Name: Lekha J

Roll Number: 231047010

JAVA ASSIGNMENT – 3

1. Illustrate the implementation of atomicity and durability related to database operations through an appropriate java program

package lekha;

import java.util.\*;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class DayCare {

private static final Connection NULL = null;

public static void main(String args[]) {

Connection con = NULL;

Scanner sc = new Scanner(System.in);

System.out.println("Book Details");

try {

String url="jdbc:sqlserver://172.16.51.64;"+" "+"databaseName=231047010;encrypt=true;trustServerCertificate=true";

String username="lekha";

String password="lekha@321";

con=DriverManager.getConnection(url,username,password);

con.setAutoCommit(false);

System.out.println("Connection established");

System.out.println("Enter the ID of the book");

int word = sc.nextInt();

if(!checkBookExists(con, word)) {

System.out.println("Enter the ID of the book");

return;

}

System.out.println("Enter the new price");

int newPrice = sc.nextInt();

if (newPrice < 0) {

throw new IllegalArgumentException("Price cannot be negative ");

}

//PreparedStatement stmt = con.prepareStatement("INSERT INTO BookMaster values(?,?,?)");

//stmt.setInt(1, word);

//stmt.setString(2, name);

//stmt.setInt(3, cost);

updateBookPrice(con, word, newPrice);

System.out.println("updated succeesfully");

//stmt.executeUpdate();

con.commit();

}catch(SQLException | IllegalArgumentException e) {

System.out.println("Error Occurred:" + e.getMessage());

try {

if(con != null) {

con.rollback();

}

}catch (SQLException rollbackException) {

rollbackException.printStackTrace();

}

//System.out.println("Error Occured" + e.getMessage());

} finally {

try {

if(con != null) {

con.setAutoCommit(true);

con.close();

System.out.println("Connection Closed");

}

} catch (SQLException closeException) {

closeException.printStackTrace();

}

}

}

private static boolean checkBookExists(Connection con, int bookId) throws SQLException {

try (PreparedStatement checkStmt = con.prepareStatement("SELECT \* FROM BookMaster WHERE BookID = ?")) {

checkStmt.setInt(1, bookId);

return checkStmt.executeQuery().next();

}

}

private static void updateBookPrice(Connection con, int bookId, int newPrice) throws SQLException {

// Update the price of the book with the given ID using PreparedStatement

try (PreparedStatement updateStmt = con.prepareStatement("UPDATE BookMaster SET Price = ? WHERE BookID = ?")) {

updateStmt.setInt(1, newPrice);

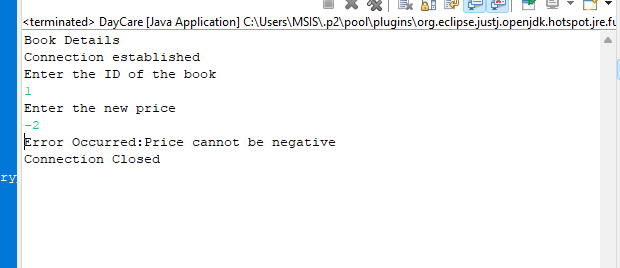
updateStmt.setInt(2, bookId);

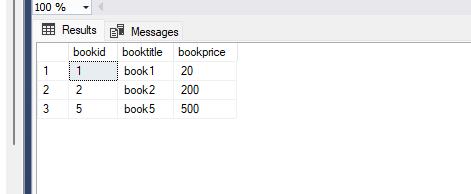
updateStmt.executeUpdate();

}

}

}





1. Stored procedure

