Homework 2

2.1

a. Use Case: Delete all messages in the queue from a specific phone number without having to listen to any of them.

1. Caller carries out Log in
2. Mailbox owner selects “delete messages”
3. Mailbox owner dials the phone number.
4. Mailbox owner presses #
5. System deletes all messages from sender.

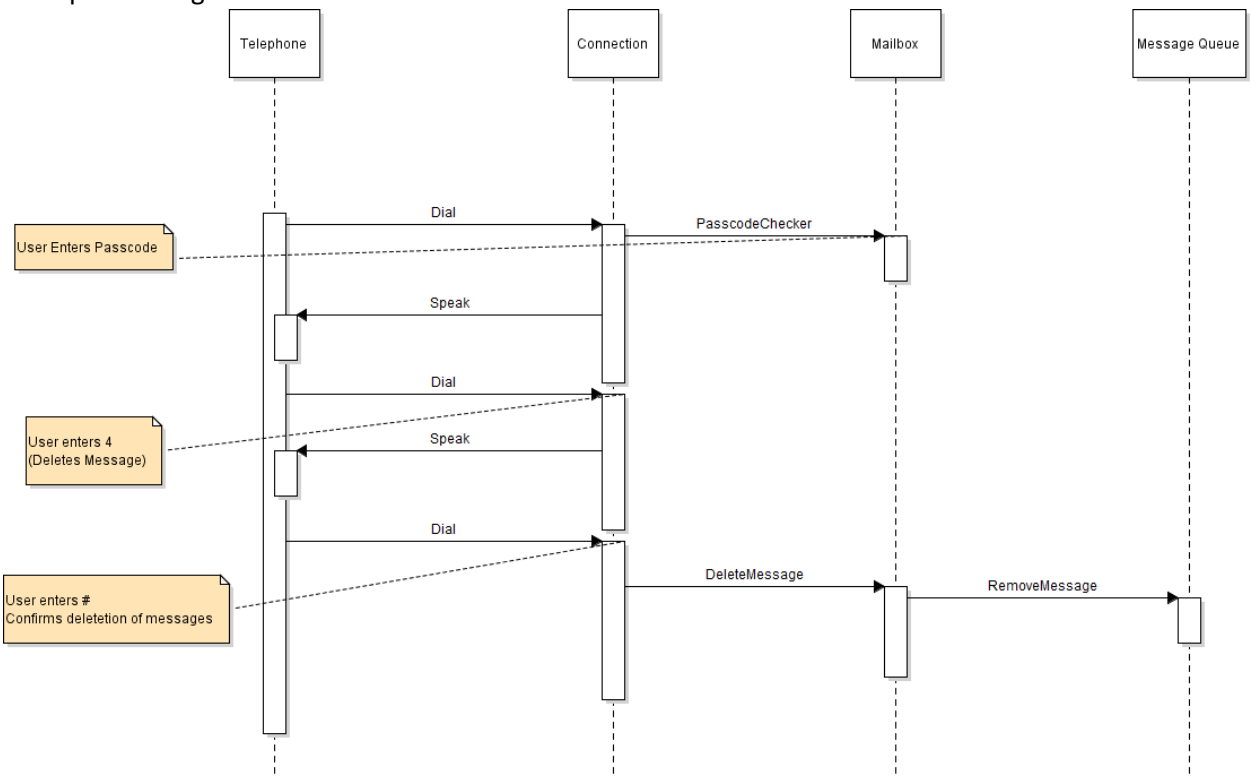
Variation #1: Hang up before confirmation

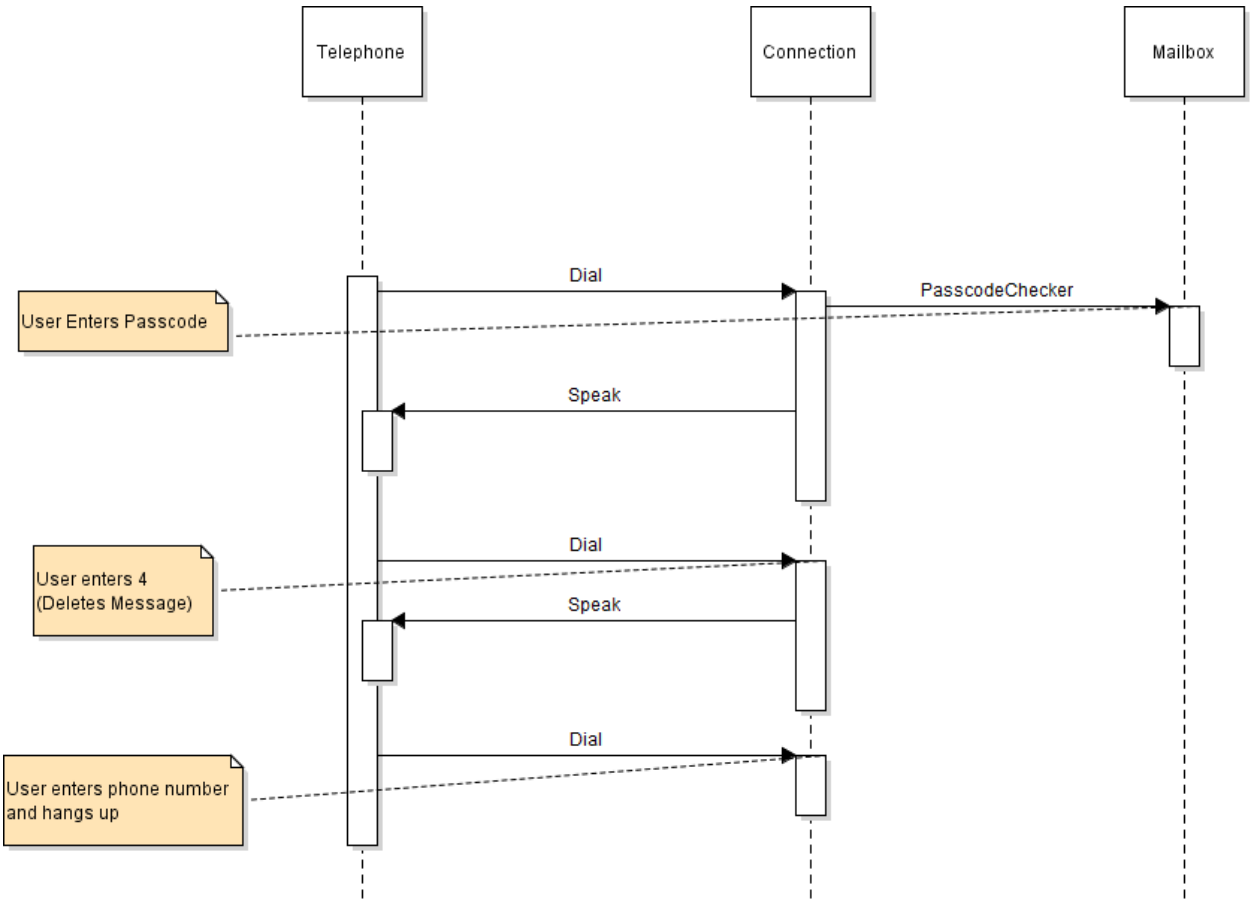
1.1 Start a number 3.

1.2 Mailbox owner hangs up.

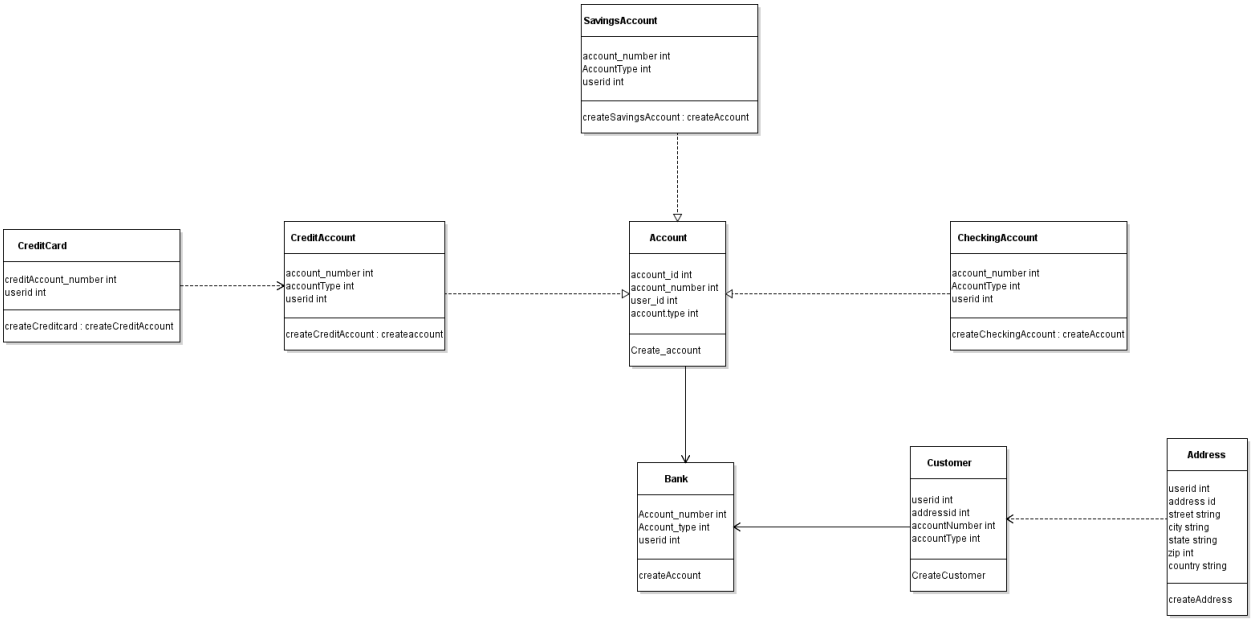
1.3 System keeps messages.

b. Sequence diagrams for the two new scenarios.




Telephone
Connection
Mailbox
Dial
PasscodeChecker
User Enters Passcode
Speak
Dial
User enters 4
(Deletes Message)
Speak
Dial
User enters phone number
and hangs up

2.2


SavingsAccount
account_number int
AccountType int
userid int
createSavingsAccount : createAccount
CreditCard
creditAccount_number int
userid int
createCreditcard : createCreditAccount
CreditAccount
account_number int
accountType int
userid int
createCreditAccount : createaccount
Account
account_id int
account_number int
user_id int
account.type int
Create_account
CheckingAccount
account_number int
AccountType int
userid int
createCheckingAccount : createAccount
Bank
Account_number int
Account_type int
userid int
createAccount
Customer
userid int
addressid int
accountNumber int
accountType int
CreateCustomer
Address
userid int
address id
street string
city string
state string
zip int
country string
createAddress

2.3

a. CRC Cards

Class: System

Responsibility:

1. Login credential authentication
2. Query rental companies' inventory

Collaborator: Inventory, Rental Company

Class: Customer

Responsibility:

1. Search for cars in the platform

Class: Inventory

Responsibility:

1. Search for available vehicles in all the rental companies

Collaborators: SystemRegistration

Class: Reservation

Responsibility:

1. Stores reservation picked by the customer

Collaborator: SystemRegistration

Class: SystemRegistration

Responsibility:

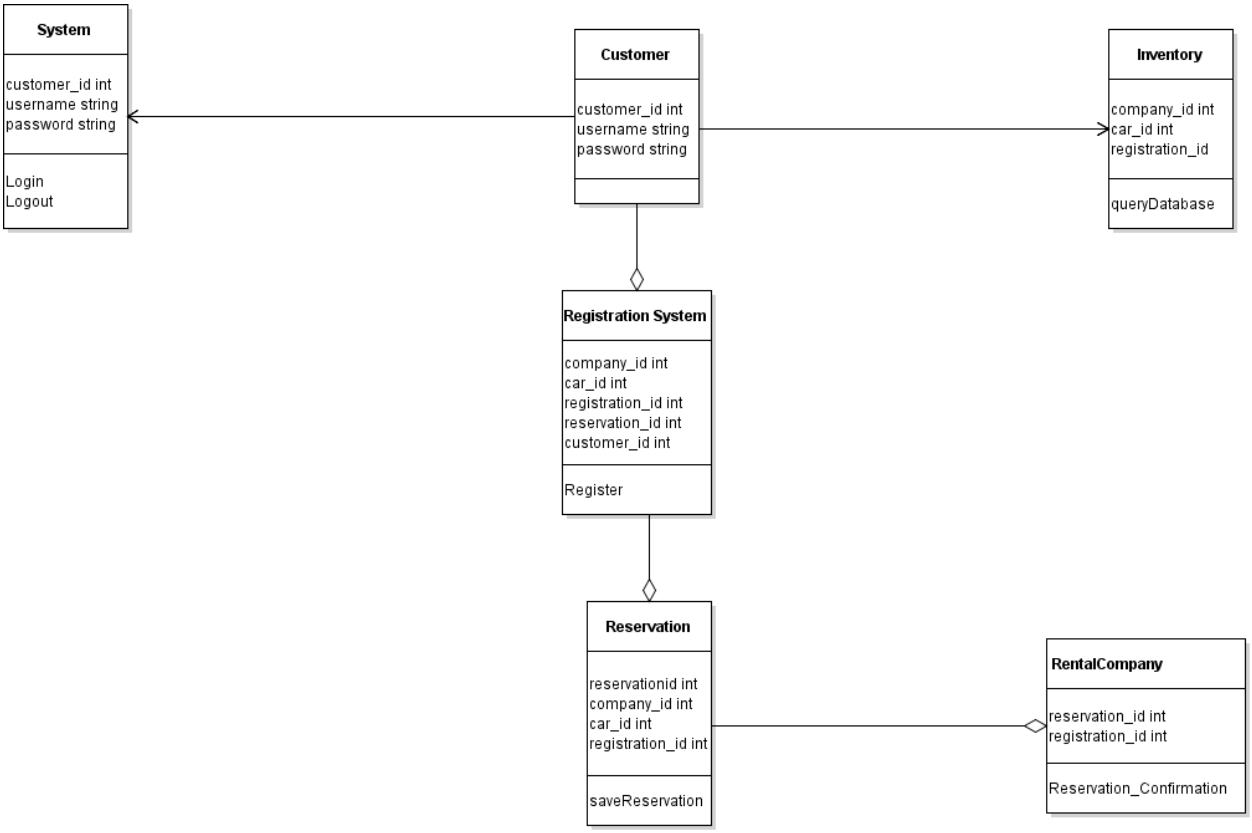
1. Manage customer registration

Class: RentalCompany

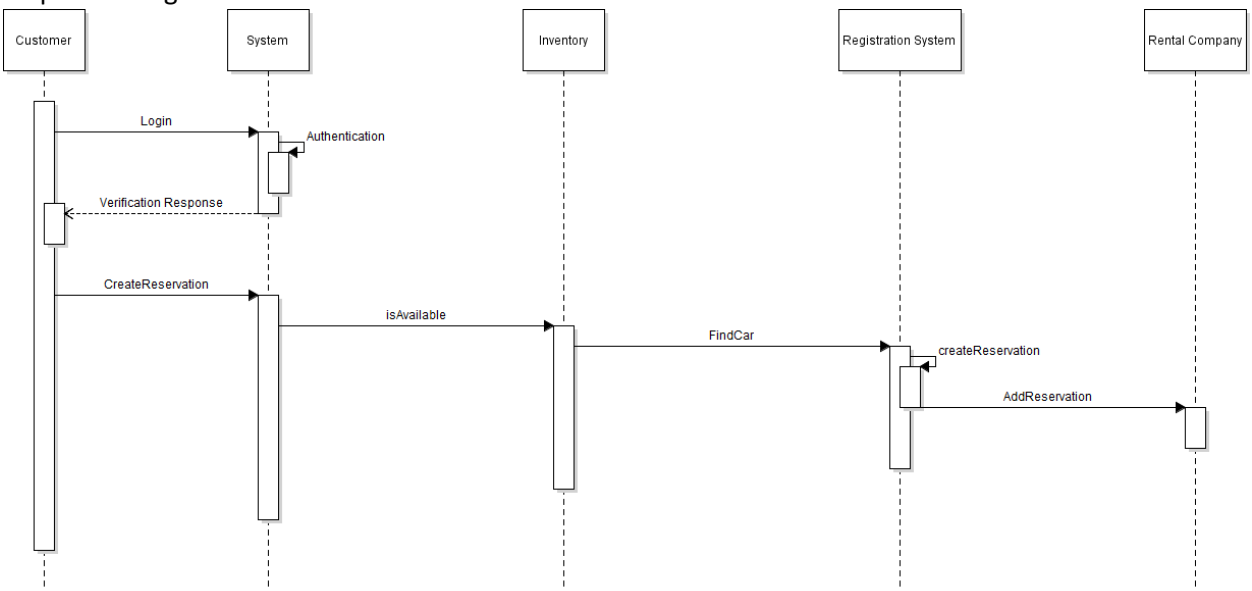
Responsibility:

1. Confirm registration

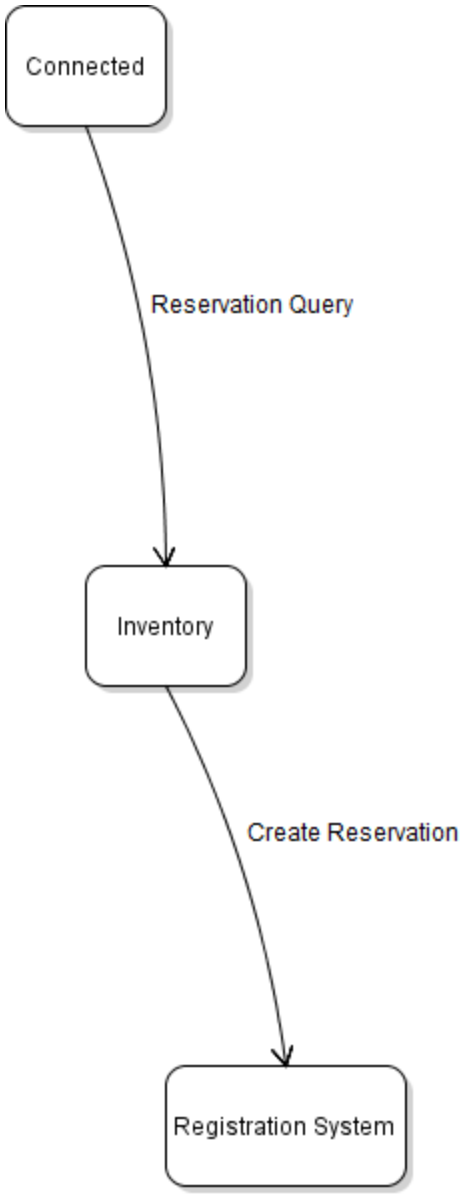
UML Class diagram



Sequence Diagram



State Diagram


Connected
Reservation Query
Inventory
Create Reservation
Registration System

2.4

CRC card

Class: Product

Responsibility:

1. Stores product information

Collaborates: Register

Class: Register

Responsibility:

1. Registers products
2. Display Summary
3. process main logic

Collaborates: Inventory

Class: Inventory

Responsibility:

1. Stores products
2. Checks if products exist
3. Gets products
4. prints products as a helper for register

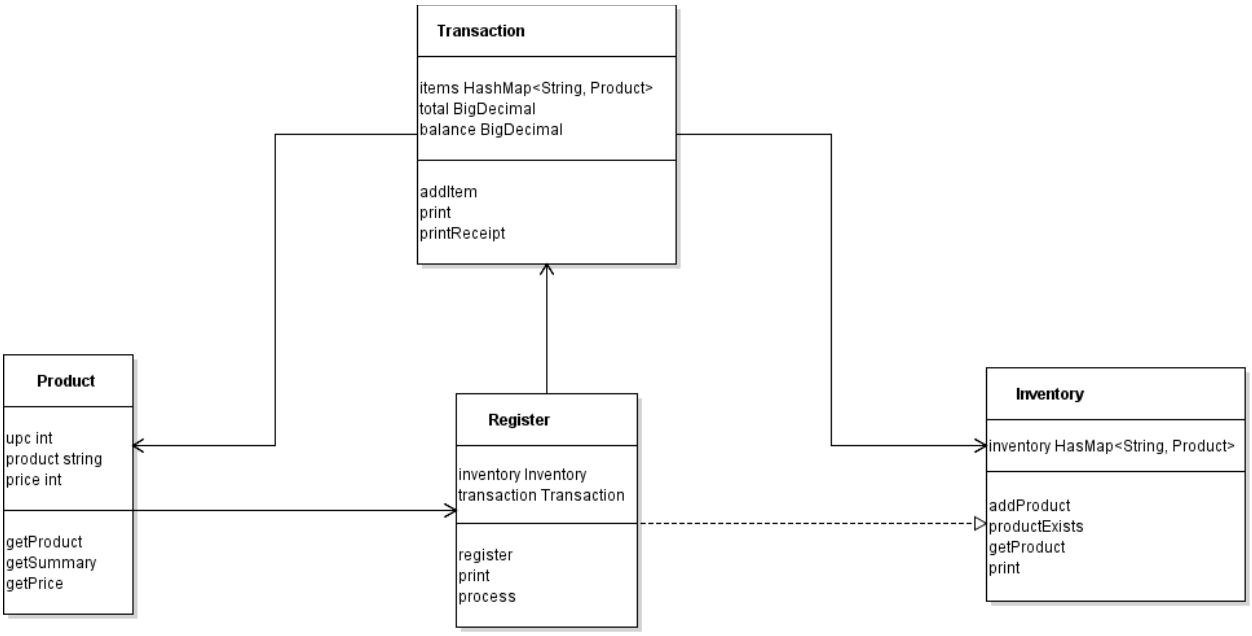
Collaborates: Register

Class: Transaction

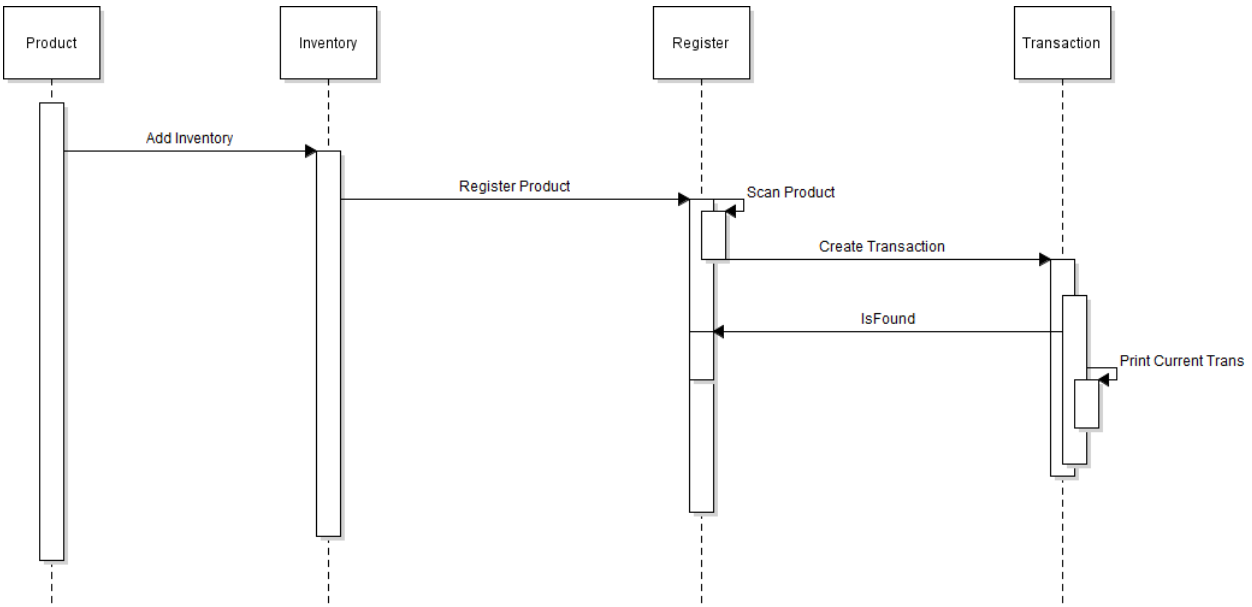
Responsibility:

1. Add already bought items
2. displays items already bought
3. prints final receipt

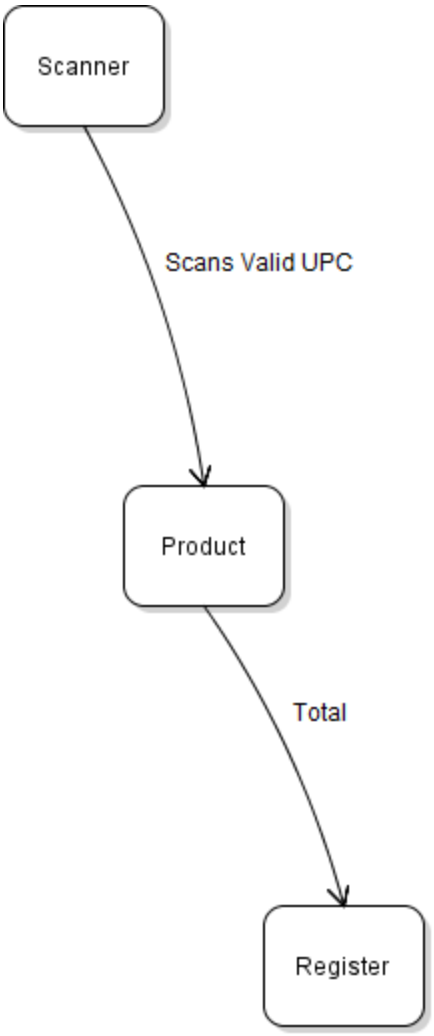
Class diagram:



Sequence Diagram



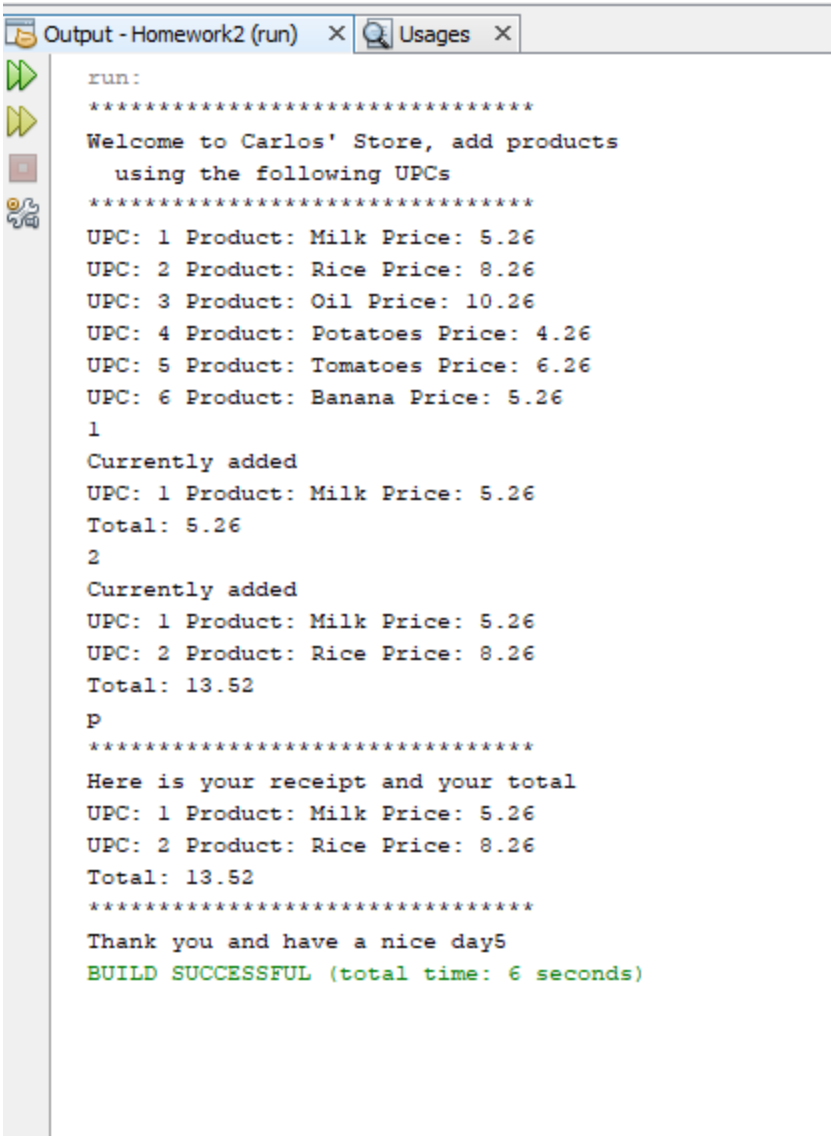
State Diagram

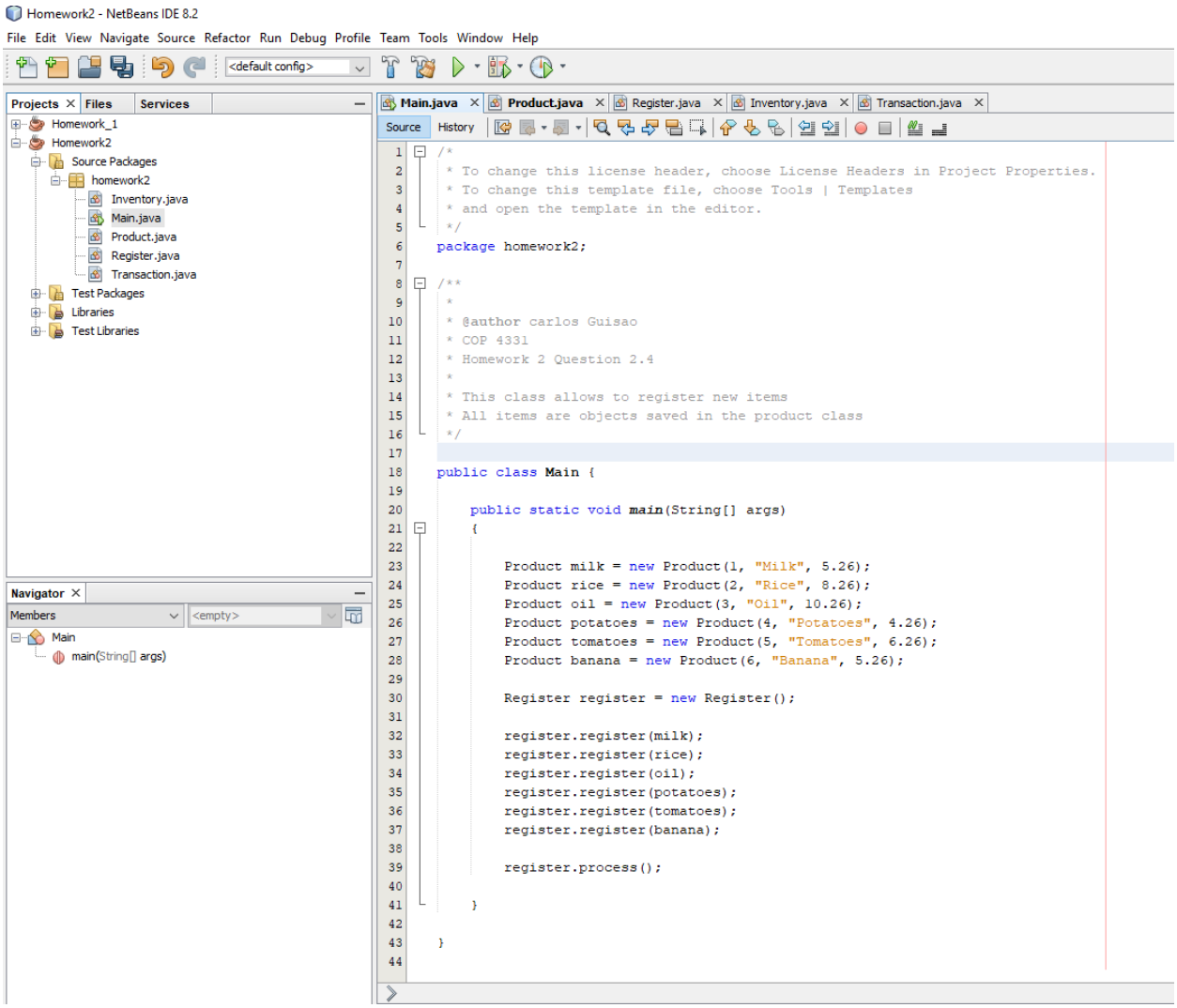


Program Code:

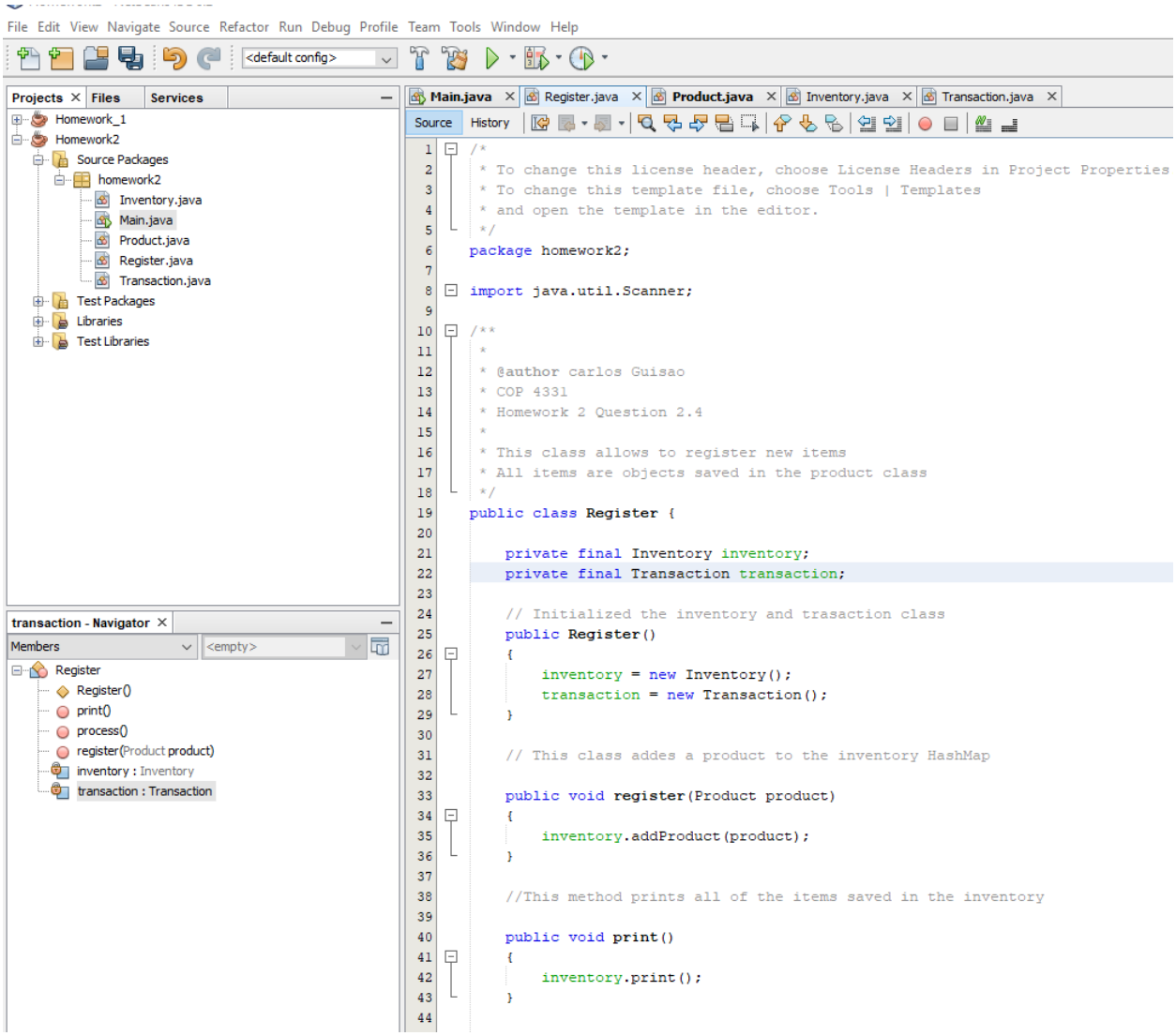
Final Output

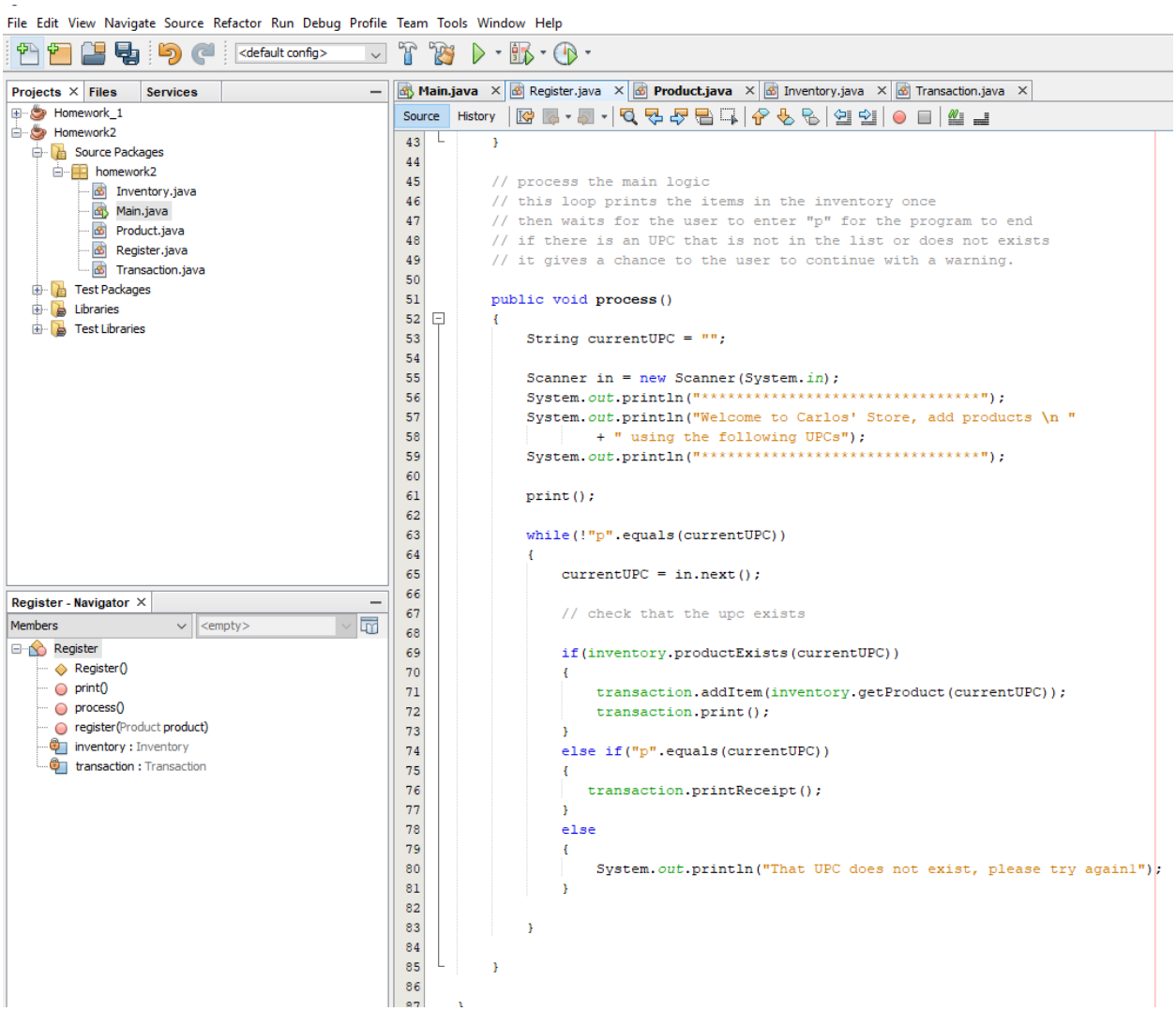
Main


Output - Homework2 (run)
Usages
run:
****
Welcome to Carlos' Store, add products
using the following UPCs
****
UPC: 1 Product: Milk Price: 5.26
UPC: 2 Product: Rice Price: 8.26
UPC: 3 Product: Oil Price: 10.26
UPC: 4 Product: Potatoes Price: 4.26
UPC: 5 Product: Tomatoes Price: 6.26
UPC: 6 Product: Banana Price: 5.26
1
Currently added
UPC: 1 Product: Milk Price: 5.26
Total: 5.26
2
Currently added
UPC: 1 Product: Milk Price: 5.26
UPC: 2 Product: Rice Price: 8.26
Total: 13.52
p
****
Here is your receipt and your total
UPC: 1 Product: Milk Price: 5.26
UPC: 2 Product: Rice Price: 8.26
Total: 13.52
****
Thank you and have a nice day5
BUILD SUCCESSFUL (total time: 6 seconds)

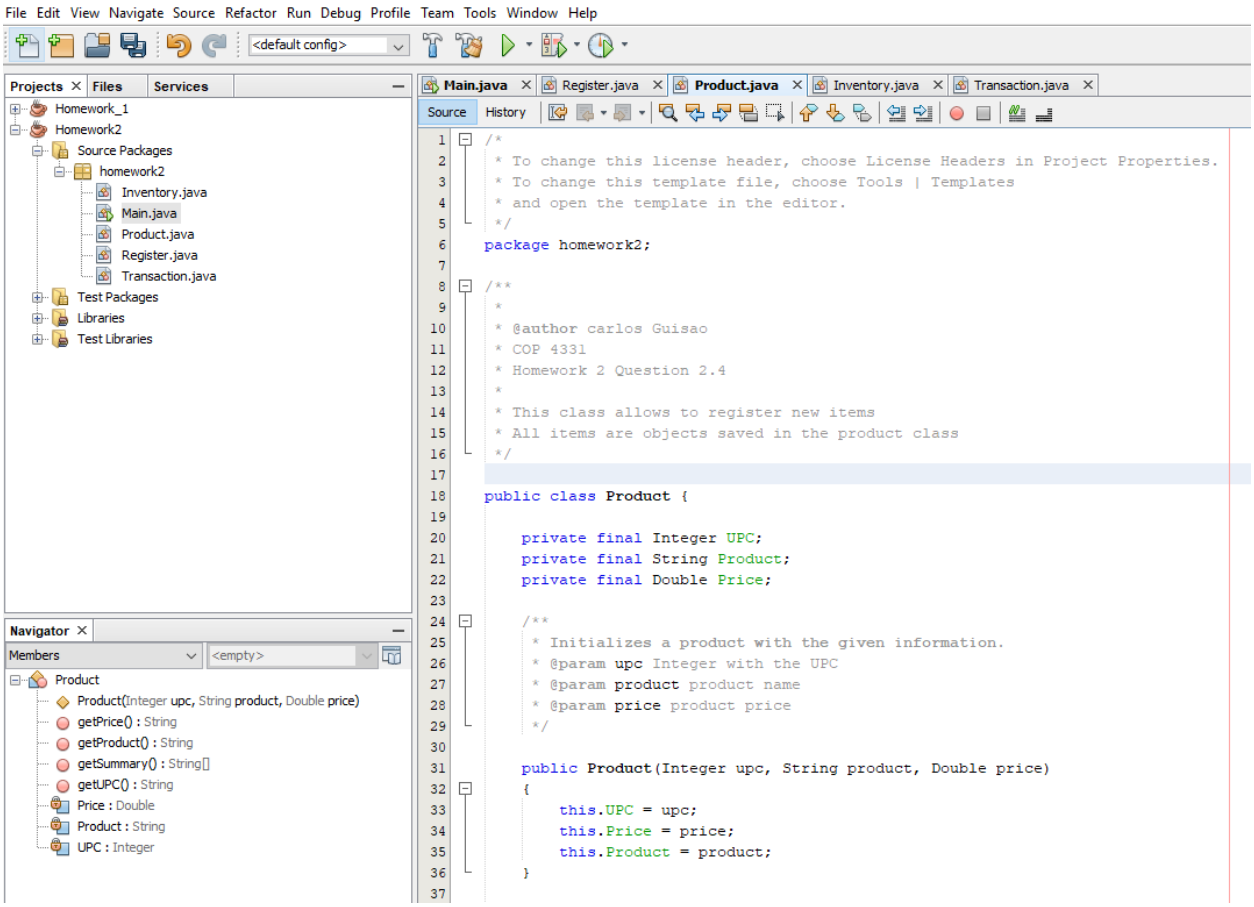


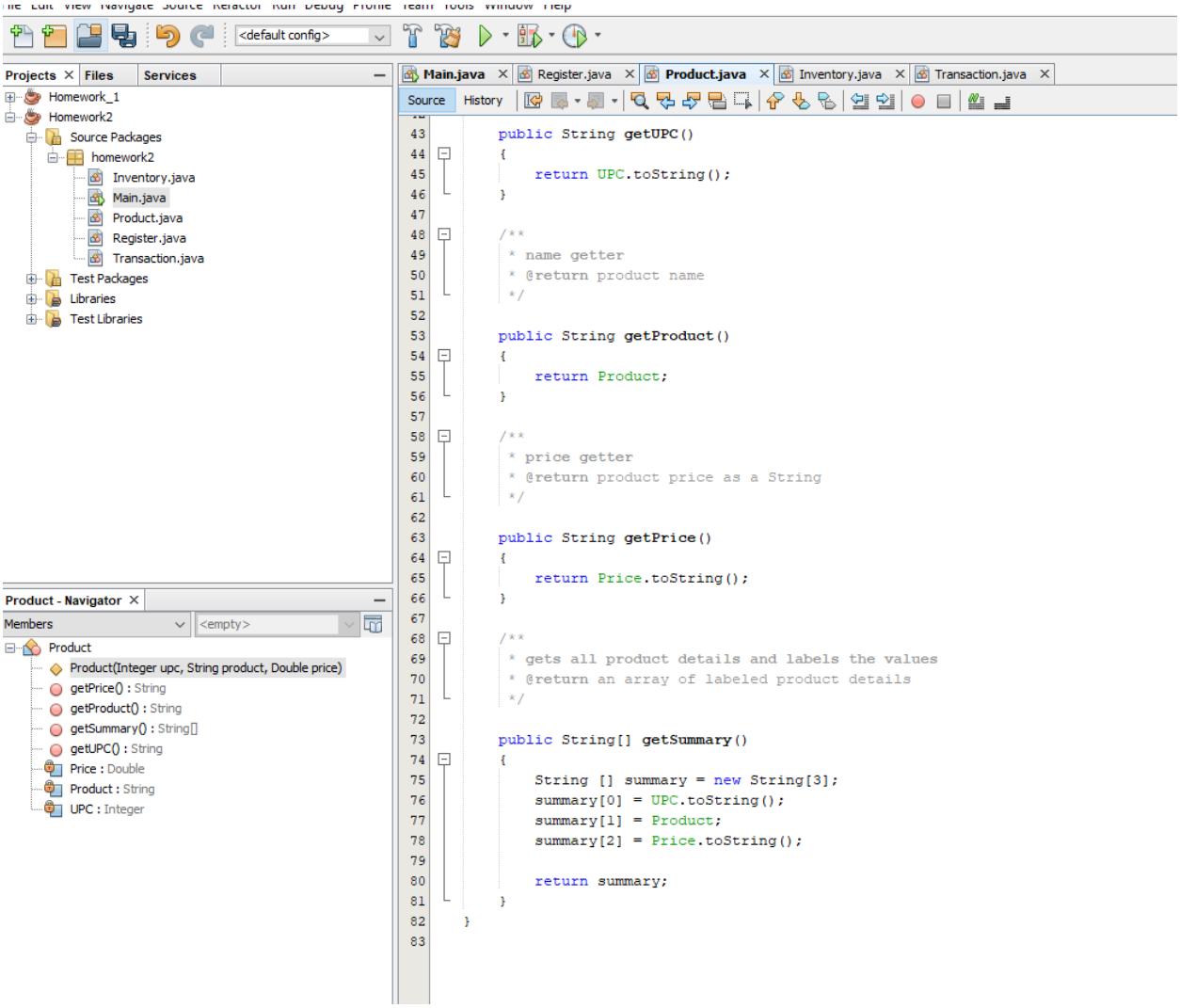
Register Class


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects
Files
Services
Homework_1
Homework2
Source Packages
homework2
Inventory.java
Main.java
Product.java
Register.java
Transaction.java
Test Packages
Libraries
Test Libraries
transaction - Navigator
Members
<empty>
Register
Register()
print()
process()
register(Product product)
inventory : Inventory
transaction : Transaction
Main.java
Register.java
Product.java
Inventory.java
Transaction.java
Source
History
1 /*
2 * To change this license header, choose License Headers in Project Properties
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6 package homework2;
7
8 import java.util.Scanner;
9
10 /**
11 *
12 * @author carlos Guisao
13 * COP 4331
14 * Homework 2 Question 2.4
15 *
16 * This class allows to register new items
17 * All items are objects saved in the product class
18 */
19 public class Register {
20
21 private final Inventory inventory;
22 private final Transaction transaction;
23
24 // Initialized the inventory and trasaction class
25 public Register()
26 {
27 inventory = new Inventory();
28 transaction = new Transaction();
29 }
30
31 // This class addes a product to the inventory HashMap
32
33 public void register(Product product)
34 {
35 inventory.addProduct(product);
36 }
37
38 //This method prints all of the items saved in the inventory
39
40 public void print()
41 {
42 inventory.print();
43 }
44

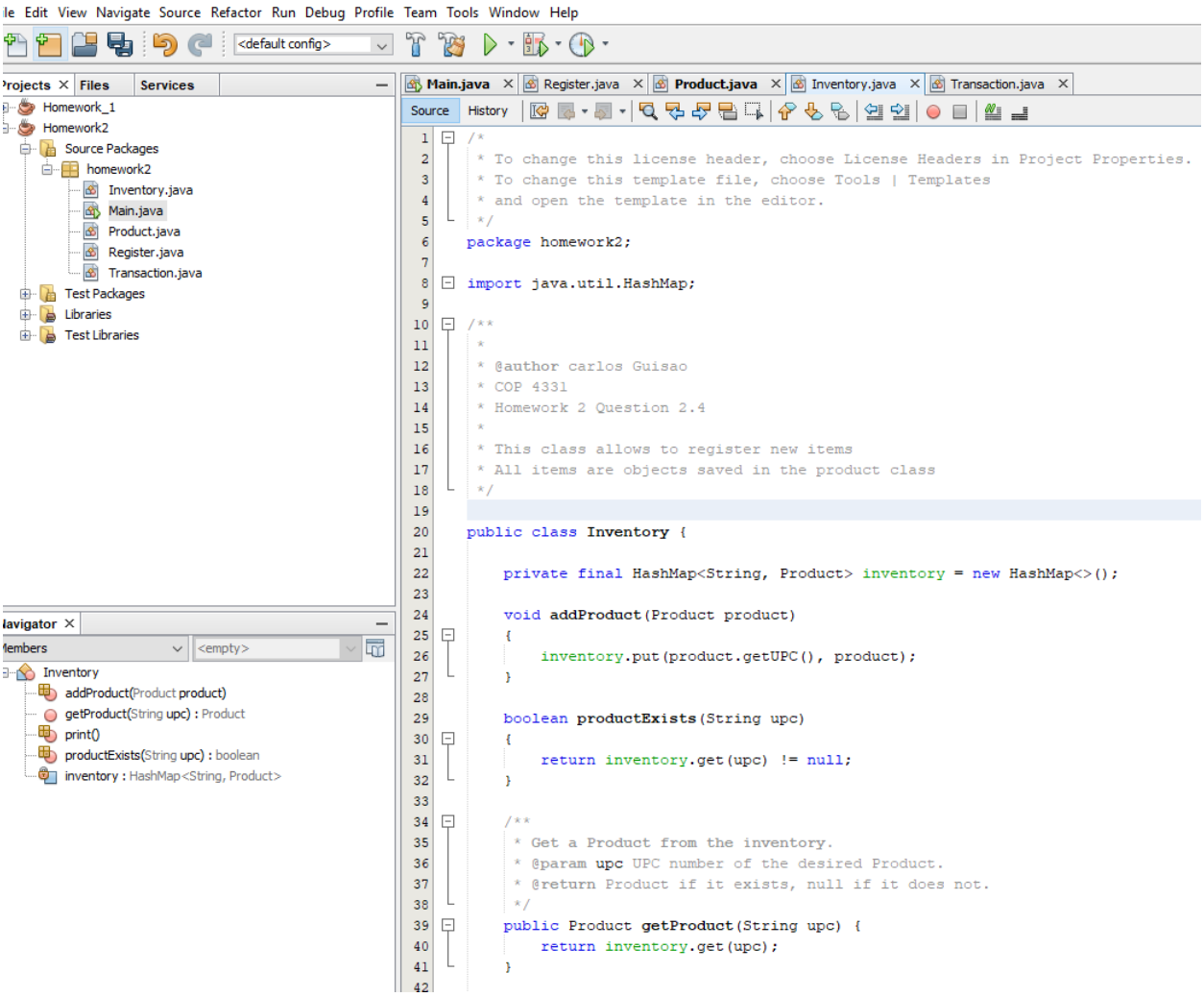


Product Class


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects Files Services
Homework_1
Homework2
Source Packages
homework2
Inventory.java
Main.java
Product.java
Register.java
Transaction.java
Test Packages
Libraries
Test Libraries
Navigator
Members
<empty>
Product
Product(Integer upc, String product, Double price)
getPrice() : String
getProduct() : String
getSummary() : String[]
getUPC() : String
Price : Double
Product : String
UPC : Integer
Main.java
Register.java
Product.java
Inventory.java
Transaction.java
Source History
1 /*
2 * To change this license header, choose License Headers in Project Properties.
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6 package homework2;
7
8 /**
9 *
10 * @author carlos Guisao
11 * COP 4331
12 * Homework 2 Question 2.4
13 *
14 * This class allows to register new items
15 * All items are objects saved in the product class
16 */
17
18 public class Product {
19
20 private final Integer UPC;
21 private final String Product;
22 private final Double Price;
23
24 /**
25 * Initializes a product with the given information.
26 * @param upc Integer with the UPC
27 * @param product product name
28 * @param price product price
29 */
30
31 public Product(Integer upc, String product, Double price)
32 {
33 this.UPC = upc;
34 this.Price = price;
35 this.Product = product;
36 }
37

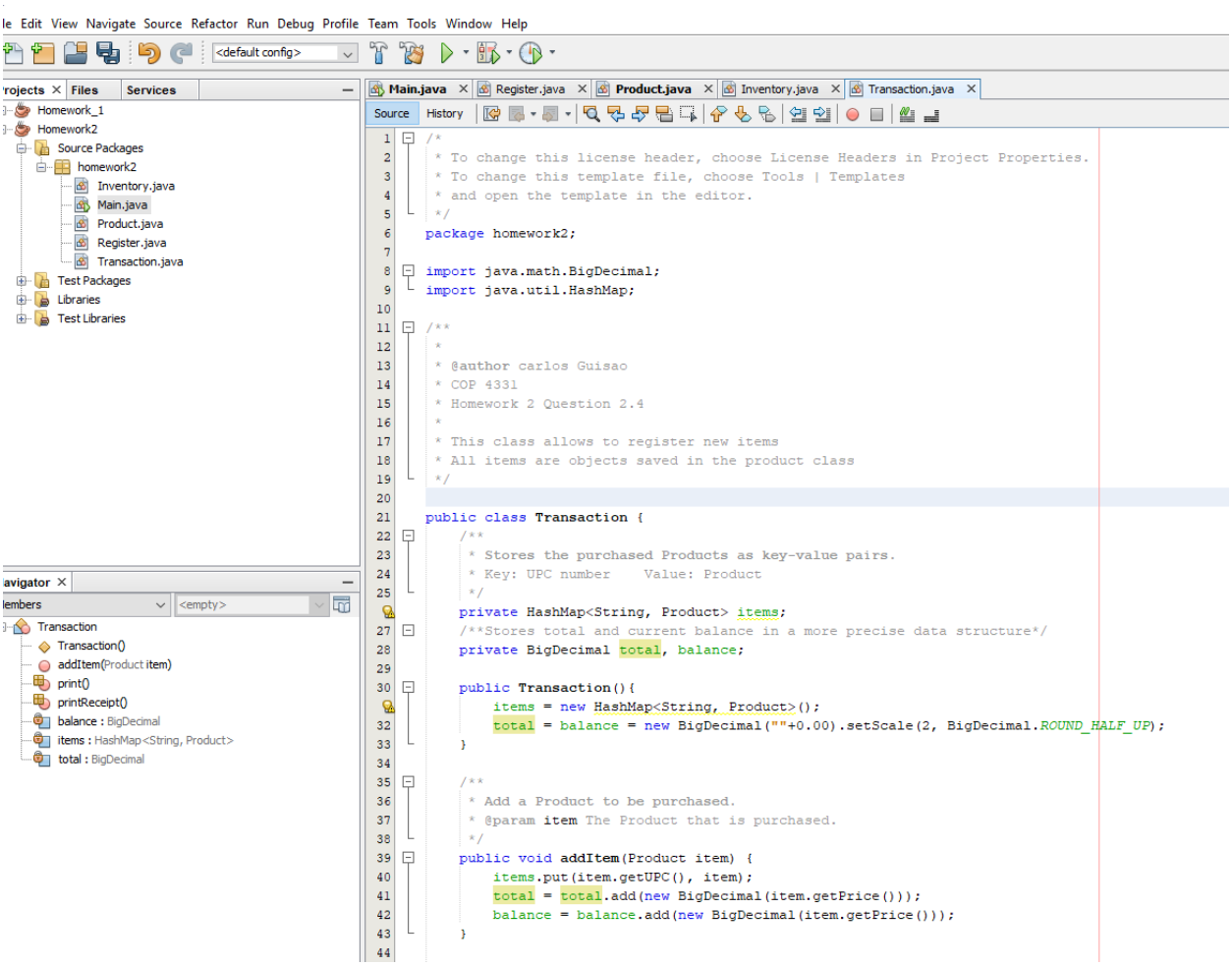


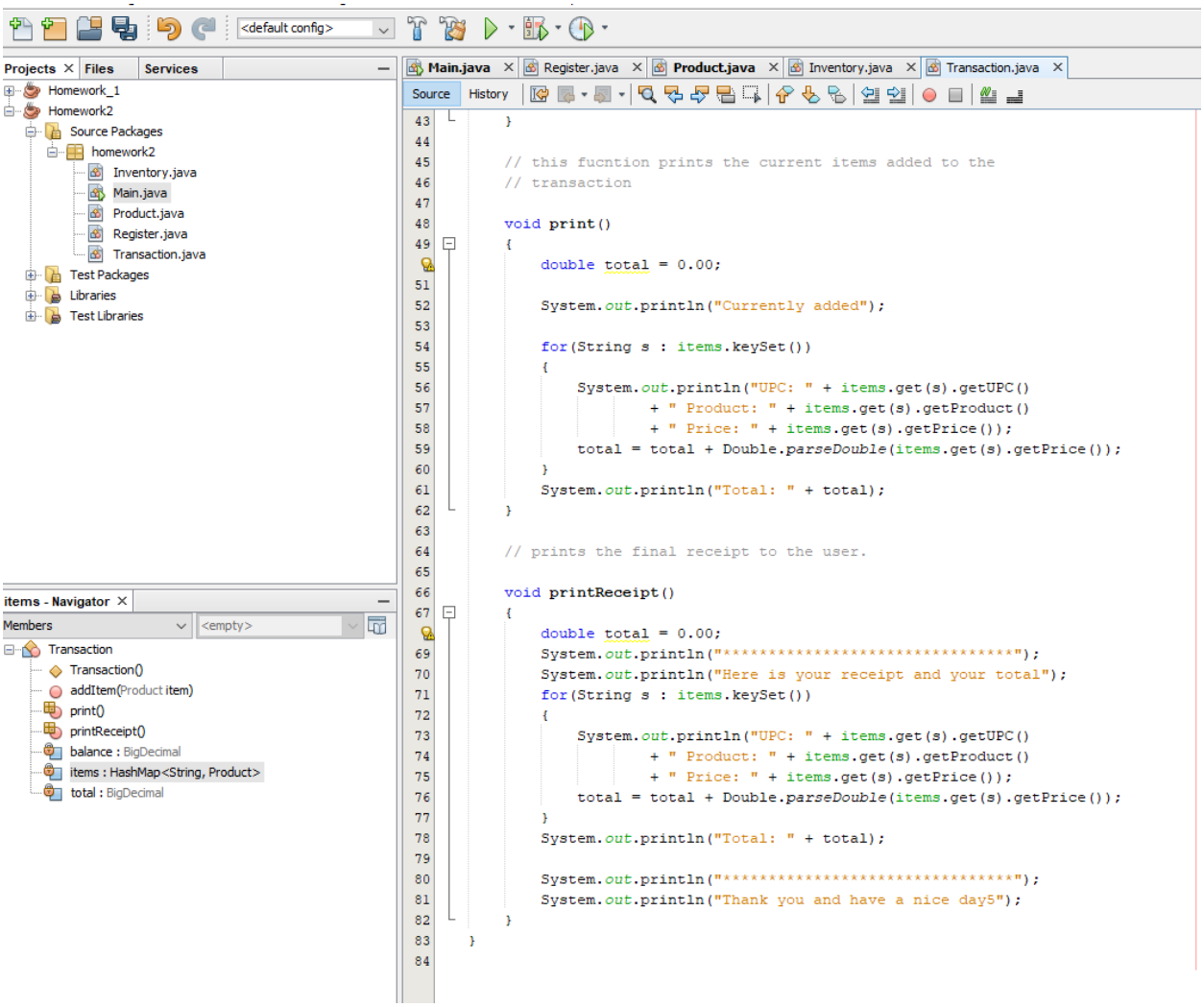
Inventory Class


le Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects
Files
Services
Homework_1
Homework2
Source Packages
homework2
Inventory.java
Main.java
Product.java
Register.java
Transaction.java
Test Packages
Libraries
Test Libraries
Main.java
Register.java
Product.java
Inventory.java
Transaction.java
Source
History
1 /*
2 * To change this license header, choose License Headers in Project Properties.
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6 package homework2;
7
8 import java.util.HashMap;
9
10 /**
11 *
12 * @author carlos Guisao
13 * COP 4331
14 * Homework 2 Question 2.4
15 *
16 * This class allows to register new items
17 * All items are objects saved in the product class
18 */
19
20 public class Inventory {
21
22 private final HashMap<String, Product> inventory = new HashMap<>();
23
24 void addProduct(Product product)
25 {
26 inventory.put(product.getUPC(), product);
27 }
28
29 boolean productExists(String upc)
30 {
31 return inventory.get(upc) != null;
32 }
33
34 /**
35 * Get a Product from the inventory.
36 * @param upc UPC number of the desired Product.
37 * @return Product if it exists, null if it does not.
38 */
39 public Product getProduct(String upc) {
40 return inventory.get(upc);
41 }
42
avigator
Members
<empty>
Inventory
addProduct(Product product)
getProduct(String upc) : Product
print()
productExists(String upc) : boolean
inventory : HashMap<String, Product>

```
41      }
42
43      // prints all of the items from the inventory
44      // it is called from the register function
45
46      void print()
47      {
48          for(String s : inventory.keySet())
49          {
50              System.out.println("UPC: " + inventory.get(s).getUPC()
51                      + " Product: " + inventory.get(s).getProduct()
52                      + " Price: " + inventory.get(s).getPrice());
53          }
54      }
55  }
56
```

Transaction Class


le Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
rojects
Files
Services
Homework_1
Homework2
Source Packages
homework2
Inventory.java
Main.java
Product.java
Register.java
Transaction.java
Test Packages
Libraries
Test Libraries
avigator
lembers
<empty>
Transaction
Transaction()
addItem(Product item)
print()
printReceipt()
balance : BigDecimal
items : HashMap<String, Product>
total : BigDecimal
Main.java
Register.java
Product.java
Inventory.java
Transaction.java
Source
History
/*
* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package homework2;
import java.math.BigDecimal;
import java.util.HashMap;
/**
*
* @author carlos Guisao
* COP 4331
* Homework 2 Question 2.4
*
* This class allows to register new items
* All items are objects saved in the product class
*/
public class Transaction {
/**
* Stores the purchased Products as key-value pairs.
* Key: UPC number Value: Product
*/
private HashMap<String, Product> items;
/**Stores total and current balance in a more precise data structure*/
private BigDecimal total, balance;
public Transaction(){
items = new HashMap<String, Product>();
total = balance = new BigDecimal(""+0.00).setScale(2, BigDecimal.ROUND_HALF_UP);
}
/**
* Add a Product to be purchased.
* @param item The Product that is purchased.
*/
public void addItem(Product item) {
items.put(item.getUPC(), item);
total = total.add(new BigDecimal(item.getPrice()));
balance = balance.add(new BigDecimal(item.getPrice()));
}



2.5

a.

Case Study: Edit Profile

- edit X's profile, that includes name, education, and work history (i.e. current and prior jobs).
- Education consists of a list of schools (just their name) the user attended.
- A job's description includes just the company name.

Use Case: User logs in and creates a new profile

1. User enters application for the first time
2. User creates an account
3. User creates username and password

Use Case: User Edit profile

1. User logs in
2. The system authenticates the user
3. User enters the application dashboard
3. The enters the edit profile module
4. The user fills out form

Case Study: Write Post

- post an update on X's own blog (like posting on the Facebook wall)

Use case: Post creation

1. User logs in
2. The system authenticates the user
3. User enters the application dashboard
4. User enters the post creation module
5. User writes post
6. System saves post and post it for everyone to see

Case Study: Read Recent Posts

- read posts written by all of user X's social network members

Use case: Read posts

1. User logs in
2. The system authenticates the user
3. User enters the application dashboard
4. User begins to scroll down and read posts

b. CRC Card

Use case: Edit Profile

Class: Registration

Responsibility:

1. Registers users
2. process main logic

Collaborates: login

Class: login

Responsibility:

1. logs users
2. Authenticates users

Collaborates: Registration

Class: userProfile

Responsibility:

1. logs users
2. Authenticates users

Collaborates: Registration, login

Class: postCreation

Responsibility:

1. user creates post

Collaborates: ReadPost

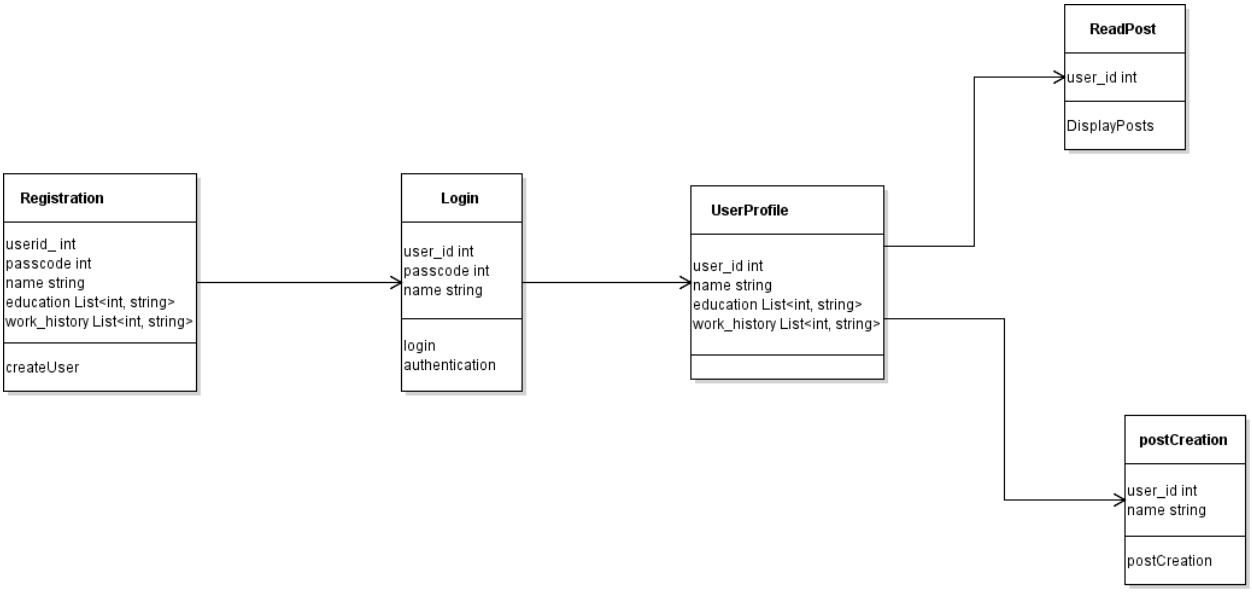
Class: ReadPost

Responsibility:

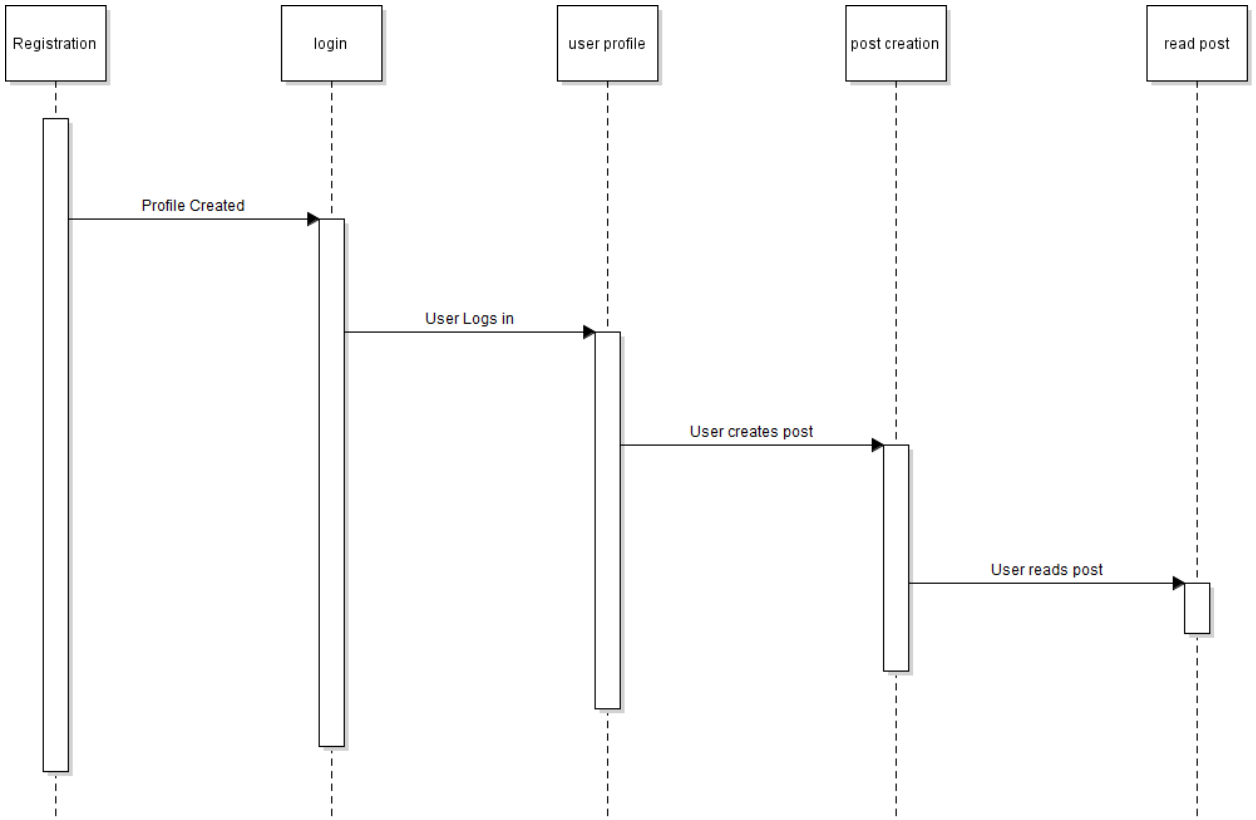
1. Dashboard

Collaborates: postCreation

Class diagram



Sequence diagram


Registration
login
user profile
post creation
read post
Profile Created
User Logs in
User creates post
User reads post