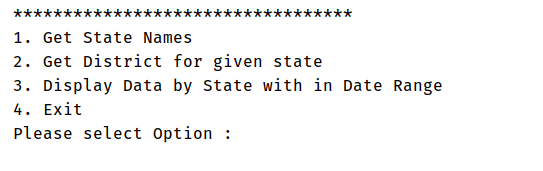
|  |
| --- |
| Covid Analysis System |

## Problem Statement

Design and implement an application program to automate the process of analysis of covid data and produce appropriate reports. The application stores a collection of covid data provided and provides a mechanism of generating the report as follows.

**The main menu contains following options:**

1. **Get States Name.**
2. **Get District name for given states**
3. **Display Data by state with in date range**
4. **Display Confirmed cases by comparing two states for a given date range.**
5. **Exit**

****

It is 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  **Controller class -> Service class ->DAO class -> Database**

### Instructions

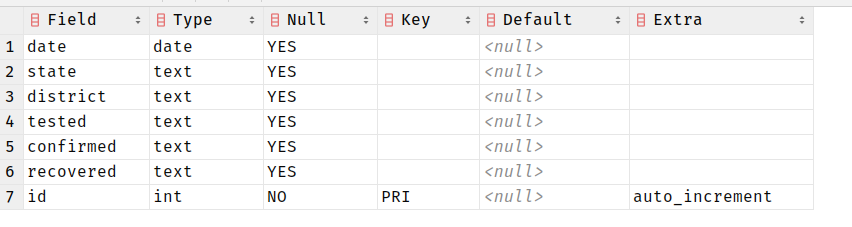
1. Java 8 features should be used to accomplish all the given requirements.
2. Create the table **‘covid\_data’**  which will be used in the application. For Table **‘covid\_data’**, please refer the SQL script provided for data.
3. Read the complete question before coding
4. All user defined exceptions must be created under com.mindtree.exceptions package
5. All exceptions must be handled in Service Class. DAO class should not handle any exception; it must throw it to Service Class.
6. Loose coupling: Create 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 all service class

* 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.

### Database design

**Create database “covid\_analysis” and create the following table used for the application:**

Table Name: **covid\_data**

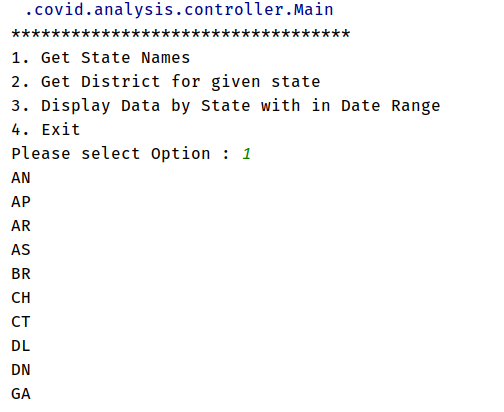
****

### Menu design sample

## Use case

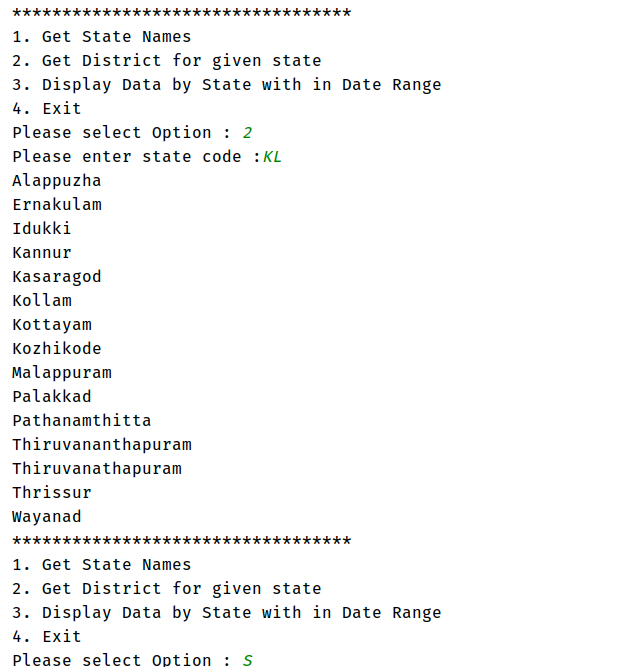
### 1. Get States Name.

* When the user selects option 1, the System will identify the unique state code and display it to the user**.**
* Please use the following Java features in implementation
  + - Stream
    - Lambda
    - Functional Interface - Predicate
    - Method Reference
* Sample output shown below.



### 2. Get District name for given states

* When the user selects Option 2, the user will input **state code**.
* Please use the following Java features in implementation
  + Stream
  + Lambda
  + Functional Interface - Predicate
  + Method Reference
* Sample output shown below.



Write code to display the output in below sorted order (**sort by** **district name**) by using the returned values if the business rules are satisfied; otherwise display the appropriate exception message.

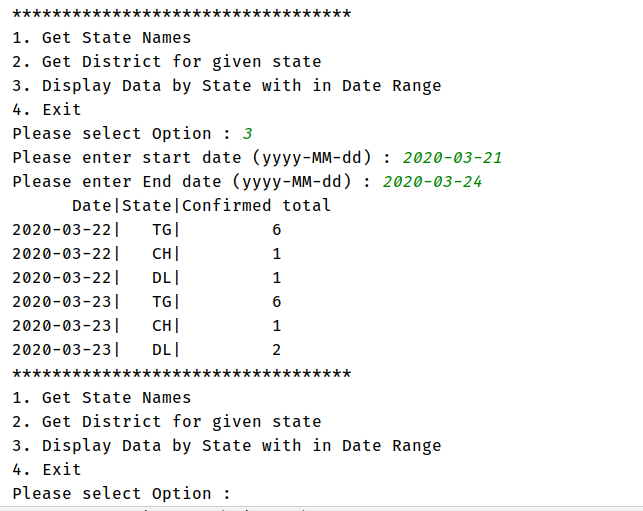
Your code should check for the business rules given below and throw appropriate user defined exception, when exception thrown corresponding message is displayed to user, which is as given in the table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Rule No** | **Business constraint** | **User defined exceptions to be thrown** | **Message to user to be displayed** |
| 1 | **State code** entered should be present in the database. | If **State code** is invalid, throw **InvalidStateCodeException** | “Invalid State code, please check your input” |

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

### 3. Display Data by state with in date range

* When the user selects Option 3, the user will input **start and end date range**.
* Please use the following Java features in implementation
  + Stream
  + Lambda
  + Functional Interface - Predicate
  + Method Reference
  + Functional Interface - BiPredicate
  + Functional Interface - BiFunction
* Output will display following information
  + Date
  + State
  + Confirmed total
  + Tested total
  + Recovered total
* Sample output shown below.



Write code to display the output in below sorted order (**sort by** **date**) by using the returned values if the business rules are satisfied; otherwise display the appropriate exception message.

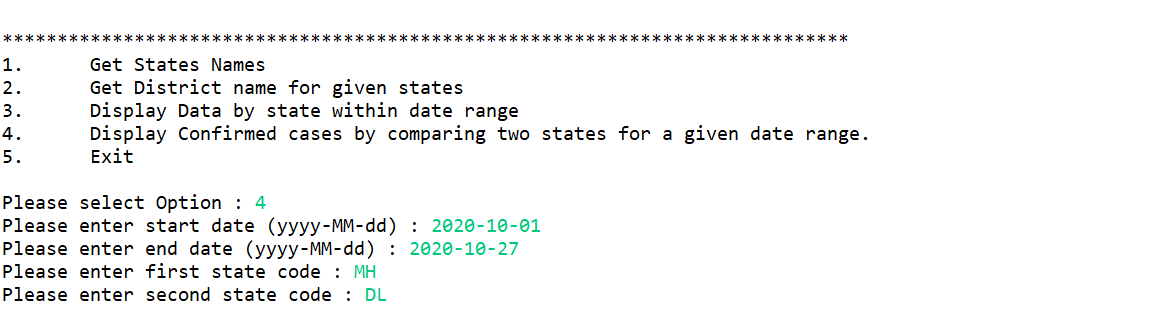
Your code should check for the business rules given below and throw appropriate user defined exception, when exception thrown corresponding message is displayed to user, which is as given in the table:

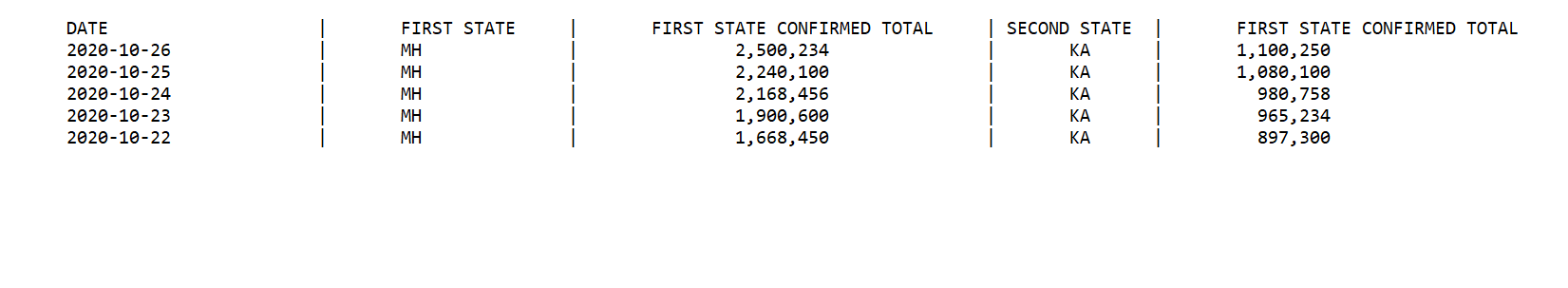
|  |  |  |  |
| --- | --- | --- | --- |
| **Rule No** | **Business constraint** | **User defined exceptions to be thrown** | **Message to user to be displayed** |
| 1 | **Start date** is not valid. | If **Start date** is invalid, throw **InvalidDateException** | “Invalid Start date, please check your input” |
| 2 | **End date** is not valid. | If **End date** is invalid, throw **InvalidDateException** | “Invalid End date, please check your input” |
| 3 | End date is before start date | If End date is before start date, throw InvalidDateRangeException | “Invalid Date Range, Please check your input” |
| 4 | If not data present | If not data available, throw NoDataFoundException | “No data present” |

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

### 4. Display Confirmed cases by comparing two states for a given date range.

* When the user selects Option 4, the user will input **start and end date range & State code (First and second)**.
* Please use the following Java features in implementation
  + Stream
  + Lambda
  + Functional Interface - Predicate
  + Method Reference
  + Functional Interface - BiPredicate
  + Functional Interface - BiFunction
* Output will display following information
  + Date
  + First State
  + First state Confirmed total
  + Second state
  + Second state confirmed total
* Sample output shown below.





### 

### 5. Exit

When user selects option 4, application should terminate

**Project contains**

1. **Player entity class in “com.mindtree.entity” package**
2. **“com.mindtree.dao” package should contain all the database interaction code.**