import java.util.Arrays;

public class MyClass {

public static void main(String args[]) {

String[] customers = { "Customer1", "Customer2", "Customer3" };

String[] factories = { "Factory1" };

String[] warehouse = { "Warehouse1" };

String[] depos = { "Depot1", "Depot2" };

float[][] customerDayWiseDemand = { { 5000, 700, 3000 }, { 3000, 3500, 2000 }, { 8000, 4000, 2000 } };

INVENTORY inventory = new INVENTORY(3500, 2000, 1500);

BOD[] bod = new BOD[10];

bod[0] = new BOD("Factory1", "Warehouse1", 2, 2000);

bod[1] = new BOD("Warehouse1", "Depot1", 5, 5000);

bod[2] = new BOD("Warehouse1", "Depot2", 7, 7000);

bod[3] = new BOD("Depot1", "Customer1", 1, 3000);

bod[4] = new BOD("Depot1", "Customer2", 2, 4000);

bod[5] = new BOD("Depot1", "Customer3", 3, 2000);

bod[6] = new BOD("Depot2", "Customer1", 2, 3000);

bod[7] = new BOD("Depot2", "Customer2", 1, 2500);

bod[8] = new BOD("Depot2", "Customer3", 4, 3000);

for (int i = 0; i < bod.length-1; i++) {

System.out.println("BOD value: " + (i+1) + " " + bod[i].getDestination() + " " + bod[i].getSource() + " " + bod[i].getLeadTime() + " " + bod[i].getFrieghtCost());

}

System.out.println("----------------All Possible Routes-----------------");

int route = 0;

ROUTES[] routes = new ROUTES[15];

int routesIndex = 0;

for (int index = 0; index < factories.length; index++) {

String element1 = factories[index];

for (int j = 0; j < warehouse.length; j++) {

String element2 = warehouse[j];

for (int h = 0; h < depos.length; h++) {

String element3 = depos[h];

for (int k = 0; k < customers.length; k++) {

String element4 = customers[k];

route++;

String actualRoute = element1 + " -> " + element2 + " -> " + element3 + " -> " + element4;

// System.out.println("Route " + route + ": " + actualRoute);

BOD customerDepoRoute = filter(element3, element4, bod);

BOD depoWareRoute = filter(element2, element3, bod);

BOD wareFactoryRoute = filter(element1, element2, bod);

int totalLeadTime = customerDepoRoute.getLeadTime() + depoWareRoute.getLeadTime() + wareFactoryRoute.getLeadTime();

float totalFrieghtCost = customerDepoRoute.getFrieghtCost() + depoWareRoute.getFrieghtCost() + wareFactoryRoute.getFrieghtCost();

// System.out.println(customerDepoRoute.getFrieghtCost() + " " + depoWareRoute.getFrieghtCost() + " " + wareFactoryRoute.getFrieghtCost());

// System.out.println("Frieght cost : " + totalFrieghtCost);

routes[routesIndex] = new ROUTES(actualRoute, element4, totalFrieghtCost, totalLeadTime);

routesIndex++;

}

}

}

}

for (int i = 0; i < routes.length-1; i++) {

if (routes[i] != null) {

System.out.println("Route: " + (i+1) + " " + routes[i].route + " " + routes[i].customer + " " + routes[i].frieghtCost + " " + routes[i].leadTime);

}

}

// Step 2: Find the optimised route for each customer

ROUTES[] optimisedRoute = new ROUTES[10];

int optRtIndex = 0;

for (int index = 0; index < customers.length; index++) {

String element = customers[index];

float min = 100000;

ROUTES optRoute = new ROUTES();

for (int i = 0; i < routes.length-1; i++) {

if (routes[i] != null) {

if (routes[i].customer == element) {

if (min > routes[i].frieghtCost) {

min = routes[i].frieghtCost;

optRoute = routes[i];

}

}

}

}

optimisedRoute[optRtIndex] = optRoute;

optRtIndex++;

}

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

for (int i = 0; i < optimisedRoute.length-1; i++) {

if (optimisedRoute[i] != null) {

System.out.println("Optimised Route: " + (i+1) + " " + optimisedRoute[i].route + " " + optimisedRoute[i].customer + " " + optimisedRoute[i].frieghtCost + " " + optimisedRoute[i].leadTime);

}

}

// Step 3: Inventory Calculation

for (int i= 0; i < customerDayWiseDemand.length; i++) {

float totalDayDemand = 0;

for (int j= 0; j < customerDayWiseDemand.length; j++) {

totalDayDemand = totalDayDemand + customerDayWiseDemand[j][i];

}

System.out.println("Day: " + (i+1) + " demand " + totalDayDemand);

float totalInhand = inventory.warehouse + inventory.depot1;

float factoryReq = totalDayDemand - totalInhand;

System.out.println("Request for production to factory at Day " + (i+1) + " is " + factoryReq);

}

}

public static BOD filter(String source, String destination, BOD[] bod) {

BOD temp = new BOD();

for (int i = 0; i < bod.length-1; i++) {

if (bod[i].getDestination().equals(destination) && bod[i].getSource().equals(source)) {

temp = bod[i];

return temp;

}

}

return temp;

}

}

class BOD {

private String source;

private String destination;

private int lead\_time;

private float frieght\_cost;

public BOD () {

}

public BOD (String newSource, String newDest, int newLeadTime, float newFrieghtCost) {

source = newSource;

destination = newDest;

lead\_time = newLeadTime;

frieght\_cost = newFrieghtCost;

}

public String getSource() {

return source;

}

public String getDestination() {

return destination;

}

public int getLeadTime() {

return lead\_time;

}

public float getFrieghtCost() {

return frieght\_cost;

}

public void setDestination(String newDest) {

destination = newDest;

}

public void setSource(String newSource) {

source = newSource;

}

public void setLeadTime(int newLeadTime) {

lead\_time = newLeadTime;

}

public void setFrieghtCost(float newFrieghtCost) {

frieght\_cost = newFrieghtCost;

}

}

class ROUTES {

String route;

String customer;

float frieghtCost;

int leadTime;

public ROUTES() {

}

public ROUTES(String newRoute, String newCust, float newCost, int newTime) {

route = newRoute;

customer = newCust;

frieghtCost = newCost;

leadTime = newTime;

}

}

class INVENTORY {

float warehouse;

float depot1;

float depot2;

public INVENTORY(float newWare, float newDepo1, float newDepo2) {

warehouse = newWare;

depot1 = newDepo1;

depot2 = newDepo2;

}

}