# Assessment Parameters

* Complete flow of the application  with validation and exception handling -70%
* Comments/best practice, coding standards – 10%
* Execution of the application (Output) – 20%
* ScreenShot should be submitted along with the solution
* The solution(Project) created by the trainee should have the name like AppName\_Empid Ex:ABCCorp\_675467
* Code with compilation errors will not be considered for evaluation

# Billing Software Application

**Problem Statement**: A Take Home company requires a billing software for its super market where different types of products are sold to the customers on retail basis. This billing software will help

1. the cashier to see the product information on entering the product code
2. calculate the total cost for each product based on the quantity
3. and display the final bill to user.

***Collection Schema***

Consider one Hash Map products which contain **product Id** as key and **Product** object as value;

**Display menu to the Cashier should be as follows**

1. **Generate Bill by entering Product code and quantity**
2. **Exit**
3. **Generate Bill by entering Product code and quantity**
4. **Enter Product Details [60 Marks]**

Following information should be accepted from the cashier when he/she bills the product chosen by the customer.

**Enter the product code**: 1001

**Enter the quantity**: 2

**Note**:

* + 1. Quantity should not be lesser than or equal to zero
    2. Validation should be taken care for all input fields(i.e. ProductCode should be 4 digit )

1. **Calculating Total and displaying the bill**

When the above details are entered by the cashier, this application should calculate the line total (product\_price \* quantity), display the following information

**Product Name: iPhone**

**Product Category: Electronics**

**Product Description: Smart Phone**

**Product Price (Rs): 35000**

**Quantity: 2**

**Line Total (Rs): 70000**

1. **If the Product Code is not available in existing HashMap collection following error message will be** **displayed**

**Sorry ! The Product Code<<product\_code>> is not available.**

* **Mark Distribution:**

|  |  |
| --- | --- |
| Validation of product code and quantity | 5 |
| Retrieving product information based on product code(DAO + Service + bean) from collection | 15 |
| Displaying product information( Client layer ) | 10 |
| Calculating line total | 10 |
| Display error message if product code not available | 5 |
| Comments and best practices/standards | 5 |
| Proper Exception handling | 5 |

Write JUNIT test cases for the method that searches the product information in the dao class  **[ 10 Marks]**

**2 Exit**

When Cashier selects this option, he/she should be able to quit from application

**Classes to be created**

**com.capgemini.takehome.ui**

class Client {

public static void main (

// User Interface which will accept the input from user

// Create object for service and execute the respective methods}

com.capgemini.takehome.bean

class Product {

// Bean classes to be created with all required properties like product id, product name ,product category ,product price etc.

}

com.capgemini.takehome.service // Service utility classes and Interfaces

interface IProductService {

Product getProductDetails (int productCode);

}

class ProductService implements IProductService {

Product getProductDetails (int productCode){ . . . . . }

}

-----------------------------------------------------------------------

com.capgemini.takehome.dao // Class to perform Data Accessing logic

interface IProductDAO{

Product getProductDetails (int productCode);

}

class ProductDAO implements IProductDAO{

Product getProductDetails (int productCode){ . . . . . }

}

----------------------------------------------------------------

com.capgemini.takehome.util **-**// package containing data access class to perform utility operations

public class CollectionUtil{

private static Map<Integer, Product> products=new HashMap<Integer, Product>();

static {

products.put(1001, new Product(1001, “iPhone”,” Electronics”, 35000));

products.put(1002, new Product(1002, “LEDTV”,” Electronics”, 45000));

products.put(1003, new Product(1003, “Teddy”,” Toys”, 800));

products.put(1004, new Product(1004, “Telescope”,” Toys”, 5000));

}

……..

}

**Add appropriate user defined exception classes and any other supporting classes required.**