**Exercise 1: Inventory Management System**

Scenario:

You are developing an inventory management system for a warehouse. Efficient data storage and retrieval are crucial.

Steps:

**1. Understand the Problem:**

* Explain why data structures and algorithms are essential in handling large inventories.

Ans -:

* Data structure represents logical arrangements of data in memory in a particular model. Data structures are essential in handling large repositories because they can be used to retrieve or manipulate on the data in a very easier way.
* Algorithms are a sequence of computational steps that transform the input into the output and it helps in solving a well-specified computational problem. Thus, algorithms are essential in handling large inventories as they specifies the Pre-requisites of the problem and also the time & space complexity of the problem.
* Discuss the types of data structures suitable for this problem.

Ans -:

* For the given problem the data structure that is required is Array, but as an array can store only numeric values thus an ArrayList is required to store alphanumeric values.
* For the given problem the data structure Vector can also be used to store alphanumeric values.

**2. Setup:**

* Create a new project for the inventory management system.
* Project created with the name “Inventory\_Management\_System” and word document “Theory.docx” and Java file “Driver.java” contained in it.

**3. Implementation:**

* Define a class Product with attributes like productId, productName, quantity, and price.
* Choose an appropriate data structure to store the products (e.g., ArrayList, HashMap).
* Implement methods to add, update, and delete products from the inventory.

**SOURCE CODE -:**

import java.util.\*;

import java.io.\*;

class Driver{

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

Product ob=new Product();

int i,ch=0;

ArrayList PD=new ArrayList();

ArrayList pd=new ArrayList();

while(ch!=5){

System.out.println("-----------------");

System.out.println("1. INSERT");

System.out.println("2. DELETE");

System.out.println("3. UPDATE");

System.out.println("4. DISPLAY");

System.out.println("5. EXIT");

System.out.print("\n Enter choice : ");

ch=sc.nextInt();

if(ch==1){

pd=ob.insert();

PD.add(pd);

}

else if(ch==2){

System.out.print("\n Enter postion : ");

i=sc.nextInt();

PD=ob.delete(PD,i);

}

else if(ch==3){

System.out.print("\n Enter postion : ");

i=sc.nextInt();

pd=ob.insert();

PD=ob.update(PD,pd,i);

}

else if(ch==4){

ob.display(PD);

}

else{

System.out.println("EXIT");

break;

}

}

}

}

class Product{

String productname;

int productid, quantity;

double price;

ArrayList insert(){

Scanner sc=new Scanner(System.in);

System.out.print("\n Enter product-name = ");

productname=sc.next();

System.out.print("\n Enter product-ID = ");

productid=sc.nextInt();

System.out.print("\n Enter product-quantity = ");

quantity=sc.nextInt();

System.out.print("\n Enter product-price = ");

price=sc.nextDouble();

ArrayList P=new ArrayList();

P.add(productid);

P.add(productname);

P.add(quantity);

P.add(price);

return P;

}

ArrayList delete(ArrayList arr,int j){

arr.remove(j);

return arr;

}

ArrayList update(ArrayList L,ArrayList li,int k){

L.set(k,li);

return L;

}

void display(ArrayList LI){

System.out.println(LI);

}

}

**OUTPUT -:**

Microsoft Windows [Version 10.0.22621.3880]

(c) Microsoft Corporation. All rights reserved.

D:\Trisita\Java\_Progs\Inventory\_Management\_System>javac Driver.java

Note: Driver.java uses unchecked or unsafe operations.

Note: Recompile with -Xlint:unchecked for details.

D:\Trisita\Java\_Progs\Inventory\_Management\_System>java Driver

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 1

Enter product-name = Rice

Enter product-ID = 211

Enter product-quantity = 4

Enter product-price = 5000

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 4

[[211, Rice, 4, 5000.0]]

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 1

Enter product-name = Grains

Enter product-ID = 232

Enter product-quantity = 6

Enter product-price = 7890.67

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 4

[[211, Rice, 4, 5000.0], [232, Grains, 6, 7890.67]]

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 3

Enter postion : 1

Enter product-name = Wheat

Enter product-ID = 222

Enter product-quantity = 5

Enter product-price = 6000

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 4

[[211, Rice, 4, 5000.0], [222, Wheat, 5, 6000.0]]

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 3

Enter postion : 1

Enter product-name = Wheat

Enter product-ID = 222

Enter product-quantity = 2

Enter product-price = 6000

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 4

[[211, Rice, 4, 5000.0], [222, Wheat, 2, 6000.0]]

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 2

Enter postion : 0

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 4

[[222, Wheat, 2, 6000.0]]

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

1. INSERT

2. DELETE

3. UPDATE

4. DISPLAY

5. EXIT

Enter choice : 5

EXIT

D:\Trisita\Java\_Progs\Inventory\_Management\_System>

**4. Analysis:**

* Analyse the time complexity of each operation (add, update, delete) in your chosen data structure.
* In add operation as no loops are present but ArrayList in-built function “add()” is called which adds a single value to the list, so constant time is required in units.
* In update operation no loops are present but ArrayList in-built function “set()” is called which replaces only one position value, so again constant time is required in units.
* In delete operation no loops are present but ArrayList in-built function “remove()” is called which removes only one position value, so again constant time is required in units.
* Discuss how you can optimize these operations.
* These operations can be optimized when we make use of JDBC, which means connection with the database, so no data will be lost and insertion, deletion and updating will take place in a very easy way.