.spring.model;

import java.util.Map;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class Showroom {
	@Value("Lexa Kiwi")
```

```
	private String showroomName;
	@Value("Melbourne")
	private String showroomLocation;

	@Value("#($(carPriceDetails : (Climber: '429000',Dusture: '857000',"
			+ "Triber: '1503000',Elantra: '910000',Sonata: '3750000')))")
	private Map<String,Double> carPriceDetails;

	public String getShowroomName() {
		return showroomName;
	}
	public void setShowroomName(String showroomName) {
		this.showroomName = showroomName;
	}
	public String getShowroomLocation() {
		return showroomLocation;
	}
	public void setShowroomLocation(String showroomLocation) {
		this.showroomLocation = showroomLocation;
	}
	public Map<String, Double> getCarPriceDetails() {
		return carPriceDetails;
	}
	public void setCarPriceDetails(Map<String, Double> carPriceDetails) {
		this.carPriceDetails = carPriceDetails;
	}

}
```

AutomobileService.java

```
package com.spring.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
```

```
import org.springframework.stereotype.Component;

import com.spring.bo.AutomobileBO;
import com.spring.config.ApplicationConfig;
import com.spring.exception.InvalidFuelTypeException;
import com.spring.model.Automobile;

@Component
public class AutomobileService {

	private AutomobileBO automobileBOObj;

	@Autowired
	public AutomobileService(AutomobileBO automobileBOObj) {
		super();
		this.automobileBOObj = automobileBOObj;
	}

	public AutomobileBO getAutomobileBOObj() {
		return automobileBOObj;
	}

	public void setAutomobileBOObj(AutomobileBO automobileBOObj) {
		this.automobileBOObj = automobileBOObj;
	}


	public double calculateOnRoadPrice(String brandName,String brandModel,String carColour,String
carFuelType,String carName)
			throws InvalidFuelTypeException {
		double amount=0;
		// fill the code

		String s1 = "Petrol";
```

```
		String s2 = "Diesel";

		if(carFuelType.equals(s1) || carFuelType.equals(s2)) {

			@SuppressWarnings("resource")
			ConfigurableApplicationContext context = new
AnnotationConfigApplicationContext(ApplicationConfig.class);

			Automobile am =(Automobile)context.getBean(Automobile.class);

			am.setBrandName(brandName);
			am.setBrandModel(brandModel);
			am.setCarColour(carColour);
			am.setCarFuelType(carFuelType);
			amount = automobileBOObj.calculateOnRoadPrice(am, carName);
		}
		else {
			throw new InvalidFuelTypeException("Fuel Type is Not Valid");
		}
		return amount;
	}

}
```

AutomobileBO.java

```
package com.spring.bo;

import java.util.Map;

import org.springframework.stereotype.Component;

import com.spring.model.Automobile;

@Component
public class AutomobileBO {
```

```
	public double calculateOnRoadPrice(Automobile carObj,String carName) {

		double amount=0;
		// fill the code

		Map<String, Double> map = carObj.getShowroomDetails().getCarPriceDetails();
		if(map.containsKey(carName)) {
			double d = map.get(carName);
			amount = d + 38200 + ((d*3)/100);
		}
		return amount;
	}

}
```

ApplicationConfig.java

```
package com.spring.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.stereotype.Component;

import com.spring.bo.AutomobileBO;
import com.spring.model.Automobile;
import com.spring.model.Showroom;
import com.spring.service.AutomobileService;

// fill the code
@ComponentScan
@Configuration
@Component
public class ApplicationConfig {
```

```
	@Bean
	public Showroom showroom() {
		return new Showroom();
	}

	@Bean
	public Automobile automobile() {
		return new Automobile(showroom());
	}

	@Bean
	public AutomobileBO automobileBO() {
		return new AutomobileBO();
	}

	public AutomobileService automobileService() {
		return new AutomobileService(automobileBO());

	}
}
```

InvalidFuelTypeException.java

```
package com.spring.exception;

public class InvalidFuelTypeException extends Exception {

	public InvalidFuelTypeException(String msg) {

		// fill the code
		super(msg);
	}

}
```

Driver.java

```
package com.spring.main;

import java.util.Scanner;

import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.stereotype.Component;

import com.spring.config.ApplicationConfig;
import com.spring.exception.InvalidFuelTypeException;
import com.spring.service.AutomobileService;

@Component
public class Driver {

	public static void main(String[] args) {

		Scanner in=new Scanner(System.in);
		// fill the code
		System.out.println("Enter the brand Name");
		String brandName = in.nextLine();
		System.out.println("Enter the brand Model");
		String brandModel = in.nextLine();
		System.out.println("Enter the preferred car colour");
		String carColour = in.nextLine();
		System.out.println("Enter the preferred fuel type");
		String carFuelType = in.nextLine();
		System.out.println("Enter the car Name");
		String carName = in.nextLine();

		@SuppressWarnings("resource")
		ConfigurableApplicationContext context = new
AnnotationConfigApplicationContext(ApplicationConfig.class);
```

```
AutomobileService as = (AutomobileService) context.getBean(AutomobileService.class);

try {
double d = as.calculateOnRoadPrice(brandName, brandModel, carColour, carFuelType, carName);
System.out.println("Estimated on road price for the preferred car is: "+d);
}
catch(InvalidFuelTypeException e) {
System.out.println(e.getMessage());
}

}

}
```

POM.XML

```
<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">
  <modelVersion>4.0.0</modelVersion>
  <groupId>LexaKiwiShowroom</groupId>
  <artifactId>LexaKiwiShowroom</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <properties>
  <maven.compiler.target>1.8</maven.compiler.target>
  <maven.compiler.source>1.8</maven.compiler.source>
  <testing.version>6.14.3</testing.version>
  </properties>

   <dependencies>
```

```
    <!-- https://mvnrepository.com/artifact/org.springframwork/spring.context -->
        <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.2.RELEASE</version>
        </dependency>


  </dependencies>


  <build>
  <plugins>


  <plugin>
  <groupId>org.apache.maven.plugin</groupId>
  <artifactId>maven-surefire-report-plugin</artifactId>
  <version>3.0.0-M4</version>
  </plugin>

  </plugins>
  </build>

</project>
```

STATIONARY BILL

Application.properties

```
#Don't change property names, you can change value

notebook.CostPerQuantity=100
pen.CostPerQuantity=50
pencil.CostPerQuantity=10
notebook.quantity=7
```

```
pen.quantity=10

pencil.quantity=1
```

Spring.xml

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

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

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

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

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

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

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

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



        <context:property-placeholder

                location="classpath:application.properties" />


  <bean id="notebook" class="com.cts.stationarybill.vo.Item" >

    <property name="itemName" value="Notebook"/>

    <property name="itemCostPerQuantity" value="${notebook.CostPerQuantity}"/>

    <property name="quantity" value="${notebook.quantity}"/>

  </bean>


  <bean id="pen" class="com.cts.stationarybill.vo.Item" >

    <property name="itemName" value="Pen"/>

    <property name="itemCostPerQuantity" value="${pen.CostPerQuantity}"/>

    <property name="quantity" value="${pen.quantity}"/>

  </bean>


  <bean id="pencil" class="com.cts.stationarybill.vo.Item" >

    <property name="itemName" value="Pencil"/>

    <property name="itemCostPerQuantity" value="${pencil.CostPerQuantity}"/>

    <property name="quantity" value="${pencil.quantity}"/>

  </bean>
```

```
  <bean id="myCart" class="com.cts.stationarybill.vo.MyCart">
    <property name="items">
      <list>
        <ref bean="notebook"/>
        <ref bean="pen"/>
        <ref bean="pencil"/>
      </list>
    </property>
  </bean>

  <bean id="cartBo" class="com.cts.stationarybill.bo.CartBO">
    <constructor-arg ref="myCart"/>
  </bean>



  <bean id="cartService" class="com.cts.stationarybill.service.CartService">
  <property name="cartBo" ref="cartBo" />
  </bean>
 </beans>
```

CartBO.java

```
package com.cts.stationarybill.bo;

import com.cts.stationarybill.vo.Item;
import com.cts.stationarybill.vo.MyCart;

import java.util.*;

public class CartBO {
	private MyCart cart;

	public CartBO() {
		// TODO Auto-generated constructor stub
	}
	public CartBO(MyCart cart) {
```

```
            super();
            this.cart = cart;
    }

    public MyCart getCart() {
            return cart;
    }
    public void setCart(MyCart cart) {
            this.cart = cart;
    }




    public double calculateBillAfterDiscount(double totalOrderBill) {

List<Item> items= cart.getItems();

double nbprice=0.0;
double penprice=0.0;
for(Item i : items)
            {
              if(i.getItemName().equals("Notebook"))
                nbprice=i.getItemCostPerQuantity();
              if(i.getItemName().equals("Pen"))
                penprice=i.getItemCostPerQuantity();
            }

int notebooks= countNoteBook();
int pens= countPen();

double disc=0.0;
            if(totalOrderBill>500)
            {
              if(notebooks>=3)
```

```
                disc+=(notebooks*0.1*nbprice);
            if(pens>=10)
                disc+=(pens*0.2*penprice);

        }

        return (totalOrderBill-disc);  //TODO, change this value
}

public int countNoteBook() {
    List<Item> items= cart.getItems();

        int count = 0;
        for(Item i : items)
        {
                if(i.getItemName().equals("Notebook"))
            {
                count=i.getQuantity();
                break;
            }
        }

        return count;  //TODO, change this value
}

public int countPen() {
    List<Item> items= cart.getItems();

        int count = 0;
        for(Item i : items)
        {
            if(i.getItemName().equals("Pen"))
            {
                count=i.getQuantity();
```

```
                    break;
                }
            }

            return count;  //TODO, change this value
        }
}
```

InvalidCostPerQuantityException.java

```
package com.cts.stationarybill.exception;

public class InvalidCostPerQuantityException extends Exception{

        /**
         *
         */
        private static final long serialVersionUID = 1L;

        public InvalidCostPerQuantityException(String message) {
                super(message);
        }
}
```

Main.java

```
package com.cts.stationarybill.main;

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

import com.cts.stationarybill.exception.InvalidCostPerQuantityException;
import com.cts.stationarybill.service.CartService;
import com.cts.stationarybill.skeletonvalidator.SkeletonValidator;

public class Main
{
   public static void main( String[] args )
```

```
  {

                SkeletonValidator validator = new SkeletonValidator();

        @SuppressWarnings("resource")
        ApplicationContext ctx=new ClassPathXmlApplicationContext("spring.xml");

        CartService service=(CartService)ctx.getBean("cartService");

        try
        {
                double totalBill=service.calculateOrderTotalBill();
        System.out.println("Your total bill is = "+totalBill);

        }
        catch(InvalidCostPerQuantityException e)
        {
                System.out.println(e.getMessage());

        }
  }
}
```

CartService.java

```
package com.cts.stationarybill.service;



import java.util.List;

import com.cts.stationarybill.bo.CartBO;
import com.cts.stationarybill.exception.InvalidCostPerQuantityException;
import com.cts.stationarybill.vo.Item;

public class CartService {
        private CartBO cartBo;
```

```
	public CartService() {
		// TODO Auto-generated constructor stub
	}



	public CartService(CartBO cartBo) {
		super();
		this.cartBo = cartBo;
	}

	public CartBO getCartBo() {
		return cartBo;
	}

	public void setCartBo(CartBO cartBo) {
		this.cartBo = cartBo;
	}
	/**
	 * Method to calculate total bill
	 *
	 * @param MyCart
	 * @return totalOrderBill
	 */
	public double calculateOrderTotalBill() throws InvalidCostPerQuantityException {
		double totalOrderBill = 0;
	   List<Item> items = cartBo.getCart().getItems();

if(validateCostPerQuantity())
{
   for(Item i:items)
      totalOrderBill+=i.getItemCostPerQuantity()*i.getQuantity();
```

```
        totalOrderBill=cartBo.calculateBillAfterDiscount(totalOrderBill);

        return totalOrderBill;
    }
    else
    {
        throw new InvalidCostPerQuantityException("Cost per quantity must be positive non zero value");
    }

        }

        /**
         * Method to validate cost per quantity and throws exception if validation failes
         *
         * @exception InvalidCostPerQuantityException
         */
        public  boolean validateCostPerQuantity() throws InvalidCostPerQuantityException{

        List<Item> items = cartBo.getCart().getItems();

    for(Item i:items)
    {
        if(i.getItemCostPerQuantity()<1)
            throw new InvalidCostPerQuantityException("Cost per quantity must be positive non zero value");
    }

        return true;
        }
}
```

SkeletonValidator.java

```
package com.cts.stationarybill.skeletonvalidator;

import java.lang.reflect.Method;
import java.util.logging.Level;
```

```
import java.util.logging.Logger;

/**
 * @author t-aarti3
 *       This class is used to verify if the Code Skeleton is intact and not
 *       modified by participants thereby ensuring smooth auto evaluation
 * */

public class SkeletonValidator {
	public SkeletonValidator() {
		validateClassName("com.cts.stationarybill.service.CartService");
		validateClassName("com.cts.stationarybill.vo.Item");
		validateClassName("com.cts.stationarybill.vo.MyCart");
		validateClassName("com.cts.stationarybill.bo.CartBO");

		validateMethodSignature(
				"calculateOrderTotalBill:double,validateCostperQuantity:boolean",
				"com.cts.stationarybill.service.CartService");
		validateMethodSignature(
				"calculateBillAfterDiscount:double,countNoteBook:int,countPen:int",
				"com.cts.stationarybill.bo.CartBO");

	}
	private static final Logger LOG = Logger.getLogger("SkeletonValidator");
	protected final boolean validateClassName(String className) {

		boolean iscorrect = false;
		try {
			Class.forName(className);
			iscorrect = true;
			LOG.info("Class Name " + className + " is correct");

		} catch (ClassNotFoundException e) {
```

```
                LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the
correct package "
                        + "and class name as provided in the skeleton");


        } catch (Exception e) {
                LOG.log(Level.SEVERE,
                        "There is an error in validating the " + "Class Name. Please manually verify
that the "
                                + "Class name is same as skeleton before uploading");
        }
        return iscorrect;
    }

    protected final void validateMethodSignature(String methodWithExcptn, String className) {
        Class cls = null;
        try {

                String[] actualmethods = methodWithExcptn.split(",");
                boolean errorFlag = false;
                String[] methodSignature;
                String methodName = null;
                String returnType = null;

                for (String singleMethod : actualmethods) {
                        boolean foundMethod = false;
                        methodSignature = singleMethod.split(":");

                        methodName = methodSignature[0];
                        returnType = methodSignature[1];
                        cls = Class.forName(className);
                        Method[] methods = cls.getMethods();
                        for (Method findMethod : methods) {
                                if (methodName.equals(findMethod.getName())) {
                                        foundMethod = true;
                                        if (!(findMethod.getReturnType().getName().equals(returnType))) {
```

```
                                                errorFlag = true;
                                                LOG.log(Level.SEVERE, " You have changed the " + "return
type in '" + methodName
                                                        + "' method. Please stick to the " + "skeleton
provided");

                                        } else {
                                                LOG.info("Method signature of " + methodName + " is
valid");
                                        }

                                }
                        }
                        if (!foundMethod) {
                                errorFlag = true;
                                LOG.log(Level.SEVERE, " Unable to find the given public method " +
methodName
                                                + ". Do not change the " + "given public method name. " +
"Verify it with the skeleton");
                        }

                }
                if (!errorFlag) {
                        LOG.info("Method signature is valid");
                }

        } catch (Exception e) {
                LOG.log(Level.SEVERE,
                                " There is an error in validating the " + "method structure. Please manually
verify that the "
                                                + "Method signature is same as the skeleton before
uploading");
        }
    }

}
```

Item.java

```
package com.cts.stationarybill.vo;


public class Item {
	// member variables
		private String itemName;
		private double itemCostPerQuantity;
		private int quantity;
		public Item() {
			// TODO Auto-generated constructor stub
		}
		// Parameterized Constructor
		public Item(String itemName, double itemCostPerQuantity, int quantity) {
			super();
			this.itemName = itemName;
			this.itemCostPerQuantity = itemCostPerQuantity;
			this.quantity = quantity;
		}

		// getter and setter methods

		/**
		 * @return the itemName
		 */
		public String getItemName() {
			return itemName;
		}

		/**
		 * @param itemName
		 *	the itemName to set
		 */
		public void setItemName(String itemName) {
			this.itemName = itemName;
```

```
}

/**
 * @return the itemCostPerQuantity
 */
public double getItemCostPerQuantity() {
	return itemCostPerQuantity;
}

/**
 * @param itemCostPerQuantity
 *          the itemCostPerQuantity to set
 */
public void setItemCostPerQuantity(double itemCostPerQuantity) {
	this.itemCostPerQuantity = itemCostPerQuantity;
}

/**
 * @return the quantity
 */
public int getQuantity() {
	return quantity;
}

/**
 * @param quantity
 *          the quantity to set
 */
public void setQuantity(int quantity) {
	this.quantity = quantity;
}
@Override
public String toString() {
```

```
		return "Item [itemName=" + itemName + ", itemCostPerQuantity=" + itemCostPerQuantity + ", quantity="
+ quantity + "]";
	}



}
```

MyCart.java

```
package com.cts.stationarybill.vo;


import java.util.List;


public class MyCart {
private List<Item> items;
public MyCart() {
	// TODO Auto-generated constructor stub
}
public MyCart(List<Item> items) {
	super();
	this.items = items;
}
public List<Item> getItems() {
	return items;
}
public void setItems(List<Item> items) {
	this.items = items;
}


}
```

POM.XML

```
<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">
	<modelVersion>4.0.0</modelVersion>
```

```
        <groupId>com.cts.stationerybill</groupId>
        <artifactId>RoughStationeryBill</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>StationeryBill</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <maven.compiler.source>1.8</maven.compiler.source>
                <maven.compiler.target>1.8</maven.compiler.target>
                <junit.jupiter.version>5.4.0</junit.jupiter.version>
        </properties>

        <dependencies>

  <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
   <version>5.2.0.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine -->
                <dependency>
                        <groupId>org.junit.jupiter</groupId>
                        <artifactId>junit-jupiter-engine</artifactId>
                        <version>${junit.jupiter.version}</version>
                        <scope>test</scope>
                </dependency>
                <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
                <dependency>
                        <groupId>org.junit.jupiter</groupId>
```

```
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit.jupiter.version}</version>
            <scope>test</scope>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.junit.platform/junit-platform-launcher -->
        <dependency>
            <groupId>org.junit.platform</groupId>
            <artifactId>junit-platform-launcher</artifactId>
            <version>1.5.1</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.junit.platform</groupId>
            <artifactId>junit-platform-commons</artifactId>
            <version>1.5.1</version>
        </dependency>

    </dependencies>
    <build>
<plugins>
<plugin>
<artifactId>maven-surefire-plugin</artifactId>
<version>2.22.1</version>
</plugin>
</plugins>
</build>
    <reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-report-plugin</artifactId>
      <version>2.19.1</version>
```

```
        </plugin>
      </plugins>
    </reporting>
</project>
```

COLLEGE ELECTION

totalVotes.properties

```
totalVotes=300
```

Application.java

```
package com.spring.config;

import java.util.ArrayList;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;

import com.spring.model.Candidate;
//Use appropriate annotation
@Configuration
@ComponentScan(basePackages = "com.spring")
@PropertySource("totalVotes.properties")
public class ApplicationConfig {

        @Bean
        public ArrayList<Candidate> getVoteList() {
                ArrayList<Candidate> voteList = new ArrayList<Candidate>();
                voteList.add(new Candidate("Rahul", "RA102021", "BBA", 110));
                voteList.add(new Candidate("Pavithra", "RA222021", "BSC", 97));
                voteList.add(new Candidate("Jerom", "RA332021", "BBA", 78));
                return voteList;
        }
}
```

ElectionDAO.java

```
package com.spring.dao;


import java.util.ArrayList;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;


import com.spring.model.Candidate;


//Use appropriate annotation
@Component
public class ElectionDAO {
	@Autowired
	private ArrayList<Candidate> voteList = new ArrayList<Candidate>();


	public Candidate candidateWithMaximumVote() {
		int max = 0;
		Candidate withMaxVote = null;
		for (Candidate candidate : voteList) {
			if (candidate.getNumberOfVotes() > max) {
				withMaxVote = candidate;
				max = candidate.getNumberOfVotes();
			}
		}
		return withMaxVote;
	}


	public int getTotalVotes() {
		return Candidate.getTotalVotes();
	}


	public int unpolledVotes() {
		return getTotalVotes() - totalpolledVotes();
```

```
	}

	public int totalpolledVotes() {
		int total = 0;
		for (Candidate candidate : voteList) {
			total += candidate.getNumberOfVotes();
		}
		return total;
	}

	public ElectionDAO() {
	}

	public ArrayList<Candidate> getVoteList() {
		return voteList;
	}

	public void setVoteList(ArrayList<Candidate> voteList) {
		this.voteList = voteList;
	}
}
```

Driver.java

```
package com.spring.main;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.spring.config.ApplicationConfig;
import com.spring.model.Candidate;
import com.spring.service.ElectionService;

public class Driver {

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

```
		ApplicationContext context = new AnnotationConfigApplicationContext(ApplicationConfig.class);
		ElectionService service = context.getBean(ElectionService.class);
		System.out.println("College Election Report");
		System.out.println("Total Number of Votes:" + service.getTotalVotes());
		System.out.println("Total Number of polled Votes:" + service.totalpolledVotes());
		System.out.println("Total Number of unpolled Votes:" + service.unpolledVotes());
		for (Candidate candidate : service.getVoteList()) {
			System.out.println(candidate);
		}
		Candidate winner = service.candidateWithMaximumVote();
		System.out.println("Election won by " + winner.getName() + " with " + winner.getNumberOfVotes() +
" votes");
	}

}
```

Candidate.java

```
package com.spring.model;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

//POJO class
//Use appropriate annotation
@Component
public class Candidate {

	private String name;
	private String candidateId;
	private String department;
	private int numberOfVotes;
	private static int totalVotes;

	public Candidate(String name, String candidateId, String department, int numberOfVotes) {
		super();
```

```
        this.name = name;
        this.candidateId = candidateId;
        this.department = department;
        this.numberOfVotes = numberOfVotes;
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public String getCandidateId() {
        return candidateId;
}

public void setCandidateId(String candidateId) {
        this.candidateId = candidateId;
}

public static int getTotalVotes() {
        return totalVotes;
}

@Value("${totalVotes}")
public void setTotalVotes(int totalVotes) {
        Candidate.totalVotes = totalVotes;
}

public String getDepartment() {
        return department;
}
```

```
	public void setDepartment(String department) {
		this.department = department;
	}

	public int getNumberOfVotes() {
		return numberOfVotes;
	}

	public void setNumberOfVotes(int numberOfVotes) {
		this.numberOfVotes = numberOfVotes;
	}

	public Candidate() {
	}

	@Override
	public String toString() {
		return "Candidate " + this.name + " got " + this.numberOfVotes + " votes";
	}
}
```

ElectionService.java

```
package com.spring.service;

import java.util.ArrayList;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import com.spring.dao.ElectionDAO;
import com.spring.model.Candidate;

//Use appropriate annotation
@Component
```

```
public class ElectionService {

	private ElectionDAO electionDAO;

	@Autowired
	public ElectionService(ElectionDAO electionDAO) {
		super();
		this.electionDAO = electionDAO;
	}

	public Candidate candidateWithMaximumVote() {
		return electionDAO.candidateWithMaximumVote();
	}

	public int unpolledVotes() {
		return electionDAO.unpolledVotes();
	}

	public int totalpolledVotes() {
		return electionDAO.totalpolledVotes();
	}

	public int getTotalVotes() {
		return electionDAO.getTotalVotes();
	}

	public ArrayList<Candidate> getVoteList() {
		return electionDAO.getVoteList();
	}

	public void setElectionDAO(ElectionDAO electionDAO) {
		this.electionDAO = electionDAO;
	}
```

```
	public ElectionDAO getElectionDAO() {
		return electionDAO;
	}
}
```

POM.XML

```
<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">
  <modelVersion>4.0.0</modelVersion>
  <groupId>CollegeElection</groupId>
  <artifactId>CollegeElection</artifactId>
  <version>0.0.1-SNAPSHOT</version>
   <dependencies>
	<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>4.3.10.RELEASE</version>
	</dependency>

	<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
	<dependency>
    <groupId>xmlunit</groupId>
    <artifactId>xmlunit</artifactId>
    <version>1.6</version>
    <scope>test</scope>
</dependency>


	<!-- https://mvnrepository.com/artifact/com.github.stefanbirkner/system-rules -->
<dependency>
```

```
    <groupId>com.github.stefanbirkner</groupId>
    <artifactId>system-rules</artifactId>
    <version>1.16.0</version>
    <scope>test</scope>
</dependency>

<dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-impl</artifactId>
            <version>2.1.13</version>
</dependency>
<!-- Thanks for using https://jar-download.com -->

<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.1</version>
</dependency>

            </dependencies>

             <build>
        <plugins>
     <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.0.0-M1</version>
    <configuration>
        <testFailureIgnore>true</testFailureIgnore>
    </configuration>



</plugin>
```

```
</plugins>
</build>

  <properties>
   <maven.compiler.source>1.8</maven.compiler.source>
   <maven.compiler.target>1.8</maven.compiler.target>
</properties>

</project>
```

EASY RECHARGE

Beans.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
        xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:context="http://www.springframework.org/schema/context"
        xmlns:jee="http://www.springframework.org/schema/jee"
xmlns:tx="http://www.springframework.org/schema/tx"
        xmlns:task="http://www.springframework.org/schema/task"
        xsi:schemaLocation="http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.2.xsd http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-
context-3.2.xsd http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd http://www.springframework.org/schema/task
http://www.springframework.org/schema/task/spring-task-3.2.xsd">

        <context:property-placeholder location="classpath:cashBack.properties" />
        <!-- fill the code -->

        <bean id="recharge" class="com.spring.model.Recharge">
          <property name="rechargePlan">
```

```
                <bean id="rechargeplanobj" class="com.spring.model.RechargePlan">
                <property name="rechargePlans">
                <map>
                <entry key="199" value="1.5GB data,28 days validity"/>
                <entry key="399" value="2GB data,58 days validity"/>
                <entry key="550" value="2.5GB data,85 days validity"/>
                <entry key="51" value="6GB data,6 days validity"/>
                <entry key="750" value="4GB data,90 days validity"/>
                </map>
              </property>
            </bean>
              </property>
              <property name="cashBackPercent" value="${cashBackPercent}" />
            </bean>
            <bean id="rechargeBoObj" class="com.spring.bo.RechargeBO"/>

            <bean id="rechargeservice" class="com.spring.service.RechargeService">
              <property name="rechargeBO" ref="rechargeBoObj"/>
            </bean>

</beans>
```

cashBack.properties

```
cashBackPercent=5
```

RechargeBO.java

```
package com.spring.bo;

import com.spring.model.Recharge;
import com.spring.model.RechargePlan;
import java.util.*;
public class RechargeBO {

        public double processRecharge(Recharge recharge){
            double amount=0;
            // fill the code
```

```
            double rechargeamount=recharge.getRechargeAmount();
            RechargePlan rechargeplanobj=recharge.getRechargePlan();
            Map<Double, String> rechargeplans=rechargeplanobj.getRechargePlans();
            boolean flag=false;
            for(Double findamount:rechargeplans.keySet())
            {
              if(findamount==rechargeamount)
              {
                flag=true;
                break;
              }
            }
            if(flag==true)
            {
            amount=(rechargeamount*recharge.getCashBackPercent())/100;
            }
            return amount;
        }
}
```

InvalidRechargePackException.java

```
package com.spring.exception;


public class InvalidRechargePackException extends Exception {
        public InvalidRechargePackException(String msg) {
        //fill the code
        super(msg);
        }
}
```

Driver.java

```
package com.spring.main;
import java.util.*;
import com.spring.service.RechargeService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
import com.spring.exception.InvalidRechargePackException;


public class Driver {
        public static void main(String[] args) {
          Scanner sc=new Scanner(System.in);
          System.out.println("Enter the Recharge Id:");
          String rechargeId=sc.next();
          System.out.println("Enter the Operator:");
          String operator=sc.next();
          System.out.println("Enter the Customer PhoneNumber:");
          long phonenumber=sc.nextLong();
          System.out.println("Enter the Recharge Amount:");
          double rechargeamount=sc.nextDouble();
          ApplicationContext ctx=new ClassPathXmlApplicationContext("beans.xml");
                RechargeService rechargeserviceobj=(RechargeService) ctx.getBean("rechargeservice");
                double amount=0;
                try
                {
                amount=rechargeserviceobj.processRecharge(rechargeId,operator,phonenumber,rechargeamount);
                }
                catch(InvalidRechargePackException exc)
                {
                  System.out.println(exc.getMessage());
                }
                if(amount!=0)
                {
                  System.out.println("Recharge of "+rechargeamount+" is successful. You got a cash back of
RS:"+amount);
                }
        }


}
```

Recharge.java

```
package com.spring.model;
```

```
//pojo class with required attributes,getters and setters
public class Recharge {

	private String rechargeId;
	private long phoneNumber;
	private double rechargeAmount;
	private RechargePlan rechargePlan;
   private int cashBackPercent;
	public int getCashBackPercent() {
		return cashBackPercent;
	}

	public void setCashBackPercent(int cashBackPercent) {

		this.cashBackPercent = cashBackPercent;
	}

	public String getRechargeId() {
		return rechargeId;
	}

	public void setRechargeId(String rechargeId) {
		this.rechargeId = rechargeId;
	}

	public RechargePlan getRechargePlan() {
		return rechargePlan;
	}

	public void setRechargePlan(RechargePlan rechargePlan) {
		this.rechargePlan = rechargePlan;
	}
```

```
	public long getPhoneNumber() {
		return phoneNumber;
	}

	public void setPhoneNumber(long phoneNumber) {
		this.phoneNumber = phoneNumber;
	}



	public double getRechargeAmount() {
		return rechargeAmount;
	}

	public void setRechargeAmount(double rechargeAmount) {
		this.rechargeAmount = rechargeAmount;
	}



}
```

RechargePlan.java

```
package com.spring.model;



import java.util.Map;

//pojo class with required attributes,getters and setters

public class RechargePlan {

private String operator;
private  Map<Double, String> rechargePlans;

public Map<Double, String> getRechargePlans() {
	return rechargePlans;
```

```
}

public void setRechargePlans(Map<Double, String> rechargePlans) {
	this.rechargePlans = rechargePlans;
}
public String getOperator() {
	return operator;
}
public void setOperator(String operator) {
	this.operator = operator;
}



}
```

RechargeService.java

```
package com.spring.service;

import com.spring.bo.RechargeBO;
import com.spring.model.Recharge;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.spring.exception.InvalidRechargePackException;

public class RechargeService {

	private RechargeBO rechargeBO;

	public RechargeBO getRechargeBO() {
		return rechargeBO;
	}

	public void setRechargeBO(RechargeBO rechargeBO) {
		this.rechargeBO = rechargeBO;
	}
```

```
	public double processRecharge(String rechargeId, String operator,long phoneNumber,double
rechargeAmount) throws InvalidRechargePackException{
		double amount=0;
		ApplicationContext ctx=new ClassPathXmlApplicationContext("beans.xml");
		Recharge rechargeobj=(Recharge) ctx.getBean("recharge");
		rechargeobj.setRechargeId(rechargeId);
		rechargeobj.setPhoneNumber(phoneNumber);
		rechargeobj.setRechargeAmount(rechargeAmount);
		amount=rechargeBO.processRecharge(rechargeobj);
		if(amount==0)
		{
		   throw new InvalidRechargePackException("Recharge Pack not available");
		}
		return amount;
	}
}
```

POM.XML

```
<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">
  <modelVersion>4.0.0</modelVersion>

  <groupId>EasyRecharge</groupId>
  <artifactId>EasyRecharge</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>EasyRecharge</name>
  <url>http://maven.apache.org</url>

  <dependencies>
	<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
```

```
  <version>4.3.10.RELEASE</version>
        </dependency>

        <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
        <dependency>
  <groupId>xmlunit</groupId>
  <artifactId>xmlunit</artifactId>
  <version>1.6</version>
  <scope>test</scope>
</dependency>



        <!-- https://mvnrepository.com/artifact/com.github.stefanbirkner/system-rules -->
<dependency>
  <groupId>com.github.stefanbirkner</groupId>
  <artifactId>system-rules</artifactId>
  <version>1.16.0</version>
  <scope>test</scope>
</dependency>

<dependency>
        <groupId>com.sun.xml.bind</groupId>
        <artifactId>jaxb-impl</artifactId>
        <version>2.1.13</version>
</dependency>
<!-- Thanks for using https://jar-download.com -->

<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency>
```

```
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.1</version>
</dependency>

            </dependencies>

             <build>
        <plugins>
     <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.0.0-M1</version>
    <configuration>
       <testFailureIgnore>true</testFailureIgnore>
    </configuration>

   </plugin>

</plugins>
</build>

   <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
</project>
```

HOSTEL MANAGEMENT

feesDiscountDetails.properties

```
feesDiscount.map={'O':'50', 'D':'40', 'A':'30', 'B':'20'}
```

StudentBO.java

```
package com.spring.bo;
```

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.stereotype.Component;

import com.spring.config.ApplicationConfig;
import com.spring.model.FeesDiscountDetails;
import com.spring.model.Student;

//use appropriate annotation to make this class as component class
@Component
public class StudentBO {

	public float calculateFee(Student obj) {
		ApplicationContext context = new AnnotationConfigApplicationContext(ApplicationConfig.class);
		FeesDiscountDetails fd = context.getBean(FeesDiscountDetails.class);
		float fee = obj.getHostel().getRent();
		if (fd.getFeesDiscount().keySet().contains(Character.toString(obj.getGrade())))
			fee -= (fee * fd.getFeesDiscount().get(Character.toString(obj.getGrade())) / 100);
		return fee;
	}

}
```

ApplicationConfig.java

```
package com.spring.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;

import com.spring.bo.StudentBO;
import com.spring.model.FeesDiscountDetails;
import com.spring.model.Hostel;
import com.spring.model.Student;
```

```
import com.spring.service.StudentService;

@Configuration
@PropertySource("feesDiscountDetails.properties")
@ComponentScan
public class ApplicationConfig {
	@Bean
	public FeesDiscountDetails feesDiscountDetails() {
		FeesDiscountDetails feesDiscountDetails = new FeesDiscountDetails();
		return feesDiscountDetails;
	}

	@Bean
	public Hostel hostel() {
		Hostel hostel = new Hostel();
		return hostel;
	}

	@Bean
	public Student student() {
		Student student = new Student(hostel());
		return student;
	}

	@Bean
	public StudentBO studentBO() {
		StudentBO studentBO = new StudentBO();
		return studentBO;
	}

	@Bean
	public StudentService studentService() {
		StudentService studentService = new StudentService(studentBO());
		return studentService;
```

```
	}

}
```

InvalidGradeException.java

```
package com.spring.exception;

public class InvalidGradeException extends Exception {

	public InvalidGradeException(String msg) {
		super(msg);
	}

}
```

Driver.java

```
package com.spring.main;

import java.util.Scanner;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.spring.config.ApplicationConfig;
import com.spring.exception.InvalidGradeException;
import com.spring.service.StudentService;

public class Driver {

	public static void main(String[] args) {
		ApplicationContext context = new AnnotationConfigApplicationContext(ApplicationConfig.class);
		StudentService studentService = context.getBean(StudentService.class);
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the student name:");
		String name = sc.nextLine();
		System.out.println("Enter the admission number:");
```

```
		String admissionNo = sc.nextLine();
		System.out.println("Enter the grade:");
		char grade = sc.nextLine().charAt(0);
		System.out.println("Enter the hostel details");
		System.out.println("Enter the hostel name:");
		String hostelName = sc.nextLine();
		System.out.println("Enter the room rent:");
		float rent = sc.nextFloat();
		try {
			System.out.println(
								"Hostel fee is Rs:" + studentService.calculateFee(name, admissionNo, grade,
hostelName, rent));
		} catch (InvalidGradeException e) {
			System.out.println(e.getMessage());
		}
	}

}
```

FeesDiscountDetails.java

```
package com.spring.model;

import java.util.Map;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

//pojo class with required attributes,getters and setters
//use appropriate annotation to make this class as component class
@Component
public class FeesDiscountDetails {

	@Value("#{${feesDiscount.map}}")
	private Map<String, Integer> feesDiscount;
```

```
	public Map<String, Integer> getFeesDiscount() {
		return feesDiscount;
	}

	public void setFeesDiscount(Map<String, Integer> feesDiscount) {
		this.feesDiscount = feesDiscount;
	}

	@Override
	public String toString() {
		return "FeesDiscountDetails [feesDiscount=" + feesDiscount + "]";
	}

}
```

Hostel.java

```
package com.spring.model;

import org.springframework.stereotype.Component;

//pojo class with required attributes,getters and setters
//use appropriate annotation to make this class as component class
@Component
public class Hostel {

	private String hostelName;
	private float rent;

	public String getHostelName() {
		return hostelName;
	}
	public void setHostelName(String hostelName) {
		this.hostelName = hostelName;
	}
	public float getRent() {
```

```
		return rent;
	}
	public void setRent(float rent) {
		this.rent = rent;
	}
	@Override
	public String toString() {
		return "Hostel [hostelName=" + hostelName + ", rent=" + rent + "]";
	}



}
```

Student.java

```
package com.spring.model;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

//pojo class with required attributes,getters and setters
//use appropriate annotation to make this class as component class
@Component
public class Student {

	private String name;
	private String admissionNo;
	private char grade;
	private Hostel hostel;

	@Autowired
	public Student(Hostel hostel) {
		super();
		this.hostel = hostel;
	}
```

```
public Hostel getHostel() {
        return hostel;
}

public void setHostel(Hostel hostel) {
        this.hostel = hostel;
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public String getAdmissionNo() {
        return admissionNo;
}

public void setAdmissionNo(String admissionNo) {
        this.admissionNo = admissionNo;
}

public char getGrade() {
        return grade;
}

public void setGrade(char grade) {
        this.grade = grade;
}

@Override
public String toString() {
```

```
		return "Student [name=" + name + ", admissionNo=" + admissionNo + ", grade=" + grade + ", hostel=" + hostel
+ "]";
	}

}
```

StudentService.java

```
package com.spring.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.stereotype.Component;

import com.spring.bo.StudentBO;
import com.spring.config.ApplicationConfig;
import com.spring.exception.InvalidGradeException;
import com.spring.model.Student;

//use appropriate annotation to make this class as component class
@Component
public class StudentService {

	public StudentBO getStudentBOObj() {
		return studentBOObj;
	}

	public void setStudentBOObj(StudentBO studentBOObj) {
		this.studentBOObj = studentBOObj;
	}

	private StudentBO studentBOObj;

	// fill the code
```

```
	@Autowired
	public StudentService(StudentBO studentBOObj) {
		super();
		this.studentBOObj = studentBOObj;
	}

	public float calculateFee(String name, String admissionNo, char grade, String hostelName, float rent)
			throws InvalidGradeException {
		String validChars = "ODABCF";
		if (!validChars.contains(Character.toString(grade))) {
			throw new InvalidGradeException("Invalid Grade");
		}
		ApplicationContext context = new AnnotationConfigApplicationContext(ApplicationConfig.class);
		Student student = context.getBean(Student.class);
		student.setName(name);
		student.setAdmissionNo(admissionNo);
		student.setGrade(grade);
		student.getHostel().setHostelName(hostelName);
		student.getHostel().setRent(rent);
		return this.studentBOObj.calculateFee(student);
	}

}
```

POM.XML

```
<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">
  <modelVersion>4.0.0</modelVersion>
  <groupId>HostelManagement</groupId>
  <artifactId>HostelManagement</artifactId>
  <version>0.0.1-SNAPSHOT</version>
   <dependencies>
	<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
```

```
    <version>4.3.10.RELEASE</version>
            </dependency>

            <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
            <dependency>
    <groupId>xmlunit</groupId>
    <artifactId>xmlunit</artifactId>
    <version>1.6</version>
    <scope>test</scope>
</dependency>



            <!-- https://mvnrepository.com/artifact/com.github.stefanbirkner/system-rules -->
<dependency>
    <groupId>com.github.stefanbirkner</groupId>
    <artifactId>system-rules</artifactId>
    <version>1.16.0</version>
    <scope>test</scope>
</dependency>

<dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-impl</artifactId>
            <version>2.1.13</version>
</dependency>
<!-- Thanks for using https://jar-download.com -->

<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency>
```

```
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.1</version>
</dependency>

            </dependencies>

             <build>
        <plugins>
     <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.0.0-M1</version>
    <configuration>
        <testFailureIgnore>true</testFailureIgnore>
    </configuration>



</plugin>




</plugins>
</build>

   <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>

</project>
```

NAKSHATRA FASHION

couponCode.properties

```
couponCodes.map={'MAX':'40', 'EPIC':'30', 'GRAB':'60', 'MARK':'55'}
```

ShipmentBO.java

```
package com.spring.bo;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.stereotype.Component;

import com.spring.config.ApplicationConfig;
import com.spring.model.CouponUtility;
import com.spring.model.Shipment;

//use appropriate annotation to make this class as component class
@Component
public class ShipmentBO {

	public double processDiscount(Shipment shipmentObj) {
		double amount = 0;
		// fill the code
		ApplicationContext ctx = new AnnotationConfigApplicationContext(ApplicationConfig.class);
		CouponUtility util = ctx.getBean(CouponUtility.class);

		int couponCode = util.getCouponCodes().get(shipmentObj.getCouponCode());
		double price = shipmentObj.getItem().getPrice();

		amount = price - (price * couponCode) / 100.0;

		return amount;
	}

}
```

ApplicationConfig.java

```
package com.spring.config;


import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;


//Use appropriate annotation
@Configuration
@ComponentScan("com.spring.*")
@PropertySource("couponCode.properties")
public class ApplicationConfig {



}
```

InvalidCouponCodeException.java

```
package com.spring.exception;


public class InvalidCouponCodeException extends Exception {


	public InvalidCouponCodeException(String msg) {
		// fill the code
		super(msg);
	}


}
```

Driver.java

```
package com.spring.main;


import java.util.Scanner;


import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
```

```
import com.spring.config.ApplicationConfig;
import com.spring.exception.InvalidCouponCodeException;
import com.spring.service.ShipmentService;

public class Driver {

	public static void main(String[] args) {
		// fill the code
		ApplicationContext ctx = new AnnotationConfigApplicationContext(ApplicationConfig.class);
		Scanner sc = new Scanner(System.in);

		System.out.println("Enter the Item Id:");
		String itemId = sc.nextLine();
		System.out.println("Enter the Item Type:");
		String itemType = sc.nextLine();
		System.out.println("Enter the Price:");
		double price = Double.parseDouble(sc.nextLine());

		System.out.println("Enter the Shipment Details");
		System.out.println("Enter the Shipment Id:");
		String shipId = sc.nextLine();
		System.out.println("Enter the Shipment Date:");
		String shipDate = sc.nextLine();
		System.out.println("Enter the Shipment Address:");
		String shipAddress = sc.nextLine();
		System.out.println("Enter the Coupon code:");
		String couponCode = sc.nextLine();
		try {
			ShipmentService service = ctx.getBean(ShipmentService.class);

			double amount = service.processDiscount(shipId, shipDate, itemId, itemType, price,
shipAddress, couponCode);
			System.out.print("Discounted Amount: " + amount);
		} catch (InvalidCouponCodeException e) {
```

```
                System.out.println(e.getMessage());
            }
        }

}
```

CouponUtility.java

```
package com.spring.model;

import java.util.Map;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

//pojo class with required attributes,getters and setters
//use appropriate annotation to make this class as component class
@Component
public class CouponUtility {
        public Map<String, Integer> getCouponCodes() {
                return couponCodes;
        }

        public void setCouponCodes(Map<String, Integer> couponCodes) {
                this.couponCodes = couponCodes;
        }

        // Fill the code
        @Value(value = "#{${couponCodes.map}}")
        private Map<String, Integer> couponCodes;
}
```

Item.java

```
package com.spring.model;

import org.springframework.stereotype.Component;
```

```
//pojo class with required attributes,getters and setters
//use appropriate annotation to make this class as component class
@Component
public class Item {

	private String itemId;
	private String itemType;
	private double price;

	public String getItemId() {
		return itemId;
	}

	public void setItemId(String itemId) {
		this.itemId = itemId;
	}

	public String getItemType() {
		return itemType;
	}

	public void setItemType(String itemType) {
		this.itemType = itemType;
	}

	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

}
```

Shipment.java

```
package com.spring.model;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

//pojo class with required attributes,getters and setters
//use appropriate annotation to make this class as component class
@Component
public class Shipment {

	private String shipmentId;
	private String shipmentDate;
	private Item item;
	private String deliveryAddress;
	private String couponCode;

	public String getCouponCode() {
		return couponCode;
	}

	public void setCouponCode(String couponCode) {
		this.couponCode = couponCode;
	}

	public String getDeliveryAddress() {
		return deliveryAddress;
	}

	public void setDeliveryAddress(String deliveryAddress) {
		this.deliveryAddress = deliveryAddress;
	}

	// fill the code
```

```
	@Autowired
	public Shipment(Item item) {
		super();
		this.item = item;
	}

	public String getShipmentId() {
		return shipmentId;
	}

	public void setShipmentId(String shipmentId) {
		this.shipmentId = shipmentId;
	}

	public String getShipmentDate() {
		return shipmentDate;
	}

	public void setShipmentDate(String shipmentDate) {
		this.shipmentDate = shipmentDate;
	}

	public Item getItem() {
		return item;
	}

	public void setItem(Item item) {
		this.item = item;
	}

}
```

ShipmentService.java

```
package com.spring.service;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.stereotype.Component;

import com.spring.bo.ShipmentBO;
import com.spring.config.ApplicationConfig;
import com.spring.exception.InvalidCouponCodeException;
import com.spring.model.CouponUtility;
import com.spring.model.Shipment;

//use appropriate annotation to make this class as component class
@Component
public class ShipmentService {

	private ShipmentBO shipmentBOObj;

	// fill the code
	@Autowired
	public ShipmentService(ShipmentBO shipmentBOObj) {
		super();
		this.shipmentBOObj = shipmentBOObj;
	}

	public ShipmentBO getShipmentBOObj() {
		return shipmentBOObj;
	}

	public void setShipmentBOObj(ShipmentBO shipmentBOObj) {
		this.shipmentBOObj = shipmentBOObj;
	}

	public double processDiscount(String shipmentId, String shipmentDate, String itemId, String itemType,
double price,
```

```
                String address, String couponCode) throws InvalidCouponCodeException {
        double amount = 0;
        // fill the code
        ApplicationContext ctx = new AnnotationConfigApplicationContext(ApplicationConfig.class);
        CouponUtility util = ctx.getBean(CouponUtility.class);

        if (!util.getCouponCodes().containsKey(couponCode))
                throw new InvalidCouponCodeException("Invalid Coupon Code");

        Shipment spmt = ctx.getBean(Shipment.class);
        ShipmentBO bo = ctx.getBean(ShipmentBO.class);

        spmt.setCouponCode(couponCode);
        spmt.setDeliveryAddress(address);
        spmt.setShipmentDate(shipmentDate);
        spmt.setShipmentId(shipmentId);
        spmt.getItem().setItemId(itemId);
        spmt.getItem().setItemType(itemType);
        spmt.getItem().setPrice(price);

        amount = bo.processDiscount(spmt);

        return amount;
    }

}
```

POM.XML

```
<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">
    <modelVersion>4.0.0</modelVersion>
    <groupId>NakshatraFashion</groupId>
    <artifactId>NakshatraFashion</artifactId>
    <version>0.0.1-SNAPSHOT</version>
```

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.3.10.RELEASE</version>
    </dependency>

    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>xmlunit</groupId>
        <artifactId>xmlunit</artifactId>
        <version>1.6</version>
        <scope>test</scope>
    </dependency>



    <!-- https://mvnrepository.com/artifact/com.github.stefanbirkner/system-rules -->
    <dependency>
        <groupId>com.github.stefanbirkner</groupId>
        <artifactId>system-rules</artifactId>
        <version>1.16.0</version>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>com.sun.xml.bind</groupId>
        <artifactId>jaxb-impl</artifactId>
        <version>2.1.13</version>
    </dependency>
```

```
        <!-- Thanks for using https://jar-download.com -->

        <!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
        <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
            <version>2.1</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.0.0-M1</version>
                <configuration>
                    <testFailureIgnore>true</testFailureIgnore>
                </configuration>



            </plugin>




        </plugins>
    </build>
    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

</project>
```

PINK FTINESS FRANCHISE

franchisePercent.properties

```
percentage=12.5
```

Beans.xml

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

	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"

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

	xmlns:jee="http://www.springframework.org/schema/jee"
xmlns:tx="http://www.springframework.org/schema/tx"

	xmlns:task="http://www.springframework.org/schema/task"

	xsi:schemaLocation="http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.2.xsd http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-
context-3.2.xsd http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-3.2.xsd http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd http://www.springframework.org/schema/task
http://www.springframework.org/schema/task/spring-task-3.2.xsd">


	<context:property-placeholder location="classpath:FranchisePercent.properties" />


	<!-- fill the code -->

	<bean name="fitnessFranchise" class="com.spring.model.FitnessFranchise" scope="prototype">

		<constructor-arg name="percentage" value="${percentage}" />

		<constructor-arg name="commonExpenses">

			<map>

				<entry key="Rent" value="12000.0"/>

				<entry key="HouseKeeping" value="4000.0"/>

				<entry key="ElectricityCharges" value="2000.0"/>

				<entry key="Maintenence" value="3000.0"/>

			</map>

		</constructor-arg>

	</bean>

	<bean name="fitnessService" class="com.spring.service.FitnessService" scope="prototype" >

		<constructor-arg ref="fitnessBO"/>
```

```
	</bean>
	<bean name="fitnessBO" class="com.spring.bo.FitnessBO" scope="prototype"/>





</beans>
```

FitnessBO.java

```
package com.spring.bo;


import com.spring.model.FitnessFranchise;


public class FitnessBO {


	public double calculateNetProfit(FitnessFranchise franchise) {


		//// Calculating netProfit based on factors


		double expense = franchise.getTotalIncome() - (franchise.getTrainerSalary()
				+ franchise.getCommonExpenses().get("Rent") +
franchise.getCommonExpenses().get("HouseKeeping")
				+ franchise.getCommonExpenses().get("ElectricityCharges")
				+ franchise.getCommonExpenses().get("Maintenence"));


		return expense;


	}
}
```

NoProfitException.java

```
package com.spring.exception;


public class NoProfitException extends Exception {


	public NoProfitException(String msg) {
```

```
		//fill the code
		super(msg);
	}
}
```

Driver.java

```
package com.spring.main;

import java.util.*;

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

import com.spring.exception.NoProfitException;
import com.spring.model.FitnessFranchise;
import com.spring.service.FitnessService;

public class Driver {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		// fill the code
		Scanner sc = new Scanner(System.in);
		ApplicationContext applicationContext = new ClassPathXmlApplicationContext("beans.xml");

		System.out.println("Pink Fitness Franchise 1 details");
		System.out.println("Enter the total location:");
		String location = sc.next();
		System.out.println("Enter the total income:");
		double totalIncome = sc.nextDouble();
		System.out.println("Enter total expense amount for trainer salary:");
		double trainerSalary = sc.nextDouble();
		FitnessFranchise fitnessFranchise1 = (FitnessFranchise)
applicationContext.getBean("fitnessFranchise");
		fitnessFranchise1.setLocation(location);
```

```
fitnessFranchise1.setTotalIncome(totalIncome);
fitnessFranchise1.setTrainerSalary(trainerSalary);
System.out.println("Pink Fitness Franchise 2 details");
System.out.println("Enter the total location:");
String location2 = sc.next();
System.out.println("Enter the total income:");
double totalIncome2 = sc.nextDouble();
System.out.println("Enter total expense amount for trainer salary:");
double trainerSalary2 = sc.nextDouble();
FitnessFranchise fitnessFranchise2 = (FitnessFranchise)
applicationContext.getBean("fitnessFranchise");
fitnessFranchise2.setLocation(location2);
fitnessFranchise2.setTotalIncome(totalIncome2);
fitnessFranchise2.setTrainerSalary(trainerSalary2);

FitnessService fitnessService = (FitnessService) applicationContext.getBean("fitnessService");
try {

    fitnessService.calculateNetProfit(fitnessFranchise1);
    System.out.println("Pink Fitness at " + fitnessFranchise1.getLocation() + " franchise Amount
is RS:"
            + fitnessFranchise1.getFranchiseAmount());
} catch (NoProfitException e) {
    System.out.println(e.getMessage());
}
try {

    fitnessService.calculateNetProfit(fitnessFranchise2);
    System.out.println("Pink Fitness at " + fitnessFranchise2.getLocation() + " franchise Amount
is RS:"
            + fitnessFranchise2.getFranchiseAmount());
} catch (NoProfitException e) {
    System.out.println(e.getMessage());
}
```

```
	}

}
```

FitnessFranchise.java

```
package com.spring.model;

import java.util.Map;
// implement the FitnessFranchisor interface
public class FitnessFranchise implements FitnessFranchisor {
	private String location;
	private double totalIncome;
	private double trainerSalary;
	private double franchiseAmount;
	private double percentage;
	private Map<String, Double> commonExpenses;

	public FitnessFranchise(double percentage, Map<String, Double> commonExpenses) {
		super();
		this.percentage = percentage;
		this.commonExpenses = commonExpenses;
	}

	public void setPercentage(double percentage) {
		this.percentage = percentage;
	}

	public double getPercentage() {
		return percentage;
	}

	public double getFranchiseAmount() {
		return franchiseAmount;
	}
```

```
public Map<String, Double> getCommonExpenses() {
	return commonExpenses;
}

public void setCommonExpenses(Map<String, Double> commonExpenses) {
	this.commonExpenses = commonExpenses;
}

public void setFranchiseAmount(double franchiseAmount) {
	this.franchiseAmount = franchiseAmount;
}


public String getLocation() {
	return location;
}

public void setLocation(String location) {
	this.location = location;
}


public double getTotalIncome() {
	return totalIncome;
}

public void setTotalIncome(double totalIncome) {
	this.totalIncome = totalIncome;
}


public double getTrainerSalary() {
	return trainerSalary;
}
```

```
	public void setTrainerSalary(double trainerSalary) {
		this.trainerSalary = trainerSalary;
	}

	public void calculateFranchiseAmount(double netProfit) {

	//Calculating Franchise Amount
		this.franchiseAmount=netProfit*(percentage)/100;

	}
}
```

FitnessFranchisor.java

```
package com.spring.model;

public interface FitnessFranchisor {

 public void calculateFranchiseAmount(double netProfit);

}
```

FitnessService.java

```
package com.spring.service;

import com.spring.bo.FitnessBO;
import com.spring.exception.NoProfitException;
import com.spring.model.FitnessFranchise;

public class FitnessService {

	private FitnessBO fitnessBO;


	public FitnessBO getFitnessBO() {
		return fitnessBO;
```

```
	}

	public void setFitnessBO(FitnessBO fitnessBO) {
		this.fitnessBO = fitnessBO;
	}
	public void calculateNetProfit(FitnessFranchise franchise) throws NoProfitException {
		//fill the code

		double netProfit=fitnessBO.calculateNetProfit(franchise);

		if(netProfit<0)
		{
			throw new NoProfitException("No profit to calculate franchise amount");
		}
		else
		{
			franchise.calculateFranchiseAmount(netProfit);
		}
	}

	public FitnessService(FitnessBO fitnessBO) {
		super();
		this.fitnessBO = fitnessBO;
	}




}
```

POM.XML

```
<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">
  <modelVersion>4.0.0</modelVersion>
  <groupId>PinkFitness</groupId>
  <artifactId>PinkFitness</artifactId>
```

```
  <version>0.0.1-SNAPSHOT</version>
   <dependencies>
            <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>4.3.10.RELEASE</version>
            </dependency>

            <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
            <dependency>
    <groupId>xmlunit</groupId>
    <artifactId>xmlunit</artifactId>
    <version>1.6</version>
    <scope>test</scope>
</dependency>


            <!-- https://mvnrepository.com/artifact/com.github.stefanbirkner/system-rules -->
<dependency>
    <groupId>com.github.stefanbirkner</groupId>
    <artifactId>system-rules</artifactId>
    <version>1.16.0</version>
    <scope>test</scope>
</dependency>

<dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-impl</artifactId>
            <version>2.1.13</version>
```

```
</dependency>
<!-- Thanks for using https://jar-download.com -->

<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
<dependency>
   <groupId>javax.xml.bind</groupId>
   <artifactId>jaxb-api</artifactId>
   <version>2.1</version>
</dependency>

            </dependencies>

             <build>
        <plugins>
    <plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-surefire-plugin</artifactId>
   <version>3.0.0-M1</version>
   <configuration>
       <testFailureIgnore>true</testFailureIgnore>
   </configuration>

  </plugin>

</plugins>
</build>

  <properties>
   <maven.compiler.source>1.8</maven.compiler.source>
   <maven.compiler.target>1.8</maven.compiler.target>
</properties>
</project>
```

CarStore Application

CarStoreApplication

```
package com.cts.carstore;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.cts.carstore.skeletonvalidator.SkeletonValidator;

@SpringBootApplication

public class CarStoreApplication {

	public static void main(String[] args) {

		SpringApplication.run(CarStoreApplication.class, args);

		new SkeletonValidator();

	}

}
```

CarStoreController

```
package com.cts.carstore.controller;

import java.util.ArrayList;

import java.util.List;

import javax.validation.Valid;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.validation.Validator;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PostMapping;

import com.cts.carstore.exception.ApplicationException;
```

```
import com.cts.carstore.model.Car;

import com.cts.carstore.model.CarSearch;

import com.cts.carstore.service.CarStoreService;

@Controller

public class CarStoreController {

	@Autowired

	private CarStoreService service;

	@Autowired

	private Validator validator;

	public CarStoreController() {

		System.out.println("in default constr of controller");

	}

	  @Autowired

	  public CarStoreController(CarStoreService service, Validator validator) {

	    this.service=service;

	    this.validator=validator;

	  }

	@GetMapping("/showCarSearchForm")

	public String showCarSearchForm(@ModelAttribute("carSearch") CarSearch carSearch) {

		//Add code here..

		carSearch=new CarSearch();

		return "carSearch"; //TODO, modify return value

	}

@PostMapping("/getCarSearchResultPage")

public String getCarSearchResultForm(@Valid @ModelAttribute("carSearch") CarSearch carSearch,
BindingResult result,ModelMap map) throws ApplicationException  {

		//Add code here..

		if(result.hasErrors())
```

```
        return "carSearch"; //TODO, modify return value
        List<Car> carList=service.getCarSearchResult(carSearch);
        map.addAttribute("carList", carList);
        return "carSearchResult";
}
@ModelAttribute("cities")
public List<String> populateCities() {
        List<String> cities = new ArrayList<String>();
        //cities.add("select");
         cities.add("Chennai");
          cities.add("Mumbai");
          cities.add("Delhi");
          cities.add("Bangalore");
          cities.add("Pune");
          cities.add("Kolkatta");
        return cities;
}
@ModelAttribute("brands")
public List<String> populateBrands() {
        List<String> brands = new ArrayList<String>();
          brands.add("Maruti Suzuki");
          brands.add("Honda");
          brands.add("Mahindra");
          brands.add("Toyota");
          brands.add("Hundai");
        return brands;
}
```

```
        @ModelAttribute("budgetUpto")
        public List<String> populateBudget() {
                List<String> budgetUpto = new ArrayList<String>();
                 budgetUpto.add("Below 3 Lakh");
                  budgetUpto.add("3 Lakh");
                  budgetUpto.add("5 Lakh");
                  budgetUpto.add("7 Lakh");
                  budgetUpto.add("10 Lakh");
                return budgetUpto;
        }
        @ModelAttribute("fuelTypes")
        public List<String> populateFuelTypes() {
                List<String> fuelTypes = new ArrayList<String>();
                 fuelTypes.add("Petrol");
                  fuelTypes.add("Diesel");
                  fuelTypes.add("CNG");
                return fuelTypes;
        }
}
```

ApplicationException

```
package com.cts.carstore.exception;
public class ApplicationException extends Exception {
        private static final long serialVersionUID = -9079454849611061074L;
        public String errorMessage;
}
```

ExceptionHandlerControllerAdvice

```
package com.cts.carstore.exception;

import java.time.LocalDateTime;

import javax.servlet.http.HttpServletRequest;

import org.springframework.http.HttpStatus;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.servlet.ModelAndView;

@ControllerAdvice
public class ExceptionHandlerControllerAdvice {

	@ExceptionHandler(ApplicationException.class)
	@ResponseStatus(value = HttpStatus.INTERNAL_SERVER_ERROR)
	public ModelAndView handleResourceNotFound(final ApplicationException
exception, final HttpServletRequest request, final Model model) {

//Add code here..
ModelAndView mv=new ModelAndView();
mv.setViewName("error");
mv.addObject("errorMessage","Low Budget-No car available below 3 lakh price");
//add attribute error and timestamp
mv.addObject("errorTime",LocalDateTime.now());

return mv; //TODO, change return value
	}
}
```

Car

```
package com.cts.carstore.model;

public class Car {
	private String brandName;
	private String modelName;
	private double price;   //price in lakh
	private String fuelType;
	private double mileage;
	private int seatingCapacity;

	public Car() {
		// TODO Auto-generated constructor stub
	}

	public Car(String brandName, String modelName, double price, String
fuelType,double mileage, int seatingCapacity) {
```

```
        super();
        this.brandName = brandName;
        this.modelName = modelName;
        this.price = price;
        this.fuelType = fuelType;
        this.mileage = mileage;
        this.seatingCapacity = seatingCapacity;
    }

    public String getBrandName() {
        return brandName;
    }

    public void setBrandName(String brandName) {
        this.brandName = brandName;
    }

    public String getModelName() {
        return modelName;
    }

    public void setModelName(String modelName) {

        this.modelName = modelName;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public String getFuelType() {
        return fuelType;
    }

    public void setFuelType(String fuelType) {
        this.fuelType = fuelType;
    }

    public int getSeatingCapacity() {
        return seatingCapacity;
    }

    public void setSeatingCapacity(int seatingCapacity) {
        this.seatingCapacity = seatingCapacity;
    }

    public double getMileage() {
        return mileage;
    }

    public void setMileage(double mileage) {
```

```
		this.mileage = mileage;
	}

}
```

CarSearch

```
package com.cts.carstore.model;

import javax.validation.constraints.*;

public class CarSearch {
	// Use validation annotations as per the requirement
	@NotEmpty(message="Customer name is required")
	private String customerName;

	@NotEmpty(message="Gender is required")
	private String gender;

	@NotEmpty(message="Mobile number is required")
	@Pattern(regexp="[7-9]{1}[0-9]{9}",message="Mobile number should be 10 digits
and starting with digit 7/8/9")
	private String mobileNumber;

	private String customerCity;
	private String brand;
	private String budgetUpto;
	private String fuelType;

	public CarSearch() {
		// TODO Auto-generated constructor stub
	}

	public String getCustomerName() {
		return customerName;
	}

	public void setCustomerName(String customerName) {
		this.customerName = customerName;
	}
	public String getGender() {
		return gender;
	}

	public void setGender(String gender) {
		this.gender = gender;
	}

	public String getMobileNumber() {

		return mobileNumber;
	}
```

```java
	public void setMobileNumber(String mobileNumber) {
		this.mobileNumber = mobileNumber;
	}

	public String getCustomerCity() {
		return customerCity;
	}

	public void setCustomerCity(String customerCity) {
		this.customerCity = customerCity;
	}

	public String getBrand() {
		return brand;
	}

	public void setBrand(String brand) {
		this.brand = brand;
	}

	public String getBudgetUpto() {
		return budgetUpto;
	}

	public void setBudgetUpto(String budgetUpto) {
		this.budgetUpto = budgetUpto;
	}

	public String getFuelType() {
		return fuelType;
	}

	public void setFuelType(String fuelType) {

		this.fuelType = fuelType;
	}

}
```

ErrorResponse

```java
package com.cts.carstore.model;
public class ErrorResponse {

	private String errorMessage;
	private String requestedURI;

	//add code here
}
```

CarStoreService

```
package com.cts.carstore.service;

import java.util.ArrayList;

import java.util.List;

import org.springframework.stereotype.Service;

import com.cts.carstore.exception.ApplicationException;

import com.cts.carstore.model.Car;

import com.cts.carstore.model.CarSearch;

@Service

public class CarStoreService {

	public List<Car> getCarSearchResult(CarSearch carSearch) throws ApplicationException {

		//Add code here..

		List<Car> list=new ArrayList<Car>();

		List<Car> available=buildCars();

		if(carSearch.getBudgetUpto().startsWith("Below"))

			throw new ApplicationException();

		int budgetLimit=Integer.parseInt(carSearch.getBudgetUpto().substring(0,2).trim());

		for(Car car:available) {

		if(car.getBrandName().equalsIgnoreCase(carSearch.getBrand())&&

			car.getFuelType().equalsIgnoreCase(carSearch.getFuelType())

				&& car.getPrice()<=budgetLimit) {

				list.add(car);

			}

		}

		return list;  //TODO, modify this return value

	}

	// DO NOT CHANGE THIS METHOD
```

```
//DO NOT CHANGE CODE WITHIN METHOD
private List<Car> buildCars() {
        List<Car> cars = new ArrayList<Car>();
        // brand,modelName,price,fuelType,seatingCapacity
        Car car1 = new Car("Maruti Suzuki", "Swift", 5.20, "Petrol", 21.21, 5);
        Car car2 = new Car("Maruti Suzuki", "Alto", 3, "Petrol", 22.05, 5);
        Car car3 = new Car("Maruti Suzuki", "Ertiga LXi", 7.5, "Petrol", 19.01, 7);
        Car car4 = new Car("Maruti Suzuki", "Ertiga VXi", 8.9, "CNG", 26.2, 7);
        Car car5 = new Car("Maruti Suzuki", "WagonR", 4.5, "Petrol", 20.52, 5);
        Car car6 = new Car("Maruti Suzuki", "WagonR LXi", 5.20, "CNG", 32.52, 5);
        Car car7 = new Car("Honda", "Amaze", 6.20, "Petrol", 18.6, 5);
        Car car8 = new Car("Honda", "Amaze1.2E MT", 7.6, "Diesel", 24.7, 5);
        Car car9 = new Car("Honda", "Amaze1.5E MT", 8, "Diesel", 25.2, 5);
        Car car10 = new Car("Honda", "City", 9.30, "Petrol", 17.4, 5);
        Car car11 = new Car("Mahindra", "KUV100", 5.6, "Petrol", 15, 6);
        Car car12 = new Car("Toyota", "Yaris", 8.8, "Petrol", 18.1, 5);
        Car car13 = new Car("Hundai", "Aura", 7.30, "CNG", 20.5, 5);
        Car car14 = new Car("Hundai", "Creta", 9.9, "Petrol", 21.5, 5);
        cars.add(car1);
        cars.add(car2);
        cars.add(car3);
        cars.add(car4);
        cars.add(car5);
        cars.add(car6);
        cars.add(car7);
        cars.add(car8);
        cars.add(car9);
        cars.add(car10);
```

```
            cars.add(car11);

            cars.add(car12);

            cars.add(car13);

            cars.add(car14);

            return cars;

        }

}
```

Application.properties

```
server.port=3030
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

---

caeSearch.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
        <br>
        <br>
        <!-- Add code here.. -->
        <h1 id="heading">Windsor Car Showroom</h1>

        <h3 style="align:center">Search Cars</h3>
        <sf:form name="form" action="getCarSearchResultPage"
                                        modelAttribute="carSearch" method="post">
                <sf:label path="customerName">Customer Name:</sf:label>
                <sf:input path="customerName" id="customerName" />
                <sf:errors path="customerName" cssStyle="color:red;"/>
                <br />
                <sf:label path="mobileNumber">Mobile Number:</sf:label>
                <sf:input path="mobileNumber" id="mobileNumber" />
                <sf:errors path="mobileNumber" cssStyle="color:red;" />
                <br />
                <sf:label path="gender">Gender:</sf:label>
                <sf:radiobutton path="gender" value="Male" />Male
                <sf:radiobutton path="gender" value="Female" />Female
```

```
            <sf:errors path="gender" cssStyle="color:red;"/>
            <br />
            <sf:label path="customerCity">Customer City:</sf:label>
            <sf:select path="customerCity" id="customerCity">
                  <sf:options items="${cities}" />
            </sf:select>
            <br />
            <sf:label path="brand">Brand:</sf:label>
            <sf:select path="brand" id="brand">
                  <sf:options items="${brands}" />
            </sf:select>
            <br />
            <sf:label path="fuelType">Fuel Type:</sf:label>
            <sf:select path="fuelType" id="fuelType">
                  <sf:options items="${fuelTypes}" />
            </sf:select>
            <br />
            <sf:label path="budgetUpto">Budget Upto:</sf:label>
            <sf:select path="budgetUpto" id="budgetUpto">
                  <sf:options items="${budgetUpto}" />
            </sf:select>
            <br />
            <input type="submit" value="CarSearch" />
            <input type="reset" value="Clear" />
      </sf:form>

</body>
</html>
```

carSearchResult.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>

<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

      <br>
      <br>
      <!--Add code here  -->
      <sf:form id="searchResult">
      <c:if test="${carList.isEmpty()==false}">
            <h3>Here are cars matching your search criteria: </h3>
            <table border="1">
```

```
                    <tr>
                          <th>Brand Name</th>
                          <th>Model Name</th>
                          <th>Price in lakh</th>
                          <th>Fuel/Transmission</th>
                          <th>Mileage</th>
                          <th>Seating Capacity</th>
                    </tr>
                    <c:forEach items="${carList}" var="car">
                          <tr>
                                <td>${car.brandName}</td>
                                <td>${car.modelName}</td>
                                <td>${car.price}</td>
                                <td>${car.fuelType}</td>
                                <td>${car.mileage}</td>
                                <td>${car.seatingCapacity}</td>
                          </tr>
                    </c:forEach>
              </table>
      </c:if>
      <c:if test="${carList.isEmpty()==true}">
              <h3 id="noResult">Sorry,No car available matching your profile.</h3>
      </c:if>
      </sf:form>
      <br/>
      <a href="showCarSearchForm" id="searchCars">Search Cars</a>
</body>
</html>
```

Error.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1" isELIgnored="false" isErrorPage="true"%>

<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<!DOCTYPE HTML>
<html>

<head>
<title>Car Search Errors</title>
</head>

<body>
      <!--Add code here..  -->
      <sf:form id="error">
      <h3>Unable to retrieve car information. Below are the error details:</h3>
      <h3>${errorMessage}</h3>
      <h3>${errorTime}</h3>
      </sf:form>
```

```
        <br/>
        <a href="showCarSearchForm" id="searchCars">Search Cars</a>
</body>
</html>
```

SnapFitnessApplication

FitnessController

```
package com.controller;

import java.util.HashMap;

import java.util.Map;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.ModelAttribute;

import javax.validation.Valid;

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

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import com.model.FitnessCenter;

import com.service.FitnessService;

//use appropriate annotation to configure SnapFitnessController as Controller

@Controller

public class FitnessController {

//          Use appropriate annotation

   @Autowired

            private FitnessService service;

            @ModelAttribute("packageList")

             public Map<String, String> buildState() {

                        Map<String, String> pairs = new HashMap();

                        pairs.put("Aerobics","Aerobics");

                        pairs.put("Zumba","Zumba");

                        pairs.put("PersonalTraining","PersonalTraining");
```

```
                pairs.put("Yoga","Yoga");
                pairs.put("RegularGYM","RegularGYM");
                return pairs;
        }
        @GetMapping("/snapEnquiryPage")
         public String snapEnquiryPage( @ModelAttribute("snap") FitnessCenter fitnessCenter){
                 return "snapEnquiryPage";
         }
   @PostMapping("/packEstimation")
        public String calculatePackageCost(@Valid @ModelAttribute("snap")FitnessCenter fitness,
BindingResult result, ModelMap model) {
        if(result.hasErrors()){
            model.addAttribute("messege", "Number of months should be minimum one");
                        return "snapEnquiryPage";
                }
                model.addAttribute("cost",service.calculatePackageCost(fitness));
                return "snapEstimation";
        }
}
```

SnapFitnessApplication

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
@SpringBootApplication
@ComponentScan({"com.*"})
public class SnapFitnessApplication {
```

```
	public static void main(String[] args) {
		SpringApplication.run(SnapFitnessApplication.class, args);
	}
}
```

FitnessCenter

```
package com.model;
import javax.validation.constraints.Min;
import org.springframework.stereotype.Component;
//pojo class with required attributes,getters and setters
//use appropriate annotation to make this class as component class
@Component
public class FitnessCenter {
	private String preferredPackage;
	private double costPerMonth;
	// use appropriate annotation for validating numberOfMonths attribute
	@Min(value= 1,message="Number of months should be minimum one")
	private int numberOfMonths;
	public String getPreferredPackage() {
		return preferredPackage;
	}

	public void setPreferredPackage(String preferredPackage) {
		this.preferredPackage = preferredPackage;
	}
	public double getCostPerMonth() {
		return costPerMonth;
```

```
	}
	public void setCostPerMonth(double costPerMonth) {
		this.costPerMonth = costPerMonth;
	}
	public int getNumberOfMonths() {
		return numberOfMonths;
	}
	public void setNumberOfMonths(int numberOfMonths) {
		this.numberOfMonths = numberOfMonths;
	}
}
```

FitnessService

```
package com.service;
import com.model.FitnessCenter;
import org.springframework.stereotype.Service;
//use appropriate annotation to configure SnapFitnessService as a Service class
@Service
public class FitnessService {
	//calculate the Package cost and return the rent amount
	public double calculatePackageCost (FitnessCenter fitness) {
		double packageCost=0.0;
		// fill the code
		String serviceName = fitness.getPreferredPackage();
		int months = fitness.getNumberOfMonths();
		if(serviceName.equals("Aerobics"))
		{
```

```
                packageCost = months*1500.0;
            }
            else if(serviceName.equals("Zumba"))
            {
                packageCost = months*1750.0;
            }
            else if(serviceName.equals("RegularGYM"))
            {
                packageCost = months*1250.0;
            }
            else if(serviceName.equals("PersonalTraining"))
            {
                packageCost = months*2500.0;
            }
            else if(serviceName.equals("Yoga"))
            {
                packageCost = months*1300.0;
            }
            return packageCost;
        }
}
```

Application.properties

```
#Do not edit the key server.port.Port number can be changed
server.port=8088
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
```

snapEnquirypage.jsp

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1><center> Snap Fitness Center </center></h1>
    <form:form modelAttribute="snap" method="post" action="/packEstimation">
        <center>
      <table>
      <tr>
      <td><form:label path="preferredPackage">Select Package</form:label></td>

      <td><form:select path="preferredPackage" id="preferredPackage"
      name="preferredPackage" items="${packageList}" required="required" /></td>
      </tr>
      <tr>
<td><form:label path="numberOfMonths">Package duration (In months)</form:label></td>
<td><form:input path="numberOfMonths" id="numberOfMonths"
             name="numberOfMonths" required="required"/>

<label>${messege}</label>
      </td>
      </tr>
      <tr>
<td><input type="submit" id="submit(Name)" name="submit" value="Package Cost" /></td>
      <td></td>
      </tr>
            </table>
            </center>
      </form:form>
</body>
</html>
```

snapEstimationpage.jsp

```
<%@page isELIgnored="false" %>
<html>
<body bgcolor="lavender">
<h3>Welcome to Snap Fitness Center.Your preferred package cost is Rs.${cost}</h3>
</body>
</html>
</html>
```

DIGITALHOMEBROKER V2

BrokerController

```
package com.controller;

import javax.validation.Valid;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;


import com.model.Home;

import com.service.BrokerService;


//use appropriate annotation to configure BrokerController as Controller

@Controller

public class BrokerController {

	//Use appropriate annotation above this property

	@Autowired

	private BrokerService service;

	@RequestMapping(value = "/searchPage", method = RequestMethod.GET)

	public String searchPage(@ModelAttribute("homeBean") Home homeBean)	{

		return "searchPage";

	}

	//invoke the service class - searchHome method.
```

```
@RequestMapping(value = "/homeList", method = RequestMethod.POST)
	public String searchHome(@ModelAttribute("homeBean") @Valid Home homeBean,
BindingResult result, ModelMap model) {
		if (result.hasErrors()) {
			return "searchPage";
		}
		model.addAttribute("homes", service.searchHome(homeBean));
		return "homeListPage";
	}
}
```

DigitalHomeBrokerApplication

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan({"com.*"})
public class DigitalHomeBrokerApplication {
	public static void main(String[] args) {
		SpringApplication.run(DigitalHomeBrokerApplication.class, args);
	}
}
```

Home

```
package com.model;
import javax.validation.constraints.Min;
```

```
import org.springframework.stereotype.Component;
//pojo class with required attributes,getters and setters
@Component
public class Home {
	private String location;
	private String address;
	private String contactNumber;
	private String bhkType;
	private String contactPerson;
	@Min(value = 5000, message = "Minimum rent should be 5000")
	private double rent;
	public double getRent() {
		return rent;
	}
	public void setRent(double rent) {
		this.rent = rent;
	}
	public String getLocation() {
		return location;
	}
	public void setLocation(String location) {
		this.location = location;	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}
```

```
	public String getContactNumber() {
		return contactNumber;
	}
	public void setContactNumber(String contactNumber) {
		this.contactNumber = contactNumber;
	}
	public String getBhkType() {
		return bhkType;
	}
	public void setBhkType(String bhkType) {
		this.bhkType = bhkType;
	}
	public String getContactPerson() {
		return contactPerson;
	}
	public void setContactPerson(String contactPerson) {
		this.contactPerson = contactPerson;
	}

	public Home(String location, String address, String contactNumber, String bhkType, String
contactPerson,double rent) {
		super();
		this.location = location;
		this.address = address;
		this.contactNumber = contactNumber;
		this.bhkType = bhkType;
		this.contactPerson = contactPerson;
		this.rent=rent;
```

```
	}
	public Home(){}
}
```

BrokerService

```
package com.service;
import java.util.ArrayList;
import org.springframework.stereotype.Service;
import com.model.Home;
//use appropriate annotation to configure BrokerService as a Service
@Service
public class BrokerService {
	ArrayList<Home> list = new ArrayList<Home>();
	// search homes/apartments from the list and return it as a ArrayList
	public ArrayList<Home> searchHome(Home homeBean) {
		ArrayList<Home> resultList = new ArrayList<Home>();


for (Home home : list) {
if (home.getLocation().equalsIgnoreCase(homeBean.getLocation())	&&
home.getBhkType().equalsIgnoreCase(homeBean.getBhkType())	&&
home.getRent() == homeBean.getRent()) {
				resultList.add(home);
				}
		}
		return resultList;
	}
	public BrokerService() {
		addHomes();
```

```
        }

        // Dont modify this code

        public ArrayList<Home> addHomes() {

list.add(new Home("Coimbatore", "#412, Shipitha cascade, Ganapathy, PIN-641001", "9917678456",
"1BHK","Malavika", 8000));

list.add(new Home("Coimbatore", "#12, shardhalotus, Ram Nagar, PIN-641001", "9917100989", "2BHK",
"Karthik",12000));

list.add(new Home("Coimbatore", "#12, Anandha nilayam,Kuniamuthur, PIN-641001", "9917100989",
"2BHK", "Karthik",12000));

list.add(new Home("Coimbatore", "#201, Shipitha cascade, Saravanampatti, PIN-641001",
"9917123456", "3BHK","Sowparnika", 17000));

list.add(new Home("Chennai", "#34/45, RV Homes, 5th cross, Anssari street, PIN-600001",
"7788900989", "1BHK","Mukunthan", 6000));

list.add(new Home("Chennai", "#212/4, 10th cross, T-nagar, PIN-60008", "9917100989", "2BHK",
"Annesh", 14000));

list.add(new Home("Chennai", "#556/4, 5th cross, Thambaram, PIN-60010", "9917100989", "2BHK",
"Jisha", 14000));

list.add(new Home("Chennai", "#556/4,Nila paradise, Thambaram, PIN-60010", "9917100989", "2BHK",
"Jisha",14000));

list.add(new Home("Chennai", "#412/4, 10th cross, Velacheri, PIN-60009", "9917156989", "3BHK",
"Rakesh", 18000));

list.add(new Home("Bangalore", "#45, 12Th main , GM Palya, PIN-560075", "9917122119", "1BHK",
"Senthil", 11000));

list.add(new Home("Bangalore", "#67/2, 3rd cross , Indra nagar, PIN-560077", "9944550989", "2BHK",
"RamKumar",17000));

list.add(new Home("Bangalore", "#302, Sakthi enclave ,1st main , Malleshpalya, PIN-560075",
"9917668889","3BHK", "Saradha", 25000));

return list;

        }

}
```

Application.properties

```
#Do not edit the key server.port.Port number can be changed
#include other needed configurations
server.port=8088
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
```

HomeListPage.jsp

```
<%@page isELIgnored="false"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>
<body bgcolor="lavender">
	<h1>
		<center>Digital Home Broker</center>
	</h1>
	<h2>
		<center>Search homes|Apartments !!!</center>
	</h2>

	<c:if test="${fn:length(homes) == 0}">

<h2><center>No home/Apartment found for this search</center></h2>
	</c:if>

	<c:if test="${fn:length(homes) > 0}">
	<table>
		<thead>
			<tr>
				<td>Location</td>
				<td>BhkType</td>
				<td>Address</td>
				<td>ContactPerson</td>
				<td>ContactNumber</td>
			</tr>
		</thead>
		<tbody>
			<c:forEach items="${homes}" var="home">
				<tr>
					<td>${home.location}</td>
					<td>${home.bhkType}</td>
					<td>${home.address}</td>
					<td>${home.contactPerson}</td>
					<td>${home.contactNumber}</td>
				</tr>
			</c:forEach>
		</tbody>
	</table>
```

```
	</c:if>

</body>
</html>
```

Searchpage.jsp

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color: lavender">
	<h1>
		<center>Digital Home Broker</center>
	</h1>
	<h2>
		<center>Search homes|Apartments !!!</center>
	</h2>
	<center>
		<table>
		<form:form action="homeList" modelAttribute="homeBean" method="post">
		<tr>
	<td><form:label path="location">Search location</form:label></td>
	<td><form:input path="location" id="location" name="location" /></td>
	</tr>
	<tr>
	<td><form:label path="bhkType">Search BHK Type</form:label></td>
	<td><form:input path="bhkType" id="bhkType" name="bhkType" /></td>
	</tr>
	<tr>
	<td><form:label path="rent">Preferred Rent</form:label></td>
	<td><form:input path="rent" id="rent" name="rent" />
	<form:errors path="rent" />
	</td>
	</tr>
	<tr>
	<td><input type="submit" id="submit" name="submit" value="Search" /></td>
	</tr>
	</form:form>
	</table>
	</center>
</body>
</html>

--------------------------------------------------------------------------------
```

FlawlessAcademyApplication

AcademyController

```
package com.controller;

import java.util.HashMap;

import java.util.Map;

import javax.validation.Valid;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import com.model.Academy;

import com.service.AcademyService;

@Controller

public class AcademyController {

	@Autowired

	private AcademyService service;

	@RequestMapping(value = "/enrollmentPage", method = RequestMethod.GET)

	public String showPage(@ModelAttribute("academyBean") Academy academyBean) {

		System.out.println("in controller");

		return "enrollmentPage";

	}


	@ModelAttribute("programList")

	public  Map<String, String> buildState() {
```

```
		Map<String, String> progMap = new HashMap<String, String>();
		progMap.put("ClassicalDance", "ClassicalDance");
		progMap.put("KarnaticVocals", "KarnaticVocals");
		progMap.put("WesternDance","WesternDance");
		progMap.put("Drawing", "Drawing");
		progMap.put("Instruments","Instruments");
		return progMap;
	}
	@RequestMapping(value = "/progEstimation", method = RequestMethod.POST)
	public String calculateProgramCost(@ModelAttribute("academyBean") @Valid Academy
academyBean, BindingResult result,
		ModelMap model) {
		if (result.hasErrors()) {
			System.out.println("Error : " + result.toString());
			return "enrollmentPage";
		}
		double cost=service.calculateProgramCost(academyBean);
		model.addAttribute("cost", cost);
		return "estimationPage";
	}
}
```

FlawlessAcademyApplication

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
@SpringBootApplication
```

```
@ComponentScan({"com.*"})
public class FlawlessAcademyApplication {
	public static void main(String[] args) {
		SpringApplication.run(FlawlessAcademyApplication.class, args);
	}
}
```

Academy

```
package com.model;
import javax.validation.constraints.Min;
import javax.validation.constraints.Max;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
@Component
public class Academy {
	private String program;
	private double costPerSession;
	private int weeksPerMonth;
	@Min(value = 2, message = "Minimum 2 Sessions/Week")
	@Max(value = 5, message = "Maximum 5 Sessions/Week")
	private int sessionsPerWeek;
	public int getSessionsPerWeek() {
		return sessionsPerWeek;
	}
	public void setSessionsPerWeek(int sessionsPerWeek) {
		this.sessionsPerWeek = sessionsPerWeek;
	}
	public Academy() {}
```

```
	public int getWeeksPerMonth() {
		return weeksPerMonth;
	}
	public void setWeeksPerMonth(int weeksPerMonth) {
		this.weeksPerMonth = weeksPerMonth;
	}
	public double getCostPerSession() {
		return costPerSession;
	}
	public void setCostPerSession(double costPerSession) {
		this.costPerSession = costPerSession;
	}
	public String getProgram() {
		return program;
	}
	public void setProgram(String program) {
		this.program = program;
	}
}
```

AcademyService

```
package com.service;
import org.springframework.stereotype.Service;
import com.model.Academy;
@Service
public class AcademyService {
	public double calculateProgramCost (Academy academyBean) {
		double cost=0.0;
```

```
academyBean.setWeeksPerMonth(4);

System.out.println(academyBean.getSessionsPerWeek()+" 
"+academyBean.getProgram()+" "+academyBean.getWeeksPerMonth());

if(academyBean.getProgram().equalsIgnoreCase("ClassicalDance") && 
academyBean.getSessionsPerWeek()>=1)

{

academyBean.setCostPerSession(150.0);

cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.ge
tWeeksPerMonth();

}

else if(academyBean.getProgram().equalsIgnoreCase("KarnaticVocals") && 
academyBean.getSessionsPerWeek()>=1 )

{

academyBean.setCostPerSession(100.0);

cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.ge
tWeeksPerMonth();

}

else if(academyBean.getProgram().equalsIgnoreCase("WesternDance") && 
academyBean.getSessionsPerWeek()>=1)

{

academyBean.setCostPerSession(125.0);

cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.ge
tWeeksPerMonth();

}

if(academyBean.getProgram().equalsIgnoreCase("Drawing") && 
academyBean.getSessionsPerWeek()>=1)

{

academyBean.setCostPerSession(130.0);

cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.ge
tWeeksPerMonth();

}
```

```
else if(academyBean.getProgram().equalsIgnoreCase("Instruments") &&
academyBean.getSessionsPerWeek()>=1)

{

        academyBean.setCostPerSession(200.0);

        cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.ge
tWeeksPerMonth();

}

        return cost;

        }

}
```

Application.properties

```
server.port=9088
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
```

---

enrollementpage.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1><center> Flawless Academy</center></h1>
<form:form method="post" action="progEstimation" modelAttribute="academyBean">
<table style="margin: 0px auto; margin-left: auto; margin-right:auto">
                    <tr>
                    <td>Select preferred program</td>
                    <td>
                    <form:select path="program" id="program">
                    <form:options items="${programList}"/>
                    </form:select>

                    </td>
                    </tr>

                    <tr>
                    <td>Number of sessions per week</td>
```

```
        <td><form:input path="sessionsPerWeek" id="sessionsPerWeek"/></td>
        <td><form:errors path="sessionsPerWeek" id="errormsg"/></td>
        <tr>
        <td><input type="submit" value="Program Cost" name="submit" /></td>
        </tr>
</table>
</form:form>
</body>
</html>
```

Estimationpage.jsp

```
<%@page isELIgnored="false" %>
<html>
<body bgcolor="lavender">
<center><h2>Welcome to Flawless Academy. Your preferred program cost is Rs.
${cost}</h2></center>
</body>
</html>
```

FarmFreshApplication

ShoppingController

```
package com.controller;

import java.util.ArrayList;

import java.util.List;

import javax.validation.Valid;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import com.exception.NoStockException;

import com.model.Product;

import com.service.ShoppingService;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;

//use appropriate annotation to configure ShoppingController as Controller

@Controller

public class ShoppingController {

	@Autowired

	private ShoppingService shoppingService;

	@RequestMapping(value = "/showPage", method = RequestMethod.GET)

	public String showPage(@ModelAttribute("product")Product product) {

		return "showPage";		}

	@ModelAttribute("productNames")
```

```
public  List<String> populateProduct(){
        List<String> productNames = new ArrayList<String>();
            productNames.add("Apple");
            productNames.add("Orange");
            productNames.add("Grapes");
            productNames.add("Mango");
            productNames.add("JackFruit");
            return productNames;
}
//invoke the service class - calculateCostAndUpdate method.
@RequestMapping(value = "/calculate", method = RequestMethod.POST)
public String calculateCost(@ModelAttribute("product")@Valid Product product, BindingResult
result, ModelMap model) throws  NoStockException {
        if(result.hasErrors()){
            return ("showPage");
        }   else {
            double cost=shoppingService.calculateCost(product);
            model.addAttribute("cost",cost);
            model.addAttribute("costPerKg",product.getCostPerKg());
            model.addAttribute("name",product.getProductName());
            model.addAttribute("quantity",product.getQuantity());
        }
        return ("billDesk");
}
@ExceptionHandler(NoStockException. class)
public ModelAndView exceptionHandler(Exception e) {
        ModelAndView mav = new ModelAndView();
        mav.addObject("message", e.getMessage());
```

```
		mav.setViewName("exceptionPage");

		return mav;

	}

}
```

FarmFreshApplication

```
package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan({"com.*"})

public class FarmFreshApplication {

	public static void main(String[] args) {

		SpringApplication.run(FarmFreshApplication.class, args);

	}

}
```

NoStockException

```
package com.exception;

public class NoStockException extends Exception{
	private static final long serialVersionUID = 1L;
public NoStockException(String msg) {

	super(msg);
}

}
```

Product

```
package com.model;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotEmpty;
import org.springframework.stereotype.Component;

@Component
public class Product {
	private String productName ;
	@Min(value=1,message="Minimum quantity should be 1Kg")
	private int quantity;
	private double costPerKg;


	public String getProductName() {
		return productName;
	}

	public void setProductName(String productName) {
		this.productName = productName;
	}

	public int getQuantity() {
		return quantity;
	}

	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}


	public double getCostPerKg() {
		return costPerKg;
	}

	public void setCostPerKg(double costPerKg) {
		this.costPerKg = costPerKg;
	}


public Product() {}


}
```

ShoppingService

```
package com.service;

import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
```

```java
import org.springframework.stereotype.Service;
import com.exception.NoStockException;
import com.model.Product;

//use appropriate annotation to configure ElectricityService as a Service
@Service
public class ShoppingService {

	public double calculateCost(Product product) throws  NoStockException {
	double totalCost=0.0;

	Map<String,Integer> stock=getProductStock();
	for (Entry<String, Integer> entry : stock.entrySet()) {
		if(product.getProductName().equalsIgnoreCase(entry.getKey())) {

			if(product.getQuantity()>entry.getValue()) {
				throw new NoStockException("No enough stock for product
"+product.getProductName());
			}
		}
	}

	if(product.getProductName().equals("Apple")) {
		product.setCostPerKg(250);
	}
	else if(product.getProductName().equals("JackFruit")) {
		product.setCostPerKg(75);
	}
	else if(product.getProductName().equals("Orange")) {
		product.setCostPerKg(90);
	}
	else if(product.getProductName().equals("Mango")) {
		product.setCostPerKg(60);
	}
	else if(product.getProductName().equals("Grapes")) {
		product.setCostPerKg(150);
	}

	totalCost=product.getQuantity()*product.getCostPerKg();

	return totalCost;
	}

	public  Map<String,Integer> getProductStock(){
		Map<String,Integer> stock = new HashMap<String,Integer>();
		stock.put("Apple",50);
		stock.put("Grapes",10);
		stock.put("Orange",30);
		stock.put("Mango",75);
		stock.put("JackFruit",25);
			return stock;
	}
}
```

Application.properties

```
server.port=9088
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
```

Billdesk.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html>
<body bgcolor="lavender"> <center>
<h2>Farm Fresh - A natural farm store!!</h2>

                    <table  >
                                 <tr>
                            <td>Product Name  <td>${name}</td>
                            </tr>
                            <tr>
                             <td>Quantity  <td>${quantity}</td>
                             </tr>
                             <tr>
                             <td>Cost Per Kg  <td>${costPerKg}</td>
                             </tr>
                             <tr>
                             <td>Total cost Rs  <td>${cost}</td>
                           </tr>
              </table>
       </center>

       </body>
</html>
```

Exceptionpage.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html>
<body bgcolor="lavender">

<center> <h2>Farm Fresh - A natural farm store!!</h2>
              <h3>
              <% if (request.getAttribute("message") != null) {  %>
              <%=request.getAttribute("message")%>
              <%  }  %>
              </h3>
              <div>
```

```
            <center> <a href="showPage"> Home</a></center>
            </div>
            </center>

</body>
</html>
```

Showpage.jsp

```
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1" isELIgnored="false"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>

<body style="background-color:lavender">
<!--  Design the page as per the requirements-->

<form:form method="post" action="calculate" modelAttribute="product">
<center><h2>Farm Fresh - A natural farm store!!</h2></center>
<table style="margin: 0px auto; margin-left: auto; margin-right:auto">
              <tr>
         <td>Select product</td>
                   <td>
                   <form:select path="productName" id="productName">
                   <form:options items="${productNames}"/>
                   </form:select>
             </td>
              </tr>
         <tr>
              <td>Quantity</td>
         <td><form:input path="quantity" id="quantity"/></td>
         <td><form:errors path="quantity"/></td>
              </tr>

       </table>
       <center><input type="submit" value="Calculate Cost" id="submit"
name="submit"/></center>
       <br>
</form:form>
</body>
</html>
```

DeliveryAgencyApplication

AgencyController

```
package com.controller;

import java.util.List;

import javax.validation.Valid;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;

import com.exception.NoOrdersFoundException;

import com.model.Order;

import com.service.AgencyService;

//use appropriate annotation to configure AgencyController as Controller

@Controller

public class AgencyController {

	//Use appropriate annotation

	@Autowired

	private AgencyService service;

	@RequestMapping(value = "/searchPage", method = RequestMethod.GET)

	public String searchPage(@ModelAttribute("orderBean") Order orderBean) {

		return "searchPage";		}

	//invoke the service class - searchDeliveryOrders method.
```

```
	@RequestMapping(value = "/orderList", method = RequestMethod.POST)

	public String searchDeliveryOrders(@ModelAttribute("orderBean") @Valid Order orderBean,
BindingResult result, ModelMap model) throws NoOrdersFoundException  {

		//fill the code

		if (result.hasErrors()) {

			return  "searchPage";

		} else {

		List<Order> ordersFound = service.searchDeliveryOrders(orderBean);

		if(ordersFound.isEmpty()) {

	throw new NoOrdersFoundException("No orders found for this search criteria");

} else {

model.addAttribute("orders", ordersFound);

return "deliveryListPage";

			}

		}

	}

	@ExceptionHandler(NoOrdersFoundException.class)

	public ModelAndView exceptionHandler(Exception e) {

		ModelAndView mav = new ModelAndView();

	   mav.addObject("exception", e);

	   mav.setViewName("exceptionPage");

	   return mav;

	}

}
```

DeliveryAgencyApplication

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan({"com.*"})

public class DeliveryAgencyApplication {

	public static void main(String[] args) {

		SpringApplication.run(DeliveryAgencyApplication.class, args);

	}

}
```

NoOrdersFoundException

```
package com.exception;

public class NoOrdersFoundException extends Exception {
	private static final long serialVersionUID = 1L;

	public NoOrdersFoundException(String msg) {
		super(msg);
	}

}
```

Order

```
package com.model;

import javax.validation.constraints.NotEmpty;

import org.springframework.stereotype.Component;


//pojo class with required attributes,getters and setters

@Component

public class Order {

	private String orderId;

	private String orderFrom;
```

```
private String deliveryTo;
// fill the code
@NotEmpty(message = "Delivery Date should be Mandatory")
private String deliveryDate;
private String deliveryStatus;
public String getOrderId() {
        return orderId;
}
public void setOrderId(String orderId) {
        this.orderId = orderId;
}
public String getOrderFrom() {
        return orderFrom;
}
public void setOrderFrom(String orderFrom) {
        this.orderFrom = orderFrom;
}
public String getDeliveryTo() {
        return deliveryTo;
}
public void setDeliveryTo(String deliveryTo) {
        this.deliveryTo = deliveryTo;
}
public String getDeliveryDate() {
        return deliveryDate;
}
public void setDeliveryDate(String deliveryDate) {
        this.deliveryDate = deliveryDate;        }
```

```
	public String getDeliveryStatus() {
		return deliveryStatus;
	}
	public void setDeliveryStatus(String deliveryStatus) {
		this.deliveryStatus = deliveryStatus;
	}
	public Order(String orderId, String orderFrom, String deliveryTo, String deliveryDate, String deliveryStatus) {
		super();
		this.orderId = orderId;
		this.orderFrom = orderFrom;
		this.deliveryTo = deliveryTo;
		this.deliveryDate = deliveryDate;
		this.deliveryStatus = deliveryStatus;
	}
	public Order()	{}
	}
```

AgencyService

```
package com.service;
import java.util.ArrayList;
import org.springframework.stereotype.Service;
import com.model.Order;
//use appropriate annotation to configure AgencyService as a Service
@Service
public class AgencyService {
	ArrayList<Order> list = new ArrayList<Order>();
	//search orders and return the ArrayList<Order>
```

```
public ArrayList<Order> searchDeliveryOrders(Order orderBean) {

ArrayList<Order> foundOrders = new ArrayList<>();

for(Order order : list) {

if(order.getDeliveryDate().equals(orderBean.getDeliveryDate()) &&
order.getDeliveryStatus().equals(orderBean.getDeliveryStatus())) {

foundOrders.add(order);

}

}

return foundOrders;

}

public AgencyService() {

addOrders();

}

public  void addOrders(){

// don't change this code

list.add(new Order("DF1234233","AJIO trends","#412, Shipitha cascade, Ganapathy, PIN-
641001","10/02/2021","pending"));

list.add(new Order("FGF7788","Amazon india","#12, shardhalotus, Ram Nagar, PIN-641001",
"02/02/2021","pending"));

list.add(new Order("AF5546","Mynthra","#12, Anandha nilayam,Kuniamuthur, PIN-
641001","08/02/2021","pending"));

list.add(new Order("DF5543","Flip cart","#201, Shipitha cascade, Saravanampatti, PIN-
641001","10/02/2021","delivered"));

list.add(new Order("DX17788","AJIO trends","#34/45, RV Homes, 5th cross, Anssari street, PIN-
600001","01/02/2021","pending"));

list.add(new Order("RF55346","Amazon india","#212/4, 10th cross, T-nagar, PIN-
60008","03/02/2021","delivered"));

list.add(new Order("DF88994","Mynthra","#556/4, 5th cross, Thambaram, PIN-
60010","12/02/2021","pending"));

list.add(new Order("FT51122","AJIO trends","#556/4,Nila paradise, Thambaram, PIN-
60010","03/02/2021","delivered"));
```

```
list.add(new Order("LK44338","Flip cart","#412/4, 10th cross, Velacheri, PIN-
60009","03/02/2021","delivered"));

list.add(new Order("JK99008","Amazon india","#45, 12Th main , GM Palya, PIN-
560075","03/02/2021","pending"));

list.add(new Order("MJ99765","Flip cart","#67/2, 3rd cross , Indra nagar, PIN-
560077","08/02/2021","delivered"));

list.add(new Order("NK66754","Mynthra","#302, Sakthi enclave ,1st main , Malleshpalya, PIN-
560075","08/02/2021","delivered"));

        }

}
```

Applications.properties

```
#Do not edit the key server.port.Port number can be changed
#include other needed configurations
server.port=8088
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
```

deliveryList.jsp

```
<%@page isELIgnored="false"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<html>
<body bgcolor="lavender">
       <h1>
              <center>Delivery Agency</center>
       </h1>

       <table>
              <tr>
                     <td>OrderId</td>
                     <td>Order From</td>
                     <td>Delivery To</td>
                     <td>Delivery Date</td>
                     <td>Status</td>
              </tr>
              <c:forEach items="${orders}" var="order">
                     <tr>
                            <td>${order.orderId}</td>
                            <td>${order.orderFrom}</td>
                            <td>${order.deliveryTo}</td>
                            <td>${order.deliveryDate}</td>
                            <td>${order.deliveryStatus}</td>
                     </tr>
              </c:forEach>
       </table>
```

```
</body>
</html>
```

---

exceptionPage.jsp

```
<%@page isELIgnored="false"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
<html>
<body bgcolor="lavender">
	<h1>
		<center>Delivery Agency</center>
	</h1>
	<center>
		<h3>${exception.message}</h3>
		<a href="/searchPage">
			<button>Go to Search</button>
		</a>
	</center>
</body>
</html>
```

Searchpage.jsp

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color: lavender">
<h1><center>Delivery Agency</center></h1>
<center>
<sf:form action="orderList" name="form" modelAttribute="orderBean" method="POST">

<fieldset>
	<label for='deliveryDate'>Search Delivery Date</label>
	<sf:input path="deliveryDate" id="deliveryDate" name="deliveryDate"
				placeholder="dd/mm/yyyy" />
	<sf:errors path="deliveryDate" />
	<br />
	<label for='deliveryStatus'>Select Delivery Status</label>
	<sf:select path="deliveryStatus" id="deliveryStatus" name="deliveryStatus">
		<sf:option label="delivered" value="delivered"/>
		<sf:option label="pending" value="pending"/>
	</sf:select>
```

```
</fieldset>
      <input type="submit" id="submit" value="Search" name="submit" />
      </sf:form>
      </center>
</body>
</html>
```

DigitalHomeBrokerApplication

BrokerController

```
package com.controller;

import java.util.ArrayList;

import java.util.List;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import com.model.Home;

import com.service.BrokerService;

//use appropriate annotation to configure BrokerController as Controller

@Controller

public class BrokerController {

	//Use appropriate annotation above this property

	@Autowired

	private BrokerService service;

	//invoke the service class - searchHome method.

	@RequestMapping(value = "/homeList", method = RequestMethod.POST)

	public String searchHome(@ModelAttribute("homeBean") Home homeBean,

 BindingResult result, ModelMap model) {

		if(result.hasErrors())

			return "searchPage";
```

```
		ArrayList<Home> listHome=service.searchHome(homeBean);
		model.addAttribute("listHome",listHome);
		//fill the code
		return "homeListPage";
	}
	@RequestMapping(value="/searchPage", method = RequestMethod.GET)
	public String searchPage() {
		return "searchPage";
	}
}
```

DigitalHomeBrokerApplication

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
@SpringBootApplication
@ComponentScan({"com.*"})
public class DigitalHomeBrokerApplication {
	public static void main(String[] args) {
		SpringApplication.run(DigitalHomeBrokerApplication.class, args);
	}
}
```

Home

```
package com.model;
import javax.validation.constraints.Size;
```

```
import org.springframework.stereotype.Component;
//pojo class with required attributes,getters and setters
@Component
public class Home {
	private String location;
	private String address;
	private String contactNumber;
	private String bhkType;
	private String contactPerson;
	@Size(min=5000,message="Minimum rent should be 5000")
	private double rent;
	public double getRent() {
		return rent;
	}
	public void setRent(double rent) {
		this.rent = rent;
	}
	public String getLocation() {
		return location;
	}
	public void setLocation(String location) {
		this.location = location;	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}
```

```
	public String getContactNumber() {
		return contactNumber;
	}
	public void setContactNumber(String contactNumber) {
		this.contactNumber = contactNumber;
	}
	public String getBhkType() {
		return bhkType;
	}
	public void setBhkType(String bhkType) {
		this.bhkType = bhkType;
	}
	public String getContactPerson() {
		return contactPerson;
	}
	public void setContactPerson(String contactPerson) {
		this.contactPerson = contactPerson;
	}

	public Home(String location, String address, String contactNumber, String bhkType, String
contactPerson,double rent) {
		super();
		this.location = location;
		this.address = address;
		this.contactNumber = contactNumber;
		this.bhkType = bhkType;
		this.contactPerson = contactPerson;
		this.rent=rent;
```

```
	}
	public Home()
	{}
}
```

BrokerService

```
package com.service;
import java.util.ArrayList;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.model.Home;
//use appropriate annotation to configure BrokerService as a Service
@Service
public class BrokerService {
	ArrayList<Home> list = new ArrayList<Home>();
	// search homes/apartments from the list and return it as a ArrayList
	public ArrayList<Home> searchHome(Home homeBean) {
		//fill the code
		ArrayList<Home> arr=new ArrayList<>();
		for(Home h:list)
		{
			if(h.getBhkType().equalsIgnoreCase(homeBean.getBhkType()) &&
h.getLocation().equalsIgnoreCase(homeBean.getLocation()) && h.getRent()==homeBean.getRent())
			{
				arr.add(h);
			}
		}
		return arr;
	}
```

```
	public BrokerService() {
		addHomes();
	}
	// Dont modify this code
	public ArrayList<Home> addHomes() {
list.add(new Home("Coimbatore", "#412, Shipitha cascade, Ganapathy, PIN-641001", "9917678456",
"1BHK","Malavika", 8000));
list.add(new Home("Coimbatore", "#12, shardhalotus, Ram Nagar, PIN-641001", "9917100989", "2BHK",
"Karthik",12000));
list.add(new Home("Coimbatore", "#12, Anandha nilayam,Kuniamuthur, PIN-641001", "9917100989",
"2BHK", "Karthik",12000));
list.add(new Home("Coimbatore", "#201, Shipitha cascade, Saravanampatti, PIN-641001",
"9917123456", "3BHK","Sowparnika", 17000));
list.add(new Home("Chennai", "#34/45, RV Homes, 5th cross, Anssari street, PIN-600001",
"7788900989", "1BHK","Mukunthan", 6000));
list.add(new Home("Chennai", "#212/4, 10th cross, T-nagar, PIN-60008", "9917100989", "2BHK",
"Annesh", 14000));
list.add(new Home("Chennai", "#556/4, 5th cross, Thambaram, PIN-60010", "9917100989", "2BHK",
"Jisha", 14000));
list.add(new Home("Chennai", "#556/4,Nila paradise, Thambaram, PIN-60010", "9917100989", "2BHK",
"Jisha",14000));
list.add(new Home("Chennai", "#412/4, 10th cross, Velacheri, PIN-60009", "9917156989", "3BHK",
"Rakesh", 18000));
list.add(new Home("Bangalore", "#45, 12Th main , GM Palya, PIN-560075", "9917122119", "1BHK",
"Senthil", 11000));
list.add(new Home("Bangalore", "#67/2, 3rd cross , Indra nagar, PIN-560077", "9944550989", "2BHK",
"RamKumar",17000));
list.add(new Home("Bangalore", "#302, Sakthi enclave ,1st main , Malleshpalya, PIN-560075",
"9917668889","3BHK", "Saradha", 25000));
return list;
	}
}
```

Application.properties

```
#Do not edit the key server.port.Port number can be changed
#include other needed configurations
server.port=8088
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
```

Homelistpage.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<body bgcolor="lavender">
<h1><center> Digital Home Broker </center></h1>
<h2><center>Search homes|Apartments !!!  </center></h2>
<table border="1" cellpadding="10">
        <thead>
            <tr>
                <th>Location</th>
                <th>BhkType</th>
                <th>Address</th>
                <th>Contact Person</th>
                <th>Contact Number</th>
            </tr>
        </thead>
        <tbody>
            <tr th:each="home : ${listHome}">
             <td th:text="${home.location}">Location</td>
                <td th:text="${home.bhkType}">BhkType</td>
                <td th:text="${home.address}">Address</td>
                <td th:text="${home.contactPerson}">Contact Person</td>
                <td th:text="${home.contactNumber}">Contact Number</td>

            </tr>
        </tbody>
    </table>

</body>
</html>
```

Searchpage.jsp

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1><center> Digital Home Broker </center></h1>
<h2><center>Search homes|Apartments !!!  </center></h2>
<form action="#" th:action="@{/homeList}" th:object="${home}"
                   method="post">

      <table border="0" cellpadding="10">
      <tr>
<td>Search Location:</td>
<td><input name="location" id="location" type="text" th:field="*{location}" /></td>
      </tr>
      <tr>
      <td>Search BHK Type:</td>
<td><input name="bhkType" id="bhkType" type="text" th:field="*{bhkType}" /></td>
      </tr>
      <tr>
      <td>Preferred Rent:</td>
<td><input min="5000" name="rent" id="rent" type="number" th:field="*{rent}" /></td>
      </tr>
      <tr>
<td><button type="submit">Search</button> </td>
      </tr>
      </table>
      </form>
</body>
</html>
```

Construction Cost Estimator - V1

Grade settings: Maximum grade: 100
Based on: Construction Cost Estimator - V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB **Maximum execution file size**: 1.50 GiB

Construction Cost Estimator

Smart Builders is one of the leading builders in Tamilnadu, they approaches you to create a separate online application for calculating construction cost based on bricks type (UnburntClayBricks/BurntClayBricks/ConcreteBricks).

Create a Spring MVC Spring Boot Application for developing a Construction Cost Estimator Application. Design a Construction Cost Estimator Page to choose an appropriate bricks Type **(UnburntClayBricks/BurntClayBricks/ConcreteBricks)** and enter the buildup area.

On clicking CalculateCost button, the application should calculate the Construction cost based on the bricks type chosen and the entered buildup area. The customer has to then be redirected to result.jsp page that displays the message "**Approximate construction cost for given parameters is Rs.<<constructionCost>>"**

Application Work Flow:

- **ConstructionController** is the Controller class.
- **Construction** is the model class with three attributes **bricksType**, **costPerSqFt** and **buildupArea** along with its getters and setters.
- **ConstructionService** is the Service class which has a method called **calculateConstructionCost** that takes Invoice as its argument and returns double.

1. Calculate the construction cost depending on the rate per square feet based on the bricksType selected and the buildupArea given by the customer.
2. This method should set the **costPerSqFt** instance variable based on bricks type value given in the below table.
3. It should be returned as the double.

Example:

- If buildupArea is 1200 and bricksType is ConcreteBricks then set costPerSqFt = 300.00
- totalCost = buildupArea * costPerSqFt = 1200 * 300.00 = 360000.00

bricksType	costPerSqFt
UnburntClayBricks	340.00
BurntClayBricks	390.00
ConcreteBricks	300.00

Initially, the customer should be routed via the ConstructionControllerâ€™s **estimatorPage** method to **estimatorpage.jsp** that allows customer to choose the bricksType and enter the buildupArea.

[Note: estimatorPage method has to be written inside the ConstructionController]

A method in the **ConstructionController** known as buildState should be annotated with the ModelAttribute "**bricksList"**. This method should populate the bricks type (**UnburntClayBricks/BurntClayBricks/ConcreteBricks**) the Map as key-value pair. Both key and value be the same buildingType. (Eg: Key- **ConcreteBricks**, Value- **ConcreteBricks**) and then return the Map. This should be then used to autopopulate the bricksType in the **estimatorpage.jsp**

[Note: buildState method should be written inside the ConstructionController]

On clicking the Calculate Cost button, the ConstructionControllerâ€™s **calculateConstructionCost** method should be called. This method takes three arguments - model attribute named "**construction"** which holds the form populated **Construction** Object, BindingResult and the ModelMap.

- This method should calculate the **constructionCost** by invoking the **calculateConstructionCost** method of the **ConstructionService**.

estimatorpage.jsp

- If the buildupArea entered by the customer is less than 500 then an error message "**Buildup Area should be minimum 500**" has to be displayed.

estimatorpage.jsp

Redirect the customer to result.jsp page with a message "**Approximate construction cost for given parameters is Rs. <<constructionCost>>**"

result.jsp

Design Constraints:

UI Design Constraints:

estimatorpage.jsp		
Component	**ID**	**Constraints**
Select	bricksType	Should be auto populated using the model attribute written above the **buildState** method inside the **ConstructionController**. Do not hard code the values
Textbox	buildupArea	Minimum value should be 500
submit	submit	-

Note:

- In **result.jsp**, the Result has to be rendered in the **<h2>** tag

Component Specification:

Controller

ConstructionController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**
service	ConstructionService	private	Use annotation to Autowire

ConstructionController			
Method Name	**Method Argument name: type**	**Return type**	**RequestMapping URL & Request Method**
estimatorPage	modelAttribute "construction": Construction	String	/ estimatorPage & GET
calculateConstructionCost	modelAttribute "construction": Construction, result:BindingResult, map:ModelMap	String	/result & POST
buildState		Map <String,String>	Should be annotated with ModelAttribute with name **"bricksList"**

Service

ConstructionService		
Method Name	**Method Argument name: type**	**Return type**
calculateConstructionCost	construction: Construction	double

Model

Construction	
AttributeName	**AttributeType**
bricksType	String
buildupArea	double
costPerSqFt	double

Overall Design Constraints:

- **ConstructionController** should be inside the package **com.controller**
- **ConstructionService** should be inside the package **com.service**
- **Construction** should be in the package **com.model**
- Use appropriate annotation to configure ConstructionService as a Service
- Use appropriate annotation to configure ConstructionController as a Controller
- ConstructionService should be autowired inside the BuildingController.
- Use annotations to implement the business Validation as specified in the screen shot **[That is, when the buildupArea is less than 500 then error message "Buildup Area should be minimum 500" should be rendered in the UI]**
- Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.
- In the pom.xml you are provided with all the dependencies needed for developing the application.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.
- Do not change or delete the class/method names or return types which are provided to you as a part of the base code skeleton.
- Please make sure that your code does not have any compilation errors while submitting your case study solution.

Submitted by sathya .

Automatic evaluation[+]

ConstructionCostEstimator/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.1.8.RELEASE</version>
```

```
9                       <relativePath/> <!-- lookup parent from repository -->
10          </parent>
11          <groupId>com.controller</groupId>
12          <artifactId>ConstructionCostEstimator</artifactId>
13          <version>0.0.1-SNAPSHOT</version>
14          <name>ConstructionCostEstimator</name>
15          <description>Demo project for Spring Boot</description>
16
17          <properties>
18                      <java.version>1.8</java.version>
19          </properties>
20
21          <dependencies>
22                      <dependency>
23                                  <groupId>org.springframework.boot</groupId>
24                                  <artifactId>spring-boot-starter-web</artifactId>
25                      </dependency>
26
27                       <dependency>
28     <groupId>org.apache.tomcat.embed</groupId>
29     <artifactId>tomcat-embed-jasper</artifactId>
30     <scope>provided</scope>
31   </dependency>
32   <dependency>
33   <groupId>javax.servlet</groupId>
34   <artifactId>servlet-api</artifactId>
35   <version>2.5</version>
36   <scope>provided</scope>
37 </dependency>
38 <dependency>
39   <groupId>javax.servlet.jsp</groupId>
40   <artifactId>jsp-api</artifactId>
41   <version>2.1</version>
42   <scope>provided</scope>
43 </dependency>
44
45 <dependency>
46   <groupId>taglibs</groupId>
47   <artifactId>standard</artifactId>
48   <version>1.1.2</version>
49 </dependency>
50 <dependency>
51   <groupId>javax.servlet</groupId>
52   <artifactId>jstl</artifactId>
53   <version>1.2</version>
54 </dependency>
55 <!-- Spring test dependencies -->
56          <dependency>
57                                  <groupId>org.mockito</groupId>
58                                  <artifactId>mockito-all</artifactId>
59                                  <version>1.10.19</version>
60                                  <scope>test</scope>
61                      </dependency>
62
63                      <!-- <dependency>
64                                  <groupId>org.seleniumhq.selenium</groupId>
65                                  <artifactId>selenium-java</artifactId>
66                                  <version>2.53.0</version>
67                      </dependency>-->
68
69                      <dependency>
70     <groupId>org.seleniumhq.selenium</groupId>
71     <artifactId>htmlunit-driver</artifactId>
72     <version>2.26</version>
73 </dependency>
74
75          <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->
```

```
<dependency>
    <groupId>org.w3c.css</groupId>
    <artifactId>sac</artifactId>
    <version>1.3</version>
</dependency>


<dependency>
                                <groupId>org.springframework</groupId>
                                <artifactId>spring-test</artifactId>
                                 <version>5.1.7.RELEASE</version>  <!-- 4.0.5 -->
                                <scope>test</scope>
                        </dependency>
<!-- End of spring test dependencies -->



                        <dependency>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-starter-test</artifactId>
                                <scope>test</scope>
                                <exclusions>
                                        <exclusion>
                                                <groupId>org.junit.vintage</groupId>
                                                <artifactId>junit-vintage-engine</artifactId>
                                        </exclusion>
                                </exclusions>
                        </dependency>
        </dependencies>

        <build>
                        <plugins>
                                <plugin>
                                        <groupId>org.springframework.boot</groupId>
                                        <artifactId>spring-boot-maven-plugin</artifactId>
                                </plugin>
                        </plugins>
        </build>

</project>


```

ConstructionCostEstimator/src/main/java/com/controller/ConstructionController.java

```
package com.controller;

import java.util.HashMap;
import java.util.Map;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.model.Construction;
import com.service.ConstructionService;

@Controller
public class ConstructionController {

```

```
	@Autowired
	private ConstructionService service;

	@RequestMapping(value = "/estimatorPage", method = RequestMethod.GET)
	public String estimatorPage(@ModelAttribute("construction") Construction construction)
	{
		return "estimatorpage";
	}

	@ModelAttribute("bricksList")
	public  Map<String, String> buildState(){

		Map<String, String> serviceMap = new HashMap<String, String>();
		serviceMap.put("UnburntClayBricks", "UnburntClayBricks");
		serviceMap.put("BurntClayBricks", "BurntClayBricks");
		serviceMap.put("ConcreteBricks", "ConcreteBricks");
		return serviceMap;
	}

	@RequestMapping(value = "/result", method = RequestMethod.POST)
	public String calculateConstructionCost(@Valid @ModelAttribute("construction") Construction construction,
			BindingResult result,ModelMap map)
	{
		if(result.hasErrors()) {
			return "estimatorpage";
	}
		else
		{
		double cost=service.calculateConstructionCost(construction);
		map.addAttribute("cost",cost);
		return "result";

		}
	}



}

```

ConstructionCostEstimator/src/main/java/com/example/demo/ConstructionCostEstimatorApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan({"com.controller","com.model","com.service"})
public class ConstructionCostEstimatorApplication {

	public static void main(String[] args) {
		SpringApplication.run(ConstructionCostEstimatorApplication.class, args);
	}

}

```

ConstructionCostEstimator/src/main/java/com/model/Construction.java

```
package com.model;

import javax.validation.constraints.Min;

import org.springframework.stereotype.Component;

```

```
@Component
public class Construction {

	private String bricksType;
	@Min(value=500, message="Buildup Area should be minimum 500")
	private double buildupArea;
	private double costPerSqFt;

	public String getBricksType() {
		return bricksType;
	}
	public void setBricksType(String bricksType) {
		this.bricksType = bricksType;
	}
	public double getBuildupArea() {
		return buildupArea;
	}
	public void setBuildupArea(double buildupArea) {
		this.buildupArea = buildupArea;
	}
	public double getCostPerSqFt() {
		return costPerSqFt;
	}
	public void setCostPerSqFt(double costPerSqFt) {
		this.costPerSqFt = costPerSqFt;
	}



}
```

ConstructionCostEstimator/src/main/java/com/service/ConstructionService.java

```
package com.service;

import org.springframework.stereotype.Service;

import com.model.Construction;

@Service
public class ConstructionService {

	public double calculateConstructionCost(Construction obj)
	{
		double totalCost=0.0;
		if(obj.getBricksType().equalsIgnoreCase("UnburntClayBricks"))
		{
			obj.setCostPerSqFt(340.00);
			totalCost=obj.getBuildupArea() * obj.getCostPerSqFt();
		}
		else if(obj.getBricksType().equalsIgnoreCase("BurntClayBricks"))
		{
			obj.setCostPerSqFt(390.00);
			totalCost=obj.getBuildupArea() * obj.getCostPerSqFt();
		}
		else
		{
			obj.setCostPerSqFt(300.00);
			totalCost=obj.getBuildupArea() * obj.getCostPerSqFt();
		}
		return totalCost;
	}

}
```

ConstructionCostEstimator/src/main/resources/application.properties

```
server.port=9067
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
```

ConstructionCostEstimator/src/main/webapp/WEB-INF/views/estimatorpage.jsp

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1><center> Construction Cost Estimator </center></h1>
<form:form method="post" action="result" modelAttribute="construction">

<table style="margin: 0px auto; margin-left: auto; margin-right:auto">

                                        <tr>
                                                        <td>Choose BricksType:</td>
                                                        <td>
                                                         <form:select path="bricksType" id="bricksType">
                                                                                <form:options items="${bricksList}"/>
                                                                        </form:select>

                                                        </td>
                                        </tr>

                                        <tr>
                                        <td>Enter Buildup Area</td><td><form:input path="buildupArea"
id="buildupArea"/></td><td><form:errors path="buildupArea"/></tr>
                                        <tr>

                                                        <td><input type="submit" value="Calculate Cost" id="submit"
/></td>
                                                        <td><input type="reset" value="Cancel"/></td>
                                        </tr>

                        </table>
</form:form>


</body>
</html>

```

ConstructionCostEstimator/src/main/webapp/WEB-INF/views/result.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor="lavender">
<h2>Approximate construction cost for given parameters is Rs.${cost}</h2>
</body>
</html>
```

Grade

Reviewed on Friday, 18 February 2022, 12:31 AM by Automatic grade
Grade 98.75 / 100
Assessment report
[-]**SOURCE CODE ANALYZER REPORT**

Error Msg: A method should have only one exit point, and that should be the last statement in the method
Variable Name:
Class Name: ConstructionController

Assessment Completed Successfully
[-]**Grading and Feedback**

Feature Test - 15.00 / 15.00(Success)
** check whether Service is configured in Controller class with the required Annotation - 4.50 / 4.50*
** check whether the Controller class with the required annotation is implemented - 3.50 / 3.50*
** check whether the Service class with the required annotation is implemented - 3.50 / 3.50*
** check whether the model class with the required annotation is implemented - 3.50 / 3.50*

Functional testing - 20.00 / 20.00(Success)
** check whether the Construction cost is calculated as per the requirement for ConcreteBricks - 7.50 / 7.50*
** check whether the Construction cost is calculated as per the requirement for UnburntClayBricks - 5.00 / 5.00*
** check whether the Construction cost is calculated as per the requirement for BurntClayBricks - 7.50 / 7.50*

Spring Testing in the Controller - 30.00 / 30.00(Success)
** check whether the request is mapped for estimatorpage and redirected to estimatorpage.jsp - 5.00 / 5.00*
** check whether the model attribute is specified above the buildState method that holds the Map containing the bricksTypes - 5.00 / 5.00*
** check whether the model attribute with the name bricksList is specified above the buildState method that holds the Map containing the bricksTypes - 7.50 / 7.50*
** check whether layering is followed that is whether construction cost is calculated by invoking the service method inside the Controller - 7.50 / 7.50*
** check whether the validations are working correctly - 5.00 / 5.00*

UI Testing - 30.00 / 30.00(Success)
** check whether UI from estimatorpage to result page is redirected after calculating the construction cost for BurntClayBricks - 5.00 / 5.00*
** check whether UI from estimatorpage to result page is redirected after calculating the construction cost for ConcreteBricks - 5.00 / 5.00*
** check whether UI from estimatorpage to result page is redirected after calculating the construction cost for UnburntClayBricks - 10.00 / 10.00*
** check whether the buildUpArea is rendered in estimatorpage as per the requirement - 5.00 / 5.00*
** check whether the bricksType is rendered in estimatorpage as per the requirement - 5.00 / 5.00*

Comments and best practices/standards - 3.75 / 5.0(Partial)

College Admission Fee Calculator - V1

Grade settings: Maximum grade: 100
Based on: College Admission Fee Calculator - V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB **Maximum execution file size**: 1.50 GiB

College Admission Fee Calculator

One of the leading College in Tamilnadu approaches you to create a separate online application for calculating admission fees based on department (EEE/ECE/CSE) chosen by the applicant.

Create a Spring MVC Spring Boot Application for developing an Admission Fee Calculator Application. Design an Admission Fees Calculator Page to choose an appropriate department (EEE/ECE/CSE) and enter the other details like admissionId, hostelPreference and firstGraduateInfo.

On clicking Calculate button, the application should calculate the admissionFees based on the department chosen by the applicant. The applicant has to then be redirected to result.jsp page that displays the message "**Admission Fees to be paid is Rs.<<admissionFees>>"**

Application Work Flow:

- **AdmissionController** is the Controller class.
- **Admission** is the model class with 5 attributes admissionId, deptName, hostelPreference, tuitionFees and firstGraduateInfo along with its getters and setters.
- **AdmissionService** is the Service class which has a method called calculateAdmissionFees that takes Admission as its argument and returns double.

1. Calculate the admissionFees depending on the deptName and the other information provided by the customer.
2. This method should set the **tuitionFees** instance variable based on the value given in the below table for each department.
3. It should be returned as the double.

deptName	tuitionFees
EEE	45000.00
ECE	50000.00
CSE	60000.00

Initially, the applicant should be routed via the AdmissionController's **feesCalculatorPage** method to **feescalculatorpage.jsp** that allows applicant to choose the deptName and enter the other inputs.

[Note: feesCalculatorPage method has to be written inside the AdmissionController]

A method in the **AdmissionController** known as buildState should be annotated with the ModelAttribute "**deptNameList"**. This method should populate the depart name **(EEE/ECE/CSE)** the Map as key-value pair. Both key and value be the same department name. (Eg: Key- **EEE**, Value- **EEE**) and then return the Map. This should be then used to autopopulate the deptName in the **feescalculatorpage.jsp**

A method in the **AdmissionController** known as **getHostelPreference** should be annotated with the ModelAttribute "**preferenceList"**. This method should populate the depart name **(YES/NO)** the Map as key-value pair. Both key and value be the same hostel preference. (Eg: Key- **YES**, Value- **YES**) and then return the Map. This should be then used to autopopulate the hostel preference in the **feescalculatorpage.jsp**

A method in the **AdmissionController** known as **getFirstGraduateInfo** should be annotated with the ModelAttribute "**graduateList"**. This method should populate the graduate info **(YES/NO)** the Map as key-value pair. Both key and value be the same department name. (Eg: Key- **YES**, Value- **YES**) and then return the Map. This should be then used to autopopulate the first graduate info in the **feescalculatorpage.jsp**

[Note: buildState, getHostelPreference and getFirstGraduateInfo methods should be written inside the AdmissionController]

On clicking the Calculate button, AdmissionController's **calculateAdmissionFees** method should be called. This method takes three arguments - model attribute named "**admission"** which holds the form populated **Admission** Object, BindingResult and the ModelMap.

- This method should calculate the **admissionFees** by invoking the **calculateAdmissionFees** method of the **AdmissionService**.
- **admissionFees** has to be calculated as shown in the below example,
- **Example:**

If applicant chosen deptName as "CSE", hostelPreference ="yes" and firstGraduateInfo ="No" then,

admissionFees = packageFees + tuitionFees + hostelFees = 30000.00 +60000.00 + 75000.00 = 165000.00

If applicant chosen deptName as "CSE", hostelPreference ="yes" and firstGraduateInfo ="yes" then **packageFees is not included for that particular applicant**

admissionFees = tuitionFees + hostelFees = 60000.00 + 75000.00 = 135000.00

If applicant chosen deptName as "CSE", hostelPreference ="No" and firstGraduateInfo ="yes" then **hostelFees and packageFees are not included for that particular applicant**

admissionFees = tuitionFees = 60000.00 = 60000.00

If applicant chosen deptName as "CSE", hostelPreference ="No" and firstGraduateInfo ="No" then **hostelFees is not included for that particular applicant**

admissionFees = tuitionFees + packageFees = 60000.00 + 30000.00 = 90000.00

Note: packageFees and hostelFees for each department is given in the below table

deptName	packageFees	hostelFees
EEE	20000.00	Rs.75000.00 for all the departments
ECE	25000.00	
CSE	30000.00	

feescalculatorpage.jsp

- If the admissionId entered by the applicant is less than 6 digits then an error message "**AdmissionId should be a 6 digit positive number"** has to be displayed.

feescalculatorpage.jsp

Redirect the customer to result.jsp page with a message "**Admission Fees to be paid is Rs.<<admissionFees>>**"

result.jsp

Design Constraints:

UI Design Constraints:

feescalculatorpage.jsp		
Component	**ID**	**Constraints**
Select	deptName	Should be auto populated using the model attribute written above

		the **buildState** method inside the **AdmissionController**. Do not hard code the values
Textbox	admissionId	Should be a 6 digit number
Select	preferHostel	Should be auto populated using the model attribute written above the **getHostelPreference** method inside the **AdmissionController**. Do not hard code the values
Select	firstGraduateInfo	Should be auto populated using the model attribute written above the **getFirstGraduateInfo** method inside the **AdmissionController**. Do not hard code the values
Submit	submit	-

Note:

- In **result.jsp**, the Result has to be rendered in the **<h2>** tag

Component Specification

Controller

AdmissionController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**
service	AdmissionService	private	Use annotation to Autowire

AdmissionController			
Method Name	**Method Argument name: type**	**Return type**	**RequestMapping URL & Request Method**
feesCalculatorPage	modelAttribute "admission": Admission	String	/feesCalculatorPage & GET
calculateAdmissionFees	modelAttribute "admission": Admission, result:BindingResult, map:ModelMap	String	/result & POST
buildState		Map <String,String>	Should be annotated with ModelAttribute with name "**deptNameList**"
getHostelPreference		Map <String,String>	Should be annotated with ModelAttribute with name "**preferenceList**"
getFirstGraduateInfo		Map <String,String>	Should be annotated with ModelAttribute with name "**graduateList**"

Service

AdmissionService		
Method Name	**Method Argument name: type**	**Return type**
calculateAdmissionFees	admission : Admission	double

Model

Admission	
AttributeName	**AttributeType**
admissionId	int
deptName	String
preferHostel	String
firstGraduateInfo	String
tuitionFees	double

Overall Design Constraints:

- **AdmissionController** should be inside the package **com.controller**
- **AdmissionService** should be inside the package **com.service**
- **Admission** should be in the package **com.model**
- Use appropriate annotation to configure AdmissionService as a Service
- Use appropriate annotation to configure AdmissionController as a Controller
- AdmissionService should be autowired inside the AdmissionController.
- Use annotations to implement the business Validation as specified in the screen shot **[That is, when the admissionId is less than 6 digit then error message "Admission ID should be a 6 digit positive number" should be rendered in the UI]**
- Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.
- In the pom.xml you are provided with all the dependencies needed for developing the application.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.
- Do not change or delete the class/method names or return types which are provided to you as a part of the base code skeleton.
- Please make sure that your code does not have any compilation errors while submitting your case study solution.

Result Description

Automatic evaluation[+]

AdmissionFeeCalculator/pom.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4         <modelVersion>4.0.0</modelVersion>
5         <parent>
6                 <groupId>org.springframework.boot</groupId>
7                 <artifactId>spring-boot-starter-parent</artifactId>
8                 <version>2.1.8.RELEASE</version>
9                 <relativePath/> <!-- lookup parent from repository -->
10        </parent>
11        <groupId>com.controller</groupId>
12        <artifactId>AdmissionFeeCalculator</artifactId>
13        <version>0.0.1-SNAPSHOT</version>
14        <name>AdmissionFeeCalculator</name>
15        <description>Demo project for Spring Boot</description>
16
17        <properties>
18                <java.version>1.8</java.version>
19        </properties>
20
21        <dependencies>
22                <dependency>
23                        <groupId>org.springframework.boot</groupId>
24                        <artifactId>spring-boot-starter-web</artifactId>
25                </dependency>
26
27                 <dependency>
28    <groupId>org.apache.tomcat.embed</groupId>
29    <artifactId>tomcat-embed-jasper</artifactId>
30    <scope>provided</scope>
31   </dependency>
32   <dependency>
33  <groupId>javax.servlet</groupId>
34  <artifactId>servlet-api</artifactId>
35  <version>2.5</version>
36  <scope>provided</scope>
37 </dependency>
38 <dependency>
39  <groupId>javax.servlet.jsp</groupId>
40  <artifactId>jsp-api</artifactId>
41  <version>2.1</version>
42  <scope>provided</scope>
43 </dependency>
44
45 <dependency>
46  <groupId>taglibs</groupId>
47  <artifactId>standard</artifactId>
48  <version>1.1.2</version>
49 </dependency>
50 <dependency>
51  <groupId>javax.servlet</groupId>
52  <artifactId>jstl</artifactId>
53  <version>1.2</version>
54 </dependency>
55 <!-- Spring test dependencies -->
56        <dependency>
57                <groupId>org.mockito</groupId>
58                <artifactId>mockito-all</artifactId>
59                <version>1.10.19</version>
60                <scope>test</scope>
```

```
61                      </dependency>
62
63                      <!-- <dependency>
64                              <groupId>org.seleniumhq.selenium</groupId>
65                              <artifactId>selenium-java</artifactId>
66                              <version>2.53.0</version>
67                      </dependency>-->
68
69                      <dependency>
70    <groupId>org.seleniumhq.selenium</groupId>
71    <artifactId>htmlunit-driver</artifactId>
72    <version>2.26</version>
73 </dependency>
74
75          <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->
76 <dependency>
77    <groupId>org.w3c.css</groupId>
78    <artifactId>sac</artifactId>
79    <version>1.3</version>
80 </dependency>
81
82
83 <dependency>
84                              <groupId>org.springframework</groupId>
85                              <artifactId>spring-test</artifactId>
86                               <version>5.1.7.RELEASE</version>  <!-- 4.0.5 -->
87                              <scope>test</scope>
88                      </dependency>
89 <!-- End of spring test dependencies -->
90
91
92
93                      <dependency>
94                              <groupId>org.springframework.boot</groupId>
95                              <artifactId>spring-boot-starter-test</artifactId>
96                              <scope>test</scope>
97                              <exclusions>
98                                      <exclusion>
99                                              <groupId>org.junit.vintage</groupId>
100                                             <artifactId>junit-vintage-engine</artifactId>
101                                     </exclusion>
102                             </exclusions>
103                     </dependency>
104         </dependencies>
105
106         <build>
107                     <plugins>
108                             <plugin>
109                                     <groupId>org.springframework.boot</groupId>
110                                     <artifactId>spring-boot-maven-plugin</artifactId>
111                             </plugin>
112                     </plugins>
113         </build>
114
115 </project>
116
117
```

AdmissionFeeCalculator/src/main/java/com/controller/AdmissionController.java

```
1 package com.controller;
2
3 import java.util.HashMap;
4 import java.util.Map;
5
6 import javax.validation.Valid;
```

```
7
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.stereotype.Controller;
10 import org.springframework.ui.ModelMap;
11 import org.springframework.validation.BindingResult;
12 import org.springframework.web.bind.annotation.ModelAttribute;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RequestMethod;
15
16 import com.model.Admission;
17 import com.service.AdmissionService;
18
19 @Controller
20 public class AdmissionController {
21
22         @Autowired
23         private AdmissionService service;
24
25         @RequestMapping(value = "/feesCalculatorPage", method = RequestMethod.GET)
26         public String feesCalculatorPage (@ModelAttribute("admission") Admission admission)
27         {
28                         return "feescalculatorpage";
29         }
30         @ModelAttribute("deptNameList")
31         public  Map<String, String> buildState(){
32
33                         Map<String, String> serviceMap = new HashMap<String, String>();
34                         serviceMap.put("EEE", "EEE");
35                         serviceMap.put("ECE", "ECE");
36                         serviceMap.put("CSE", "CSE");
37                         return serviceMap;
38         }
39
40         @ModelAttribute("preferenceList")
41         public  Map<String, String> getHostelPreference(){
42
43                         Map<String, String> serviceMap = new HashMap<String, String>();
44                         serviceMap.put("YES", "YES");
45                         serviceMap.put("NO", "NO");
46                         return serviceMap;
47         }
48
49         @ModelAttribute("graduateList")
50         public  Map<String, String> getFirstGraduateInfo(){
51
52                         Map<String, String> serviceMap = new HashMap<String, String>();
53                         serviceMap.put("YES", "YES");
54                         serviceMap.put("NO", "NO");
55                         return serviceMap;
56         }
57
58         @RequestMapping(value = "/result", method = RequestMethod.POST)
59         public String calculateAdmissionFees(@Valid @ModelAttribute("admission") Admission admission,
60                                 BindingResult result,ModelMap map)
61         {
62                         if(result.hasErrors()) {
63                                         return "feescalculatorpage";
64         }
65                         else
66                         {
67                         double cost=service.calculateAdmissionFees(admission);
68                         map.addAttribute("cost",cost);
69                         return "result";
70
71                         }
72         }
73
```

```
}

```

AdmissionFeeCalculator/src/main/java/com/example/demo/AdmissionFeeCalculatorApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan({"com.controller","com.model","com.service"})
public class AdmissionFeeCalculatorApplication {

	public static void main(String[] args) {
		SpringApplication.run(AdmissionFeeCalculatorApplication.class, args);
	}

}

```

AdmissionFeeCalculator/src/main/java/com/model/Admission.java

```
package com.model;


import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import org.springframework.stereotype.Component;

@Component
public class Admission {

    @NotNull
	@Size(min=6, max=6, message = "Admission Id should be a 6 digit positive number")
	private String admissionId;
	private String deptName;
	private String preferHostel;
	private String firstGraduateInfo;
	private double tuitionFees;



	public String getAdmissionId() {
		return admissionId;
	}
	public void setAdmissionId(String admissionId) {
		this.admissionId = admissionId;
	}
	public String getDeptName() {
		return deptName;
	}
	public void setDeptName(String deptName) {
		this.deptName = deptName;
	}
	public String getPreferHostel() {
		return preferHostel;
	}
	public void setPreferHostel(String preferHostel) {
		this.preferHostel = preferHostel;
	}
	public String getFirstGraduateInfo() {
		return firstGraduateInfo;
	}
	public void setFirstGraduateInfo(String firstGraduateInfo) {
		this.firstGraduateInfo = firstGraduateInfo;
```

```
45          }
46          public double getTuitionFees() {
47                      return tuitionFees;
48          }
49          public void setTuitionFees(double tuitionFees) {
50                      this.tuitionFees = tuitionFees;
51          }
52
53
54
55 }
56
```

AdmissionFeeCalculator/src/main/java/com/service/AdmissionService.java

```
1 package com.service;
2
3 import org.springframework.stereotype.Service;
4
5 import com.model.Admission;
6
7 @Service
8 public class AdmissionService {
9
10          public double calculateAdmissionFees(Admission obj)
11          {
12                      double admissionFees;
13                      if(obj.getDeptName().equalsIgnoreCase("EEE"))
14                      {
15                                  obj.setTuitionFees(45000.00);
16                      }
17                      else if(obj.getDeptName().equalsIgnoreCase("ECE"))
18                      {
19                                  obj.setTuitionFees(50000.00);
20                      }
21                      else
22                      {
23                                  obj.setTuitionFees(60000.00);
24                      }
25
26                      if(obj.getDeptName().equalsIgnoreCase("EEE"))
27                      {
28                                  if(obj.getPreferHostel().equalsIgnoreCase("Yes"))
29                                  {
30                                              if(obj.getFirstGraduateInfo().equalsIgnoreCase("Yes"))
31                                              {
32                                                          admissionFees=obj.getTuitionFees()+75000.00;
33                                              }
34                                              else
35                                              {
36
        admissionFees=obj.getTuitionFees()+75000.00+20000.00;
37                                              }
38                                  }
39                                  else
40                                  {
41                                              if(obj.getFirstGraduateInfo().equalsIgnoreCase("Yes"))
42                                              {
43                                                          admissionFees=obj.getTuitionFees();
44                                              }
45                                              else
46                                              {
47                                                          admissionFees=obj.getTuitionFees()+20000.00;
48                                              }
49                                  }
50                      }
51                      else if(obj.getDeptName().equalsIgnoreCase("ECE"))
52                      {
```

```
53                                  if(obj.getPreferHostel().equalsIgnoreCase("Yes"))
54                                  {
55                                          if(obj.getFirstGraduateInfo().equalsIgnoreCase("Yes"))
56                                          {
57                                                  admissionFees=obj.getTuitionFees()+75000.00;
58                                          }
59                                          else
60                                          {
61
        admissionFees=obj.getTuitionFees()+75000.00+25000.00;
62                                          }
63                                  }
64                                  else
65                                  {
66                                          if(obj.getFirstGraduateInfo().equalsIgnoreCase("Yes"))
67                                          {
68                                                  admissionFees=obj.getTuitionFees();
69                                          }
70                                          else
71                                          {
72                                                  admissionFees=obj.getTuitionFees()+25000.00;
73                                          }
74                                  }
75                          }
76
77                          else
78                          {
79                                  if(obj.getPreferHostel().equalsIgnoreCase("Yes"))
80                                  {
81                                          if(obj.getFirstGraduateInfo().equalsIgnoreCase("Yes"))
82                                          {
83                                                  admissionFees=obj.getTuitionFees()+75000.00;
84                                          }
85                                          else
86                                          {
87
        admissionFees=obj.getTuitionFees()+75000.00+30000.00;
88                                          }
89                                  }
90                                  else
91                                  {
92                                          if(obj.getFirstGraduateInfo().equalsIgnoreCase("Yes"))
93                                          {
94                                                  admissionFees=obj.getTuitionFees();
95                                          }
96                                          else
97                                          {
98                                                  admissionFees=obj.getTuitionFees()+30000.00;
99                                          }
100                                 }
101                         }
102                         return admissionFees;
103         }
104
105 }
106
107
```

AdmissionFeeCalculator/src/main/resources/application.properties

```
1 server.port=9069
2 spring.mvc.view.prefix = /WEB-INF/views/
3 spring.mvc.view.suffix = .jsp
4 spring.mvc.static-path-pattern=/resources/**
```

AdmissionFeeCalculator/src/main/webapp/WEB-INF/views/feescalculatorpage.jsp

```
1 <%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
```

```
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1><center> Admission Fees Calculator </center></h1>
<form:form method="post" action="result" modelAttribute="admission">

<table style="margin: 0px auto; margin-left: auto; margin-right:auto">
                <tr>
                        <td>Enter Admission ID</td><td><form:input path="admissionId"
id="admissionId"/></td><td><form:errors path="admissionId"/></tr>
                        <tr>
                        <tr>
                                <td>Choose Department:</td>
                                <td>
                                 <form:select path="deptName" id="deptName">
                                        <form:options items="${deptNameList}"/>
                                </form:select>

                                </td>
                        </tr>

                        <tr>
                                <td>Hostel Preference:</td>
                                <td>
                                 <form:select path="preferHostel" id="preferHostel">
                                        <form:options items="${preferenceList}"/>
                                </form:select>

                                </td>
                        </tr>

                        <tr>
                                <td>Are you first graduate in your family:</td>
                                <td>
                                 <form:select path="firstGraduateInfo" id="firstGraduateInfo">
                                        <form:options items="${graduateList}"/>
                                </form:select>

                                </td>
                        </tr>

                                <td><input type="submit" value="CalculateFees" id="submit"
/></td>
                                <td><input type="reset" value="Cancel"/></td>
                        </tr>

        </table>
</form:form>


</body>
</html>

```

AdmissionFeeCalculator/src/main/webapp/WEB-INF/views/result.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor="lavender">
```

```
<h2>Admission Fees to be paid is Rs.${cost}</h2>
</body>
</html>
```

Grade

Reviewed on Friday, 18 February 2022, 12:27 AM by Automatic grade

Grade 98.75 / 100

Assessment report

[-]**SOURCE CODE ANALYZER REPORT**

Error Msg: A method should have only one exit point, and that should be the last statement in the method
Variable Name:
Class Name: AdmissionController

Assessment Completed Successfully

[-]**Grading and Feedback**

Feature Test - 15.00 / 15.00(Success)
* * check whether Service is configured in Controller class with the required Annotation - 4.50 / 4.50*
* * check whether the Controller class with the required annotation is implemented - 3.50 / 3.50*
* * check whether the Service class with the required annotation is implemented - 3.50 / 3.50*
* * check whether the model class with the required annotation is implemented - 3.50 / 3.50*

Functional testing - 15.00 / 15.00(Success)
* * check whether the admission fees is calculated as per the requirement for CSE department - 5.00 / 5.00*
* * check whether the admission fees is calculated as per the requirement for ECE department - 5.00 / 5.00*
* * check whether the admission fees is calculated as per the requirement for EEE department - 5.00 / 5.00*

Spring Testing in the Controller - 35.00 / 35.00(Success)
* * check whether the request is mapped for feesCalculatorPage and redirected to feescalculatorpage.jsp - 5.00 / 5.00*
* * check whether the model attribute is specified above the buildState method that holds the Map containing the deptName - 5.00 / 5.00*
* * check whether the model attribute is specified above the getHostelPreference method that holds the Map containing the hostel preference - 2.50 / 2.50*
* * check whether the model attribute is specified above the getFirstGraduateInfo method that holds the Map containing the first graduate info - 2.50 / 2.50*
* * check whether the model attribute with the name deptNameList is specified above the buildState method that holds the Map containing the department name - 5.00 / 5.00*
* * check whether the model attribute with the name preferenceList is specified above the getHostelPreference method that holds the Map containing the hostel preference - 2.50 / 2.50*
* * check whether the model attribute with the name graduateList is specified above the getFirstGraduateInfo method that holds the Map containing the first graduate info - 2.50 / 2.50*
* * check whether layering is followed that is whether admission fees is calculated by invoking the service method inside the Controller - 5.00 / 5.00*
* * check whether the validations are working correctly - 5.00 / 5.00*

UI Testing - 30.00 / 30.00(Success)
* * check whether deptName in feescalculatorpage is auto populated as per the requirement - 3.00 / 3.00*
* * check whether preferHostel in feescalculatorpage is auto populated as per the requirement - 3.00 / 3.00*

** check whether firstGraduateInfo in feescalculatorpage is auto populated as per the requirement - 3.00 / 3.00*
** check whether the type attribute of admissionId in feescalculatorpage is as per the requirement - 1.00 / 1.00*
** check whether UI from feescalculatorpage to result page is redirected after calculating the admission fees for EEE department - 5.00 / 5.00*
** check whether UI from feescalculatorpage to result page is redirected after calculating the admission fees for ECE department - 5.00 / 5.00*
** check whether UI from feescalculatorpage to result page is redirected after calculating the admission fees for CSE department - 5.00 / 5.00*
** check whether UI from feescalculatorpage to result page is redirected after calculating the admission fees for CSE department with first graduate offer - 5.00 / 5.00*

Comments and best practices/standards - 3.75 / 5.0(Partial)

Grade settings: Maximum grade: 100
Based on: Sai Home Builders - V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB **Maximum execution file size**: 1.50 GiB

Sai Home Builders

Sai Home Builders is one of the leading builders in Tamilnadu, they approaches you to create a separate online application for calculating building coverage ratio based on property type (Commercial/Residential) and inform the customer whether the plan is approved or not.

Create a Spring MVC Spring Boot Application for developing a Building Coverage Ratio Application. Design a Building Coverage Ration Estimator Page to choose an appropriate building Type **("Commercial", "Residential")** and enter the building area and site area.

On clicking Calculate button, the application should calculate the BCR based on the building type chosen and the entered building area and site area. The customer has to then be redirected to **result.jsp** page that displays the message "**Welcome to Sai Home Builders. Building Coverage Ratio for your property is <<ratio>>, Your plan is <<approvalStatus>>"**

Application Work Flow:

- **BuildingController** is the Controller class.
- **Building** is the model class with four attributes buildingType, buildingArea, siteArea and approvalStatus along with its getters and setters.
- **BuildingService** is the Service class which has a method called **calculateBCR** that takes Building as its argument and returns double.

1. Calculate the BCR depending on the buildingArea and siteArea for the buildingType chosen by the customer.
2. This method should set the approvalStatus instance variable as "**Approved or NotApproved"** based on BCR value given in the below table.
3. It should be returned as the double.

Example:

Consider building type is ="**Commercial"** with building area is **500.0** and site area is **1000.0**

BCR = (buildingArea / siteArea) * 100.0 = (500.0/1000.0) *100.0 = 50.0

If buildingType is "commercial" and BCR is greater than 85.5 set,

approvalStatus = Not Approved else set,

approvalStatus = Approved

Note: siteArea should be always greater than buildingArea

buildingType	BCR	approvalStatus
Commercial	>85.5	NotApproved
Residential	>75.5	NotApproved

Initially, the customer should be routed via the BuildingControllerâ€™s **calculateBCRPage** method to **BCRpage.jsp** that allows customer to choose the buildingType and enter the buildingArea and siteArea.

[Note: calculateBCRPage method has to be written inside the BuildingController]

A method in the **BuildingController** known as buildState should be annotated with the ModelAttribute "**buildingList"**. This method should populate the building types (Commercial, Residential) the Map as key-value pair. Both key and value be the same buildingType. (Eg: Key- Residential, Value- Residential, Key- Commercial, Value- Commercial) and then return the Map. This should be then used to autopopulate the buildingType in the **BCRpage.jsp**

[Note: buildState method should be written inside the BuildingController]

On clicking the Calculate button, the BuildingControllerâ€™s **calculateBuildingCoverageRatio** method should be called. This method takes three arguments - model attribute named "**building"** which holds the form populated Building Object, **BindingResult** and the **ModelMap**.

- This method should calculate the **BCR** by invoking the **calculateBuildingCoverageRatio** method of the **BuildingService**.

BCRpage.jsp

- If the buildingArea entered by the customer is less than 1 then an error messages "Building Area should be minimum one" has to be displayed.

- If the siteArea entered by the customer is less than 1 then an error messages "Site Area should be minimum one" has to be displayed.

BCRpage.jsp

Redirect the customer to result.jsp page with a message "**Welcome to Sai Home Builders. Building Coverage Ratio for your property is <<ratio>>, Your plan is <<approvalStatus>>"**

result.jsp

Design Constraints:

UI Design Constraints:

BCRpage.jsp		
Component	**ID**	**Constraints**
Select	buildingType	Should be auto populated using the model attribute written above the buildState method inside the BuildingController. Do not hard code the values
Textbox	buildingArea	Minimum value should be 1
Textbox	siteArea	Minimum value should be 1
submit	submit	-

Note:

- **In result.jsp, the Result has to be rendered in the <h2> tag**

Component Specification:

Controller

BuildingController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**
service	BuildingService	private	Use annotation to Autowire

BuildingController			
Method Name	**Method Argument name: type**	**Return type**	**RequestMapping URL & Request Method**
calculateBCRPage	modelAttribute "building":Building	String	/BCRPage & GET
calculateBuildingCoverageRatio	modelAttribute "building":Building, result:BindingResult, map:ModelMap	String	/result & POST
buildState		Map <String,String>	Should be annotated with ModelAttribute with name "buildingList"

Service

BuildingService		
Method Name	**Method Argument name: type**	**Return type**
calculateBuildingCoverageRatio	building:Building	double

Model

Building	
AttributeName	**AttributeType**
buildingType	String
buildingArea	double
siteArea	double
approvalStatus	String

Overall Design Constraints:

- **BuildingController** should be inside the package **com.controller**
- **BuildingService** should be inside the package **com.service**
- **Building** should be in the package **com.model**
- Use appropriate annotation to configure **BuildingService** as a **Service**
- Use appropriate annotation to configure **BuildingController** as a **Controller**
- **BuildingService** should be **autowired** inside the **BuildingController**.
- Use annotations to implement the business Validation as specified in the screen shot **[That is, when the buildingArea or siteArea is less than 1 then error message "building area / site area should be minimum one" should be rendered in the UI]**
- **Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.**
- In the **pom.xml** you are provided with all the dependencies needed for developing the application.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.
- Do not change or delete the class/method names or return types which are provided to you as a part of the base code skeleton.
- Please make sure that your code does not have any compilation errors while submitting your case study solution.

Automatic evaluation[+]

SaiHomeBuilders/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.1.8.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.controller</groupId>
	<artifactId>SaiHomeBuilders</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>SaiHomeBuilders</name>
	<description>Demo project for Spring Boot</description>

	<properties>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

```

```
27                          <dependency>
28     <groupId>org.apache.tomcat.embed</groupId>
29     <artifactId>tomcat-embed-jasper</artifactId>
30     <scope>provided</scope>
31   </dependency>
32   <dependency>
33  <groupId>javax.servlet</groupId>
34  <artifactId>servlet-api</artifactId>
35  <version>2.5</version>
36  <scope>provided</scope>
37 </dependency>
38 <dependency>
39  <groupId>javax.servlet.jsp</groupId>
40  <artifactId>jsp-api</artifactId>
41  <version>2.1</version>
42  <scope>provided</scope>
43 </dependency>
44
45 <dependency>
46  <groupId>taglibs</groupId>
47  <artifactId>standard</artifactId>
48  <version>1.1.2</version>
49 </dependency>
50 <dependency>
51  <groupId>javax.servlet</groupId>
52  <artifactId>jstl</artifactId>
53  <version>1.2</version>
54 </dependency>
55 <!-- Spring test dependencies -->
56         <dependency>
57                                     <groupId>org.mockito</groupId>
58                                     <artifactId>mockito-all</artifactId>
59                                     <version>1.10.19</version>
60                                     <scope>test</scope>
61                         </dependency>
62
63                         <!-- <dependency>
64                                     <groupId>org.seleniumhq.selenium</groupId>
65                                     <artifactId>selenium-java</artifactId>
66                                     <version>2.53.0</version>
67                         </dependency>-->
68
69                         <dependency>
70     <groupId>org.seleniumhq.selenium</groupId>
71     <artifactId>htmlunit-driver</artifactId>
72     <version>2.26</version>
73 </dependency>
74
75         <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->
76 <dependency>
77     <groupId>org.w3c.css</groupId>
78     <artifactId>sac</artifactId>
79     <version>1.3</version>
80 </dependency>
81
82
83 <dependency>
84                                     <groupId>org.springframework</groupId>
85                                     <artifactId>spring-test</artifactId>
86                                      <version>5.1.7.RELEASE</version>  <!-- 4.0.5 -->
87                                     <scope>test</scope>
88                         </dependency>
89 <!-- End of spring test dependencies -->
90
91
92
93                         <dependency>
```

```
94                                  <groupId>org.springframework.boot</groupId>
95                                  <artifactId>spring-boot-starter-test</artifactId>
96                                  <scope>test</scope>
97                                  <exclusions>
98                                          <exclusion>
99                                                  <groupId>org.junit.vintage</groupId>
100                                                 <artifactId>junit-vintage-engine</artifactId>
101                                         </exclusion>
102                                 </exclusions>
103                         </dependency>
104         </dependencies>
105
106         <build>
107                         <plugins>
108                                 <plugin>
109                                         <groupId>org.springframework.boot</groupId>
110                                         <artifactId>spring-boot-maven-plugin</artifactId>
111                                 </plugin>
112                         </plugins>
113         </build>
114
115 </project>
116
117
```

SaiHomeBuilders/src/main/java/com/controller/BuildingController.java

```
1 package com.controller;
2
3 import java.util.HashMap;
4 import java.util.Map;
5
6 import javax.validation.Valid;
7
8 import org.springframework.validation.BindingResult;
9 import org.springframework.web.bind.annotation.ModelAttribute;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RequestMethod;
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.stereotype.Controller;
14 import org.springframework.ui.ModelMap;
15
16 import com.model.Building;
17 import com.service.BuildingService;
18
19 @Controller
20 public class BuildingController {
21
22         @Autowired
23         private BuildingService service;
24
25         @RequestMapping(value = "/BCRPage", method = RequestMethod.GET)
26         public String calculateBCRPage(@ModelAttribute("building") Building building)
27         {
28                         return "BCRpage";
29         }
30
31         @ModelAttribute("buildingList")
32         public  Map<String, String> buildState(){
33
34                         Map<String, String> serviceMap = new HashMap<String, String>();
35                         serviceMap.put("Commercial", "Commercial");
36                         serviceMap.put("Residential", "Residential");
37                         return serviceMap;
38         }
39
40         @RequestMapping(value = "/result", method = RequestMethod.POST)
```

```
	public String calculateBuildingCoverageRatio(@Valid @ModelAttribute("building") Building building,
		BindingResult result,ModelMap map)
	{
		if(result.hasErrors()) {
			return "BCRpage";
	}
		else
		{
		double cost=service.calculateBCR(building);
		String status =building.getApprovalStatus();
		map.addAttribute("status",status);
		map.addAttribute("cost",cost);
		return "result";

		}
	}

}

```

SaiHomeBuilders/src/main/java/com/example/demo/SaiHomeBuildersApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan({"com.controller","com.model","com.service"})
public class SaiHomeBuildersApplication {

	public static void main(String[] args) {
		SpringApplication.run(SaiHomeBuildersApplication.class, args);
	}

}

```

SaiHomeBuilders/src/main/java/com/model/Building.java

```
package com.model;

import javax.validation.constraints.Min;

import org.springframework.stereotype.Component;

@Component
public class Building {

	private String buildingType;
	@Min(value=1, message="Building Area should be minimum one")
	private double buildingArea;
	@Min(value=1, message="Site Area should be minimum one")
	private double siteArea;
	private String approvalStatus;

	public String getBuildingType() {
		return buildingType;
	}
	public void setBuildingType(String buildingType) {
		this.buildingType = buildingType;
	}
	public double getBuildingArea() {
		return buildingArea;
	}
```

```
26          public void setBuildingArea(double buildingArea) {
27                      this.buildingArea = buildingArea;
28          }
29          public double getSiteArea() {
30                      return siteArea;
31          }
32          public void setSiteArea(double siteArea) {
33                      this.siteArea = siteArea;
34          }
35          public String getApprovalStatus() {
36                      return approvalStatus;
37          }
38          public void setApprovalStatus(String approvalStatus) {
39                      this.approvalStatus = approvalStatus;
40          }
41
42
43
44 }
45
```

SaiHomeBuilders/src/main/java/com/service/BuildingService.java

```
 1 package com.service;
 2
 3 import org.springframework.stereotype.Service;
 4
 5 import com.model.Building;
 6
 7 @Service
 8 public class BuildingService {
 9
10          public double calculateBCR(Building obj)
11          {
12                      double bcr=0.0;
13                      bcr = (obj.getBuildingArea()/obj.getSiteArea())*100.0;
14                      if(obj.getBuildingType().equals("Commercial"))
15                      {
16                                  if(bcr>85.5) {
17                                              obj.setApprovalStatus("Not Approved");
18                                  }
19                                  else {
20                                              obj.setApprovalStatus("Approved");
21                                  }
22                      }
23                      else if(obj.getBuildingType().equals("Residential"))
24                      {
25                                  if(bcr>75.5) {
26                                              obj.setApprovalStatus("Not Approved");
27                                  }
28                                  else {
29                                              obj.setApprovalStatus("Approved");
30                                  }
31                      }
32                      return bcr;
33          }
34
35 }
36
37
```

SaiHomeBuilders/src/main/resources/application.properties

```
1 server.port=9066
2 spring.mvc.view.prefix = /WEB-INF/views/
3 spring.mvc.view.suffix = .jsp
4 spring.mvc.static-path-pattern=/resources/**
```

SaiHomeBuilders/src/main/webapp/WEB-INF/views/BCRpage.jsp

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1><center> Building Coverage Ration Estimator </center></h1>
<form:form method="post" action="result" modelAttribute="building">

<table style="margin: 0px auto; margin-left: auto; margin-right:auto">

                                        <tr>
                                                        <td>Choose BuildingType:</td>
                                                        <td>
                                                         <form:select path="buildingType" id="buildingType">
                                                                                <form:options items="${buildingList}"/>
                                                                        </form:select>

                                                        </td>
                                        </tr>

                                        <tr>
                                        <td>Enter Building Area</td><td><form:input path="buildingArea"
id="buildingArea"/></td><td><form:errors path="buildingArea"/></tr>
                                        <tr>
                                        <tr>
                                        <td>Enter Site Area</td><td><form:input path="siteArea"
id="siteArea"/></td><td><form:errors path="siteArea"/></tr>
                                        <tr>
                                                        <td><input type="submit" value="CalculateRatio" id="submit"
/></td>
                                                        <td><input type="reset" value="Cancel"/></td>
                                        </tr>

                        </table>
</form:form>


</body>
</html>

```

SaiHomeBuilders/src/main/webapp/WEB-INF/views/result.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor="lavender">
<h2>Welcome to Sai Home Builders. Building Coverage Ratio for your property is ${cost}, Your plan is
${status}</h2>
</body>
</html>
```

Grade

Reviewed on Friday, 18 February 2022, 12:30 AM by Automatic grade
Grade 98.75 / 100

Assessment report

[-]SOURCE CODE ANALYZER REPORT

Error Msg: A method should have only one exit point, and that should be the last statement in the method
Variable Name:
Class Name: BuildingController

Assessment Completed Successfully

[-]Grading and Feedback

Feature Test - 15.00 / 15.00(Success)
* * check whether Service is configured in Controller class with the required Annotation - 4.50 / 4.50*
* * check whether the Controller class with the required annotation is implemented - 3.50 / 3.50*
* * check whether the Service class with the required annotation is implemented - 3.50 / 3.50*
* * check whether the model class with the required annotation is implemented - 3.50 / 3.50*

Functional testing - 15.00 / 15.00(Success)
* * check whether the building coverage ratio for the commercial building is calculated as per the requirement - 7.50 / 7.50*
* * check whether the building coverage ratio for the Residential building is calculated as per the requirement - 7.50 / 7.50*

Spring Testing in the Controller - 35.00 / 35.00(Success)
* * check whether the request is mapped for BCRpage and redirected to BCR jsp page - 7.50 / 7.50*
* * check whether the model attribute is specified above the buildState method that holds the Map containing the buildingTypes - 7.50 / 7.50*
* * check whether the model attribute with the name buildingList is specified above the buildState method that holds the Map containing the buildingTypes - 7.50 / 7.50*
* * check whether layering is followed that is whether BCR is calculated by invoking the service method inside the Controller - 7.50 / 7.50*
* * check whether the validations are working correctly - 5.00 / 5.00*

UI Testing - 30.00 / 30.00(Success)
* * check whether UI from BCRpage to result page is redirected after calculating the BCR for Residential building - 10.00 / 10.00*
* * check whether UI from BCRpage to result page is redirected after calculating the BCR for Commercial building - 10.00 / 10.00*
* * check whether buildingType is rendered in BCR page as expected - 4.00 / 4.00*
* * check whether the BCR jsp page is rendered - 3.00 / 3.00*
* * check whether siteArea is rendered in BCR page as expected - 3.00 / 3.00*

Comments and best practices/standards - 3.75 / 5.0(Partial)

Delivery Agency - V1

Grade settings: Maximum grade: 100
Based on: Delivery Agency - V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB

Delivery Agency

Click here to download the code skeleton

Delivery Agency is a web application to track the delivery details of an order. Along with that they need to search the delivered and pending orders on a particular date. Help them by developing an application

Create a Spring MVC Spring Boot Application for developing a Delivery Agency Application. Design a search page to delivered/pending orders based on date and delivery status

On clicking search button, the application should list the order based on the given search criteria. Then the control has to be redirected to deliveryListPage.jsp page that displays the list of orders in tabular format

Application Work Flow:

- **AgencyController** is the Controller class.
- **Order** is the model class with attributes and its getters and setters.
- **AgencyService** is the Service class that has a method called **searchDeliveryOrders** that takes **Order** as its argument and returns ArrayList<Order>.
- Attribute ArrayList<Order> list; defined in the service class. And the value for this list was populated through **addOrders** method. Donâ€™t change the code in this method.
- Through searchDeliveryOrders method, have to search the **Order** objects based on the given input **delivery date** and **status**. (Iterate the list, get the **Order** objects which have the same date and status entered in UI)

Initially, the searchPage.jsp should be routed via the AgencyController's searchPage method ,that allows user to enter the delivery date and status.

[Note: searchPage method has to be written inside the **AgencyController]**

In AgencyService you are provided with a method **addOrders** which populates the list attribute with few Order objects.

On clicking the search button, the **AgencyController's searchDeliveryOrders** method should be called. This method takes three arguments - model attribute named â€œ**orderBean**â€• which holds the form populated **Order** Object, BindingResult and the ModelMap.

- This method should search the Order objects from the list based on the delivery date and status

Screen designs and Expected Output:

searchPage.jsp

delivery date UI component is mandatory. if its empty display message as **"Delivery Date should be Mandatory"** has to be displayed in the searchPage.jsp. To do this validation use appropriate annotation above **deliveryDate** attribute in **Order** model class.

deliveryListPage.jsp

If there is no **Order** objects available based on the search then raise the user-defined **NoOrdersFoundException** with error message **â€œNo orders found for this search criteriaâ€•** Redirect the user to **exceptionPage.jsp** page (display the error message **â€œNo orders found for this search criteria**

Exception flow:

exceptionPage.jsp

By clicking the **â€œGo to Searchâ€•** button in **exceptionPage.jsp**, it should redirect to searchPage method (in AgencyController)

Design Constraints:

UI Design Constraints:

searchPage.jsp		
Component	**ID**	**Constraints**
textbox	deliveryDate	Name and id attribute should be deliveryDate, Default placeholder value should be "dd/mm/yyyy"
Drop down	deliveryStatus	Name and id attribute should be deliveryStatus. Populate the values as "delivered" and "pending"
submit	submit	Name and id attribute should be submit

Component Specification

Controller

AgencyController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**

AgencyService	service	private	Use annotation to Autowire

AgencyController			
Method Name	**Method Argument name:type**	**Return type**	**RequestMapping URL & Request Method**
searchPage	modelAttribute â€œOrderBeanâ€ • :Order	String	/searchPage & GET
searchDeliveryOrders	modelAttribute â€œorderBeanâ€ • :Order, result:BindingResult, model:ModelMap	String	/orderList & POST

Service:

AgencyService		
Method Name	**Method Argument name:type**	**Return type**
searchDeliveryOrders	orderBean:Order	ArrayList<Order>

Model:

OrderBean	
AttributeName	**AttributeType**
orderId	String
orderFrom	String
deliveryTo	String
deliveryDate	String
deliveryStatus	String

Overall Design Constraints:

- **AgencyController** should be inside the package **com.controller**
- **AgencyService** should be inside the package **com.service**
- **Order** should be in the package **com.model**
- **NoOrdersFoundException** class should in package **com.exception.**
- Use appropriate annotation to configure **AgencyService** as a **Service**
- Use appropriate annotation to configure **AgencyController** as a **Controller**
- AgencyService should be **autowired** inside the **AgencyController**.
- Use appropriate annotation to handle **Exceptions** in **AgencyController** class.
- Use annotations to implement the business Validation as specified in the screenshot [That is when the **deliveryDate** is empty then error message **"Delivery Date should be Mandatory"** should be rendered in the UI.]

- In the pom.xml you are provided with all the dependencies needed for developing the application.
- Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.

- **Do not change or delete the class/method names or return types that are provided to you as a part of the base code skeleton.**
- Please make sure that your code does not have any compilation errors while submitting your case study solution.

Automatic evaluation[+]

DeliveryAgency/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.8.RELEASE</version>
            <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>DeliveryAgency</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>DeliveryAgency</name>
    <description>Demo project for Spring Boot</description>
    <properties>
            <java.version>1.8</java.version>
    </properties>

    <dependencies>
            <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-web</artifactId>
            </dependency>

             <dependency>
   <groupId>org.apache.tomcat.embed</groupId>
   <artifactId>tomcat-embed-jasper</artifactId>
   <scope>provided</scope>
  </dependency>

  <dependency>
 <groupId>javax.servlet</groupId>
 <artifactId>servlet-api</artifactId>
 <version>2.5</version>
```

```
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>jsp-api</artifactId>
  <version>2.1</version>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>taglibs</groupId>
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
<!-- Spring test dependencies -->
     <dependency>
                                <groupId>org.mockito</groupId>
                                <artifactId>mockito-all</artifactId>
                                <version>1.10.19</version>
                                <scope>test</scope>
                    </dependency>

                    <!-- <dependency>
                                <groupId>org.seleniumhq.selenium</groupId>
                                <artifactId>selenium-java</artifactId>
                                <version>2.53.0</version>
                    </dependency>-->

                    <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>htmlunit-driver</artifactId>
    <version>2.26</version>
</dependency>

     <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->
<dependency>
    <groupId>org.w3c.css</groupId>
    <artifactId>sac</artifactId>
    <version>1.3</version>
</dependency>


<dependency>
                                <groupId>org.springframework</groupId>
                                <artifactId>spring-test</artifactId>
                                 <version>5.1.7.RELEASE</version>  <!-- 4.0.5 -->
                                <scope>test</scope>
                    </dependency>
<!-- End of spring test dependencies -->



                    <dependency>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-starter-test</artifactId>
                                <scope>test</scope>
                                <exclusions>
                                            <exclusion>
```

```
                                        <groupId>org.junit.vintage</groupId>
                                        <artifactId>junit-vintage-engine</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
    </dependencies>

    <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
    </build>

</project>
```

DeliveryAgency/src/main/java/com/controller/AgencyController.java

```
package com.controller;

import java.util.List;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.servlet.ModelAndView;
import com.model.Order;
import com.service.AgencyService;
import com.exception.NoOrdersFoundException;

//use appropriate annotation to configure AgencyController as Controller
@Controller
public class AgencyController {

    //Use appropriate annotation
    @Autowired
    private AgencyService service;

    @RequestMapping(value="/searchPage",method=RequestMethod.GET)
    public String searchPage(@ModelAttribute("orderBean")Order orderBean){
        return "searchPage";
    }


    //invoke the service class - searchDeliveryOrders method.
    @RequestMapping(value="/orderList",method=RequestMethod.POST)
    public String searchDeliveryOrders(@ModelAttribute("orderBean") @Valid Order orderBean,
BindingResult result,
                        ModelMap model) throws NoOrdersFoundException {
                //fill the code
                if(result.hasErrors()){
                    return "searchPage";
                } else{
                    List<Order> ordersFound=service.searchDeliveryOrders(orderBean);
                    if(ordersFound.isEmpty()){
```

```
                        throw new NoOrdersFoundException("No orders found for this search criteria");
                    } else{
                        model.addAttribute("orders", ordersFound);
                        return "deliveryListPage";
                    }
                }
    }

    //use appropriate annotation to handle the Exception
    @ExceptionHandler(NoOrdersFoundException.class)
    public ModelAndView exceptionHandler(Exception e) {

                //fill the code
                ModelAndView mav =new ModelAndView();
                mav.addObject("exception",e);
                mav.setViewName("exceptionPage");
                return mav;
    }
}

```

DeliveryAgency/src/main/java/com/example/demo/DeliveryAgencyApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan({"com.*"})
public class DeliveryAgencyApplication {

    public static void main(String[] args) {
                SpringApplication.run(DeliveryAgencyApplication.class, args);
    }

}

```

DeliveryAgency/src/main/java/com/exception/NoOrdersFoundException.java

```
package com.exception;

public class NoOrdersFoundException extends Exception{
    private static final long serialVersionUID = 1L;
public NoOrdersFoundException(String msg) {

    //fill the code
        super(msg);
}

}

```

DeliveryAgency/src/main/java/com/model/Order.java

```
package com.model;

import javax.validation.constraints.NotEmpty;
import org.springframework.stereotype.Component;

//pojo class with required attributes,getters and setters
@Component
public class Order {
    private String orderId;
```

```
10      private String orderFrom;
11      private String deliveryTo;
12      // fill the code
13
14      @NotEmpty(message="Delivery Date should be Mandatory")
15      private String deliveryDate;
16
17      private String deliveryStatus;
18      public String getOrderId() {
19                      return orderId;
20      }
21      public void setOrderId(String orderId) {
22                      this.orderId = orderId;
23      }
24      public String getOrderFrom() {
25                      return orderFrom;
26      }
27      public void setOrderFrom(String orderFrom) {
28                      this.orderFrom = orderFrom;
29      }
30      public String getDeliveryTo() {
31                      return deliveryTo;
32      }
33      public void setDeliveryTo(String deliveryTo) {
34                      this.deliveryTo = deliveryTo;
35      }
36      public String getDeliveryDate() {
37                      return deliveryDate;
38      }
39      public void setDeliveryDate(String deliveryDate) {
40                      this.deliveryDate = deliveryDate;
41      }
42
43      public String getDeliveryStatus() {
44                      return deliveryStatus;
45      }
46      public void setDeliveryStatus(String deliveryStatus) {
47                      this.deliveryStatus = deliveryStatus;
48      }
49      public Order(String orderId, String orderFrom, String deliveryTo, String deliveryDate, String
deliveryStatus) {
50                      super();
51                      this.orderId = orderId;
52                      this.orderFrom = orderFrom;
53                      this.deliveryTo = deliveryTo;
54                      this.deliveryDate = deliveryDate;
55                      this.deliveryStatus = deliveryStatus;
56      }
57
58      public Order()          {}
59
60  }
61
```

DeliveryAgency/src/main/java/com/service/AgencyService.java

```
1  package com.service;
2
3  import java.util.ArrayList;
4  import com.model.Order;
5  import org.springframework.stereotype.Service;
6
7  //use appropriate annotation to configure AgencyService as a Service
8  @Service
```

```java
public class AgencyService {
    ArrayList<Order> list = new ArrayList<Order>();

    //search orders and return the ArrayList<Order>
    public ArrayList<Order> searchDeliveryOrders(Order orderBean) {
                //fill the code
                ArrayList<Order> foundOrders = new ArrayList<>();
                for(Order order:list){
                    if(order.getDeliveryDate().equals(orderBean.getDeliveryDate()) &&
order.getDeliveryStatus().equals(orderBean.getDeliveryStatus())){
                        foundOrders.add(order);
                    }
                }
                return foundOrders;

    }

    public AgencyService() {
                 addOrders();
    }

    public  void addOrders(){                          // don't change this code
                list.add(new Order("DF1234233","AJIO trends","#412, Shipitha cascade, Ganapathy, PIN-
641001","10/02/2021","pending"));
                list.add(new Order("FGF7788","Amazon india","#12, shardhalotus, Ram Nagar, PIN-
641001", "02/02/2021","pending"));
                list.add(new Order("AF5546","Mynthra","#12, Anandha nilayam,Kuniamuthur, PIN-
641001","08/02/2021","pending"));
                list.add(new Order("DF5543","Flip cart","#201, Shipitha cascade, Saravanampatti, PIN-
641001","10/02/2021","delivered"));

                list.add(new Order("DX17788","AJIO trends","#34/45, RV Homes, 5th cross, Anssari
street, PIN-600001","01/02/2021","pending"));
                list.add(new Order("RF55346","Amazon india","#212/4, 10th cross, T-nagar, PIN-
60008","03/02/2021","delivered"));
                list.add(new Order("DF88994","Mynthra","#556/4, 5th cross, Thambaram, PIN-
60010","12/02/2021","pending"));
                list.add(new Order("FT51122","AJIO trends","#556/4,Nila paradise, Thambaram, PIN-
60010","03/02/2021","delivered"));
                list.add(new Order("LK44338","Flip cart","#412/4, 10th cross, Velacheri, PIN-
60009","03/02/2021","delivered"));

                list.add(new Order("JK99008","Amazon india","#45, 12Th main , GM Palya, PIN-
560075","03/02/2021","pending"));
                list.add(new Order("MJ99765","Flip cart","#67/2, 3rd cross , Indra nagar, PIN-
560077","08/02/2021","delivered"));
                list.add(new Order("NK66754","Mynthra","#302, Sakthi enclave ,1st main , Malleshpalya,
PIN-560075","08/02/2021","delivered"));


    }
}


```

DeliveryAgency/src/main/resources/application.properties

```
#Do not edit the key server.port.Port number can be changed
#include other needed configurations
server.port=8088
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**

```

DeliveryAgency/src/main/webapp/WEB-INF/views/deliveryListPage.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor="lavender">
<h1><center> Delivery Agency </center></h1>
<table>
   <tr>
      <td>OrderId</td>
      <td>Order From</td>
      <td>Delivery To</td>
      <td>Delivery Date</td>
      <td>Status</td>
   </tr>
   <c:forEach items="${orders}" var="order">
      <tr>
         <td>${order.orderId}</td>
         <td>${order.orderFrom}</td>
         <td>${order.deliveryTo}</td>
         <td>${order.deliveryDate}</td>
         <td>${order.deliveryStatus}</td>
      </tr>
    </c:forEach>
</table>
</body>
</html>
```

DeliveryAgency/src/main/webapp/WEB-INF/views/exceptionPage.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html>
<body bgcolor="lavender">
<h1><center> Delivery Agency </center></h1>
<center>
    <h3>${exception.message}</h3>
    <a href="/searchPage">
       <button>Go to Search</button>
    </a>
</center>
</body>
</html>
```

DeliveryAgency/src/main/webapp/WEB-INF/views/searchPage.jsp

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1>
   <center>Delivery Agency</center>
</h1>
<center>
  <sf:form action="orderList" name="form" modelAttribute="orderBean"
   method="POST">

   <fieldset>
```

```
        <label for='deliveryDate'>Search Delivery Date</label>
        <sf:input path="deliveryDate" id="deliveryDate" name="deliveryDate"
            placeholder="dd/mm/yyyy" />
        <sf:errors path="deliveryDate" />
        <br/>
        <label for='deliveryStatus'>Select Delivery Status</label>
        <sf:select path="deliveryStatus" id="deliveryStatus" name="deliveryStatus">
            <sf:option label="delivered" value="delivered"/>
            <sf:option label="pending" value="pending"/>
        </sf:select>
    </fieldset>
    <input type="submit" id="submit" value="Search" name="submit" />
    </sf:form>
    </center>
</body>
</html>

```

Grade

Reviewed on Wednesday, 12 January 2022, 6:39 AM by Automatic grade
Grade 95.75 / 100
Assessment report
FAILURE TEST CASES-UI
Fail 1 --test5ForSearchPageUIForDeliveryDateValidation::Check for the file searchPage.jsp/Check whether you have provided the components in searchPage.jsp as per the requirement

[-]SOURCE CODE ANALYZER REPORT

```
Error Msg: A method should have only one exit point, and that should be the last statement
in the method
Variable Name:
Class Name: AgencyController
```

[-]Grading and Feedback

```
Feature Test - 18.0 / 18.0(Success)
          * check whether the Controller / Service class with the required annotation is
implemented - 3.0 / 3.0
          * check whether the ExceptionHandler annotation is present above the Method - 6.0
/ 6.0
          * check whether the Service class with the required annotation is implemented -
3.0 / 3.0
          * check whether the model class with the required annotation is implemented - 3.0
/ 3.0

Functional testing - 18.0 / 18.0(Success)
          * check whether the searchDeliveryOrders method in AgencyService class with
pending status - 6.0 / 6.0
          * check the searchDeliveryOrders method in AgencyService class with pending
status  - 6.0 / 6.0
          * check searchDeliveryOrders method in AgencyService class  with delivered status
- 6.0 / 6.0

Spring Testing in the Controller - 28.0 / 28.0(Success)
          * check whether the request is mapped for searchPage and redirected to
searchPage.jsp page - 6.0 / 6.0
          * check whether the request is mapped for orderList and redirected to
deliveryListPage.jsp page - 6.0 / 6.0
          * check Whether exception was thrown properly - 10.0 / 10.0
          * check Delivery date UI component validated properly - 6.0 / 6.0

UI Testing - 28.0 / 31.0(Success)
```

** check whether the searchPage is rendered with proper constraints for Date - 5.0 / 5.0*
** check whether the searchPage is rendered with proper constraints for Status - 5.0 / 5.0*
** check whether UI from searchPage to deliveryListPage.jsp is redirected and rendered the required output - 10.0 / 10.0*
** check whether error message is displayed correctly when the list is empty - 8.0 / 8.0*
** check whether the deliveryDate field validated properly - 0.0 / 3.0*

Comments and best practices/standards - 3.75 / 5.0(Partial)

Flooring Cost Estimator - V1

Grade settings: Maximum grade: 100
Based on: Spring MVC GLOBAL METADATA_Report
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB **Maximum execution file size**: 1.50 GiB

Flooring Cost Estimator

One of the leading builders in Tamilnadu approaches you to create a separate online application for calculating flooring cost based on tiles brand (Kajaria/Bajaj/Nitco/OrientBell/Johnson) and the total square feet value chosen by the customer.

Create a Spring MVC Spring Boot Application for developing a Flooring Cost Estimator Application. Design a Flooring Cost Estimator Page to choose an appropriate tiles brand (Kajaria/Bajaj/Nitco/OrientBell/Johnson) and enter the total square feet.

On clicking CalculateCost button, the application should calculate the Flooring cost based on the tilesBrand chosen for the entered total square feet value. The customer has to then be redirected to result.jsp page that displays the message "**Approximate flooring cost for the given Square feet is Rs.<<flooringCost>>"**

Application Work Flow:

- **ContractController** is the Controller class.
- **FlooringContract** is the model class with three attributes tilesBrandName, costPerSqFt and totalSqFt along with its getters and setters.
- **ContractService** is the Service class which has a method called calculateFlooringCost that takes FlooringContract as its argument and returns double.

1. Calculate the flooring cost depending on the tiles rate per square feet and average flooring cost for the totalSqFt and tiles brand chosen by the customer.
2. This method should set the **costPerSqFt** instance variable based on tiles value given in the below table.
3. It should be returned as the double.

tilesBrandName	costPerSqFt
Kajaria	180.00
Bajaj	175.00
Nitco	160.00
OrientBell	155.00
Johnson	200.00

Initially, the customer should be routed via the ContractController's **estimatorPage** method to **estimatorpage.jsp** that allows customer to choose the tilesBrandName and enter the totalSqFt.

[Note: estimatorPage method has to be written inside the ConstructionController]

A method in the **ContractController** known as buildState should be annotated with the ModelAttribute "**brandNameList"**. This method should populate the tiles brand name **(Kajaria/Bajaj/Nitco/OrientBell/Johnson)** the Map as key-value pair. Both key and value be the same tilesBrandName. (Eg: Key- **Kajaria**, Value- **Kajaria**) and then return the Map. This should be then used to autopopulate the tilesBrandName in the **estimatorpage.jsp**

[Note: buildState method should be written inside the ContractController]

On clicking the Calculate button, the ContractController's **calculateFlooringCost** method should be called. This method takes three arguments - model attribute named "**contract"** which holds the form populated **FlooringContract** Object, BindingResult and the ModelMap.

- This method should calculate the **flooringCost** by invoking the **calculateFlooringCost** method of the **ContractService**.
- **flooringCost** has to be calculated as shown in the below example,
- **Example:**

If customer choose tilesBrand as "Johnson" and total square feet value as 1300

tilesCost= costPerSqFt * totalSqFt = 200.00 * 1300 = 260000.00

flooringCost = tilesCost + avgFlooringCost = 260000.00 + 94500.00 = 354500.00

Note: avgFlooringCost is given in the below table

totalSqFt	avgFlooringCost
>=100 & <=400	42000.00
>400 & <=700	73500.00
>700 & <=1400	94500.00
>1400	147000.00

- If the totalSqFt entered by the customer is less than 100 then an error message "**Total Square Feet should be minimum 100**" has to be displayed.

estimatorpage.jsp

Redirect the customer to result.jsp page with a message "**Approximate flooring cost for the given Square feet is Rs.<<flooringCost>>**"

result.jsp

Design Constraints:

UI Design Constraints:

estimatorpage.jsp		
Component	**ID**	**Constraints**
Select	tilesBrandName	Should be auto populated using the model attribute written above the **buildState** method inside the **ContractController**. Do not hard code the values
Textbox	totalSqFt	Minimum value should be 100
submit	submit	-

Note:

- In **result.jsp**, the Result has to be rendered in the **<h2>** tag

Component Specification

Controller

ContractController

AttributeName	AttributeType	Access Specifier	Constraints
service	ContractService	private	Use annotation to Autowire

ContractController			
Method Name	**Method Argument name: type**	**Return type**	**RequestMapping URL & Request Method**
estimatorPage	modelAttribute "contract" FlooringContract	String	/estimatorPage & GET
calculateFlooringCost	modelAttribute "contract": FlooringContract, result:BindingResult, map:ModelMap	String	/result & POST
buildState		Map <String,String>	Should be annotated with ModelAttribute with name "**brandNameList"**

Service

ContractService		
Method Name	**Method Argument name: type**	**Return type**
calculateFlooringCost	flooringContract : FlooringContract	double

Model

FlooringContract	
AttributeName	**AttributeType**
tilesBrandName	String
totalSqFt	double
costPerSqFt	double

Overall Design Constraints:

- **ContractController** should be inside the package **com.controller**
- **ContractService** should be inside the package **com.service**
- **FlooringContract** should be in the package **com.model**
- Use appropriate annotation to configure ContractService as a Service
- Use appropriate annotation to configure ContractController as a Controller
- ContractService should be autowired inside the ContractController.
- Use annotations to implement the business Validation as specified in the screen shot **[That is, when the totalSqFt is less than 100 then error message "Total Square Feet should be minimum 100" should be rendered in the UI]**

- Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.
- In the pom.xml you are provided with all the dependencies needed for developing the application.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.
- Do not change or delete the class/method names or return types which are provided to you as a part of the base code skeleton.
- Please make sure that your code does not have any compilation errors while submitting your case study solution.

Automatic evaluation[+]

FlooringCostEstimator/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.1.8.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.controller</groupId>
	<artifactId>FlooringCostEstimator</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>FlooringCostEstimator</name>
	<description>Demo project for Spring Boot</description>

	<properties>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		 <dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
  </dependency>
  <dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>jsp-api</artifactId>
  <version>2.1</version>
```

```
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>taglibs</groupId>
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
<!-- Spring test dependencies -->
        <dependency>
                                <groupId>org.mockito</groupId>
                                <artifactId>mockito-all</artifactId>
                                <version>1.10.19</version>
                                <scope>test</scope>
                        </dependency>

                        <!-- <dependency>
                                <groupId>org.seleniumhq.selenium</groupId>
                                <artifactId>selenium-java</artifactId>
                                <version>2.53.0</version>
                        </dependency>-->

                        <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>htmlunit-driver</artifactId>
    <version>2.26</version>
</dependency>

        <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->
<dependency>
    <groupId>org.w3c.css</groupId>
    <artifactId>sac</artifactId>
    <version>1.3</version>
</dependency>


<dependency>
                                <groupId>org.springframework</groupId>
                                <artifactId>spring-test</artifactId>
                                 <version>5.1.7.RELEASE</version>  <!-- 4.0.5 -->
                                <scope>test</scope>
                        </dependency>
<!-- End of spring test dependencies -->



                        <dependency>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-starter-test</artifactId>
                                <scope>test</scope>
                                <exclusions>
                                        <exclusion>
                                                <groupId>org.junit.vintage</groupId>
                                                <artifactId>junit-vintage-engine</artifactId>
                                        </exclusion>
                                </exclusions>
                        </dependency>
        </dependencies>

        <build>
                        <plugins>
                                <plugin>
```

```
109                                                 <groupId>org.springframework.boot</groupId>
110                                                 <artifactId>spring-boot-maven-plugin</artifactId>
111                                         </plugin>
112                                 </plugins>
113                 </build>
114
115 </project>
116
117
```

FlooringCostEstimator/src/main/java/com/controller/ContractController.java

```
1 package com.controller;
2
3 import java.util.HashMap;
4 import java.util.Map;
5
6 import javax.validation.Valid;
7
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.stereotype.Controller;
10 import org.springframework.ui.ModelMap;
11 import org.springframework.validation.BindingResult;
12 import org.springframework.web.bind.annotation.ModelAttribute;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RequestMethod;
15
16 import com.model.FlooringContract;
17 import com.service.ContractService;
18
19 @Controller
20 public class ContractController {
21
22         @Autowired
23         private ContractService service;
24
25         @RequestMapping(value = "/estimatorPage", method = RequestMethod.GET)
26         public String estimatorPage(@ModelAttribute("contract") FlooringContract contract)
27         {
28                 return "estimatorpage";
29         }
30
31         @ModelAttribute("brandNameList")
32         public  Map<String, String> buildState(){
33
34                 Map<String, String> serviceMap = new HashMap<String, String>();
35                 serviceMap.put("Kajaria", "Kajaria");
36                 serviceMap.put("Bajaj", "Bajaj");
37                 serviceMap.put("Nitco", "Nitco");
38                 serviceMap.put("OrientBell", "OrientBell");
39                 serviceMap.put("Johnson", "Johnson");
40                 return serviceMap;
41         }
42
43         @RequestMapping(value = "/result", method = RequestMethod.POST)
44         public String calculateFlooringCost(@Valid @ModelAttribute("contract") FlooringContract contract,
45                         BindingResult result,ModelMap map)
46         {
47                 if(result.hasErrors()) {
48                         return "estimatorpage";
49         }
50                 else
51                 {
52                 double cost=service.calculateFlooringCost(contract);
53                 map.addAttribute("cost",cost);
54                 return "result";
55
56                 }
```

```
57          }
58
59
60
61 }
62
```

FlooringCostEstimator/src/main/java/com/example/demo/FlooringCostEstimatorApplication.java

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.context.annotation.ComponentScan;
6
7 @SpringBootApplication
8 @ComponentScan({"com.controller","com.model","com.service"})
9 public class FlooringCostEstimatorApplication {
10
11          public static void main(String[] args) {
12                      SpringApplication.run(FlooringCostEstimatorApplication.class, args);
13          }
14
15 }
16
```

FlooringCostEstimator/src/main/java/com/model/FlooringContract.java

```
1 package com.model;
2
3 import javax.validation.constraints.Min;
4
5 import org.springframework.stereotype.Component;
6
7 @Component
8 public class FlooringContract {
9
10          private String tilesBrandName;
11          @Min(value=100, message="Total Square Feet should be minimum 100")
12          private double totalSqFt;
13          private double costPerSqFt;
14
15          public String getTilesBrandName() {
16                      return tilesBrandName;
17          }
18          public void setTilesBrandName(String tilesBrandName) {
19                      this.tilesBrandName = tilesBrandName;
20          }
21          public double getTotalSqFt() {
22                      return totalSqFt;
23          }
24          public void setTotalSqFt(double totalSqFt) {
25                      this.totalSqFt = totalSqFt;
26          }
27          public double getCostPerSqFt() {
28                      return costPerSqFt;
29          }
30          public void setCostPerSqFt(double costPerSqFt) {
31                      this.costPerSqFt = costPerSqFt;
32          }
33
34
35
36 }
37
38
```

FlooringCostEstimator/src/main/java/com/service/ContractService.java

```java
package com.service;

import org.springframework.stereotype.Service;

import com.model.FlooringContract;

@Service
public class ContractService {

	public double calculateFlooringCost(FlooringContract obj)
	{
		double tilesCost,flooringCost,avgCost;
		if(obj.getTotalSqFt() >=100 && obj.getTotalSqFt() <=400)
		{
			avgCost=42000.0;
		}
		else if(obj.getTotalSqFt() >400 && obj.getTotalSqFt() <=700)
		{
			avgCost=73500.0;
		}
		else if(obj.getTotalSqFt() >700 && obj.getTotalSqFt() <=1400)
		{
			avgCost=94500.0;
		}
		else
		{
			avgCost=147000.00;
		}
		if(obj.getTilesBrandName().equalsIgnoreCase("Kajaria"))
		{
			obj.setCostPerSqFt(180.0);
			tilesCost=obj.getCostPerSqFt() * obj.getTotalSqFt();
			flooringCost=tilesCost+avgCost;
		}
		else if(obj.getTilesBrandName().equalsIgnoreCase("Bajaj"))
		{
			obj.setCostPerSqFt(175.0);
			tilesCost=obj.getCostPerSqFt() * obj.getTotalSqFt();
			flooringCost=tilesCost+avgCost;
		}
		else if(obj.getTilesBrandName().equalsIgnoreCase("Nitco"))
		{
			obj.setCostPerSqFt(160.0);
			tilesCost=obj.getCostPerSqFt() * obj.getTotalSqFt();
			flooringCost=tilesCost+avgCost;
		}
		else if(obj.getTilesBrandName().equalsIgnoreCase("OrientBell"))
		{
			obj.setCostPerSqFt(155.0);
			tilesCost=obj.getCostPerSqFt() * obj.getTotalSqFt();
			flooringCost=tilesCost+avgCost;
		}
		else
		{
			obj.setCostPerSqFt(200.0);
			tilesCost=obj.getCostPerSqFt() * obj.getTotalSqFt();
			flooringCost=tilesCost+avgCost;
		}
		return flooringCost;
	}

}
```

FlooringCostEstimator/src/main/resources/application.properties

```
server.port=9068
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
```

FlooringCostEstimator/src/main/webapp/WEB-INF/views/estimatorpage.jsp

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1><center> Flooring Cost Estimator </center></h1>
<form:form method="post" action="result" modelAttribute="contract">

<table style="margin: 0px auto; margin-left: auto; margin-right:auto">

                                        <tr>
                                                        <td>Choose Tiles Brand:</td>
                                                        <td>
                                                         <form:select path="tilesBrandName" id="tilesBrandName">
                                                                                <form:options items="${brandNameList}"/>
                                                                        </form:select>

                                                        </td>
                                        </tr>

                                        <tr>
                                        <td>Enter Total Square Feet</td><td><form:input path="totalSqFt"
id="totalSqFt"/></td><td><form:errors path="totalSqFt"/></tr>
                                        <tr>

                                                        <td><input type="submit" value="CalculateCost" id="submit"
/></td>
                                                        <td><input type="reset" value="Cancel"/></td>
                                        </tr>

                        </table>
</form:form>


</body>
</html>

```

FlooringCostEstimator/src/main/webapp/WEB-INF/views/result.jsp

```
<%@page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<body bgcolor="lavender">
<h2>Approximate flooring cost for the given Square feet is Rs.${cost}</h2>
</body>
</html>
```

Grade

Reviewed on Thursday, 27 January 2022, 3:25 PM by Automatic grade

Grade 98.75 / 100

Assessment report

[-]SOURCE CODE ANALYZER REPORT

Error Msg: A method should have only one exit point, and that should be the last statement in the method
Variable Name:
Class Name: ContractController

Assessment Completed Successfully

[-]Grading and Feedback

Feature Test - 15.00 / 15.00(Success)
* * check whether Service is configured in Controller class with the required Annotation - 4.50 / 4.50*
* * check whether the Controller class with the required annotation is implemented - 3.50 / 3.50*
* * check whether the Service class with the required annotation is implemented - 3.50 / 3.50*
* * check whether the model class with the required annotation is implemented - 3.50 / 3.50*

Functional testing - 25.00 / 25.00(Success)
* * check whether the flooring cost is calculated as per the requirement for KajariaTiles - 5.00 / 5.00*
* * check whether the flooring cost is calculated as per the requirement for BajajTiles - 5.00 / 5.00*
* * check whether the flooring cost is calculated as per the requirement for NitcoTiles - 5.00 / 5.00*
* * check whether the flooring cost is calculated as per the requirement for OrientBellTiles - 5.00 / 5.00*
* * check whether the flooring cost is calculated as per the requirement for JohnsonTiles - 5.00 / 5.00*

Spring Testing in the Controller - 25.00 / 25.00(Success)
* * check whether the request is mapped for estimatorpage and redirected to estimatorpage.jsp - 5.00 / 5.00*
* * check whether the model attribute is specified above the buildState method that holds the Map containing the tilesBrandName - 5.00 / 5.00*
* * check whether the model attribute with the name brandNameList is specified above the buildState method that holds the Map containing the tilesBrandName - 5.00 / 5.00*
* * check whether layering is followed that is whether flooring cost is calculated by invoking the service method inside the Controller - 5.00 / 5.00*
* * check whether the validations are working correctly - 5.00 / 5.00*

UI Testing - 30.00 / 30.00(Success)
* * check whether UI from estimatorpage to result page is redirected after calculating the flooring cost for KajariaTiles - 5.00 / 5.00*
* * check whether UI from estimatorpage to result page is redirected after calculating the flooring cost for BajajTiles - 5.00 / 5.00*
* * check whether UI from estimatorpage to result page is redirected after calculating the flooring cost for NitcoTiles - 5.00 / 5.00*
* * check whether UI from estimatorpage to result page is redirected after calculating the flooring cost for OrientBellTiles - 5.00 / 5.00*
* * check whether UI from estimatorpage to result page is redirected after calculating the flooring cost for JohnsonTiles - 5.00 / 5.00*
* * check whether totalSqFt is rendered in estimatorpage as per the requirement - 2.00 / 2.00*
* * check whether tilesBrandName is rendered in estimatorpage as per the requirement - 3.00 / 3.00*

Comments and best practices/standards - 3.75 / 5.0(Partial)

Snap Fitness Center - V1

Grade settings: Maximum grade: 100
Based on: Snap Fitness Center - V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB **Maximum number of processes**: 10000

Snap Fitness center

Snap Fitness center is a famous GYM in the city. It offers various packages to the customers to join in their center. They are in need of a web application for their new customers to show their available packages and the estimated cost for it. Help them by developing an application

Create a Spring MVC Spring Boot Application for developing a Snap fitness canter Application. Design a Snap Enquiry page to choose an appropriate package **(â€œAerobicsâ€• , â€□ Zumbaâ€□ , â€œPersonalTrainingâ€□ , â€œYogaâ€□ , â€œRegularGYMâ€□)** and enter the Number of months

On clicking Package cost button, the application should calculate the package cost depending on the package type is chosen and the cost per month which the customers need to pay for the package The customer has to then be redirected to **snapEstimation.jsp** page that displays the message **â€œWelcome to Snap Fitness Center. Your preferred package cost is RS. <<cost>>â€•**

Application Work Flow:

- **FitnessController** is the Controller class.
- **FitnessCenter** is the model class with three attributes **preferredPackage**, **costPerMonth** and **numberOfMonths** along with its getters and setters.
- **FitnessService** is the Service class , has a method called calculatePackageCost that takes **FitnessCenter** as its argument and returns double.
- This method needs to set the **costPerMonth** instance variable based on the cost per month (in Rs) value given in the below mentioned table.
- Calculate the package cost depending on the costPerMonth for the preferredPackage chosen and the numberOfMonths.
- Package cost should be returned as double.

Package Type	cost per month(in Rs)
Aerobics	1500.0
Zumba	1750.0
RegularGYM	1250.0
PersonalTraining	2500.0
Yoga	1300.0

Initially, the customer should be routed via the FitnessController's snapEnquiryPage method to snapEnquirypage.jsp that allows user to choose the preferredPackage and enter the duration in hours.

[Note: snapEnquiryPage method has to be written inside the **FitnessController]**

A method in the FitnessController known as buildState should be annotated with the ModelAttribute **â€œpackageListâ€•** . This method should populate the preferred packages (Aerobics, Zumba,RegularGYM,PersonalTraining,Yoga) in the Map as key-value pair. Both key and value be the same preferredPackage. (Example: Key- Aerobics, Value- Aerobics, Key- Zumba, Value- Zumba, etc) and then return the Map. This should be then used to auto-populate the preferredPackage in the snapEnquiryPage.jsp

[Note: buildState method should be written inside the FitnessController]

On clicking the Package Cost button, the **FitnessController's calculatePackageCost** method should be called. This method takes three arguments - model attribute named â€œ**snap**â€• which holds the form populated **FitnessCenter** Object, BindingResult and the ModelMap.

- This method should calculate the package cost by invoking the **calculatePackageCost** method of the **FitnessService**.

Screen designs and Expected Output:

If the numberOfMonths entered by the customer is less than 1 then an error message **"Number of months should be minimum one"** has to be displayed in the snapEnquiryPage.jsp. To do this validation use appropriate annotation above **numberOfMonths** attribute in **FitnessCenter** model class.

snapEnquiryPage.jsp

- Redirect the user to **snapEstimation.jsp** page with a message **â€œWelcome to Snap Fitness center. Your estimated package cost is Rs. <<cost>>â€•**

snapEstimation.jsp

Welcome to Snap Fitness Center. Your preferred package cost is Rs. 3750.0

Design Constraints:

UI Design Constraints:

snapFitnesspage.jsp		
Component	**ID**	**Constraints**
Select	preferredPackage	Should be auto-populated using the model attribute written above the buildState method inside the FitnessController. Do not hard code the values
Textbox	numberOfMonths	Should not be less than 1
submit	submit (Name)	-

Note: In snapEstimation.jsp, the Result has to be rendered in the **<h2>** tag

Component Specification

Controller

FitnessController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**
FitnessService	service	private	Use annotation to Autowire

FitnessController			
Method Name	**Method Argument name:type**	**Return type**	**RequestMapping URL & Request Method**
snapEnquiryPage	modelAttribute â€œsnapâ€ • :FitnessCenter	String	/snapEnquiryPage & GET
calculatePackageCost	modelAttribute â€œsnapâ€ • :FitnessCenter , result:BindingResult, model:ModelMap	String	/snapEstimation & POST
buildState		Map <String,String>	Should be annotated with ModelAttribute with name â€œpackageListâ€ •

Service:

FitnessService		
Method Name	**Method Argument name:type**	**Return type**
calculatePackageCost	fitness :FitnessCenter	double

Model:

FitnessCenter	
AttributeName	**AttributeType**
preferredPackage	String
costPerMonth	double
numberOfMonths	int

Overall Design Constraints:

- FitnessController should be inside the package com.controller
- FitnessService should be inside the package com.service
- FitnessCenter should be in the package com.model
- Use appropriate annotation to configure FitnessService as a Service
- Use appropriate annotation to configure FitnessController as a Controller
- SnapFitnessService should be autowired inside the FitnessController.
- Use annotations to implement the business Validation as specified in the screenshot [That is when the duration in hours is less than 1 then error message **"Number of months should be minimum one"** should be rendered in the UI]
- Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.
- In the pom.xml you are provided with all the dependencies needed for developing the application.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.

- **Do not change or delete the class/method names or return types that are provided to you as a part of the base code skeleton.**
- Please make sure that your code does not have any compilation errors while submitting your case study solution.

Result Description

Submitted by sathya .

Automatic evaluation[+]

SnapFitness/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.1.8.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.example</groupId>
	<artifactId>BuffetRestaurant</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>BuffetRestaurant</name>
	<description>Demo project for Spring Boot</description>

	<properties>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		 <dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
   </dependency>
   <dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>jsp-api</artifactId>
  <version>2.1</version>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>taglibs</groupId>
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
<!-- Spring test dependencies -->
```

```
        <dependency>
                        <groupId>org.mockito</groupId>
                        <artifactId>mockito-all</artifactId>
                        <version>1.10.19</version>
                        <scope>test</scope>
                </dependency>

                <!-- <dependency>
                        <groupId>org.seleniumhq.selenium</groupId>
                        <artifactId>selenium-java</artifactId>
                        <version>2.53.0</version>
                </dependency>-->

                <dependency>
   <groupId>org.seleniumhq.selenium</groupId>
   <artifactId>htmlunit-driver</artifactId>
   <version>2.26</version>
</dependency>

        <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->
<dependency>
   <groupId>org.w3c.css</groupId>
   <artifactId>sac</artifactId>
   <version>1.3</version>
</dependency>


<dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-test</artifactId>
                         <version>5.1.7.RELEASE</version>  <!-- 4.0.5 -->
                        <scope>test</scope>
                </dependency>
<!-- End of spring test dependencies -->



                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                        <exclusions>
                                <exclusion>
                                        <groupId>org.junit.vintage</groupId>
                                        <artifactId>junit-vintage-engine</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

</project>

```

SnapFitness/src/main/java/com/controller/FitnessController.java

```
package com.controller;

import java.util.HashMap;
```

```
import java.util.Map;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.model.FitnessCenter;
import com.service.FitnessService;

@Controller
public class FitnessController {

	@Autowired
	private FitnessService service;

	@RequestMapping(value = "/snapEnquiryPage", method = RequestMethod.GET)
	public String showPage(@ModelAttribute("snap") FitnessCenter fitness) {
		System.out.println("in controller");
		return "snapEnquiryPage";
	}


	@ModelAttribute("packageList")
	public  Map<String, String> buildState(){

		Map<String, String> packMap = new HashMap<String, String>();
		packMap.put("Aerobics", "Aerobics");
		packMap.put("Zumba", "Zumba");
		packMap.put("RegularGYM","RegularGYM");
		packMap.put("Yoga", "Yoga");
		packMap.put("PersonalTraining", "PersonalTraining");

		return packMap;

	}

	@RequestMapping(value = "/packEstimation", method = RequestMethod.POST)
	public String calculatePackageCost(@ModelAttribute("snap") @Valid FitnessCenter fitness,
BindingResult result,
			ModelMap model) {

		if (result.hasErrors()) {
			System.out.println("Error : " + result.toString());
			return "snapEnquiryPage";
		}
		double cost=service.calculatePackageCost(fitness);
		model.addAttribute("cost", cost);
		return "snapEstimation";
	}


}

```

SnapFitness/src/main/java/com/example/demo/SnapFitnessApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
```

```
@SpringBootApplication
@ComponentScan({"com.*"})
public class SnapFitnessApplication {

	public static void main(String[] args) {
		SpringApplication.run(SnapFitnessApplication.class, args);
	}

}
```

SnapFitness/src/main/java/com/model/FitnessCenter.java

```
package com.model;

import javax.validation.constraints.Min;

import org.springframework.stereotype.Component;

@Component
public class FitnessCenter {

	private String preferredPackage;
	private double costPerMonth;
	@Min(value = 1, message = "Number of Months should be minimum one")
	private int numberOfMonths;


	public String getPreferredPackage() {
		return preferredPackage;
	}
	public void setPreferredPackage(String preferredPackage) {
		this.preferredPackage = preferredPackage;
	}
	public double getCostPerMonth() {
		return costPerMonth;
	}
	public void setCostPerMonth(double costPerMonth) {
		this.costPerMonth = costPerMonth;
	}
	public int getNumberOfMonths() {
		return numberOfMonths;
	}
	public void setNumberOfMonths(int numberOfMonths) {
		this.numberOfMonths = numberOfMonths;
	}




}
```

SnapFitness/src/main/java/com/service/FitnessService.java

```
package com.service;

import org.springframework.stereotype.Service;

import com.model.FitnessCenter;

@Service
public class FitnessService {

	public double calculatePackageCost (FitnessCenter fitness) {
```

```
11
12                      double packageCost=0.0;
13
14                      if(fitness.getPreferredPackage().equalsIgnoreCase("Aerobics") &&
fitness.getNumberOfMonths()>=1)
15                      {
16                              fitness.setCostPerMonth(1500.0);
17                              packageCost=fitness.getCostPerMonth()*fitness.getNumberOfMonths();
18                      }
19          else if(fitness.getPreferredPackage().equalsIgnoreCase("Zumba") &&
fitness.getNumberOfMonths()>=1 )
20          {
21                      fitness.setCostPerMonth(1750.0);
22                      packageCost=fitness.getCostPerMonth()*fitness.getNumberOfMonths();
23          }
24          else if(fitness.getPreferredPackage().equalsIgnoreCase("RegularGYM") &&
fitness.getNumberOfMonths()>=1)
25          {
26                      fitness.setCostPerMonth(1250.0);
27                      packageCost=fitness.getCostPerMonth()*fitness.getNumberOfMonths();
28          }
29          if(fitness.getPreferredPackage().equalsIgnoreCase("PersonalTraining") &&
fitness.getNumberOfMonths()>=1)
30          {
31                      fitness.setCostPerMonth(2500.0);
32                      packageCost=fitness.getCostPerMonth()*fitness.getNumberOfMonths();
33          }
34 else if(fitness.getPreferredPackage().equalsIgnoreCase("Yoga") && fitness.getNumberOfMonths()>=1)
35 {
36          fitness.setCostPerMonth(1300.0);
37          packageCost=fitness.getCostPerMonth()*fitness.getNumberOfMonths();
38 }
39          return packageCost;
40
41          }
42
43 }
44
```

SnapFitness/src/main/resources/application.properties

```
1 server.port=9088
2 spring.mvc.view.prefix = /WEB-INF/views/
3 spring.mvc.view.suffix = .jsp
4 spring.mvc.static-path-pattern=/resources/**
5
```

SnapFitness/src/main/webapp/WEB-INF/views/snapEnquiryPage.jsp

```
1 <%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
2 <%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
3 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
4 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
5    pageEncoding="ISO-8859-1" isELIgnored="false"%>
6 <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
7
8 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
9 <html>
10 <body style="background-color:lavender">
11 <h1><center> Snap Fitness Center </center></h1>
12 <form:form method="post" action="packEstimation" modelAttribute="snap">
13 <table style="margin: 0px auto; margin-left: auto; margin-right:auto">
14
15                      <tr>
16                                      <td>Select Package</td>
17                                      <td>
```

```
                                        <form:select path="preferredPackage" id="preferredPackage">
                                                        <form:options items="${packageList}"/>
                                                </form:select>

                                        </td>
                                </tr>

                                <tr>
                                <td>Package duration (In months)</td>
                                <td><form:input path="numberOfMonths" id="numberOfMonths"/></td>
                                <td><form:errors path="numberOfMonths" id="errormsg"/></td>
                                </tr>
                                <tr>
                                        <td><input type="submit" value="Package Cost" name="submit"
/></td>

                                </tr>

                        </table>
</form:form>


</body>
</html>

```

SnapFitness/src/main/webapp/WEB-INF/views/snapEstimation.jsp

```
<%@page isELIgnored="false" %>
<html>
<body bgcolor="lavender">
<h2>Welcome to Snap Fitness Center. Your preferred package cost is Rs. ${cost} </h2>
</body>
</html>
```

Grade

Reviewed on Friday, 18 February 2022, 1:09 PM by Automatic grade
Grade 100 / 100
Assessment report
Assessment Completed Successfully
[-]**Grading and Feedback**

```
Feature Test - 7.50 / 7.50(Success)
        * check whether the Controller / Service class with the required annotation is
implemented - 2.50 / 2.50
        * check whether the Controller class with the required annotation is implemented
- 1.50 / 1.50
        * check whether the Service class with the required annotation is implemented -
1.50 / 1.50
        * check whether the model class with the required annotation is implemented -
2.00 / 2.00

Functional testing - 30.00 / 30.00(Success)
        * check whether the package cost for the Aerobics type is calculated properly -
6.00 / 6.00
        * check whether the package cost for the Zumba type is calculated properly - 6.00
/ 6.00
        * check whether the package cost for the Yoga type is calculated properly - 6.00
/ 6.00
        * check whether the package cost for the PersonalTraining type is calculated
properly - 6.00 / 6.00
        * check whether the package cost for the RegularGYM type is calculated properly -
6.00 / 6.00

Spring Testing in the Controller - 45.00 / 45.00(Success)
```

** check whether the request is mapped for snapEnquiryPage and redirected to snapEnquiryPage.jsp page - 6.00 / 6.00*
** check whether the request is mapped for snapEstimation and redirected to snapEstimation.jsp page - 6.00 / 6.00*
** check whether the model attribute is specified with the name above the buildState method that holds the Map containing the Preferred package - 6.00 / 6.00*
** check whether the buildstate method is configured properly - 7.00 / 7.00*
** check whether the request is mapped and redirected to the snapEstimation page for valid details - 10.00 / 10.00*
** check whether the request is mapped and redirected to the same page for Invalid details - 10.00 / 10.00*

UI Testing - 17.50 / 17.50(Success)
** Check whether preferredPackage Aerobics is autopopulated as per the requirement - 3.00 / 3.00*
** Check whether preferredPackage Zumba is autopopulated as per the requirement - 3.00 / 3.00*
** Check whether preferredPackage PersonalTraining is autopopulated as per the requirement - 3.00 / 3.00*
** Check whether preferredPackage Yoga is autopopulated as per the requirement - 5.00 / 5.00*
** Check whether preferredPackage RegularGYM is autopopulated as per the requirement - 0.50 / 0.50*
** check whether the snapEnquiryPage jsp is rendered with proper constraints for numberOfMonths - 0.50 / 0.50*
** check whether UI from snapEnquiryPage to snapEstimator page is redirected after calculating the package cost for Zumba - 0.50 / 0.50*
** check whether UI from snapEnquiryPage to snapEstimator page is redirected after calculating the package cost for PersonalTrianing - 0.50 / 0.50*
** check whether UI from snapEnquiryPage to snapEstimator page is redirected after calculating the package cost for Aerobics - 1.00 / 1.00*
** check whether the numberOfMonths field validated properly - 0.50 / 0.50*

Test Case Passed

Digital Home Brokering - V1

Grade settings: Maximum grade: 100
Based on: Digital Home Brokering - V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB **Maximum number of processes**: 10000

Digital Home Broker

Click here to download the code skeleton

Develop a web application for Digital home broker. Through this application anyone can search rented home/apartments based on location, BHK type and rent.

Create a Spring MVC Spring Boot Application for developing a Digital home broker Application. Design a search page to search homes/apartments based on location,bhkType and rent

On clicking search button, the application should list the available home/apartments based on the given search criteria. The customer has to then be redirected to homeListPage.jsp page that displays the list of available homes in a tabular format.

Application Work Flow:

- **BrokerController** is the Controller class.
- **Home** is the model class with attributes **location,address,bhkType,contactPerson,contactNumber and rent** along with its getters and setters.
- BrokerService is the Service class which has a method called **searchHome** that takes **Home** as its argument and returns ArrayList<Home>.
- **Attribute ArrayList<Home> list;** defined in service class and the value for this list was populated through **addHomes()** method. Donâ€™t change the code in this method.
- Through **searchHome** method, have to search the **Home** objects based on the given input location, bhkType and rent. (Iterate the list, get the Home objects which have the same location,bhkType and rent entered by customer)

Initially, the customer should be routed via the **BrokerController's searchPage** method to searchPage.jsp that allows user to choose the location, bhkType and rent.

[Note: searchPage method has to be written inside the **BrokerController]**

A method in the **BrokerService** known as **addHomes** is already available. which is used to add the **Home** objects into the ArrayList<Home>

On clicking the program search button, the **BrokerController's searchHome** method should be called. This method takes three arguments - model attribute named â€œ**homeBean**â€• which holds the form populated **Home** Object, BindingResult and the ModelMap.

- This method should search the Home objects from the list based on the location, bhkType and rent. **(Should be case insensitive search for location and bhkType)**

Screen designs and Expected Output:

If the rent entered by the customer is less than 5000 then an error message **"Minimum rent should be 5000"** has to be displayed in the searchPage.jsp. To do this validation use appropriate annotation above **rent** attribute in **Home** model class.

searchPage.jsp

homeListPage.jsp

localhost:9088/homeList

Digital Home Broker

Search homes|Apartments !!!

Location	BhkType	Address	ContactPerson
Chennai	2BHK	#212/4, 10th cross, T-nagar, PIN-60008	Annesh
Chennai	2BHK	#556/4, 5th cross, Thambaram, PIN-60010	Jisha
Chennai	2BHK	#556/4,Nila paradise, Thambaram, PIN-60010	Jisha

- If there are no Home objects available based on the search then display the message **â€œNo home/Apartment found for this searchâ€•** in **homeListPage.jsp**

localhost:9088/homeList

Digital Home Broker

Search homes|Apartments !!!

No home/Apartment found for this searc

Design Constraints:

UI Design Constraints:

searchPage.jsp		
Component	**ID**	**Constraints**
Textbox	location	Name and id attribute should be â€œlocationâ€ •
Textbox	bhkType	Name and id attribute should be â€œbhkTypeâ€ •
Textbox	rent	Should not be less than 5000
submit	submit	Name and id attribute value should be submit

Component Specification

Controller

BrokerController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**
BrokerService	service	private	Use annotation to Autowire

BrokerController			
Method Name	**Method Argument name:type**	**Return type**	**RequestMapping URL & Request Method**
searchPage	modelAttribute â€œhomeBeanâ€ • :Home	String	/searchPage & GET
searchHome	modelAttribute â€œhomeBeanâ€ • :Home, result:BindingResult, model:ModelMap	String	/homeList & POST

Service:

BrokerService		
Method Name	**Method Argument name:type**	**Return type**
searchHome	homeBean:Home	ArrayList<Home>

Model:

Home	
AttributeName	**AttributeType**
location	String
address	String
bhkType	String
contactPerson	String
contactNumber	String
rent	double

Overall Design Constraints:

- **BrokerController classshould** be inside the package **com.controller**

- **BrokerService** class should be inside the package **com.service**

- **Home** class should be in the package **com.model**
- Use appropriate annotation to configure **BrokerService** as a Service
- Use appropriate annotation to configure **BrokerController** as a Controller
- **BrokerService** should be autowired inside the **BrokerController**.
- Use annotations to implement the business Validation as specified in the screenshot [That is when the rent is less than 5000 then error message **"Minimum rent should be 5000"** should be rendered in the UI.]
- Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.
- In the pom.xml you are provided with all the dependencies needed for developing the application.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.

- **Do not change or delete the class/method names or return types that are provided to you as a part of the base code skeleton.**
- Please make sure that your code does not have any compilation errors while submitting your case study solution.

Result Description

Submitted by sathya .

Automatic evaluation[+]

DigitalHomeBroker/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
	<version>2.1.8.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.example</groupId>
	<artifactId>DigitalHomeBroker</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>FlawlessAcademy</name>
	<description>Demo project for Spring Boot</description>
	<properties>
		<java.version>1.8</java.version>
	</properties>

	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
```

```
23                                      <artifactId>spring-boot-starter-web</artifactId>
24                          </dependency>
25
26                           <dependency>
27     <groupId>org.apache.tomcat.embed</groupId>
28     <artifactId>tomcat-embed-jasper</artifactId>
29     <scope>provided</scope>
30   </dependency>
31
32   <dependency>
33   <groupId>javax.servlet</groupId>
34   <artifactId>servlet-api</artifactId>
35   <version>2.5</version>
36   <scope>provided</scope>
37 </dependency>
38 <dependency>
39   <groupId>javax.servlet.jsp</groupId>
40   <artifactId>jsp-api</artifactId>
41   <version>2.1</version>
42   <scope>provided</scope>
43 </dependency>
44
45 <dependency>
46   <groupId>taglibs</groupId>
47   <artifactId>standard</artifactId>
48   <version>1.1.2</version>
49 </dependency>
50 <dependency>
51   <groupId>javax.servlet</groupId>
52   <artifactId>jstl</artifactId>
53   <version>1.2</version>
54 </dependency>
55 <!-- Spring test dependencies -->
56          <dependency>
57                                      <groupId>org.mockito</groupId>
58                                      <artifactId>mockito-all</artifactId>
59                                      <version>1.10.19</version>
60                                      <scope>test</scope>
61                          </dependency>
62
63                          <!-- <dependency>
64                                      <groupId>org.seleniumhq.selenium</groupId>
65                                      <artifactId>selenium-java</artifactId>
66                                      <version>2.53.0</version>
67                          </dependency>-->
68
69                          <dependency>
70     <groupId>org.seleniumhq.selenium</groupId>
71     <artifactId>htmlunit-driver</artifactId>
72     <version>2.26</version>
73 </dependency>
74
75          <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->
76 <dependency>
77     <groupId>org.w3c.css</groupId>
78     <artifactId>sac</artifactId>
79     <version>1.3</version>
80 </dependency>
81
82
83 <dependency>
84                                      <groupId>org.springframework</groupId>
85                                      <artifactId>spring-test</artifactId>
86                                       <version>5.1.7.RELEASE</version>  <!-- 4.0.5 -->
87                                      <scope>test</scope>
88                          </dependency>
89 <!-- End of spring test dependencies -->
```

```
90
91
92
93                      <dependency>
94                              <groupId>org.springframework.boot</groupId>
95                              <artifactId>spring-boot-starter-test</artifactId>
96                              <scope>test</scope>
97                              <exclusions>
98                                      <exclusion>
99                                              <groupId>org.junit.vintage</groupId>
100                                             <artifactId>junit-vintage-engine</artifactId>
101                                     </exclusion>
102                             </exclusions>
103                     </dependency>
104         </dependencies>
105
106         <build>
107                     <plugins>
108                             <plugin>
109                                     <groupId>org.springframework.boot</groupId>
110                                     <artifactId>spring-boot-maven-plugin</artifactId>
111                             </plugin>
112                     </plugins>
113         </build>
114
115 </project>
116
```

DigitalHomeBroker/src/main/java/com/controller/BrokerController.java

```
1 package com.controller;
2
3 import java.util.ArrayList;
4 import javax.validation.Valid;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.ui.ModelMap;
8 import org.springframework.validation.BindingResult;
9 import org.springframework.web.bind.annotation.ModelAttribute;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RequestMethod;
12
13 import com.model.Home;
14 import com.service.BrokerService;
15
16 @Controller
17 public class BrokerController {
18
19         @Autowired
20         private BrokerService service;
21
22         @RequestMapping(value = "/searchPage", method = RequestMethod.GET)
23         public String searchPage(@ModelAttribute("homeBean") Home homeBean) {
24                         System.out.println("in controller");
25                         return "searchPage";
26         }
27
28         @RequestMapping(value = "/homeList", method = RequestMethod.POST)
29         public String searchHome(@ModelAttribute("homeBean")@Valid Home homeBean, BindingResult
result,
30                                 ModelMap model) {
31                         String str="";
32                         if (result.hasErrors()) {
33                                 System.out.println("Error : " + result.toString());
34                         str="searchPage";
35                         }else{
36                         ArrayList<Home> list=service.searchHome(homeBean);
```

```
37                      model.addAttribute("homeList", list);
38                      str= "homeListPage";
39                      }
40                      return str;
41          }
42 }
43
```

DigitalHomeBroker/src/main/java/com/example/demo/DigitalHomeBrokerApplication.java

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.context.annotation.ComponentScan;
6
7 @SpringBootApplication
8 @ComponentScan({"com.*"})
9 public class DigitalHomeBrokerApplication {
10
11          public static void main(String[] args) {
12                      SpringApplication.run(DigitalHomeBrokerApplication.class, args);
13          }
14
15 }
16
```

DigitalHomeBroker/src/main/java/com/model/Home.java

```
1 package com.model;
2
3 import javax.validation.constraints.Min;
4
5 import org.springframework.stereotype.Component;
6
7 @Component
8 public class Home {
9
10          private String location;
11          private String address;
12          private String contactNumber;
13          private String bhkType;
14          private String contactPerson;
15          @Min (value=5000, message="Minimum rent should be 5000")
16          private double rent;
17          public double getRent() {
18                      return rent;
19          }
20          public void setRent(double rent) {
21                      this.rent = rent;
22          }
23          public String getLocation() {
24                      return location;
25          }
26          public void setLocation(String location) {
27                      this.location = location;
28          }
29          public String getAddress() {
30                      return address;
31          }
32          public void setAddress(String address) {
33                      this.address = address;
34          }
35          public String getContactNumber() {
36                      return contactNumber;
37          }
```

```
	public void setContactNumber(String contactNumber) {
		this.contactNumber = contactNumber;
	}
	public String getBhkType() {
		return bhkType;
	}
	public void setBhkType(String bhkType) {
		this.bhkType = bhkType;
	}
	public String getContactPerson() {
		return contactPerson;
	}
	public void setContactPerson(String contactPerson) {
		this.contactPerson = contactPerson;
	}

	public Home(String location, String address, String contactNumber, String bhkType, String contactPerson,double rent) {
		super();
		this.location = location;
		this.address = address;
		this.contactNumber = contactNumber;
		this.bhkType = bhkType;
		this.contactPerson = contactPerson;
		this.rent=rent;
	}
	public Home()
	{}

}
```

DigitalHomeBroker/src/main/java/com/service/BrokerService.java

```
package com.service;

import java.util.ArrayList;
import org.springframework.stereotype.Service;
import com.model.Home;


@Service
public class BrokerService {
	ArrayList<Home> list = new ArrayList<Home>();

	public ArrayList<Home> searchHome(Home homeBean) {
		ArrayList<Home> newList=new ArrayList<Home>();
	System.out.println(list);

	for(int i=0;i<list.size();i++) {
		if(list.get(i).getLocation().equalsIgnoreCase(homeBean.getLocation()) && list.get(i).getBhkType().equalsIgnoreCase(homeBean.getBhkType())&& list.get(i).getRent()==homeBean.getRent()) {
			newList.add(list.get(i));
		}
	}
		return newList;

	}

	public BrokerService() {
		addHomes();
	}

	public  ArrayList<Home> addHomes(){
```

```
30                      list.add(new Home("Coimbatore","#412, Shipitha cascade, Ganapathy, PIN-
641001","9917678456","1BHK","Malavika",8000));
31                      list.add(new Home("Coimbatore","#12, shardhalotus, Ram Nagar, PIN-
641001","9917100989","2BHK","Karthik",12000));
32                      list.add(new Home("Coimbatore","#12, Anandha nilayam,Kuniamuthur, PIN-
641001","9917100989","2BHK","Karthik",12000));
33                      list.add(new Home("Coimbatore","#201, Shipitha cascade, Saravanampatti, PIN-
641001","9917123456","3BHK","Sowparnika",17000));
34
35                      list.add(new Home("Chennai","#34/45, RV Homes, 5th cross, Anssari street, PIN-
600001","7788900989","1BHK","Mukunthan",6000));
36                      list.add(new Home("Chennai","#212/4, 10th cross, T-nagar, PIN-
60008","9917100989","2BHK","Annesh",14000));
37                      list.add(new Home("Chennai","#556/4, 5th cross, Thambaram, PIN-
60010","9917100989","2BHK","Jisha",14000));
38                      list.add(new Home("Chennai","#556/4,Nila paradise, Thambaram, PIN-
60010","9917100989","2BHK","Jisha",14000));
39                      list.add(new Home("Chennai","#412/4, 10th cross, Velacheri, PIN-
60009","9917156989","3BHK","Rakesh",18000));
40
41                      list.add(new Home("Bangalore","#45, 12Th main , GM Palya, PIN-
560075","9917122119","1BHK","Senthil",11000));
42                      list.add(new Home("Bangalore","#67/2, 3rd cross , Indra nagar, PIN-
560077","9944550989","2BHK","RamKumar",17000));
43                      list.add(new Home("Bangalore","#302, Sakthi enclave ,1st main , Malleshpalya, PIN-
560075","9917668889","3BHK","Saradha",25000));
44                      return list;
45
46          }
47 }
48
49
```

DigitalHomeBroker/src/main/resources/application.properties

```
1 server.port=9088
2 spring.mvc.view.prefix = /WEB-INF/views/
3 spring.mvc.view.suffix = .jsp
4 spring.mvc.static-path-pattern=/resources/**
5
```

DigitalHomeBroker/src/main/webapp/WEB-INF/views/homeListPage.jsp

```
1 <%@page isELIgnored="false" %>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
4 <html>
5 <body bgcolor="lavender">
6 <h1><center> Digital Home Broker </center></h1>
7 <h2><center>Search homes|Apartments !!!  </center></h2>
8 <center>
9                                   <c:if test="${fn:length(homeList) == 0}">
10                                  <center><h2> No home/Apartment found for this search</h2></center>
11                                  </c:if>
12                                  <c:if test="${fn:length(homeList) > 0}">
13                                  <table border="1" cellpadding="6">
14                                  <tr>
15                                              <th>Location</th>
16                                              <th>BhkType</th>
17                                              <th>Address</th>
18                                              <th>ContactPerson</th>
19                                              <th>ContactNumber</th>
20                                              <th>Rent</th>
21                                  </tr>
22                                  <c:forEach var="obj" items="${homeList}">
23
24                                              <tr>
```

```
                                                <td>${obj.location}</td>
                                                <td>${obj.bhkType}</td>
                                                <td>${obj.address }</td>
                                                <td>${obj.contactPerson }</td>
                                                <td>${obj.contactNumber }</td>
                                                <td>${obj.rent }</td>
                                        </tr>
                                </c:forEach>
                        </table></center>
                        </c:if>
</body>
</html>

```

DigitalHomeBroker/src/main/webapp/WEB-INF/views/searchPage.jsp

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false"%>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<body style="background-color:lavender">
<h1><center> Digital Home Broker </center></h1>
<h2><center>Search homes|Apartments !!!  </center></h2>

<form:form method="post" action="/homeList" modelAttribute="homeBean">
<table style="margin: 0px auto; margin-left: auto; margin-right:auto">

                                <tr>
                                        <td>Search location</td>
                                        <td><form:input path="location" id="location" /></td>
                                </tr>
                                <tr>
                                <td> Search BHK Type</td>
                                <td><form:input path="bhkType" id="bhkType"/></td>
                                <tr>
                                <tr>
                                <td>Preferred Rent</td>
                                <td><form:input path="rent" id="rent"/></td>
                        <td><form:errors path="rent" id="errormsg"/></td>
                                <tr>
                                        <td><input type="submit" value="Search" name="submit" /></td>

                                </tr>

                        </table>
</form:form>


</body>
</html>

```

Grade

Reviewed on Friday, 18 February 2022, 12:20 AM by Automatic grade
Grade 100 / 100
Assessment report
Assessment Completed Successfully
[+]**Grading and Feedback**

Farm Fresh Online store - V1

Grade settings: Maximum grade: 100
Based on: Farm Fresh Online store - V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB **Maximum execution file size**: 1.50 GiB **Maximum number of processes**: 10000

Farm-Fresh online store

FarmFresh-A natural farm store, sells organic farm fresh fruits through online, they approach you to create a separate online application for calculating total bill amount for the order placed by the customer.

Create a Spring MVC Spring Boot Application for developing an Online Application. Design a showPage to choose an appropriate product**(Apple/Mango/Orange/Grapes/JackFruit/)**and enter the quantity in kg.

On clicking CalculateCost button, the application should calculate the total bill amount based on the product chosen for the quantity. The customer has to then be redirected to billDesk.jsp page that displays the product chosen and its cost.

Application Work Flow:

- **ShoppingController** is the Controller class.
- **Product** is the model class with 3 attributes productName, quantity, and costPerKg along with its getters and setters.
- **ShoppingService** is the Service class which has a method called **calculateCost** that takes **Product** as its argument and returns double.
- **ShoppingService** class has getProductStock() method. It return the Map <String,Integer>. The product and its quantity is shown in the below table

productName	Quantity
Apple	50
Orange	30
JackFruit	25
Mango	75
Grapes	10

o Compare the quantity entered by the user with the quantity in the Map for the relevant product and check if the needed quantity is available. If available, then calculate the bill amount. Else throw a user defined exception NoStockException with a message **â€œNo enough stock for product <<productName>>â€•**

o This method should set the value for **costPerKg** instance variable based on productName. To do this consider the below table.

Product Name	Cost Per Kg
Apple	250.00
Orange	90.00
JackFruit	75.00
Mango	60.00
Grapes	150.00

o Output should be returned as double.

Initially, the customer should be routed via the **ShoppingControllerâ€™s showPage** method to **showPage.jsp** that allows customer to choose the product and enter the quantity in kg.

[Note: showPage method has to be written inside the ShoppingController]

A method in the **ShoppingController** known as populateProduct should be annotated with the ModelAttribute â€œ**productNames**â€• . This method should populate the product name**(Apple/Orange/Grapes/Mango/JackFruit) in** the List<String>. This should be then used to auto-populate the productName in the **showPage.jsp**

[Note:populateProduct method should be written inside the ShoppingController]

On clicking the CalculateCost button, the **ShoppingControllerâ€™s calculateCost** method should be called. This method takes three arguments - model attribute named â€œ**product**â€• which holds the form populated Product, BindingResult and the ModelMap.

- This method should calculate the totalCost by invoking the **calculateCost** method of the **ShoppingService**.
- totalCost has to be calculated as shown in the below example,

Example:

If customer choose product as â€œOrangeâ€• and quantity as 15, set costPerKg=90.0, Check the stock Map. For orange stock has the quantity of 30 kg.

cost = quantity *costPerKg= 15 *90 = 1350.0

If customer choose product as â€œGrapesâ€• and quantity as 25, then check the quantity in the stock map. For Grapes the quantity is 10Kg. So raise NoStockException with message **â€œNo enough stock for product <<productName>>â€•**

- If the quantity entered by the customer is less than 1 then an error message **â€œMinimum Quantity should be 1Kg"** has to be displayed.

showPage.jsp

- If the quantity entered by the customer is less than 1 then an error message **â€œMinimum quantity should be 1 Kg**" has to be displayed.

Redirect the customer to billDesk.jsp and display the output as like in the below screenshot

billDesk.jsp

exceptionPage.jsp

Design Constraints:

UI Design Constraints:

showPage.jsp		
Component	**ID**	**Constraints**
DropDown	productName	Should be auto-populated using the model attribute written above the **populateProduct** method inside the **ShoppingController**. Do not hard code the values
Textbox	quantity	Minimum quantity should be 1Kg
submit	submit	Both id and name attribute value should be "submit"

Component Specification

Controller

ShoppingController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**
ShoppingService	ShoppingService	private	Use annotation to Autowire

ShoppingController			
Method Name	**Method Argument name: type**	**Return type**	**RequestMapping URL & Request Method**
showPage	modelAttribute â€œproductâ€ • :Product	String	/showPage& GET
calculateCost	modelAttribute â€œproductâ€ • :Product, result:BindingResult, model:ModelMap	String	/calculate& POST
populateProduct		List <String>	Should be annotated with ModelAttribute with name â€œ**productNames**â€ •
exceptionHandler	Exception exception		Use appropriate annotation to handle the exception

Service

ShoppingService		
Method Name	**Method Argument name: type**	**Return type**
calculateCost	product:Product	Double
getProductStock		Map<String,Integer>

Model

Product	
AttributeName	**AttributeType**
productName	String
costPerKg	double
quantity	int

Overall Design Constraints:

Â· **ShoppingController** should be inside the package **com.controller**

Â· **ShoppingService** should be inside the package **com.service**

Â· **Product** should be in the package **com.model**

Â· Use appropriate annotation to configure ShoppingService as a Service

Â· Use appropriate annotation to configure ShoppingControlleras a Controller

Â· **ShoppingService** should be autowired inside the **ShoppingController**.

Â· Use annotations to implement the business Validation as specified in the screen shot [That is, when the quantity is less than 1then an error message **"Minimum quantity should be 1 Kg"** should be rendered in the UI]

Â· Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.

Â· In the pom.xml you are provided with all the dependencies needed for developing the application.

Â· You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.

Â· Adhere to the design specifications mentioned in the case study.

Â· Do not change or delete the class/method names or return types which are provided to you as a part of the base code skeleton.

Â· Please make sure that your code does not have any compilation errors while submitting your case study solution.

Flawless Academy - V1

Grade settings: Maximum grade: 100
Based on: Flawless Academy - V1
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 240 s **Maximum memory used**: 1.50 GiB **Maximum execution file size**: 1.50 GiB

Flawless Academy

Click here to download the code skeleton

Flawless Academy offers programs for dance, vocals, and instruments. They are in need of a web application to customers to show their available packages and the estimated cost for it. Help them by developing an application

Create a Spring MVC Spring Boot Application for developing a flawless Academy Application. Design a enrollment page to choose an appropriate package **(â€œClassicalDanceâ€• , â€□ KarnaticVocalsâ€□ , â€œWesternDanceâ€□ , â€œDrawingâ€□ , â€œInstrumentsâ€•)** and enter the Number of sessions per week

On clicking program cost button, the application should calculate the program cost depending on the program chosen and the cost per session which the customers need to pay for the package The customer has to then be redirected to estimationPage.jsp page that displays the message **â€œ**Welcome to Flawless Academy. Your preferred program cost is Rs. **<<cost>>â€•**

Application Work Flow:

- **AcademyController** is the Controller class.
- **Academy** is the model class with three attributes program, costPerSession, sessionsPerWeek and weeksPerMonth along with its getters and setters.
- **AcademyService** is the Service class which has a method called calculateProgramCost that takes Academy object as its argument and returns double.
- This method needs to set the **costPerSession** instance variable based on the cost per session (in Rs) value given in the below mentioned table.
- Calculate the program cost depending on the **costPerSession** for the program chosen and the **sessionsPerWeek**.
- By default the **weeksPerMonth** value should be 4.
- Program cost should be returned as double.

Program	Cost per session (in Rs)
ClassicalDance	150.0
KarnaticVocals	100.0
WesternDance	125.0
Drawing	130.0
Instruments	200.0

Initially, the customer should be routed via the AcademyController's **showPage** method to e**nrollmentPage.jsp** that allows user to choose the program and enter the **sessionsPerWeek**

[Note: showPage method has to be written inside the AcademyController]

A method in the AcademyController known as **buildState** should be annotated with the ModelAttribute **â€œprogramListâ€•** . This method should populate the preferred

programs **(ClassicalDance, KarnaticVocals,WesternDance,Drawing,Instruments)** in the Map as key-value pair. Both key and value be the same program. (Example: Key- ClassicalDance, Value- ClassicalDance, Key- KarnaticVocals, Value- KarnaticVocals, etc) and then return the Map. This should be then used to auto-populate the program in the **enrollmentPage**.jsp

[Note: buildState method should be written inside the AcademyController]

On clicking the program Cost button, the AcademyController's calculateProgramCost method should be called. This method takes three arguments - model attribute named â€œ**academyBean**â€• which holds the form populated Academy Object, BindingResult and the ModelMap.

- This method should calculate the program cost by invoking the **calculateProgramCost** method of the **AcademyService**.

Calculation of program cost:
If the preferred program is "Drawing" and sessionsPerWeek is 3.
program cost= sessionsPerWeek* costPerSession * weeksPerMonth (by default weeksPerMonth is 4)
program cost= 3 * 130.0 * 4 => 1560.0

If the **sessionsPerWeek** entered by the customer is less than 2 then an error message **"Minimum 2 Sessions/Week"** has to be displayed in the **enrollmentPage**.**jsp**. If the **sessionsPerWeek** entered by the customer is greater than 5 then an error message **"Maximum 5 Sessions/Week"** has to be displayed in the **enrollmentPage**.**jsp** To do this validation use appropriate annotation above **sessionsPerWeek** attribute in **Academy** model class.

Screen designs and Expected Output:

enrollmentPage.jsp

- Redirect the user to **estimationPage.jsp** page with a message **â€œWelcome to Flawless Academy. Your preferred program cost is Rs. << cost>>â€**

estimationPage.jsp

Design Constraints:

UI Design Constraints:

enrollmentPage.jsp		
Component	**ID**	**Constraints**
Select	program	Should be auto-populated using the model attribute written above the buildState method inside the AcademyController. Do not hard code the values
Textbox	sessionsPerWeek	Should not be less than 2 and should not be greater than 5
submit	submit	Name and id attribute should be submit -

Note: In **estimationPage.jsp**, the Result has to be rendered in the **<h2>** tag

Component Specification

Controller

AcademyController			
AttributeName	**AttributeType**	**Access Specifier**	**Constraints**
AcademyService	service	private	Use annotation to Autowire

AcademyController			
Method Name	**Method Argument name:type**	**Return type**	**RequestMapping URL & Request Method**

showPage	modelAttribute â€œacademyBeanâ€ • : Academy	String	/enrollmentPage & GET
calculateProgramCost	modelAttribute â€œacademyBeanâ€ • :Academy, result:BindingResult, model:ModelMap	String	/progEstimation & POST
buildState		Map <String,String>	Should be annotated with ModelAttribute with name â€œprogramListâ€ •

Service:

AcademyService		
Method Name	**Method Argument name:type**	**Return type**
calculateProgramCost	academyBean:Academy	double

Model:

Academy	
AttributeName	**AttributeType**
program	String
costPerSession	double
sessionsPerWeek	int

Overall Design Constraints:

- **AcademyController** should be inside the package **com.controller**
- **AcademyService** should be inside the package **com.service**
- **Academy** should be in the package **com.model**
- Use appropriate annotation to configure **AcademyService** as a Service
- Use appropriate annotation to configure **cademyController** as a Controller
- **AcademyService** should be **autowired** inside the**AcademyController**.
- Use annotations to implement the business Validation as specified in the screenshot [That is when the **sessionsPerWeek** is less than 2 then error message **"Minimum 2 Sessions/Week"** and when sessionsPerWeek greater than 5 then error message **â€œMaximum 5 Sessions/Weekâ€•** should be rendered in the UI.]
- Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.
- In the pom.xml you are provided with all the dependencies needed for developing the application.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.

- **Do not change or delete the class/method names or return types that are provided to you as a part of the base code skeleton.**
- Please make sure that your code does not have any compilation errors while submitting your case study solution.

Result Description

Submitted by sathya .

Automatic evaluation[+]

FlawlessAcademy/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.8.RELEASE</version>
            <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>FlawlessAcademy</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>FlawlessAcademy</name>
    <description>Demo project for Spring Boot</description>
    <properties>
            <java.version>1.8</java.version>
    </properties>

    <dependencies>
            <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-web</artifactId>
            </dependency>

             <dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
  </dependency>
  <dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>jsp-api</artifactId>
  <version>2.1</version>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>taglibs</groupId>
```

```
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
<!-- Spring test dependencies -->
     <dependency>
                                <groupId>org.mockito</groupId>
                                <artifactId>mockito-all</artifactId>
                                <version>1.10.19</version>
                                <scope>test</scope>
                    </dependency>

                    <!-- <dependency>
                                <groupId>org.seleniumhq.selenium</groupId>
                                <artifactId>selenium-java</artifactId>
                                <version>2.53.0</version>
                    </dependency>-->

                    <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>htmlunit-driver</artifactId>
    <version>2.26</version>
</dependency>

     <!-- https://mvnrepository.com/artifact/org.w3c.css/sac -->
<dependency>
    <groupId>org.w3c.css</groupId>
    <artifactId>sac</artifactId>
    <version>1.3</version>
</dependency>


<dependency>
                                <groupId>org.springframework</groupId>
                                <artifactId>spring-test</artifactId>
                                 <version>5.1.7.RELEASE</version>  <!-- 4.0.5 -->
                                <scope>test</scope>
                    </dependency>
<!-- End of spring test dependencies -->



                    <dependency>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-starter-test</artifactId>
                                <scope>test</scope>
                                <exclusions>
                                            <exclusion>
                                                        <groupId>org.junit.vintage</groupId>
                                                        <artifactId>junit-vintage-engine</artifactId>
                                            </exclusion>
                                </exclusions>
                    </dependency>
                    <!-- https://mvnrepository.com/artifact/org.sonarsource.scanner.maven/sonar-maven-plugin -->
<dependency>
    <groupId>org.sonarsource.scanner.maven</groupId>
    <artifactId>sonar-maven-plugin</artifactId>
    <version>3.8.0.2131</version>
```

```
</dependency>

    </dependencies>

    <build>
                <plugins>
                            <plugin>
                                        <groupId>org.springframework.boot</groupId>
                                        <artifactId>spring-boot-maven-plugin</artifactId>
                            </plugin>
                </plugins>
    </build>

</project>

```

FlawlessAcademy/src/main/java/com/controller/AcademyController.java

```
package com.controller;

import java.util.HashMap;
import java.util.Map;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.model.Academy;
import com.service.AcademyService;

@Controller
public class AcademyController {

    @Autowired
    private AcademyService service;

    @RequestMapping(value = "/enrollmentPage", method = RequestMethod.GET)
    public String showPage(@ModelAttribute("academyBean") Academy academyBean) {
                System.out.println("in controller");
                return "enrollmentPage";
    }
    @ModelAttribute("programList")
    public  Map<String, String> buildState(){

                Map<String, String> progMap = new HashMap<String, String>();
                progMap.put("ClassicalDance", "ClassicalDance");
                progMap.put("KarnaticVocals", "KarnaticVocals");
                progMap.put("WesternDance","WesternDance");
                progMap.put("Drawing", "Drawing");
                progMap.put("Instruments","Instruments");

                return progMap;

    }

    @RequestMapping(value = "/progEstimation", method = RequestMethod.POST)
    public String calculateProgramCost(@ModelAttribute("academyBean") @Valid Academy
academyBean, BindingResult result,
                            ModelMap model) {
                if (result.hasErrors()) {
```

```
                                System.out.println("Error : " + result.toString());
                                return "enrollmentPage";
                        }
                        double cost=service.calculateProgramCost(academyBean);
                        model.addAttribute("cost", cost);
                        return "estimationPage";
        }


}

```

FlawlessAcademy/src/main/java/com/example/demo/FlawlessAcademyApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan({"com.*"})
public class FlawlessAcademyApplication {

    public static void main(String[] args) {
                        SpringApplication.run(FlawlessAcademyApplication.class, args);
    }

}

```

FlawlessAcademy/src/main/java/com/model/Academy.java

```
package com.model;

import javax.validation.constraints.Min;
import javax.validation.constraints.Max;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class Academy {

    private String program;

    private double costPerSession;
    private int weeksPerMonth;
    @Min(value = 2, message = "Minimum 2 Sessions/Week")
    @Max(value = 5, message = "Maximum 5 Sessions/Week")
    private int sessionsPerWeek;

    public int getSessionsPerWeek() {
                        return sessionsPerWeek;
    }

    public void setSessionsPerWeek(int sessionsPerWeek) {
                        this.sessionsPerWeek = sessionsPerWeek;
    }

    public Academy()
    {

    }

```

```
    public int getWeeksPerMonth() {
            return weeksPerMonth;
    }
    public void setWeeksPerMonth(int weeksPerMonth) {
            this.weeksPerMonth = weeksPerMonth;
    }
    public double getCostPerSession() {
            return costPerSession;
    }
    public void setCostPerSession(double costPerSession) {
            this.costPerSession = costPerSession;
    }

    public String getProgram() {
            return program;
    }
    public void setProgram(String program) {
            this.program = program;
    }




}


```

FlawlessAcademy/src/main/java/com/service/AcademyService.java

```
package com.service;

import org.springframework.stereotype.Service;

import com.model.Academy;

@Service
public class AcademyService {

    public double calculateProgramCost (Academy academyBean) {

            double cost=0.0;
            academyBean.setWeeksPerMonth(4);
            System.out.println(academyBean.getSessionsPerWeek()+" "+academyBean.getProgram()+" "+academyBean.getWeeksPerMonth());
            if(academyBean.getProgram().equalsIgnoreCase("ClassicalDance") && academyBean.getSessionsPerWeek()>=1)
            {
                    academyBean.setCostPerSession(150.0);

    cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.getWeeksPerMonth();
            }
    else if(academyBean.getProgram().equalsIgnoreCase("KarnaticVocals") && academyBean.getSessionsPerWeek()>=1 )
    {
            academyBean.setCostPerSession(100.0);

    cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.getWeeksPerMonth();
    }
    else if(academyBean.getProgram().equalsIgnoreCase("WesternDance") && academyBean.getSessionsPerWeek()>=1)
    {
            academyBean.setCostPerSession(125.0);
```

```
28
        cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.getWeeksPerMonth();
29      }
30      if(academyBean.getProgram().equalsIgnoreCase("Drawing") && academyBean.getSessionsPerWeek()>=1)
31      {
32                      academyBean.setCostPerSession(130.0);
33
        cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.getWeeksPerMonth();
34      }
35  else if(academyBean.getProgram().equalsIgnoreCase("Instruments") && academyBean.getSessionsPerWeek()>=1)
36  {
37      academyBean.setCostPerSession(200.0);
38
        cost=academyBean.getCostPerSession()*academyBean.getSessionsPerWeek()*academyBean.getWeeksPerMonth();
39  }
40      return cost;
41
42      }
43
44  }
45
```

FlawlessAcademy/src/main/resources/application.properties

```
1  server.port=9088
2  spring.mvc.view.prefix = /WEB-INF/views/
3  spring.mvc.view.suffix = .jsp
4  spring.mvc.static-path-pattern=/resources/**
5
6
```

FlawlessAcademy/src/main/webapp/WEB-INF/views/enrollmentPage.jsp

```
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2     pageEncoding="ISO-8859-1" isELIgnored="false"%>
3  <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
4
5  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
6  <html>
7  <body style="background-color:lavender">
8  <h1><center> Flawless Academy</center></h1>
9  <form:form method="post" action="progEstimation" modelAttribute="academyBean">
10 <table style="margin: 0px auto; margin-left: auto; margin-right:auto">
11
12
13                              <tr>
14                                      <td>Select preferred program</td>
15                                      <td>
16                                       <form:select path="program" id="program">
17                                                      <form:options items="${programList}"/>
18                                              </form:select>
19
20                                      </td>
21                              </tr>
22
23                              <tr>
24                              <td>Number of sessions per week</td>
25                              <td><form:input path="sessionsPerWeek" id="sessionsPerWeek"/></td>
26                              <td><form:errors path="sessionsPerWeek" id="errormsg"/></td>
27                              <tr>
```

```
                                        <td><input type="submit" value="Program Cost" name="submit" />></td>

                            </tr>

                    </table>
</form:form>


</body>
</html>

```

FlawlessAcademy/src/main/webapp/WEB-INF/views/estimationPage.jsp

```
<%@page isELIgnored="false" %>
<html>
<body bgcolor="lavender">
<center><h2>Welcome to Flawless Academy. Your preferred program cost is Rs. ${cost}</h2></center>
</body>
</html>
```

Grade

Reviewed on Friday, 18 February 2022, 1:09 PM by Automatic grade

Grade 100 / 100

Assessment report

Assessment Completed Successfully

[-]**Grading and Feedback**

Feature Test - 35.00 / 35.00(Success)
* * check whether the controller annotation is present above the Controller - 8.00 / 8.00*
* * check whether the ModelAttribute annotation is present above the Method - 8.00 / 8.00*
* * check whether the Service is autowired inside the Controller - 7.00 / 7.00*
* * check whether Service annotation is present above the Service - 6.00 / 6.00*
* * check whether the ModelAttribute annotation has the appropriate value above the Method - 6.00 / 6.00*

Functional testing - 25.00 / 25.00(Success)
* * check whether the program cost for the ClassicalDance type is calculated properly - 5.00 / 5.00*
* * check whether the program cost for the KarnaticVocals type is calculated properly - 5.00 / 5.00*
* * check whether the program cost for the WesternDance type is calculated properly - 5.00 / 5.00*
* * check whether the program cost for the Drawing type is calculated properly - 5.00 / 5.00*
* * check whether the program cost for the Instruments is calculated properly - 5.00 / 5.00*

Spring Testing in the Controller - 25.00 / 25.00(Success)
* * check whether the request is mapped for enrollmentPage and redirected to enrollmentPage.jsp page - 5.00 / 5.00*
* * check whether the request is mapped for estimationPage and redirected to estimationPage.jsp page - 5.00 / 5.00*
* * check whether the model attribute is specified with the name above the buildState method that holds the Map containing the Preferred program - 5.00 / 5.00*
* * check whether the request is mapped and redirected to the estimationPage page for valid details - 5.00 / 5.00*
* * check whether the request is mapped and redirected to the same page for Invalid details - 5.00 / 5.00*

UI Testing - 15.00 / 15.00(Success)

** check whether the enrollmentPage is rendered for the program type ClassicalDance - 2.00 / 2.00*
** check whether the enrollmentPage is rendered for the program type KarnaticVocals - 2.00 / 2.00*
** check whether the enrollmentPage is rendered for the program type WesternDance - 2.00 / 2.00*
** check whether the enrollmentPage is rendered for the program type Drawing - 2.00 / 2.00*
** check whether the enrollmentPage is rendered for the program type Instruments - 2.00 / 2.00*
** check whether the enrollmentPage is rendered with proper constraints for sessionsPerWeek - 1.00 / 1.00*
** check whether UI from enrollmentPage to estimationPage page is redirected after calculating the program cost for ClassicalDance - 1.00 / 1.00*
** check whether UI from enrollmentPage to estimationPage page is redirected after calculating the program cost for KarnaticVocals - 1.00 / 1.00*
** check whether UI from enrollmentPage to estimationPage page is redirected after calculating the program cost for WesternDance - 1.00 / 1.00*
** check whether the sessionsPerWeek field validated properly - 0.50 / 0.50*
** check whether the sessionsPerWeek field in Enrollment Page is validated properly - 0.50 / 0.50*