**Important Instructions:**

* Please read the document thoroughly before you code.
* Import the given skeleton code into your Eclipse.
* Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.
* You can create any number of private methods inside the given class.
* You can test your code from main() method of the program

**Time: 2 hours**

**Assessment Coverage:**

* **Setter Injection**
* **Injecting Collections**
* **Reading Properties file**

ABC company is looking out solution providers to build an IT system which can help them to run their business online. To begin with, the company wants to automate few of its functionalities related to Stationery Bill management. Develop a software component that can do the following:

1. There can be one or more than one product item added to the cart. **Calculate the total order amount** from all the product items added to the cart, for the requested quantity.
2. Based on the Total Order Amount, the retail company opts to **offer a discount** based on certain business rules. Calculate the Discount amount, based on total order amount.
3. Check the **number of product items ordered** in bulk (the business rule will explain the bulk order based on the quantity requested in subsequent sections).

**Technical Requirements:**

You are required to develop an App which can provide the following service.

**API 1 (Requirement 1):**

Create class CartBO and write the below **public** methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ClassName** | **Method Name** | **Input Parameters** | **Output Parameters** | **Logic** |
| CartBO | countNoteBook | ---- | int | This method iterates through items in myCart and returns quantity of notebooks ordered |
| CartBO | countPen | ----- | int | This method iterates through items in myCart and returns quantity of pen ordered |
| CartBO | calculateBillAfterDiscount | double totalOrderBill | double | This method calculates final total bill if bill has discount applicable as per below business rule and returns the final bill amount |

**Business Rules:**

**If total bill is more than 500 and**

|  |  |  |
| --- | --- | --- |
| **Item Ordered** | **quantity** | **Business Condition** |
| Notebook | >= 3 | 10% discount on each notebook |
| Pen | >= 10 | 20% discount on each pen |

Create another class **CartService** and write the below **public** methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ClassName** | **Method Name** | **Input Parameters** | **Output Parameters** | **Logic** |
| CartService | calculateOrderTotalBill | --- | double | This method first validates cost per quantity by invoking validateCostperQuantity() method, if validation success then calculates total order bill by iterating through list of items and invokes calculateBillAfterDiscount on cartBo object and returns final bill amount . If validation of cost per quantity failed, throw exception with message “Cost per quantity must be positive non zero value” |
| CartService | validateCostPerQuantity | --- | boolean | This method validates cost per quantity , if it is zero or negative , throws user defined exception InvalidCostPerQuantityException with message “Cost per quantity must be positive non zero value” |

**Example**

Total Order Bill= sum of cost (itemCostPerQuantity \* quantity) from each order line item

Example: Each row below correspond to one line item.

|  |  |  |
| --- | --- | --- |
| **itemName** | **itemCostPerQuantity** | **Quantity** |
| Notebook | 100 | 4 |
| Pen | 50 | 10 |
| Pencil | 10 | 1 |

Total Order Bill= (100 \* 4) + (50\*10) +(10\*1)= 910.

|  |  |
| --- | --- |
| **Total Bill (bill)** | **Discount** |
| bill >500 and  No. of notebook>=3  No of Pen>=10 | 10% on each notebook  20% on each Pen |

Total Order Bill= 770

**API 2 (Requirement 2):** Create user defined exception class **InvalidCostPerQuantityException** that extends Exception and write the below **public constructor**:

|  |  |  |
| --- | --- | --- |
| **ClassName** | **Constructor** | **Logic** |
| InvalidCostperQuantityException | public InvalidCostperQuantityException(String message) | Passes the message to super class. |

**API 3 (Requirement 3):** Prepare spring.xml that contains the beans for Item, MyCart, CartService and CartBO. Remember there will be 3 or more Item beans which will be injected inside MyCart as a <list>

Note: Use below bean id/names to respective objects, for Item class , you can use any bean Id/name

|  |  |
| --- | --- |
| **bean id** | **Class to which the bean belongs** |
| myCart | com.cts.stationerybill.vo.MyCart |
| cartBo | com.cts.stationerybill.bo.CartBO |
| cartService | com.cts.stationerybill.service.CartService |

**Refer this sample input output.**

**SAMPLE INPUT1**

If below values are set for Item object

|  |  |  |
| --- | --- | --- |
| **itemName** | **itemCostPerQuantity** | **quantity** |
| Notebook | 100 | 4 |
| Pen | 50 | 10 |
| Pencil | 10 | 1 |

**SAMPLE OUTPUT1**

Your total bill is = 770.0

**SAMPLE INPUT2**

If below values are set for Item object

|  |  |  |
| --- | --- | --- |
| **itemName** | **itemCostPerQuantity** | **quantity** |
| Notebook | 100 | 4 |
| Pen | 0 | 10 |
| Pencil | 10 | 1 |

**SAMPLE OUTPUT2**

Cost per quantity must be positive non zero value

Flow : Main function is to generate the CartService Bean using the Configuration file spring.xml, using which the calculateOrderTotalBill() method should be invoked to find the Bill amount for the order.

Service method calculateOrderTotalBill() would check the costPerQuantity using validateCostPerQuantity() method and throws relevant custom exception:InvalidCostPerQuantityException.

Service method calls the CartBo business Object’s method calculateBillAfterDiscount() which takes total OrderBillAmount as input argument and finds the Total Order’s Bill Amount after applying the relevant discount.

All values for CostPerQuantity & quantity fields are set through property setter using application.properties file.

**General Design Constraints:**

* Ensure that all the Java Coding Standards are followed.
* Read quantity and cost per quantity from **application.properties** file into spring.xml.
* Don't change property names from properties file, values you can specify.

#### StationaryBill/pom.xml

1 <project xmlns="http://maven.apache.org/POM/4.0.0"

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

3 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

4 <modelVersion>4.0.0</modelVersion>

5

6 <groupId>com.cts.stationerybill</groupId>

7 <artifactId>RoughStationeryBill</artifactId>

8 <version>0.0.1-SNAPSHOT</version>

9 <packaging>jar</packaging>

10

11 <name>StationeryBill</name>

12 <url>http://maven.apache.org</url>

13

14 <properties>

15 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

16 <maven.compiler.source>1.8</maven.compiler.source>

17 <maven.compiler.target>1.8</maven.compiler.target>

18 <junit.jupiter.version>5.4.0</junit.jupiter.version>

19 </properties>

20

21 <dependencies>

22

23 <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

24 <dependency>

25 <groupId>org.springframework</groupId>

26 <artifactId>spring-context</artifactId>

27 <version>5.2.0.RELEASE</version>

28 </dependency>

29 <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine -->

30 <dependency>

31 <groupId>org.junit.jupiter</groupId>

32 <artifactId>junit-jupiter-engine</artifactId>

33 <version>${junit.jupiter.version}</version>

34 <scope>test</scope>

35 </dependency>

36 <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->

37 <dependency>

38 <groupId>org.junit.jupiter</groupId>

39 <artifactId>junit-jupiter-api</artifactId>

40 <version>${junit.jupiter.version}</version>

41 <scope>test</scope>

42 </dependency>

43

44 <!-- https://mvnrepository.com/artifact/org.junit.platform/junit-platform-launcher -->

45 <dependency>

46 <groupId>org.junit.platform</groupId>

47 <artifactId>junit-platform-launcher</artifactId>

48 <version>1.5.1</version>

49 <scope>test</scope>

50 </dependency>

51

52 <dependency>

53 <groupId>org.junit.platform</groupId>

54 <artifactId>junit-platform-commons</artifactId>

55 <version>1.5.1</version>

56 </dependency>

57

58 </dependencies>

59 <build>

60 <plugins>

61 <plugin>

62 <artifactId>maven-surefire-plugin</artifactId>

63 <version>2.22.1</version>

64 </plugin>

65 </plugins>

66 </build>

67 <reporting>

68 <plugins>

69 <plugin>

70 <groupId>org.apache.maven.plugins</groupId>

71 <artifactId>maven-surefire-report-plugin</artifactId>

72 <version>2.19.1</version>

73 </plugin>

74 </plugins>

75 </reporting>

76 </project>

77

#### StationaryBill/src/main/java/com/cts/stationarybill/bo/CartBO.java

1 *package* com.cts.stationarybill.bo;

2

3 *import* com.cts.stationarybill.vo.Item;

4 *import* com.cts.stationarybill.vo.MyCart;

5 *import* java.util.\*;

6

7 *public* *class* CartBO {

8 *private* MyCart cart;

9

10

11 *public* CartBO() {

12 // TODO Auto-generated constructor stub

13 }

14 *public* CartBO(MyCart cart) {

15 *super*();

16 *this*.cart = cart;

17 }

18

19 *public* MyCart getCart() {

20 *return* cart;

21 }

22 *public* *void* setCart(MyCart cart) {

23 *this*.cart = cart;

24 }

25

26

27

28 *public* *double* calculateBillAfterDiscount(*double* totalOrderBill) {

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

30 *double* nbprice=0.0;

31 *double* penprice=0.0;

32 *for*(Item i:items){

33 *if*(i.getItemName().equals("Notebook")){

34 nbprice=i.getItemCostPerQuantity();

35 }

36 *if*(i.getItemName().equals("Pen")){

37 penprice=i.getItemCostPerQuantity();

38 }

39 }

40 *int* notebooks=countNoteBook();

41 *int* pens=countPen();

42

43 *double* disc=0.0;

44 *if*(totalOrderBill>500){

45 *if*(notebooks>=3){

46 disc+=(notebooks\*0.1\*nbprice);

47 }

48 *if*(pens>=10){

49 disc+=(pens\*0.2\*penprice);

50 //System.out.println(disc);

51 }

52 }

53

54 //Code here..

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

56 }

57

58 *public* *int* countNoteBook() {

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

60

61 *int* count = 0;

62 *for*(Item i:items){

63 *if*(i.getItemName().equals("Notebook")){

64 count=i.getQuantity();

65 *break*;

66 }

67 }

68

69 //Code here..

70

71 *return* count; //TODO, change this value

72 }

73

74 *public* *int* countPen() {

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

76

77 *int* count = 0;

78 *for*(Item i:items){

79 *if*(i.getItemName().equals("Pen")){

80 count=i.getQuantity();

81 *break*;

82 }

83 }

84

85 //Code here..

86

87 *return* count; //TODO, change this value

88 }

89 }

90

#### StationaryBill/src/main/java/com/cts/stationarybill/exception/InvalidCostPerQuantityException.java

1 *package* com.cts.stationarybill.exception;

2

3 *public* *class* InvalidCostPerQuantityException *extends* Exception{

4

5 /\*\*

6 \*

7 \*/

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

9

10 *public* InvalidCostPerQuantityException(String message) {

11 *super*(message);

12 //code here..

13 }

14 }

15

#### StationaryBill/src/main/java/com/cts/stationarybill/main/Main.java

1 *package* com.cts.stationarybill.main;

2

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

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

5

6 *import* com.cts.stationarybill.exception.InvalidCostPerQuantityException;

7 *import* com.cts.stationarybill.service.CartService;

8 *import* com.cts.stationarybill.skeletonvalidator.SkeletonValidator;

9 *import* com.cts.stationarybill.vo.MyCart;

10 *import* com.cts.stationarybill.vo.Item;

11

12

13 *public* *class* Main

14 {

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

16 {

17 SkeletonValidator validator = *new* SkeletonValidator();

18

19 @SuppressWarnings("resource")

20 ApplicationContext ctx=*new* ClassPathXmlApplicationContext("spring.xml");

21

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

23 *try*{

24 *double* totalBill=service.calculateOrderTotalBill();

25 System.out.println("Your total bill is = "+totalBill);

26 }

27 *catch*(InvalidCostPerQuantityException e){

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

29 }

30 }

31 }

32

#### StationaryBill/src/main/java/com/cts/stationarybill/service/CartService.java

1 *package* com.cts.stationarybill.service;

2

3 *import* java.util.List;

4

5 *import* com.cts.stationarybill.bo.CartBO;

6 *import* com.cts.stationarybill.exception.InvalidCostPerQuantityException;

7 *import* com.cts.stationarybill.vo.Item;

8 *import* java.util.\*;

9

10 *public* *class* CartService {

11 *private* CartBO cartBo;

12 *public* CartService() {

13 // TODO Auto-generated constructor stub

14 }

15

16

17 *public* CartService(CartBO cartBo) {

18 *super*();

19 *this*.cartBo = cartBo;

20 }

21

22 *public* CartBO getCartBo() {

23 *return* cartBo;

24 }

25

26 *public* *void* setCartBo(CartBO cartBo) {

27 *this*.cartBo = cartBo;

28 }

29 /\*\*

30 \* Method to calculate total bill

31 \*

32 \* @param MyCart

33 \* @return totalOrderBill

34 \*/

35 *public* *double* calculateOrderTotalBill() throws InvalidCostPerQuantityException {

36 *double* totalOrderBill = 0;

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

38 *if*(validateCostPerQuantity()){

39 *for*(Item i:items){

40 totalOrderBill+=i.getItemCostPerQuantity()\*i.getQuantity();

41 }

42 totalOrderBill=cartBo.calculateBillAfterDiscount(totalOrderBill);

43 *return* totalOrderBill;

44 }*else*{

45 *throw* *new* InvalidCostPerQuantityException("Cost per quantity must be positive non zero value");

46 }

47

48 //Code here..

49 //TODO, change this value

50

51 }

52

53 /\*\*

54 \* Method to validate cost per quantity and throws exception if validation failes

55 \*

56 \* @exception InvalidCostPerQuantityException

57 \*/

58 *public* *boolean* validateCostPerQuantity() throws InvalidCostPerQuantityException{

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

60 *for*(Item i:items){

61 *if*(i.getItemCostPerQuantity()<1){

62 *throw* *new* InvalidCostPerQuantityException("Cost per quantity must be positive non zero value");

63 }

64

65 }

66 *return* *true*;

67 //Code here..

68

69 }

70 }

71

#### StationaryBill/src/main/java/com/cts/stationarybill/skeletonvalidator/SkeletonValidator.java

1 *package* com.cts.stationarybill.skeletonvalidator;

2

3 *import* java.lang.reflect.Method;

4 *import* java.util.logging.Level;

5 *import* java.util.logging.Logger;

6

7 /\*\*

8 \* @author t-aarti3

9 \* This class is used to verify if the Code Skeleton is intact and not

10 \* modified by participants thereby ensuring smooth auto evaluation

11 \* \*/

12

13 *public* *class* SkeletonValidator {

14 *public* SkeletonValidator() {

15 validateClassName("com.cts.stationarybill.service.CartService");

16 validateClassName("com.cts.stationarybill.vo.Item");

17 validateClassName("com.cts.stationarybill.vo.MyCart");

18 validateClassName("com.cts.stationarybill.bo.CartBO");

19

20 validateMethodSignature(

21 "calculateOrderTotalBill:double,validateCostperQuantity:boolean",

22 "com.cts.stationarybill.service.CartService");

23 validateMethodSignature(

24 "calculateBillAfterDiscount:double,countNoteBook:int,countPen:int",

25 "com.cts.stationarybill.bo.CartBO");

26

27 }

28 *private* *static* *final* Logger LOG = Logger.getLogger("SkeletonValidator");

29 *protected* *final* *boolean* validateClassName(String className) {

30

31 *boolean* iscorrect = *false*;

32 *try* {

33 Class.forName(className);

34 iscorrect = *true*;

35 LOG.info("Class Name " + className + " is correct");

36

37 } *catch* (ClassNotFoundException e) {

38 LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the correct package "

39 + "and class name as provided in the skeleton");

40

41 } *catch* (Exception e) {

42 LOG.log(Level.SEVERE,

43 "There is an error in validating the " + "Class Name. Please manually verify that the "

44 + "Class name is same as skeleton before uploading");

45 }

46 *return* iscorrect;

47 }

48

49 *protected* *final* *void* validateMethodSignature(String methodWithExcptn, String className) {

50 Class cls = *null*;

51 *try* {

52

53 String[] actualmethods = methodWithExcptn.split(",");

54 *boolean* errorFlag = *false*;

55 String[] methodSignature;

56 String methodName = *null*;

57 String returnType = *null*;

58

59 *for* (String singleMethod : actualmethods) {

60 *boolean* foundMethod = *false*;

61 methodSignature = singleMethod.split(":");

62

63 methodName = methodSignature[0];

64 returnType = methodSignature[1];

65 cls = Class.forName(className);

66 Method[] methods = cls.getMethods();

67 *for* (Method findMethod : methods) {

68 *if* (methodName.equals(findMethod.getName())) {

69 foundMethod = *true*;

70 *if* (!(findMethod.getReturnType().getName().equals(returnType))) {

71 errorFlag = *true*;

72 LOG.log(Level.SEVERE, " You have changed the " + "return type in '" + methodName

73 + "' method. Please stick to the " + "skeleton provided");

74

75 } *else* {

76 LOG.info("Method signature of " + methodName + " is valid");

77 }

78

79 }

80 }

81 *if* (!foundMethod) {

82 errorFlag = *true*;

83 LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName

84 + ". Do not change the " + "given public method name. " + "Verify it with the skeleton");

85 }

86

87 }

88 *if* (!errorFlag) {

89 LOG.info("Method signature is valid");

90 }

91

92 } *catch* (Exception e) {

93 LOG.log(Level.SEVERE,

94 " There is an error in validating the " + "method structure. Please manually verify that the "

95 + "Method signature is same as the skeleton before uploading");

96 }

97 }

98

99 }

100

#### StationaryBill/src/main/java/com/cts/stationarybill/vo/Item.java

1 *package* com.cts.stationarybill.vo;

2

3 *public* *class* Item {

4 // member variables

5 *private* String itemName;

6 *private* *double* itemCostPerQuantity;

7 *private* *int* quantity;

8 *public* Item() {

9 // TODO Auto-generated constructor stub

10 }

11 // Parameterized Constructor

12 *public* Item(String itemName, *double* itemCostPerQuantity, *int* quantity) {

13 *super*();

14 *this*.itemName = itemName;

15 *this*.itemCostPerQuantity = itemCostPerQuantity;

16 *this*.quantity = quantity;

17 }

18

19 // getter and setter methods

20

21 /\*\*

22 \* @return the itemName

23 \*/

24 *public* String getItemName() {

25 *return* itemName;

26 }

27

28 /\*\*

29 \* @param itemName

30 \* the itemName to set

31 \*/

32 *public* *void* setItemName(String itemName) {

33 *this*.itemName = itemName;

34 }

35

36 /\*\*

37 \* @return the itemCostPerQuantity

38 \*/

39 *public* *double* getItemCostPerQuantity() {

40 *return* itemCostPerQuantity;

41 }

42

43 /\*\*

44 \* @param itemCostPerQuantity

45 \* the itemCostPerQuantity to set

46 \*/

47 *public* *void* setItemCostPerQuantity(*double* itemCostPerQuantity) {

48 *this*.itemCostPerQuantity = itemCostPerQuantity;

49 }

50

51 /\*\*

52 \* @return the quantity

53 \*/

54 *public* *int* getQuantity() {

55 *return* quantity;

56 }

57

58 /\*\*

59 \* @param quantity

60 \* the quantity to set

61 \*/

62 *public* *void* setQuantity(*int* quantity) {

63 *this*.quantity = quantity;

64 }

65 @Override

66 *public* String toString() {

67 *return* "Item [itemName=" + itemName + ", itemCostPerQuantity=" + itemCostPerQuantity + ", quantity="

68 + quantity + "]";

69 }

70

71

72 }

73

#### StationaryBill/src/main/java/com/cts/stationarybill/vo/MyCart.java

1 *package* com.cts.stationarybill.vo;

2

3 *import* java.util.List;

4

5 *public* *class* MyCart {

6 *private* List<Item> items;

7 *public* MyCart() {

8 // TODO Auto-generated constructor stub

9 }

10 *public* MyCart(List<Item> items) {

11 *super*();

12 *this*.items = items;

13 }

14 *public* List<Item> getItems() {

15 *return* items;

16 }

17 *public* *void* setItems(List<Item> items) {

18 *this*.items = items;

19 }

20

21 }

22

#### StationaryBill/src/main/resources/application.properties

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

2

3 notebook.CostPerQuantity=100

4 pen.CostPerQuantity=50

5 pencil.CostPerQuantity=10

6 notebook.quantity=7

7 pen.quantity=10

8 pencil.quantity=1

#### StationaryBill/src/main/resources/spring.xml

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

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

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

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

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

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

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

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

9

10

11

12 <context:property-placeholder

13 location="classpath:application.properties" />

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

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

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

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

18 </bean>

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

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

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

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

23 </bean>

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

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

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

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

28 </bean>

29 <bean id="myCart" class="com.cts.stationarybill.vo.MyCart">

30 <property name="items">

31 <list>

32 <ref bean="notebook"/>

33 <ref bean="pen"/>

34 <ref bean="pencil"/>

35 </list>

36 </property>

37 </bean>

38 <bean id="cartBo" class="com.cts.stationarybill.bo.CartBO">

39 <constructor-arg ref="myCart"/>

40 </bean>

41 <bean id="cartService" class="com.cts.stationarybill.service.CartService">

42 <property name="cartBo" ref="cartBo"/>

43 </bean>

44

45

46 </beans>

47

## Grade

Reviewed on Wednesday, 28 April 2021, 9:47 PM by Automatic grade  
**Grade** 95 / 100  
**Assessment report**  
[[+]](javascript:void(0);)**SOURCE CODE ANALYZER REPORT**  
[[+]](javascript:void(0);)**Grading and Feedback**