import java.util.HashMap;

import java.util.NoSuchElementException;

import java.util.Scanner;

import java.util.Vector;

public class Main extends FunctionImpl {

private static HashMap<String, Object> map = new HashMap<String, Object>();

Vector<Object> soldobjects = new Vector<Object>();

Vector<Object> deletedobjects = new Vector<Object>();

private static Main instance = null;

private static String value;

public Main() {

}

public static synchronized Main returnMain() {

if (instance == null) {

instance = new Main();

}

return instance;

}

public HashMap<String, Object> returnMap() {

return map;

}

public Vector<Object> returnVector() {

return soldobjects;

}

public Vector<Object> returnVectordeleted() {

return deletedobjects;

}

public static void main(String[] args) {

System.out.println("Welcome to Inventory Management System");

System.out.println("Please choose from following:");

System.out.println(" ");

System.out.println("1. create itemName costPrice sellingPrice --> For creating new object");

System.out.println(" ");

System.out.println("2. delete itemName --> For removing object");

System.out.println(" ");

System.out.println("3. updateBuy itemName quantity --> For increasing quantity of object after purchased additional quantity");

System.out.println(" ");

System.out.println("4. updateSell itemName quantity --> For reducing quantity of object after sold");

System.out.println(" ");

System.out.println("5. updateSellPrice itemName newSellPrice --> For changing selling price of particular object");

System.out.println(" ");

System.out.println("6. report --> To generate the report");

System.out.println(" ");

Main function = new Main();

while(true) {

Scanner input = new Scanner(System.in);

String command = input.nextLine();

String[] commandToArray = command.split(" ");

if(commandToArray[0].equals("create") && commandToArray.length == 4) {

try {

value = commandToArray[1];

function.create(value, Double.parseDouble(commandToArray[2]), Double.parseDouble(commandToArray[3]));

System.out.println("Object created successfully");

continue;

} catch(IllegalStateException e) {

System.out.println(e.getMessage());

} catch(Exception e) {

System.out.println("Invalid command, Please try again");

}

}

else if(commandToArray[0].equals("delete") && commandToArray.length == 2) {

try {

value = commandToArray[1];

function.delete(value);

System.out.println("Object removed successfully");

continue;

}

catch (NoSuchElementException e) {

System.out.println(e.getMessage());

continue;

} catch(Exception e) {

System.out.println("Invalid command, Please try again.");

}

}

else if(commandToArray[0].equals("updateBuy") && commandToArray.length == 3) {

try {

value = commandToArray[1];

function.updateBuy(value, Integer.parseInt(commandToArray[2]));

System.out.println("current quantity is " + map.get(value).quantity);

continue;

} catch(NoSuchElementException e) {

System.out.println(e.getMessage());

continue;

} catch(IllegalStateException e) {

System.out.println(e.getMessage());

continue;

} catch (Exception e) {

System.out.println("Invaid command, Please try again.");

}

}

else if(commandToArray[0].equals("updateSell") && commandToArray.length == 3) {

try {

value = commandToArray[1];

function.updateSell(value, Integer.parseInt(commandToArray[2]));

System.out.println("current quantity is " + map.get(value).quantity);

continue;

} catch(NoSuchElementException e) {

System.out.println(e.getMessage());

continue;

} catch(IllegalStateException e) {

System.out.println(e.getMessage());

continue;

} catch (Exception e) {

System.out.println("Invaid command, Please try again.");

}

}

else if(commandToArray[0].equals("report") && commandToArray.length == 1) {

ReportImpl report = new ReportImpl();

report.report();

}

else if(commandToArray[0].equals("updateSellPrice") && commandToArray.length == 3) {

try {

value = commandToArray[1];

function.updateSellPrice(value, Double.parseDouble(commandToArray[2]));

} catch(Exception e) {

System.out.println(e.getMessage());

}

}

else {

System.out.println("Invalid command, Please try Again");

continue;

}

}

}

}

import java.util.HashMap;

import java.util.NoSuchElementException;

import java.util.Scanner;

import java.util.Vector;

public class Main extends FunctionImpl {

private static HashMap<String, Object> map = new HashMap<String, Object>();

Vector<Object> soldobjects = new Vector<Object>();

Vector<Object> deletedobjects = new Vector<Object>();

private static Main instance = null;

private static String value;

public Main() {

}

public static synchronized Main returnMain() {

if (instance == null) {

instance = new Main();

}

return instance;

}

public HashMap<String, Object> returnMap() {

return map;

}

public Vector<Object> returnVector() {

return soldobjects;

}

public Vector<Object> returnVectordeleted() {

return deletedobjects;

}

public static void main(String[] args) {

System.out.println("Welcome to Inventory Management System");

System.out.println("Please choose from following:");

System.out.println(" ");

System.out.println("1. create itemName costPrice sellingPrice --> For creating new object");

System.out.println(" ");

System.out.println("2. delete itemName --> For removing object");

System.out.println(" ");

System.out.println("3. updateBuy itemName quantity --> For increasing quantity of object after purchased additional quantity");

System.out.println(" ");

System.out.println("4. updateSell itemName quantity --> For reducing quantity of object after sold");

System.out.println(" ");

System.out.println("5. updateSellPrice itemName newSellPrice --> For changing selling price of particular object");

System.out.println(" ");

System.out.println("6. report --> To generate the report");

System.out.println(" ");

Main function = new Main();

while(true) {

Scanner input = new Scanner(System.in);

String command = input.nextLine();

String[] commandToArray = command.split(" ");

if(commandToArray[0].equals("create") && commandToArray.length == 4) {

try {

value = commandToArray[1];

function.create(value, Double.parseDouble(commandToArray[2]), Double.parseDouble(commandToArray[3]));

System.out.println("Object created successfully");

continue;

} catch(IllegalStateException e) {

System.out.println(e.getMessage());

} catch(Exception e) {

System.out.println("Invalid command, Please try again");

}

}

else if(commandToArray[0].equals("delete") && commandToArray.length == 2) {

try {

value = commandToArray[1];

function.delete(value);

System.out.println("Object removed successfully");

continue;

}

catch (NoSuchElementException e) {

System.out.println(e.getMessage());

continue;

} catch(Exception e) {

System.out.println("Invalid command, Please try again.");

}

}

else if(commandToArray[0].equals("updateBuy") && commandToArray.length == 3) {

try {

value = commandToArray[1];

function.updateBuy(value, Integer.parseInt(commandToArray[2]));

System.out.println("current quantity is " + map.get(value).quantity);

continue;

} catch(NoSuchElementException e) {

System.out.println(e.getMessage());

continue;

} catch(IllegalStateException e) {

System.out.println(e.getMessage());

continue;

} catch (Exception e) {

System.out.println("Invaid command, Please try again.");

}

}

else if(commandToArray[0].equals("updateSell") && commandToArray.length == 3) {

try {

value = commandToArray[1];

function.updateSell(value, Integer.parseInt(commandToArray[2]));

System.out.println("current quantity is " + map.get(value).quantity);

continue;

} catch(NoSuchElementException e) {

System.out.println(e.getMessage());

continue;

} catch(IllegalStateException e) {

System.out.println(e.getMessage());

continue;

} catch (Exception e) {

System.out.println("Invaid command, Please try again.");

}

}

else if(commandToArray[0].equals("report") && commandToArray.length == 1) {

ReportImpl report = new ReportImpl();

report.report();

}

else if(commandToArray[0].equals("updateSellPrice") && commandToArray.length == 3) {

try {

value = commandToArray[1];

function.updateSellPrice(value, Double.parseDouble(commandToArray[2]));

} catch(Exception e) {

System.out.println(e.getMessage());

}

}

else {

System.out.println("Invalid command, Please try Again");

continue;

}

}

}

}

public class Object {

String item;

double costPrice;

double sellingPrice;

int quantity;

public Object(String Item, double CostPrice, double SellingPrice , int quantity){

this.item = Item;

this.costPrice = CostPrice;

this.sellingPrice = SellingPrice;

this.quantity = quantity;

}

public String getItem() {

return item;

}

public void setItem(String item) {

this.item = item;

}

public double getCostPrice() {

return costPrice;

}

public void setCostPrice(double costPrice) {

this.costPrice = costPrice;

}

public double getSellingPrice() {

return sellingPrice;

}

public void setSellingPrice(double sellingPrice) {

this.sellingPrice = sellingPrice;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

}

import java.util.HashMap;

import java.util.TreeMap;

import java.util.Vector;

public class ReportImpl implements Report{

HashMap<String, Object> map;

Vector<Object> deletedobjects;

Vector<Object> soldobjects;

public void report() {

map = Main.returnMain().returnMap();

TreeMap<String, Object> sorted = new TreeMap<>(map);

double value1 = 0;

System.out.println(String.format("%25s", "INVENTORY REPORT"));

System.out.println(String.format("%-12s %-12s %-10s %-21s %-15s", "item Name", "Bought At", "Sold At", "Available Quantity", "value"));

System.out.println(String.format("%-12s %-12s %-10s %-21s %-22s", "---------", "---------", "-------", "------------------", "-----"));

for (String key : sorted.keySet()) {

double value = sorted.get(key).costPrice \* sorted.get(key).quantity;

System.out.println(String.format("%-12s %-12s %-12s %-20s %-12s", sorted.get(key).item, sorted.get(key).costPrice, sorted.get(key).sellingPrice, sorted.get(key).quantity, value));

value1 = value1 + value;

}

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

System.out.println(String.format("%-59s %s", "Total Value",(float) value1));

float profit =(float) calculateProfit();

System.out.println(String.format("%-59s %s", "Profit since previous report", profit));

Main.returnMain().returnVector().removeAllElements();

}

public double calculateProfit() {

soldobjects = Main.returnMain().returnVector();

deletedobjects = Main.returnMain().returnVectordeleted();

double profit = 0;

for(int i = 0; i < soldobjects.size();i++) {

double amount = (soldobjects.get(i).getSellingPrice() - soldobjects.get(i).getCostPrice());

amount = amount \* soldobjects.get(i).getQuantity();

profit = profit + amount;

}

for(int i = 0; i < deletedobjects.size();i++) {

double amount = deletedobjects.get(i).getCostPrice() \* deletedobjects.get(i).getQuantity();

profit = profit - amount;

}

return profit;

}

}