**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**
* **Java 8 Date API, Stream API**

A college Library is looking out solution providers to build an IT system which can help them to run their activities online. The college wants to automate few of its functionalities related to Book Library Management. Develop a software component that can do the following:

1. A staff of library in the college can issue a book or collect book returned by student. The student can avail a book from library by providing the student id which should be validated using ValidateStudentId method
2. Once the Book name is provided ,it should be validated using ValidateBookName .
3. The program should issue book by checking number of copies available of the given book and return on the returnDate of the book and also update no. of copies.
4. The program should update/increase Number of copies available of the book on returning the book.
5. It should also Calculate Fine if the actual return date exceeds the expected return date. The software assumes **fine charge as 50/- per day delayed**

**Skeleton File for Development:**

Import the below attached skeleton code into your eclipse project and implement the required functionalities

****

**Technical Requirements:**

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

**API 1 (Requirement 1):**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ClassName** | **Method Name** | **Input Parameters** | **Output Parameters** | **Logic** |
| LibraryBO | validateStudentId | int | boolean | This method iterates through student ids in stDetails ,member of StudentInfo and throws user defined InvalidStudentIdException if student id is invalid else returns true. |
| LibraryBO | validateBook | String | boolean | This method iterates through book names in bookDetails ,member of BookInfo and throws user defined InvalidBookNameException if bookname is invalid else returns true. |
| LibraryBO | checkNoOfCopies | String | int | This method checks no of copies using the book name and returns the current value of no. Of copies. It should also check If no. of copies >0 then updates no of copies left after issuing the book . |
| LibraryBO | updateNoOfCopies | String | Int | This method updates/increases no. of copies available of the given book name and returns the same. |

**Business Rules:**

**Return date is 7 days after issued Date**

**Fine 50/- per day delayed**

Create another class LibraryService and write the below public methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ClassName** | **Method Name** | **Input Parameters** | **Output Parameters** | **Logic** |
| LibraryService | validateStudentId | Int | boolean | This method first validates Student id . It returns true if the student id is valid else raises the exception raised by validateStudentId() of LibraryBo |
| LibraryService | validateBookName | String | boolean | The method validates the book name .It returns true if the book name is valid else raises the exception raised by validateBookName() of LibraryBo . |
| LibraryService | issueBook | Int, String | String | It checks no. of copies of the book available and returns the return date of the book issued with appropriate message considering issue date as current date.  If current value of no. of copies is 0,it should return another appropriate message. |
| LibraryService | returnBook | Int,  String,  LocalDate, LocalDate | String | This method is called when the student returns the book. It accepts student id, book name , issue date and return Date.It invokes updateNoOfCopies method of business object class to update/increase no of copies available and then Invokes calculateFine method to calculate the fine amount if the actual return date is greater than expected return date .It then returns the fine amount with Student Id ,Book Name ,Issue Date and Return date in dd/MM/yyyy format. |
| **LibraryService** | calculateFine | LocalDate,  LocalDate | int | This method calculates fine using the issueDate and returnDate and returns the same. |

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

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

Create another user defined exception class **InvalidBookNameException** that extends Exception and write the below **public constructor**:

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

**API 3 (Requirement 3):** Prepare beans.xml that contains the beans for StudentInfo, BookInfo ,LibraryBo LibraryService. StudentInfo contains a Map<Integer,String> called stDetails of student id and phone that should be initialized from beans.xml. Similarly BookInfo also contains a Map<String,Integer> called bkDetails of book name and no. of copies that should be initialized in bean.xml .

**Note: Use below bean id/names to respective objects**

|  |  |
| --- | --- |
| **bean id** | **Class to which the bean belongs** |
| stinfo | com.cts.library.model.StudentInfo |
| bkinfo | com.cts.library.model.BookInfo |
| libBo | com.cts.library.bo.LibraryBo |
| libService | com.cts.library.service.LibraryService |

**Refer this sample input output.**

**Example 1:**

**Business Logic :** i) Consider the Issue Date ie current date is 02/11/2020

ii) Return Date=Issue Date+7

**SAMPLE INPUT 1**

Choose the option:- 1 . Issue Book 2. Return Book

1

Enter Student id

1111

Enter Book Name

Spring MVC

Sorry!!!The Book is not available

Would you like to continue?yes/no

yes

Choose the option:- 1 . Issue Book 2. Return Book

2

Enter Student id

1111

Enter Book Name

Spring MVC

Enter Issue Date(yyyy-MM-dd)

2021-03-12

Enter Return Date(yyyy-MM-dd)

2021-03-21

Student Id: 1111

Book Name :Spring MVC

IssueDate :12/03/2021

Return Date :21/03/2021

Fine : 100Rs.

Would you like to continue?yes/no

yes

Choose the option:- 1 . Issue Book 2. Return Book

1

Enter Student id

1111

Enter Book Name

Spring MVC

**SAMPLE OUTPUT 1**

The book Issued successfully.Return date is 31/03/2021

Would you like to continue?yes/no

**SAMPLE INPUT 2**

Choose the option:- 1 . Issue Book 2. Return Book

1

Enter Student id

1101

**SAMPLE OUTPUT 2**

The given Student id does not exist!!!

**SAMPLE INPUT 3**

Choose the option:- 1 . Issue Book 2. Return Book

1

Enter Student id

1111

Enter Book Name

Let Us C

**SAMPLE OUTPUT 3**

The given Book name does not exist!!!

**SAMPLE INPUT 4**

Choose the option:- 1 . Issue Book 2. Return Book

1

Enter Student id

1111

Enter Book Name

Spring MVC

**SAMPLE OUTPUT 4**

Sorry!!!The Book is not available

**Business Logic**:- Total Fine =(ReturnDate-(IssueDate+7))\*50

**SAMPLE INPUT 5**

Choose the option:- 1 . Issue Book 2. Return Book

2

Enter Student id

1111

Enter Book Name

Pro Spring

Enter Issue Date(yyyy-MM-dd)

2000-11-02

Enter Return Date(yyyy-MM-dd)

2000-11-12

**SAMPLE OUTPUT 5**

Student Id: 1111

Book Name :Pro Spring

IssueDate :02/11/2000

Return Date :12/11/2000

Fine : 150Rs.

**SAMPLE INPUT 6**

Choose the option:- 1 . Issue Book 2. Return Book

2

Enter Student id

1111

Enter Book Name

Pro Spring

Enter Issue Date(yyyy-MM-dd)

2000-11-02

Enter Return Date(yyyy-MM-dd)

2000-11-08

**SAMPLE OUTPUT 6**

Student Id: 1111

Book Name :Pro Spring

IssueDate :02/11/2000

Return Date :08/11/2000

Fine : 0Rs.

**SAMPLE INPUT 7**

Choose the option:- 1 . Issue Book 2. Return Book

2

Enter Student id

1011

**SAMPLE OUTPUT 7**

The given Student id does not exist!!!

**SAMPLE INPUT 8**

Choose the option:- 1 . Issue Book 2. Return Book

2

Enter Student id

1111

Enter Book Name

The Complete Reference

**SAMPLE OUTPUT 8**

The given Book name does not exist!!!

**SAMPLE INPUT 9**

Choose the option:- 1 . Issue Book 2. Return Book

3

**SAMPLE OUTPUT 9**

Invalid Choice

**Flow :**

Main function is to generate the LibraryService Bean using the Configuration file beans.xml. It provides two options to the user namely **1.Issue Book 2.Return Book**. If the user chooses **option 1** then It should accept Student id as input and call service method validateStudentId() to validate student id. If the student id is valid then it accepts Book name and calls a service method validateBookName() to validate Book name. If the book name is valid it invokes service method issueBook() which checks the availability of the book (by checking no. of copies), updates no.of copies and returns the return date with appropriate message. If the book is not available then also it returns a suitable message.

If the user chooses **option 2,** then It should accept Student id as input and call service method validateStudentId() to validate student id. If the student id is valid then it accepts Book name and calls a service method validateBookName() to validate Book name. If the book name is valid it accepts remaining details and invokes service method called returnBook().

If the user provides **any option other than 1 or 2**, the application should give some suitable message.

**General Design Constraints:**

* Ensure that all the Java Coding Standards are followed.

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

#### LibraryApp/pom.xml

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

2 <modelVersion>4.0.0</modelVersion>

3 <groupId>com.cts</groupId>

4 <artifactId>LibraryApp\_Skeleton</artifactId>

5 <version>0.0.1-SNAPSHOT</version>

6 <description>Library App Skeleton</description>

7 <dependencies>

8

9 <dependency>

10 <groupId>org.springframework</groupId>

11 <artifactId>spring-context</artifactId>

12 <version>5.2.3.RELEASE</version>

13 </dependency>

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

15 <dependency>

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

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

18 <version>5.5.2</version>

19 <scope>test</scope>

20 </dependency>

21 </dependencies>

22 </project>

#### LibraryApp/src/main/java/com/cts/library/bo/LibraryBo.java

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

2

3 *import* com.cts.library.exception.InvalidBookNameException;

4 *import* com.cts.library.exception.InvalidStudentIdException;

5 *import* com.cts.library.model.BookInfo;

6 *import* com.cts.library.model.StudentInfo;

7

8 *public* *class* LibraryBo {

9 *private* StudentInfo stInfo;

10 *private* BookInfo bkInfo;

11

12 *public* StudentInfo getStInfo() {

13 *return* stInfo;

14 }

15

16 *public* *void* setStInfo(StudentInfo stInfo) {

17 *this*.stInfo = stInfo;

18 }

19

20 *public* BookInfo getBkInfo() {

21 *return* bkInfo;

22 }

23

24 *public* *void* setBkInfo(BookInfo bkInfo) {

25 *this*.bkInfo = bkInfo;

26 }

27 /\*\*

28 \* Method to validate student id and throws exception if validation fails

29 \*

30 \* @exception InvalidStudentIdException

31 \*/

32

33 *public* *void* validateStudentId(*int* stid) throws InvalidStudentIdException

34 {//Insert code here..

35

36 }

37

38 /\*\*

39 \* Method to validate book name and throws exception if validation fails

40 \*

41 \* @exception InvalidBookNameException

42 \*/

43 *public* *void* validateBookName(String bookname) throws InvalidBookNameException

44 {//Insert code here..

45

46 }

47

48 /\*\*

49 \* Method checks no. Of copies of the given Bookname and return updated noOfCopies

50 \*/

51

52 *public* *int* checkNoOfCopies(String bookname)

53 {//Insert code here..

54

55 *return* 0;

56

57 }

58 *public* *int* updateNoOfCopies(String bookname) {

59 //Insert code here...

60 *return* -1;

61

62 }

63

64 }

65

#### LibraryApp/src/main/java/com/cts/library/exception/InvalidBookNameException.java

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

2

3 *public* *class* InvalidBookNameException *extends* Exception {

4 *public* InvalidBookNameException(String msg) {

5 //Insert code here..

6 *super*(msg);

7 }

8 }

9

#### LibraryApp/src/main/java/com/cts/library/exception/InvalidStudentIdException.java

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

2

3 *public* *class* InvalidStudentIdException *extends* Exception{

4

5

6 *public* InvalidStudentIdException(String msg)

7 {

8 //Insert code here..

9 *super*(msg);

10

11 }

12 }

#### LibraryApp/src/main/java/com/cts/library/main/LibraryMgmtApplication.java

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

2

3 *import* java.util.Scanner;

4

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

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

7

8 *import* com.cts.library.service.LibraryService;

9 *import* com.cts.library.skeletonvalidator.SkeletonValidator;

10

11 *public* *class* LibraryMgmtApplication {

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

13 String choice = "yes";

14 SkeletonValidator validator = *new* SkeletonValidator();

15 ApplicationContext ctx = *new* ClassPathXmlApplicationContext("beans.xml");

16 // Insert code here..

17 *do* {

18 System.out.println("Choose the option:- 1 . Issue Book 2. Return Book");

19 Scanner sc = *new* Scanner(System.in);

20 *int* ch = sc.nextInt();

21 *if* (ch == 1) {

22 // Insert code here...

23

24 } *else* *if* (ch == 2) {

25 // Insert code here...

26 } *else* {

27 System.out.println("Invalid Choice");

28 }

29 System.out.println("Would you like to continue?yes/no");

30 choice = sc.next();

31

32 } *while* (choice.equalsIgnoreCase("yes"));

33 }

34 }

35

#### LibraryApp/src/main/java/com/cts/library/model/BookInfo.java

1 *package* com.cts.library.model;

2

3 *import* java.util.Map;

4

5 *public* *class* BookInfo {

6 Map<String,Integer> bookDetails;

7

8 *public* Map<String, Integer> getBookDetails() {

9 *return* bookDetails;

10 }

11

12 *public* *void* setBookDetails(Map<String, Integer> bookDetails) {

13 *this*.bookDetails = bookDetails;

14 }

15 }

16

#### LibraryApp/src/main/java/com/cts/library/model/StudentInfo.java

1 *package* com.cts.library.model;

2

3 *import* java.util.Map;

4

5 *public* *class* StudentInfo {

6 Map<Integer,String> stDetails;

7

8 *public* Map<Integer, String> getStDetails() {

9 *return* stDetails;

10 }

11

12 *public* *void* setStDetails(Map<Integer, String> stDetails) {

13 *this*.stDetails = stDetails;

14 }

15 }

16

#### LibraryApp/src/main/java/com/cts/library/service/LibraryService.java

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

2

3 *import* java.time.LocalDate;

4

5 *import* com.cts.library.bo.LibraryBo;

6 *import* com.cts.library.exception.InvalidBookNameException;

7 *import* com.cts.library.exception.InvalidStudentIdException;

8

9 *public* *class* LibraryService {

10 *private* LibraryBo libBo;

11

12 *public* LibraryBo getLibBo() {

13 *return* libBo;

14 }

15

16 *public* *void* setLibBo(LibraryBo libBo) {

17 *this*.libBo = libBo;

18 }

19

20 /\*\*

21 \* Method to call validateStudentId and handles exception and returns false if

22 \* validation fails

23 \*

24 \* @exception InvalidStudentIdException

25 \*/

26

27 *public* *boolean* validateStudentId(*int* stid) throws InvalidStudentIdException

28 {// Insert code here..

29 *return* *false*;

30 }

31

32 /\*\*

33 \* Method to call validateBookName and handles exception and returns false if

34 \* validation fails

35 \*

36 \* @exception InvalidBookNameException

37 \*/

38

39 *public* *boolean* validateBookName(String bookname) throws InvalidBookNameException

40 {

41 // Insert code here..

42 *return* *false*;

43 }

44

45 /\*\*

46 \* Method to check NoOfCopies and return appropriate message if the book is

47 \* available

48 \*

49 \*/

50 *public* String issueBook(*int* stid, String bookname) {// Insert code here;

51 *return* *null*;

52 }

53

54 /\*\*

55 \* Method to calculate fine on the basis of issue date and return date and

56 \* return appropriate message

57 \*

58 \*/

59

60 *public* String returnBook(*int* stid, String bookname, LocalDate issueDate, LocalDate returnDate) {

61 // Insert code here;

62

63 *return* *null*;

64

65 }

66

67 /\*\*

68 \* Method to calculate fine on the basis of issue date and return date and

69 \* return the fine

70 \*/

71

72 *public* *int* calculateFine(LocalDate issueDate, LocalDate returnDate) {

73 // Insert code here;

74

75 *return* 0;

76 }

77 }

78

#### LibraryApp/src/main/java/com/cts/library/skeletonvalidator/SkeletonValidator.java

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

2

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

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

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

6

7 *public* *class* SkeletonValidator {

8 *public* SkeletonValidator() {

9 validateClassName("com.cts.library.service.LibraryService");

10 validateClassName("com.cts.library.bo.LibraryBo");

11 validateClassName("com.cts.library.model.BookInfo");

12 validateClassName("com.cts.library.model.StudentInfo");

13

14

15

16 validateMethodSignature(

17 "validateStudentId:boolean,validateBookName:boolean,issueBook:java.lang.String,"

18 + "returnBook:java.lang.String,calculateFine:int",

19 "com.cts.library.service.LibraryService");

20 validateMethodSignature(

21 "validateStudentId:void,validateBookName:void,checkNoOfCopies:int,updateNoOfCopies:int",

22 "com.cts.library.bo.LibraryBo");

23

24 }

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

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

27

28 *boolean* iscorrect = *false*;

29 *try* {

30 Class.forName(className);

31 iscorrect = *true*;

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

33

34 } *catch* (ClassNotFoundException e) {

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

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

37

38 } *catch* (Exception e) {

39 LOG.log(Level.SEVERE,

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

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

42 }

43 *return* iscorrect;

44 }

45

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

47 Class cls = *null*;

48 *try* {

49

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

51 *boolean* errorFlag = *false*;

52 String[] methodSignature;

53 String methodName = *null*;

54 String returnType = *null*;

55

56 *for* (String singleMethod : actualmethods) {

57 *boolean* foundMethod = *false*;

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

59

60 methodName = methodSignature[0];

61 returnType = methodSignature[1];

62 cls = Class.forName(className);

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

64 *for* (Method findMethod : methods) {

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

66 foundMethod = *true*;

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

68 errorFlag = *true*;

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

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

71

72 } *else* {

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

74 }

75

76 }

77 }

78 *if* (!foundMethod) {

79 errorFlag = *true*;

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

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

82 }

83

84 }

85 *if* (!errorFlag) {

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

87 }

88

89 } *catch* (Exception e) {

90 LOG.log(Level.SEVERE,

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

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

93 }

94 }

95

96 }

97

#### LibraryApp/src/main/resources/beans.xml

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

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

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

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

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

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

7 http://www.springframework.org/schema/aop/spring-aop-3.2.xsd

8 http://www.springframework.org/schema/beans

9 http://www.springframework.org/schema/beans/spring-beans-3.2.xsd

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

11 http://www.springframework.org/schema/context/spring-context-3.2.xsd

12

13 http://www.springframework.org/schema/tx

14 http://www.springframework.org/schema/tx/spring-tx-3.2.xsd">

15

16

17 <!-- Insert code here -->

18 <bean class="com.cts.library.model.StudentInfo" id="stinfo">

19

20 </bean>

21 <bean class="com.cts.library.model.BookInfo" id="bkinfo">

22

23 </bean>

24 <bean class="com.cts.library.bo.LibraryBo" id="libBo">

25

26 </bean>

27 <bean class="com.cts.library.service.LibraryService" id="libService">

28

29 </bean>

30 </beans>

## Grade

Reviewed on Wednesday, 12 May 2021, 6:24 PM by Automatic grade  
**Grade** 15 / 100  
**Assessment report**  
Fail 1 --testValidateBookName1::Validating Book Name failed!!!  
Fail 2 --testValidateBookName2::Expected com.cts.library.exception.InvalidBookNameException to be thrown, but nothing was thrown.  
Fail 3 --testValidateBookNameBo::Expected com.cts.library.exception.InvalidBookNameException to be thrown, but nothing was thrown.  
Fail 4 --testValidateStudentId1::Validating Student Id failed!!!  
Fail 5 --testValidateStudentId2::Expected com.cts.library.exception.InvalidStudentIdException to be thrown, but nothing was thrown.  
Fail 6 --testCheckNoOfCopies::Check Number Of Copies in Business Object class failed!!!  
Fail 7 --testReturnBook::Return Book logic of service class failed!!!  
Fail 8 --testValidateStudentIdBo::Expected com.cts.library.exception.InvalidStudentIdException to be thrown, but nothing was thrown.  
Fail 9 --testIssueBook::Issue Book in service class failed!!!  
Fail 10 --testCalculateFine::Calculate fine in service class failed!!!  
Fail 11 --testUpdateNoOfCopies::Update Number Of Copies in Business Object class failed!!!  
[[+]](javascript:void(0);)**SOURCE CODE ANALYZER REPORT**  
[[+]](javascript:void(0);)**Grading and Feedback**

**Result Description**

**Fail 1 --testValidateBookName1::Validating Book Name failed!!!**

**Fail 2 --testValidateBookName2::Expected com.cts.library.exception.InvalidBookNameException to be thrown, but nothing was thrown.**

**Fail 3 --testValidateBookNameBo::Expected com.cts.library.exception.InvalidBookNameException to be thrown, but nothing was thrown.**

**Fail 4 --testValidateStudentId1::Validating Student Id failed!!!**

**Fail 5 --testValidateStudentId2::Expected com.cts.library.exception.InvalidStudentIdException to be thrown, but nothing was thrown.**

**Fail 6 --testCheckNoOfCopies::Check Number Of Copies in Business Object class failed!!!**

**Fail 7 --testReturnBook::Return Book logic of service class failed!!!**

**Fail 8 --testValidateStudentIdBo::Expected com.cts.library.exception.InvalidStudentIdException to be thrown, but nothing was thrown.**

**Fail 9 --testIssueBook::Issue Book in service class failed!!!**

**Fail 10 --testCalculateFine::Calculate fine in service class failed!!!**

**Fail 11 --testUpdateNoOfCopies::Update Number Of Copies in Business Object class failed!!!**

**SOURCE CODE ANALYZER REPORT**

**Error Msg: Avoid unused imports such as 'com.cts.library.service.LibraryService'**

**Variable Name:**

**Class Name: LibraryMgmtApplication**

**Error Msg: Avoid unused local variables such as 'validator'.**

**Variable Name: validator**

**Class Name: LibraryMgmtApplication**

**Error Msg: Avoid unused local variables such as 'ctx'.**

**Variable Name: ctx**

**Class Name: LibraryMgmtApplication**

**Grading and Feedback**

**FunctionalTest - 10.0 / 90.0(Failed)**

**\* Test case to check bean creation inside spring configuration file - 10.0 / 10.0**

**\* Test case to check student Id is validated - 0.0 / 5.0**

**\* Test case to check custom exception called InvalidStudentIdException is raised if student Id is not valid - 0.0 / 5.0**

**\* Test case to check Book Name is validated - 0.0 / 5.0**

**\* Test case to check custom exception called InvalidBookNameException is raised if book name is not valid - 0.0 / 5.0**

**\* Test case to check ValidateStudentId of BO class throws InvalidStudentIdException if student id is not valid - 0.0 / 10.0**

**\* Test case to check ValidateBookName of BO class throws InvalidBookNameException if book name is not valid - 0.0 / 10.0**

**\* Test case to issue the book depending on No of copies available and return the Return Date - 0.0 / 10.0**

**\* Test case to check that the data specifically issue Date and return Date is displayed in dd/MM/yyyy format - 0.0 / 5.0**

**\* Test case to check the fine calculation - 0.0 / 10.0**

**\* Test case to check that number of copies of the book is updated properly while issuing the book - 0.0 / 5.0**

**Comments and best practices/standards - 5.0 / 10.0(Partial)**