**Case Study 1:**

CREATE TABLE categories(

categoryId INT AUTO\_INCREMENT PRIMARY KEY,

categoryName VARCHAR(100) NOT NULL

) ;

INSERT INTO categories VALUES ( default, ‘Electronics’);

INSERT INTO categories VALUES ( default, ‘Clothings’);

INSERT INTO categories VALUES ( default, ‘Books’);

INSERT INTO categories VALUES ( default, ‘Toys’);

CREATE TABLE products(

productId INT AUTO\_INCREMENT PRIMARY KEY,

productName varchar(100) not null,

categoryId INT,

CONSTRAINT fk\_category

FOREIGN KEY (categoryId)

REFERENCES categories(categoryId)

);

Com.psl.prodApp.dao

Interface ProductDao

{

Product addProduct(Product obj);

HashMap<Integer,Product> getAllProducts();

ArrayList<Product> getProductById(int productId);

Product deleteProduct(int productid);

}

Class ProductDaoImpl implements ProductDao {

//implement all methods from dao interface

//you can implement more methods if needed

}

Com.psl.prodApp.service

Interface ProductService

{

Product addProduct(Product obj);

HashMap<Integer,Product> getAllProducts();

ArrayList<Product> getProductById(int productId);

Product deleteProduct(int productid);

}

Class ProductServiceImpl implements ProductService

{

}

Com.psl.prodApp.dto

Class Product

{

Int productid, categoryId;

String productName;

//implement get set methods, constructor and toString()

}

Class Category

{

Int categoryId;

String categoryName;

//implement get set methods, constructors and toString()

}

Com.psl.prodApp.util

Class DBUtil {

//obtain database connection

}

Com.psl.prodApp.pl

Class MainClient

{

// write a switch case in do while for below cases

1. addProduct()

First display category details to client, accept ProductName and category Id from client. Pass it as product object to addProduct method. After successful insertion productid and productName should be printed as a success message

1. getAllProducts()

Show all products to client, category wise.

1. getProductById()

show product details by accepting productid from client

1. delete product details from Product table by accepting productid from client.

}

Perform Exception Handling wherever necessary.

**Case Study 2:**

CREATE TABLE BusDetails ( busId int AUTO\_INCREMENT PRIMARY KEY, busType text(20), FROM text(20), To text(20), Date Date, noOfSeats int);

INSERT INTO BusDetails VALUES(110, ‘AC-Shivneri’, ‘Pune’,’Dadar’,’26-Mar-2020’,50);

INSERT INTO BusDetails VALUES(111, ‘Ashwamedh’, ‘Dadar’,’Pune’,’27-Mar-2020’,40);

INSERT INTO BusDetails VALUES(112, ‘AC-Shivneri’, ‘Borivali’,’Pune’,’27-Mar-2020’,40);

INSERT INTO BusDetails VALUES(113, ‘Shivshahi’, ‘Pune’,’Thane’,’26-Mar-2020’,40);

INSERT INTO BusDetails VALUES(114, ‘AC-Shivneri’, ‘Pune’,’Thane’,’26-Mar-2020’,40);

INSERT INTO BusDetails VALUES(115, ‘AC-Shivneri’, ‘Pune’,’Borivali’,26-Mar-2020’,40);

Perform below operations on the table given.

1. HashMap<Integer,Bus>showAllBusDetails();
2. HashMap<Integer,Bus> getBusDetails(String from,String to);
3. Bus bookBus(busId);

Payment is out of scope for booking, update noOfSeats after successful booking and return the busDetails to client.

**Case study 3 :**

CREATE TABLE UserGameDetails( userId int PRIMARY KEY, username text(20), balance int );

INSERT INTO UserGameDetails values(101, ‘Pooja’,800);

INSERT INTO UserGameDetails values(102, ‘Ram’,400);

INSERT INTO UserGameDetails values(103, ‘Priti’,700);

INSERT INTO UserGameDetails values(104, ‘kiran’,800);

INSERT INTO UserGameDetails values(105, ‘Prashant’,800);

INSERT INTO UserGameDetails values(106, ‘Poorvi’,800);

In dao class create HashMap < String, Integer> map = new HashMap<String, Integer> ();

String -> game name

Integer->game price

HashMap.put(“Mario”,100); -> like this add more values to HashMap

In client class first display Games with price details.

Ask user do you want to play game?

If answer is yes , ask user to enter userid and game name.

If userId entered by user is valid as per the userGameDetails table then update the balance in the table by deducting the game amount. Display a message on client side ‘name of user thanks for playing game name and your current balance is balance’.

**Case study 4:**

Class Department

{

Private int deptId;

Private String deptName;

Private int strength;

//add get set methods, constructors and toString()

}

Class Student

{

Private int stuId;

Private String stuName;

Private String deptName;

//add get set methods, constructors and toString()

}

Create 2 HashMaps

HashMap<Integer,Department>departments

Key will be departmentId and value will Department object.

Department obj1 = new Department(101, ‘Electronics’,40);

departments.put(obj1.getStuId(),obj1);

like this add more values to this map

HashMap<Integer,Student>Students

Key will be stuId and value will be studentObject

Perform below operations

1. Accept student details from user , validate deptName entered by user from the departments HashMap, if deptName is valid then only add accepted details to Student object and add Student object to Students HashMap
2. Delete student-> accept stuId from user and delete the details of student from the HashMap students
3. Show details of students with respect to deptName