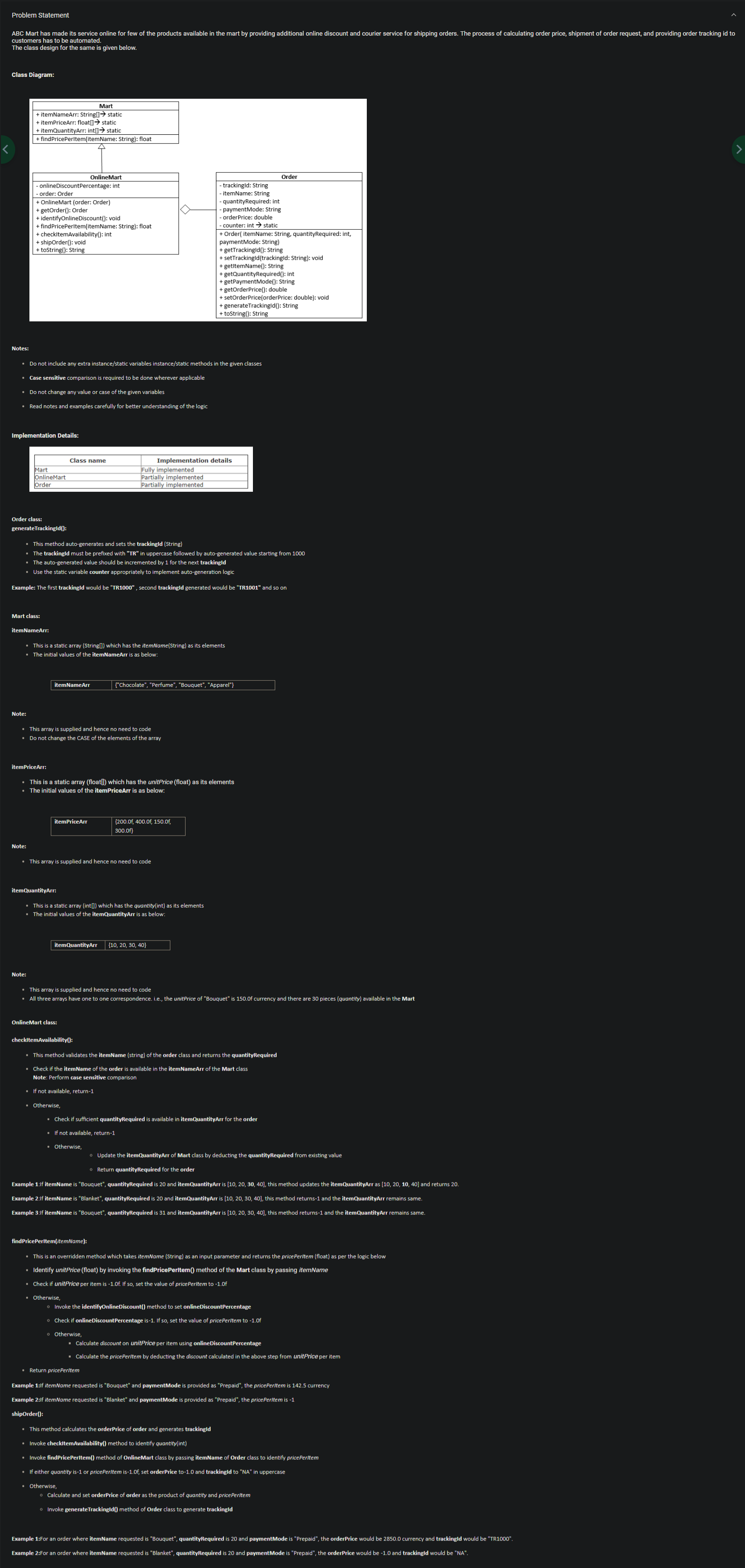
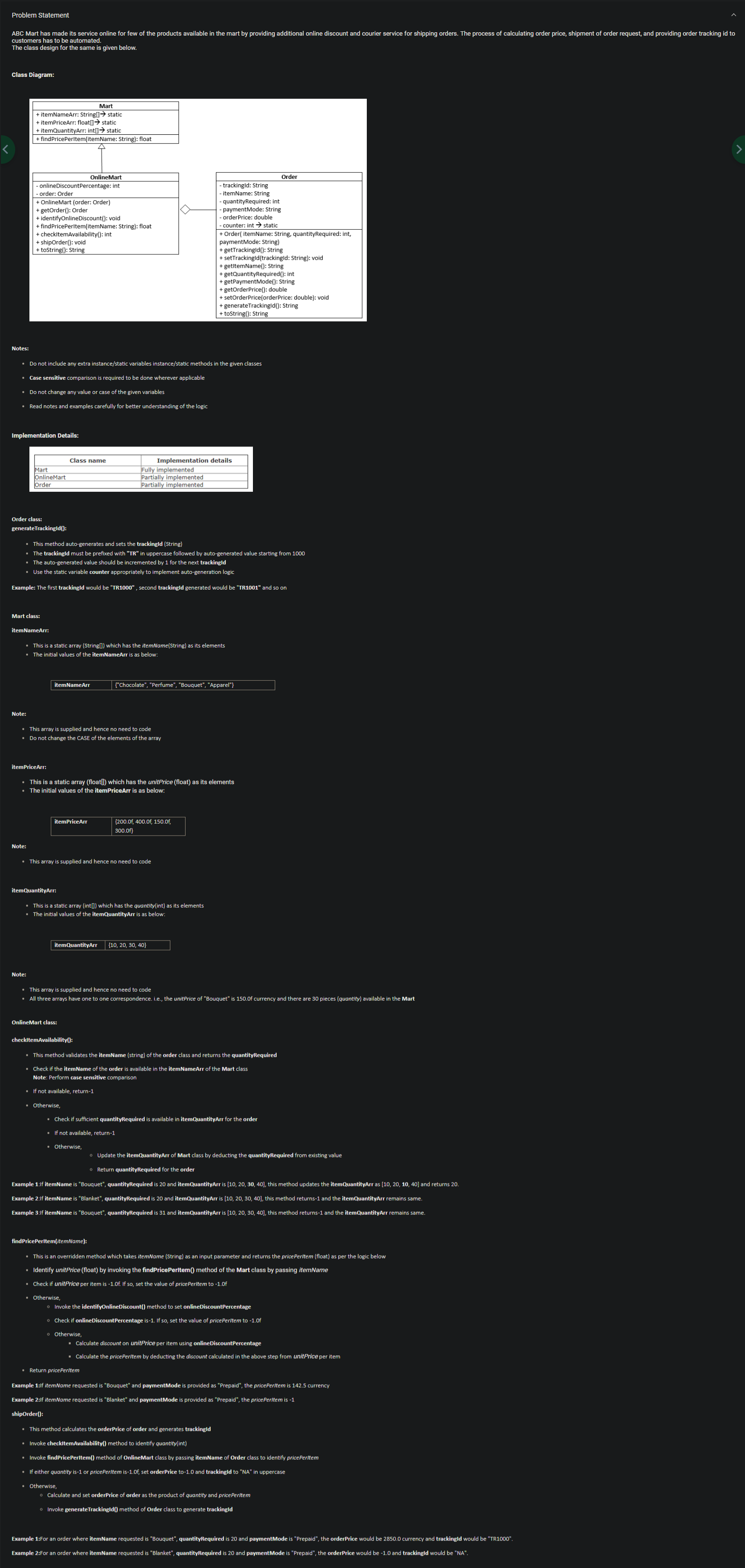


class Player {  
 private int noOfMatches;  
 private int pointsEarned;  
  
 public Player(int noOfMatches){  
 this.noOfMatches=noOfMatches;  
 }  
  
 public int getPointsEarned(){  
 return this.pointsEarned;  
 }  
  
 public void setPointsEarned(int pointsEarned){  
 this.pointsEarned = pointsEarned;  
 }  
  
 public int getNoOfMatches(){  
 return this.noOfMatches;  
 }  
  
 //To\_Trainees  
 public Boolean validateNoOfMatches(){  
 //write your logic here  
 boolean x= false;  
 if (getNoOfMatches()>0 && getNoOfMatches()<=100)  
 {  
 x= true;  
 }  
 else x= false;  
 //change return statement accordingly  
 return x;  
 }  
  
 @Override  
 public String toString() {  
 return "Player (noOfMatches=" + this.noOfMatches + ")";  
 }  
}  
  
class Batsman extends Player {  
 private int runsScored;  
 private int centuryCount;  
 private int batsmanRank;  
  
 public Batsman(int noOfMatches,int runsScored,int centuryCount){  
 super(noOfMatches);  
 this.runsScored = runsScored;  
 this.centuryCount = centuryCount;  
 }  
 public int getRunsScored(){  
 return this.runsScored;  
 }  
  
 public int getCenturyCount(){  
 return this.centuryCount;  
 }  
  
 public int getBatsmanRank(){  
 return this.batsmanRank;  
 }  
  
 @Override  
 public String toString() {  
 return "Batsman (Player (noOfMatches=" + this.getNoOfMatches() + ")+runsScored=" + this.runsScored + ", centuryCount="  
 + this.centuryCount + ")";  
 }  
  
 //To\_Trainee  
 public Boolean validateBatsmanRecord(){  
 //write your logic here  
 if(centuryCount>=0 && centuryCount<=getNoOfMatches())  
 {  
 int totalRun = centuryCount\*100;  
 if(totalRun <=this.runsScored) {  
 return true;  
 }  
 }  
  
 //change return statement accordingly  
 return false;  
 }  
  
 //To\_Trainees  
 public void calculatePoints(int index){  
  
 //write your logic here  
 if(!super.validateNoOfMatches() || !validateBatsmanRecord() )  
 {  
 setPointsEarned(-1);  
 batsmanRank=-1;  
 }  
 else {  
 setPointsEarned((runsScored\*2)+(centuryCount\*25));  
 int rank = PlayerDetails.*rankPlayer*(getPointsEarned(), index);  
 batsmanRank = rank;  
 }  
  
 }  
}  
  
class PlayerDetails {  
 static int [] *playersPointsArr*={1000,934,800,550};  
  
 public static void swap(int[] numbers, int firstIndex, int secondIndex) {  
 int temp = numbers[firstIndex];  
 numbers[firstIndex] = numbers[secondIndex];  
 numbers[secondIndex] = temp;  
 }  
  
 public static int[] sort(int[] pointsArr){  
 for(int index2=0;index2<PlayerDetails.*playersPointsArr*.length;index2++) {  
 boolean swapped = false;  
 for(int index3=0;index3<(PlayerDetails.*playersPointsArr*.length- index2 - 1);index3++) {  
 if (PlayerDetails.*playersPointsArr*[index3] < PlayerDetails.*playersPointsArr*[index3 + 1]) {  
 *swap*(PlayerDetails.*playersPointsArr*, index3, index3 + 1);  
 swapped = true;  
 }  
 }  
 if (swapped == false)  
 break;  
 }  
 return pointsArr;  
 }  
  
 //To\_Trainees  
 public static Integer rankPlayer(int pointsEarned, int index){  
 //write your code here  
 *playersPointsArr*[index]= pointsEarned; //update poitsEarened  
 //change return statement accordingly  
 *sort*(*playersPointsArr*);// sort playerPointsArr  
 for(int i=0;i<*playersPointsArr*.length;i++) {  
  
 if(*playersPointsArr*[i]==pointsEarned) {  
 int playerRank = i+1;  
 return playerRank;  
 }  
 }  
  
 return null;  
 }  
  
}  
  
class Tester {  
 public static void main(String[] args){  
 Batsman obj1 = new Batsman(10, 420, 3);  
 obj1.calculatePoints(0);  
  
 System.*out*.println("Batsman Points:" + obj1.getPointsEarned());  
 System.*out*.println("Batsman Rank:"+ obj1.getBatsmanRank());  
 System.*out*.println("Player Points Array:");  
 for(int index=0; index < PlayerDetails.*playersPointsArr*.length;index++){  
 System.*out*.print(PlayerDetails.*playersPointsArr*[index]+" ");  
 }  
  
 }  
}

—————————————————————————————————————————————————



class Mart {  
 public static String [] *itemNameArr*= {"Chocolate","Perfume","Bouquet","Apparel"};  
 public static float [] *itemPriceArr*= {200.0f,400.0f,150.0f,300.0f};  
 public static int [] *itemQuantityArr*= {10,20,30,40};  
  
 public float findPricePerItem(String itemName) {  
 float priceItem = -1.0f;  
 for(int index=0; index<Mart.*itemNameArr*.length;index++) {  
 if(itemName.equals(Mart.*itemNameArr*[index])) {  
 priceItem=Mart.*itemPriceArr*[index];  
 }  
 }  
 return priceItem;  
 }  
}  
  
class OnlineMart extends Mart{  
 private int onlineDiscountPercentage;  
 private Order order;  
  
 public OnlineMart(Order order){  
 this.order=order;  
 }  
  
 public Order getOrder(){  
 return this.order;  
 }  
  
 public void identifyOnlineDiscount() {  
 if (this.order.getPaymentMode().equals("Prepaid")){  
 this.onlineDiscountPercentage=5;  
 }else if (this.order.getPaymentMode().equals("COD")) {  
 this.onlineDiscountPercentage=2;  
 }  
 else {  
 this.onlineDiscountPercentage=-1;  
 }  
 }  
  
 //To\_Trainee  
 @Override  
 public float findPricePerItem(String itemName) {  
 float pricePerItem = 0.0f;  
  
 //write your code here  
 float unitPrice = super.findPricePerItem(itemName);  
 if(unitPrice==-1.0f)  
 {  
 pricePerItem=-1.0f;  
 }  
 else  
 {  
 identifyOnlineDiscount();  
 if(onlineDiscountPercentage==-1)  
 {  
 pricePerItem=-1.0f;  
 }  
 else  
 {  
 float discount = (unitPrice\*onlineDiscountPercentage)/100;  
 pricePerItem=unitPrice-discount;  
 }  
 }  
 return pricePerItem;  
 }  
  
 @Override  
 public String toString() {  
 return "OnlineMart (Order ( itemName=" + this.order.getItemName()  
 + ", quantityRequired=" + this.order.getQuantityRequired() + ", paymentMode="  
 + this.order.getPaymentMode() + "))";  
 }  
  
 //To\_Trainee  
 public int checkItemAvailability() {  
 //write your code here  
 String itemName = order.getItemName();  
 int quantityRequired = order.getQuantityRequired();  
 for(int i=0;i<Mart.*itemNameArr*.length;i++)  
 {  
 if(itemName.equals(Mart.*itemNameArr*[i]))  
 {  
 if(quantityRequired<=*itemQuantityArr*[i])  
 {  
 Mart.*itemQuantityArr*[i] -= quantityRequired;  
 return quantityRequired;  
 }  
 }  
 }  
 //change return statement accordingly  
 return -1;  
 }  
  
 //To\_Trainee  
 public void shipOrder() {  
  
 //write your code here  
 int quantity = checkItemAvailability();  
 float pricePerItem = findPricePerItem(order.getItemName());  
 if(quantity==-1 || findPricePerItem(order.getItemName())==-1.0f)  
 {  
 order.setOrderPrice(-1.0);  
 order.setTrackingId("NA");  
 }  
 else  
 {  
 order.setOrderPrice(quantity\*pricePerItem);  
 order.generateTrackingId();  
 }  
  
 }  
}  
  
class Order{  
 private static int *counter*=1000;  
 private String trackingId;  
 private String itemName;  
 private int quantityRequired;  
 private String paymentMode;  
 private double orderPrice;  
  
 public Order(String itemName, int quantityRequired, String paymentMode){  
 this.itemName = itemName;  
 this.quantityRequired = quantityRequired;  
 this.paymentMode = paymentMode;  
 }  
  
 public void setTrackingId(String trackingId) {  
 this.trackingId = trackingId;  
 }  
  
 public String getTrackingId() {  
 return this.trackingId;  
 }  
 public String getItemName(){  
 return this.itemName;  
 }  
 public int getQuantityRequired(){  
 return this.quantityRequired;  
 }  
  
 @Override  
 public String toString() {  
 return "Order (trackingId=" + this.trackingId + ", itemName=" + this.itemName  
 + ", quantityRequired=" + this.quantityRequired + ", paymentMode="  
 + this.paymentMode + ", orderPrice=" + this.orderPrice + ")";  
 }  
  
 public String getPaymentMode(){  
 return this.paymentMode;  
 }  
  
 public double getOrderPrice(){  
 return this.orderPrice;  
 }  
 public void setOrderPrice(double orderPrice){  
 this.orderPrice=orderPrice;  
 }  
  
 //To\_Trainee  
 public void generateTrackingId(){  
 //write your code here  
 setTrackingId("TR"+*counter*++);  
 }  
}  
  
class Tester {  
 public static void main(String[] args) {  
 Order orderObj = new Order("Bouquet", 20, "Prepaid");  
 OnlineMart onlineMartObject = new OnlineMart(orderObj);  
 onlineMartObject.shipOrder();  
 System.*out*.println("Tracking ID :" + onlineMartObject.getOrder().getTrackingId());  
 System.*out*.println("Order Price :" + onlineMartObject.getOrder().getOrderPrice());  
 }  
}