**Book Model: -**

package com.book.model  
  
data class Book (var bookID:String,var title:String,var author:String,var category: String,var price:Float)

Book Validation: -

package com.book.app

import com.book.model.\*

class BookValidation{

// validation for price

fun validatePrice(price:Float){

if(price<0)

{

throw InvalidBookException("Invalid Book Exception $price Do not enter Negative values")

}

}

//validation for category

fun validateCategory(category: String){

if(category!="Science" && category!="Others" && category!="Fiction" && category!="Technology")

{

throw InvalidBookException("Invalid Book Exception $category must be science or others or fiction or technology")

}

}

//validate for bookid

fun validateBookID(bookID:String){

if(!(bookID.startsWith('B') && bookID.length==4)){

throw InvalidBookException("Invalid book exception $bookID book id strat with captial B and length should be 4")

}

}

}

**BookInvalidException: -**

package com.book.app

import java.lang.Exception

class InvalidBookException(message:String):Exception(message)

**Book Service: -**

package com.book.service

import com.book.model.Book

import com.book.util.BookUtil

class BookService {

var BookService=BookUtil()

fun addBook(book:Book){

BookService.bookInsert(book)

}

fun searchByTitle(title:String){

BookService.searchByTitle(title)

}

fun searchByAuthor(author:String){

BookService.searchByAuthor(author)

}

fun display(){

BookService.displayAllDetails()

}

fun bookModify(id:String,column:String,value:String){

BookService.modifyBook(id,column,value)

}

fun deleteBook(bookID: String){

BookService.delete(bookID)

}

fun displaySpecificBookID(bookID:String){

BookService.displaySpecificBook(bookID)

}

}

**Book Util: -**

package com.book.util

import com.book.model.Book

class BookUtil {

var dbConnection=dbConnection()

var connection=dbConnection.connect()

var books= mutableListOf<Book>()

fun bookInsert(book: Book){

var preparedStatement=connection.prepareStatement("insert into Book values(?,?,?,?,?)")

preparedStatement.setString(1,book.bookID)

preparedStatement.setString(2,book.title)

preparedStatement.setString(3,book.author)

preparedStatement.setString(4,book.category)

preparedStatement.setFloat(5,book.price)

var result=preparedStatement.executeUpdate()

if(result>0){

println("Inserted")

}

else{

println("Not Inserted")

}

}

fun displayAllDetails(){

var preparedStatement=connection.prepareStatement("select \* from Book")

var res=preparedStatement.executeQuery()

while(res.next()){

var bookID=res.getString("bookID")

var title=res.getString("title")

var author=res.getString("author")

var category=res.getString("category")

var price=res.getFloat("price")

books.add(Book(bookID,title,author, category, price))

}

for(book in books){

println(book)

}

}

fun searchByTitle(title:String){

var preparedStatement=connection.prepareStatement("select \* from Book where title=(?)")

val res=preparedStatement.executeQuery()

while(res.next()){

var bookID=res.getString("bookID")

var title=res.getString("title")

var author=res.getString("author")

var category=res.getString("category")

var price=res.getFloat("price")

books.add(Book(bookID,title,author, category, price))

}

for(book in books){

println(book)

}

}

fun searchByAuthor(author:String){

var preparedStatement=connection.prepareStatement("select \* from Book where author=(?)")

val res=preparedStatement.executeQuery()

while(res.next()){

var bookID=res.getString("bookID")

var title=res.getString("title")

var author=res.getString("author")

var category=res.getString("category")

var price=res.getFloat("price")

books.add(Book(bookID,title,author, category, price))

}

for(book in books){

println(book)

}

}

fun delete(bookID:String){

var preparedStatement=connection.prepareStatement("delete from Book where bookID=?")

preparedStatement.setString(1,bookID)

var res=preparedStatement.executeUpdate()

if(true){

println("Deleted")

}

else{

println("Not Deleted")

}

}

fun displaySpecificBook(bookID: String)

{

val prestmt=connection.prepareStatement("select \* from books where bookId=? ")

prestmt.setString(1,bookID)

val result=prestmt.executeQuery()

val books= mutableListOf<Book>()

while(result.next()){

val bookId=result.getString("bookId")

val title=result.getString("title")

val author=result.getString("author")

val category=result.getString("category")

val price=result.getFloat("price")

books.add(Book(bookId,title,author,category,price))

}

for(book in books){

println(book)

}

}

fun modifyBook(id:String,column:String,value:String){

var message=""

if(column=="title"){

val preparedStatement=connection.prepareStatement("update Book set title=? where bookID=?")

preparedStatement.setString(1,value)

preparedStatement.setString(2,id)

val res=preparedStatement.executeUpdate()

if(res>0)

message="title modified"

}

else if(column=="author"){

val preparedStatement=connection.prepareStatement("update Book set author=? where bookID=?")

preparedStatement.setString(1,value)

preparedStatement.setString(2,id)

val res=preparedStatement.executeUpdate()

if(res>0)

message="author modified"

}

else if(column=="category"){

val preparedStatement=connection.prepareStatement("update Book set category=? where bookID=?")

preparedStatement.setString(1,value)

preparedStatement.setString(2,id)

val res=preparedStatement.executeUpdate()

if(res>0)

message="category modified"

}

else if(column=="price"){

val preparedStatement=connection.prepareStatement("update Book set price=? where bookID=?")

preparedStatement.setFloat(1,value.toFloat())

preparedStatement.setString(2, id)

val res=preparedStatement.executeUpdate()

if(res>0)

message="price modified"

}

else{

message="book not found"

}

println( message)

}

}

**Book dbConnection: -**

package com.book.util

import java.sql.Connection

import java.sql.DriverManager

class dbConnection{

fun connect():Connection{

var myurl="jdbc:mysql://localhost:3306/kotlindb"

var connection=DriverManager.getConnection(myurl,"root","hema")

return connection

}

}

**Book App: -**

package com.book.app

import com.book.model.Book

import java.util.Scanner

import com.book.service.\*

fun main() {

var bookService=BookService()

var bookValidation=BookValidation()

var scanner=Scanner(System.`in`)

while(true) {

println(

"1. Add book\n"+ "2. Modify book details\n" +

"3. Delete book details\n" +

"4. Display All the book details\n" +

"5. Display specific book details\n" +

"6. Search book by title\n" +

"7. Search book by author\n" +

"8. Exit"

)

println("----------------------------------------------------------------------------------------")

print("Enter the choice :")

var choice = scanner.nextInt()

when (choice) {

1-> {

print("Enter the bookId :")

var bookID = scanner.next()

try{

bookValidation.validateBookID(bookID)

}

catch(e:InvalidBookException){

println(e.message)

}

print("Enter the title :")

var title=scanner.nextLine()

scanner.next()

print("Enter the author :")

var author=scanner.nextLine()

scanner.next()

print("Enter the category :")

var category=scanner.next()

try {

bookValidation.validateCategory(category)

}

catch (e:InvalidBookException){

println(e.message)

}

print("Enter the price :")

var price=scanner.nextFloat()

try {

bookValidation.validatePrice(price)

}

catch (e:InvalidBookException){

println(e.message)

}

var book=Book(bookID, title, author, category, price)

bookService.addBook(book)

break

}

2->{

println("Enter book Id to modify")

var id=scanner.nextLine()

println("Enter which column you want to edit")

var column=scanner.nextLine()

println("Enter which the value to be updated")

var value=scanner.nextLine()

bookService.bookModify(id,column,value)

}

3->{

print("Enter the bookId :")

var bookID=scanner.next()

bookService.deleteBook(bookID)

}

4->{

bookService.display()

}

5->{

}

6->{

print("Enter the book title you want to search :")

var title=scanner.nextLine()

bookService.searchByTitle(title)

}

7->{

print("Enter the book author you want to search :")

var author=scanner.nextLine()

bookService.searchByAuthor(author)

}

8->{

//exit

System.exit(0)

}

else->{

println("Enter wrong details ")

break

}

}

}

}