COMP503/ENSE502/ENSE602 _ Assignment 3
Sem 2 2019

Product Recommender Algorithm

Due date: Friday, 11th October 2019 at 5:00pm

This assignment has 100 marks and is worth 10% of your final grade.

Brief

For this assignment, you will individually develop an Online Shop in Java that implements a Product Recommender Algorithm (PRA) that recommends related products based on your purchases. The PRA works by analysing transaction histories of Online Shop customers, observing the frequency products are purchased together. Products with high purchase frequencies with your current purchases are recommended to you.

Example

A three-digit integer number represents each product offered by the Online Shop. For example, the numbers 123, 187, 199, 200, 865 represent unique types of products sold by the Online Shop (Note: each number represents a product code!). A customer's purchase history is therefore a list of numbers. Here are the purchase histories from four different customers:

1. [187, 199, 200]
2. [187, 199, 123, 865]
3. [199, 123, 865]
4. [123, 199, 865, 187]

Suppose you purchase product 187. The PRA analyses each customer's transaction histories. If 187 is found in the history then it keeps track of the other products that customer purchased, creating an observation map from products to frequencies:

123 -> 2 (e.g. 123 was purchased with 187 by customers 2. and 4.)
199 -> 3 (e.g. 199 was purchased with 187 by customers 1., 2. and 3.)
200 -> 1 (e.g. 200 was purchased with 187 by customers 1.)
865 -> 2 (e.g. 185 was purchased with 187 by customers 2. and 4.)

The PRA will then return products 123, 865 and 199 as recommendations since the observation frequency is greater than/equal to 2. For larger product databases and more customers, we might have recommendations based on higher frequencies.

COMP503/ENSE502/ENSE602 _ Assignment 3
Sem 2 2019

Methodology and Marking Scheme

You will develop Java classes that implement the Online Shop. For full marks, your classes must adhere to correct OOP design principles. All source code must be documented according to Java doc standards.

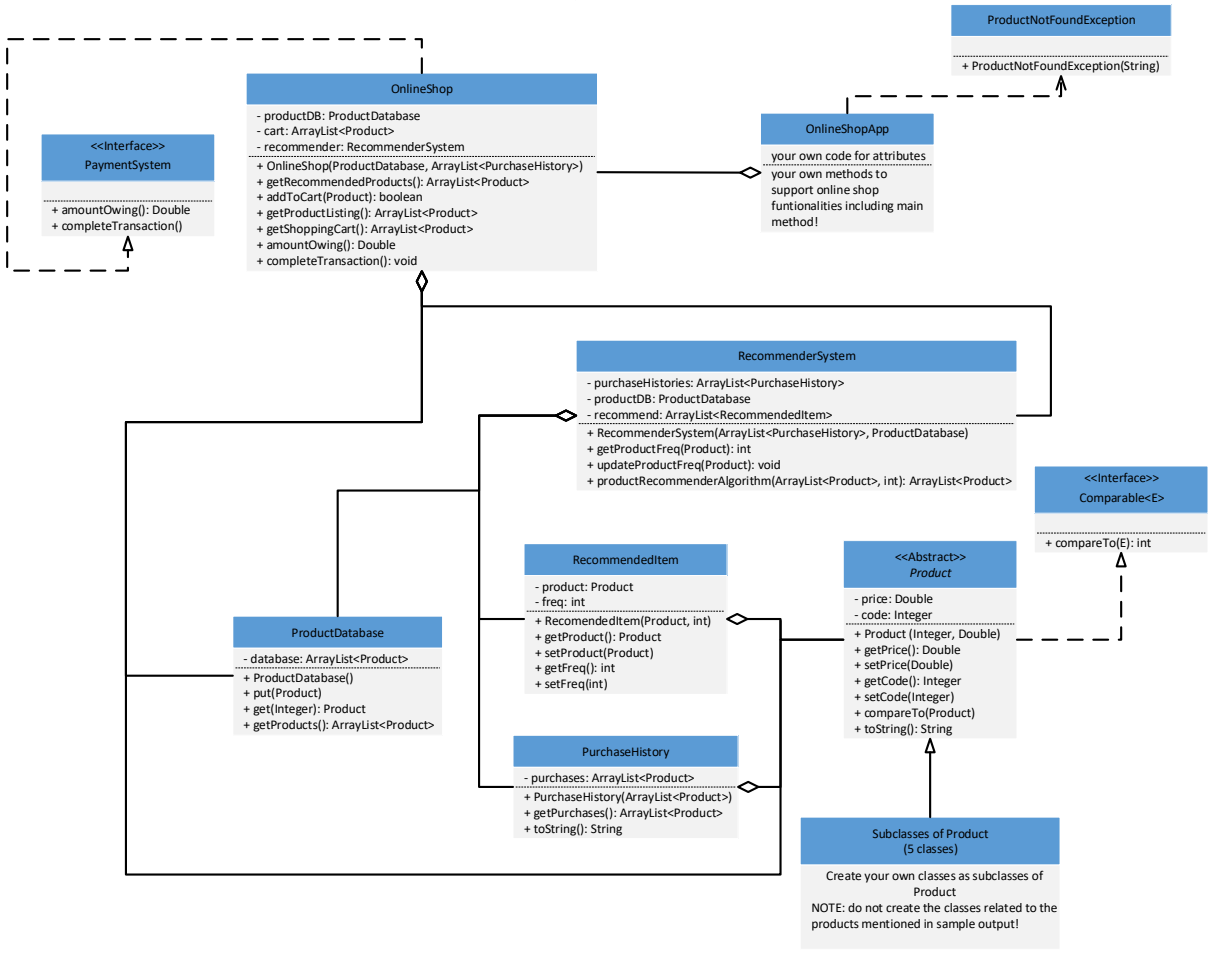
You have been provided with an outline of 13 classes which contain the core functionalities of the Online Shop: OnlineShop, OnlineShopApp, ProductNotFoundException, RecommenderSystem, Product, RecommendedItem, ProductDatabase, PurchaseHistory and 5 concrete classes as subclasses of Product class (these classes should be your own classes). The relationships between classes are described below using an UML Class Diagram (Figure 1). Complete the assignment by using the given UML class diagram and following the below steps to develop the 13 classes with the provided instance variables and methods using appropriate access modifiers (as shown in the diagram), taking the time to test your methods as you progress.

Note1: You need to provide attributes and methods that are listed in UML diagram even those which are not defined in the step sections. Some of them are clear and that's why did not mention in steps. (Please follow method signatures in UML diagram)

Note2: You can develop your own methods if you think it is required to have more modular and OO code!

Note3: The OnlineShopApp class in UML has not specific attribute and method as you need to write your own code and define several methods to make good online shop application. Try to make modular code. Rather having a long method doing lots of functionalities, try to have more methods that each can do specific function.

Figure 2: UML Diagram of Online Shop Application with Recommender System



COMP503/ENSE502/ENSE602 _ Assignment 3
Sem 2 2019

Modeling products

Design an abstract **Product** class that:

- stores a product's code as an Integer and its price.
- implements the **Comparable** interface

Extend the Product class by *at least five* concrete classes. Each subclass must have

- two or more attributes,
- a constructor with input parameters for all attributes, including inherited attributes
- a toString method

The product database and purchase history classes

Design **ProductDatabase** class. Products are stored in this class, which comprises:

- an ArrayList collection, storing Product objects.
- A constructor initializes the ArrayList of Products.
- A **put** method that provided the functionality for adding products to the database.
- A **get** method that returns a product from the database based on product code received as input parameter.
- A **getProducts** method that returns the ***unique (not repeated)*** products stored in the database as an ArrayList.

Note: You do not need to remove products from the database.

Design **PurchaseHistory** class. This class models the purchase history of a customer. It contains an ArrayList of product objects, supplied by the constructor. The class has a single get method for the ArrayList but no set or other updating methods.

Payment System Functionality

The **PaymentSystem** is modelled as an **interface**, comprising two methods:

1. `public Double amountOwing();`
2. `public void completeTransaction();`

The payment system is implemented by the Online Shop, which provides method bodies to compute the amount the customer owes. The complete transaction method simply clears the shopping cart.

The RecommendedItem

Design a **RecommendedItem** class that:

- stores a product's object and its frequencies in a purchase histories of an online shop (i.e. frequencies mean number of times an object purchased by customers in the online shop).

COMP503/ENSE502/ENSE602 _ Assignment 3
Sem 2 2019
The Recommender System

The **RecommenderSystem** class has

- instance variables for the product database
- an ArrayList containing customer purchases history objects.
- The constructor initializes both instance variables.
- a single method with the signature

  ```
  public ArrayList<Product> prAlgorithm(ArrayList<Product> cart, int freq)
  ```

  with the customer's shopping cart and a frequency as input, returning a list of recommended products (Hint: use objects of RecommendedItem. You may need to create ArrayList of RecommendedItem objects and make that ArrayList as an attribute of class to be accessible via the other methods in the class). Consider the following input:

Method Invocation	Output
praAlgorithm([187],2)	[123,199,865]
praAlgorithm([187],3)	[199]
praAlgorithm([200],2)	[]
praAlgorithm([200],1)	[199,187]
praAlgorithm([123],2)	[199,865,187]
praAlgorithm([123],3)	[199,865]

Now look at the example and try to calculate the output yourself. Use this process to devise an algorithm to compute the recommendations.

Online Shop

The **OnlineShop** is modelled by a Java class that

- contains a **ProductDatabase** and a **RecommenderSystem**.
- maintains a shopping cart that contains products the customer has selected to purchase.
- implements a **PaymentSystem**
  - to compute the amount owing and,
  - completing the customer's transaction (which also clears the cart).

Online Shop Application (Program Interaction)

The Online Shop Application provides interaction between the customer and an Online Shop object. It provides menu interactions for

1) Adding a product to the cart, which lists all products in the database, from which the customer can select from
2) Viewing the products in the shopping cart

3) Finalize purchases, which uses a payment system to compute the total amount owing and to process the transaction. When the payment is completed, the customer is given product suggestions based on what they have purchased.

*All product lists displayed in 1) to 4) **must be sorted by price**, in ascending order.*

How to get started

The first step to getting started on the assignment is creating the **Product** abstract class and deciding how the class should be extended using at least 5 subclasses. Think of interesting examples of products to sell. **They must be different from the ones shown in the console sample provided; otherwise you will get zero marks for that portion of the assignment.** The point here is to think about developing your own classes.

Once you have finished this, you can instantiate products with product codes 123, 187, 199, 200 and 865 (from the example) and generate a sample product database. I suggest that you write a static method in your Online Shop Application class, which initialises and returns a product database object with these five products using the following signature:

```
public static ProductDatabase generateSampleDatabase()
```

The text file **purchase-history.txt** is provided to you with this assignment. It contains the purchase histories of the four customers shown in the example.
In your Online Store Application class, write a static method which reads this text file, and has the following signature:

```
public static ArrayList<PurchaseHistory> readPurchaseHistoryData(ProductDatabase pb,String
filename) throws ProductNotFoundException,IOException,NumberFormatException
```

This method reads the text file and throws the following exceptions:

1) IOException, thrown if the file cannot be opened/read
2) NumberFormatException, thrown if integer parsing fails
3) ProductNotFoundException, your own exception, which is thrown if a product code does not appear in the database.

You also need to write a method called **writePurchaseHistoryData** to write the new purchase history from console into the **purchase-history.txt**. you need to keep the history data and add to the existing file. The method writes to the text file and throws the IOException, if the file cannot be opened/written.
Your main method should **try** to read/write the purchase history data, handling exceptions in an appropriate way (potentially by terminating execution with a meaningful message, but not by an uncaught exception).
Note: Your application should also elegantly handle exceptions occurring from bad user input.

COMP503/ENSE502/ENSE602 _ Assignment 3
Sem 2 2019

Bonus Part (additional 10 marks for assignments)

If you are interested, develop a GUI for this application. You need to have your own design for GUI functionalities. Based on your design you will get the bonus mark which will be added to your assignment mark.

Marking Scheme

Criteria:	Weight:	Grade A Range 100 ≥ x ≥ 80%	Grade B Range 80 > x ≥ 65%	Grade C Range 65 > x ≥ 50%	Grade D Range 50 > x ≥ 0%
Product and subclasses	**20%**	OOP paradigm consistently used for implementation of all product classes. Meaningful subclasses are defined.	Inconsistent use of OOP paradigm. Correct implementation of product class functionality	Inconsistent use of OOP paradigm or poor product subclasses. Correct implementation of product class functionality	Absent product classes or code does not compile
Product Database & Purchase History	**10%**	OOPS paradigm consistently used. Good use of collections in both classes, with correct functionality implementation	Inconsistent use of OOP paradigm. Correct use of collections with correct functionality implementation	Inconsistent use of OOP paradigm. Incorrect use of collections but with mostly correct functionality implementation	Absent product database or purchase history or code does not compile
Online Shop	**10%**	OOP paradigm consistently used for implementation of all Online Shop functionality.	Inconsistent use of OOP paradigm but correct implementation of Online Shop functionality	Some basic functionality of Online Shop/poor usage of collections	Absent functionality of Online Shop or code does not compile.
Recommender System	**10%**	OOP paradigm consistently used. Correct implementation of algorithm with good choice of collections.	Inconsistent use of OOP paradigm. Correct implementation of algorithm with good choice of collections.	Incorrect implementation of algorithm/poor choice of collections	Absent functionality of the recommender system or code does not compile
Online Store App Runtime Demonstration: -Uniqueness -Purpose -Context -Interactive	**20%**	The object-oriented program is unique, purposeful and provides an elegant implementation of the Online Shop Application. Program is interactive. All exceptions are handled. Purchase history text file is read Sample product database generated.	The object-oriented program is unique, purposeful. Reasonable implementation of the Online Shop Application. Program is interactive. Some errors reading text file or generating product databases.	The object-oriented program features an incomplete demonstration of the Online Shop Application Program is not interactive	Absent functionality of Online Shop Application or code does not run after compiling
Code Quality: -Whitespace -Naming -Reuse -Modularity -Encapsulation	**15%**	Whitespace is comprehensively consistent. All naming is sensible and meaningful. Code reuse is present. Code is modular. Code is well encapsulated.	Whitespace is comprehensively consistent. Majority of naming is sensible. Code is modular. Code is encapsulated.	Whitespace is comprehensively consistent. Code has some modularity. Code has some encapsulation.	Whitespace is inconsistent and hence code is difficult to read.
Documentation Standards: -Algorithms Commented -Javadoc	**15%**	Entire codebase has comprehensive Javadoc commenting. Algorithms are well commented.	Majority of the codebase features Javadoc commenting. Majority of algorithms are commented.	Some Javadoc comments present. Some algorithms are commented.	No Javadoc comments present.

COMP503/ENSE502/ENSE602 _ Assignment 3
Sem 2 2019

Javadoc Commenting

1. Your classes must have commenting of the form:

```
/**
 * Comment describing the class.
 * @author yourname studentnumber
 **/
```

2. All methods must be commented with appropriate Javadocs metatags. For example:

```
/**
      * A comment to describe the method
      * @param a first parameter description
      * @param b second parameter description
      * @return a description of the returned result
      * @author Jamal studentnumber
      * */
```

Authenticity

Remember, it is unacceptable to hand in any code which has previously been submitted for assessment (for any paper, including Programming 2) or available online, and all work submitted must be unique and your own!

Submission Instructions

Submit the following documents as an archive **.zip** file of your documents on Blackboard before the deadline:

- Your full java project for assignment.
- Sample console output demonstrating your program in use (**.txt** file)

Zip structure and file naming requirements. Please ensure your submission matches the following:

- **lastname-firstname-studentid.zip**
  - **Project folder (can be find in your workspace)**
  - **studentid-console-sample.txt**

Replace the underlined text with your personal details.

An extension will only be considered with a Special Consideration Form approved by the School Registrar.

You will receive your marked assignment via Blackboard. Please look over your entire assignment to make sure that it has been marked correctly. If you have any concerns, you must raise them with the lecturer. You have **one week** to raise any concerns regarding your mark. After that time, your mark cannot be changed.

Do not go to the lecturer because you do not like your mark. Only go if you feel something has been marked incorrectly.

COMP503/ENSE502/ENSE602 _ Assignment 3
Sem 2 2019

Dr. Seyedjamal Zolhavarieh, Auckland University of Technology, szolhava@aut.ac.nz