**Capabilities covered:**

* Analyze complex business scenarios and create a data model
* Use MySQL to build SQL engine for an application
* Use Java to implement business layer of your application
* Use Eclipse IDE for programming Java application
* Test, and Debug your application to make application ready for deployment
* Follow coding standard and maintain high quality by analyzing various reports generated through code quality tools
* Automate the build using Maven

**Problem Statement: Book Store**

XYZ book store wants to automate their book sales process. The customer can view all the books available in the shop and can buy a book. **Read the complete document before coding.**

**Assumption:** A customer can purchase only one book per time.

Design and implement console based application program to automate the process of purchasing books. **The main menu contains following options**

1. **Display book details**
2. **Purchase a book**
3. **Exit**

Above menu must be redisplayed after each operation is completed, except for the **‘Exit’** option. The details for each option are specified in the rest of the document.

Your application program should follow the control flow as, start from class **BookStoreSystemClient -> BookStoreSystemManager->DAO class -> Database**

**Instructions:**

1. BookStoreSystemClient class contains main method, which would interact with user for receiving input and displaying the output
2. BookStoreSystemManager class should contain all business logic and would interact with DAO class for any data from database
3. DAO class must implement an interface and contain all DB transaction logic
4. **Upload your table script with inserted values along with project solution.**
5. **Exception handling:**
   * All user defined exceptions must be created under com.mindtree.exceptions package
   * All exceptions must be handled in Manager Class. DAO class should not handle any exception; it must throw it to Manager Class.
6. **Loose coupling:**
   * Create a separate Dao interfaces and implementation classes. Database interaction code should be only in DAO classes.
   * Dao classes should not have UI code.
7. **Write Junit test cases for ‘BookStoreSystemManager’ class**

* Create a class called ‘ManagerTest’ in com.mindtree.tester package. In the ManagerTest class, create the test cases using the steps given below:
* Create test case methods with exactly the same names given in the table below with appropriate annotations.
* The test case method naming convention is as follows:
  + The test case testing valid data is **testFunctionalityNameValidData**
  + The test case testing a particular invalid attribute is **testFunctionalityNameInvalidAttributeName or testFunctionalityNameNoRecords**
* In the test case methods, do not use try - catch blocks to handle the Exception, use throws Exception to handle the same.
* For a test case testing valid data, assertion has to be done. Just invoking the Manager Class method is not considered proper. (Any of assertTrue / assertFalse / assertEquals /assertNotSame can be used with proper logic.)
* For a test case testing invalid data, assertion need not be done. The specific user-defined exception that is expected for the test case needs to be mentioned with the annotation.

1. **Database design:**

Create database **“BookStore\_DB**” and create the following two tables used for the application: **“Book” and “Purchase”.** Listed below is the table design with constraints.

Table Name: **Book**

|  |  |  |
| --- | --- | --- |
| **Column name** | **Data type** | **Constraint** |
| Book\_id | int | Primary key |
| Book\_name | varchar(60) | Not null |
| Author\_name | varchar(50) | Not null |
| Publisher | varchar(50) | Not null |
| Category | varchar(20) | Not null |
| price | int | Not null |

Below is the content of table **‘Book’,** which must be manually inserted into table**:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Book\_Id** | **Book\_name** | **Author\_name** | **Publisher** | **Category** | **Price** |
| 1001 | Java: The Complete Reference | Herbert Schildt | Tata | Technology | 615 |
| 1002 | Matlab Programming | Singh Y. Kirani | Subhash | Technology | 232 |
| 1003 | Half Girlfriend | Chetan Bhagat | Tata | General | 95 |
| 1004 | Forge your Future | A.P.J. Abdul Kalam | Swapna | General | 200 |
| 1005 | The Fault in our Stars | John Green | Pai | Kids | 159 |
| 1006 | My Little Book | Randall Munroe | Pai | Kids | 420 |

Table Name: **Purchase**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Data type** | **constraint** | **Description** |
| Purchase\_no | Int | Primary key | Auto increment |
| Book\_id | Int | Foreign key | References Book\_id column of Book |
| Customer\_name | varchar(20) | Not null |  |
| Customer\_mobileno | varchar(20) | Not null |  |
| Purchase\_date | Date | Not null |  |
| Amount | int | Not null |  |

The values for **‘Purchase’** table must be inserted through your application.

1. **Entity classes:**

Create following entity classes under package com.mindtree.entity

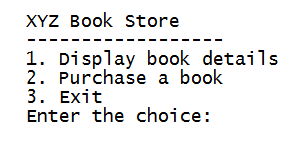
1. **Book**

****

1. **Purchase**

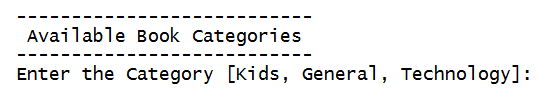
****

**BookStoreSystemClient** class displays the Menu as below:

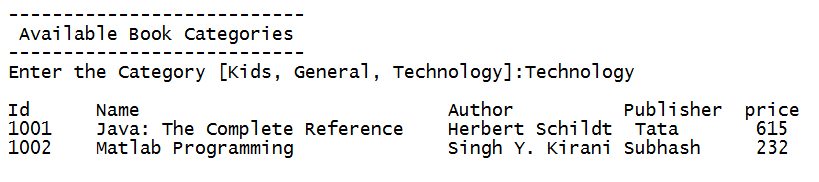
****

1. **Display book details**

* When user selects option 1, it must display the available categories and prompt the user to enter value for category
* Sample output shown below.



Write code to display the output in below format **(sorted by book name, use collection framework)** by using the returned value if selected category of books available in database otherwise display an appropriate exception/error message.



The **BookStoreSystemManager class** code should check for the business rule given below and throw appropriate user defined exception, when exception thrown corresponding message should be displayed to user, which is as given in the table:

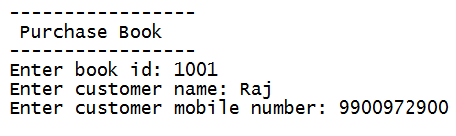
|  |  |  |  |
| --- | --- | --- | --- |
| **Rule No** | **Business constraint** | **User defined exceptions to be thrown** | **Message to user to be displayed** |
| 1 | Category entered by the user should be a value present in the database | If category entered is not present, throw **IvalidCategoryException** | “Invalid category name please check your input” |

Test the manager class by using sample data in the table given below corresponding for each of the test case.

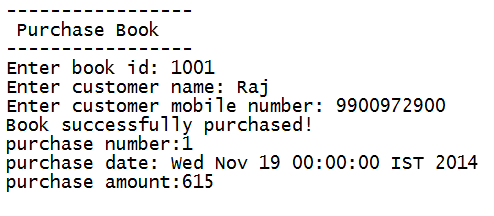
|  |  |  |
| --- | --- | --- |
| **SL NO** | **Test case method** | **Category** |
| 1 | testDisplayBooks | Technology |
| 2 | testDisplayBooksInvalidCategory | Management |

1. **Purchase a book:**

* When user selects option 2, user will input the **Book\_id, Customer\_name, and Customer\_mobileno.**
* Sample output shown below.



Write code to display the output in the below format by using the returned value if business rules are satisfied, otherwise display the appropriate exception message.



**BookStoreSystemManager class** code should check for the business rules given below and throw appropriate user defined exception, when exception thrown corresponding message is displayed to user, which is as given in the table:

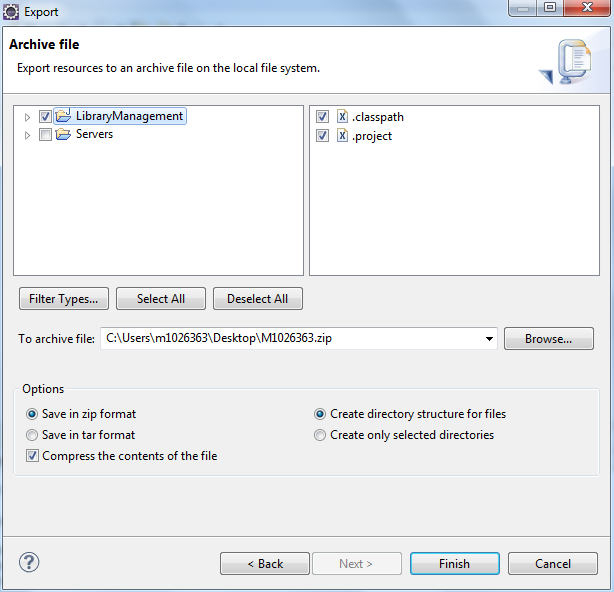
|  |  |  |  |
| --- | --- | --- | --- |
| **Rule No** | **Business constraint** | **User defined exceptions to be thrown** | **Message to user to be displayed** |
| 1 | Customer mobile No entered by the user should be of 10 digits | If rule is violated, throw InvalidMobileNoException | “Invalid mobile no, please check your input” |
| 2 | Book id must be valid | If rule is violated, throw InvalidBookIdException | “Invalid book id, please check your input” |

Test the manager class by using sample data in the table given below corresponding for each of the test case.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL NO** | **Test case method** | **BookID** | **Customer Name** | **Mobile No** |
| 1 | testPurchaseBook | 1001 | Raj | 9900972900 |
| 2 | testPurchaseBookInvalidBookId | 1009 | Raj | 9900972900 |
| 3 | testPurchaseBookInvalidMobileNo | 1001 | Raj | 9900972 |

**4. Exit**

When user selects option 3, application should terminate



Select the location to store the zip file in and click on ‘Finish’.

* Name the ZIP file as MXXXXXX.zip (It will be your employee ID) and add following files
  + Modified/Updated Eclipse projects
  + DB script
  + Upload MXXXXXX.zip as your final deliverable.