ICT373– Assignment 2 | Semester 2, 2023

Magazine Service Program Document

Date of Submission: 2/6/2023

Author: Jin Cherng Chong

Java Files: AssociateCustomer, AssociateDatabase, Customer, Magazine, PayingCustomer, Supplement, SupplementDatabase, MagazineServiceController, MagazineService, BillingHistory, MagazineServiceForm

Requirements/Specifications

Assumptions:

- Aud Is currency
- Billing history includes cost of magazine for each customer that PayingCustomer is paying for as well
- Month is 4 weeks
- Adding Customer/Adding Supplement is part of Create Mode. Edit mode is pure editing/deleting
- Adding subscription/Delete subscription is found in edit mode because in order to add a subscription it must already be created. Hence it really is an editing functionality
- Editing customer information means editing customer class information only not subtypes of customer class
- “Save existing magazine service” button will override/create the saveFile.DAT in the magazineService root directory. This file enables persistence of objects
- Users can load this saveFile.dat file to see the persistence of objects/saved magazine service

This program is used to manage a magazine service. The client can use this program to handle the magazine service. The program allows the client to add customers to the magazine service, magazine and supplements. The customers can subscribe to the service which allows for the client to know how much money is owed by to them by the customers. This helps makes it easy to monitor and manage the service.

User guide

To test out the magazine service program-

Head to create mode and load the saveFile.dat

OR

Begin in the create mode and

1. Create Magazine
2. Create Supplement/Customer

Structure/Design

Design description:

Singleton design pattern:

The Singleton design pattern has been successfully implemented in my MagazineService class. The MagazineService.java file fully utilizes this pattern. By employing the Singleton design pattern, we ensure that only one instance of the class can exist during runtime. In the case of the MagazineService, all GUI modifications operate on a single MagazineService object, which is made possible by the Singleton pattern.

The advantage of using Singleton in this scenario is that it enforces the rule of having only one MagazineService object. The constructor itself is private, preventing the programmer from creating multiple instances of the MagazineService class. Instead, a public getInstance method is provided, acting as the constructor function and returning the single instance of the MagazineService. This way, any manipulation of the MagazineService will directly affect the only existing object.

Façade design pattern

The Facade pattern has been implemented in the Customer class and Associate class. The MagazineService class serves as the interface for manipulating various data structures, including the supplementAvailable ArrayList and customerList. The MagazineService handles different complexities, such as ensuring that when a supplement is removed or deleted, it is also removed from the customers who are subscribed to it.

Simply deleting a supplement is not sufficient to remove it from the subscribed customers. This is because the customers still hold a reference to the deleted supplement. The garbage collector is only triggered when all references to an object are removed. Therefore, the responsibility falls on the MagazineService to remove the supplement and all references associated with it from the customer class.

Lack of returning objects

The program aims to minimize the need to return mutable objects. Returning an object from a class can violate the principles of encapsulation, which is designed to provide controlled access to modify an object. When an object is returned, it essentially provides a reference to the object, allowing the programmer to bypass encapsulation and modify the data members directly.

In order to avoid potential violations of encapsulation, the program implements strategies to prevent the direct return of objects. For example, in the case of the MagazineService class, the supplementDatabase needs to be sent to the MagazineService, as it is responsible for iterating through each element and printing them. To address this, a GetReadOnlyInterestedSupplements()

method has been intentionally designed to provide an unmodifiable list of the supplementDatabase to the MagazineService. This enforces encapsulation by allowing read-only access to the supplementDatabase.

Another example can be found in the AssociateDatabase class, where the data structure itself is never returned. Instead, the class provides the ContainsAssociate(...) method. When a programmer wants to check if an element is present in the AssociateDatabase, they pass the customer object to this method and let the class handle the check internally. By not returning the object to the programmer directly, the AssociateDatabase class promotes encapsulation.

Similarly, the MagazineService class has several methods that return data structures different from those encapsulated within the class. For instance, the GetNamesAllCustomers() method returns a string ArrayList containing all the names of the customers back to the controller. Since string data types are immutable, any modification made to the string representing a customer's name does not affect the customer object itself. By avoiding the direct return of the data structure for customer objects, encapsulation is upheld.

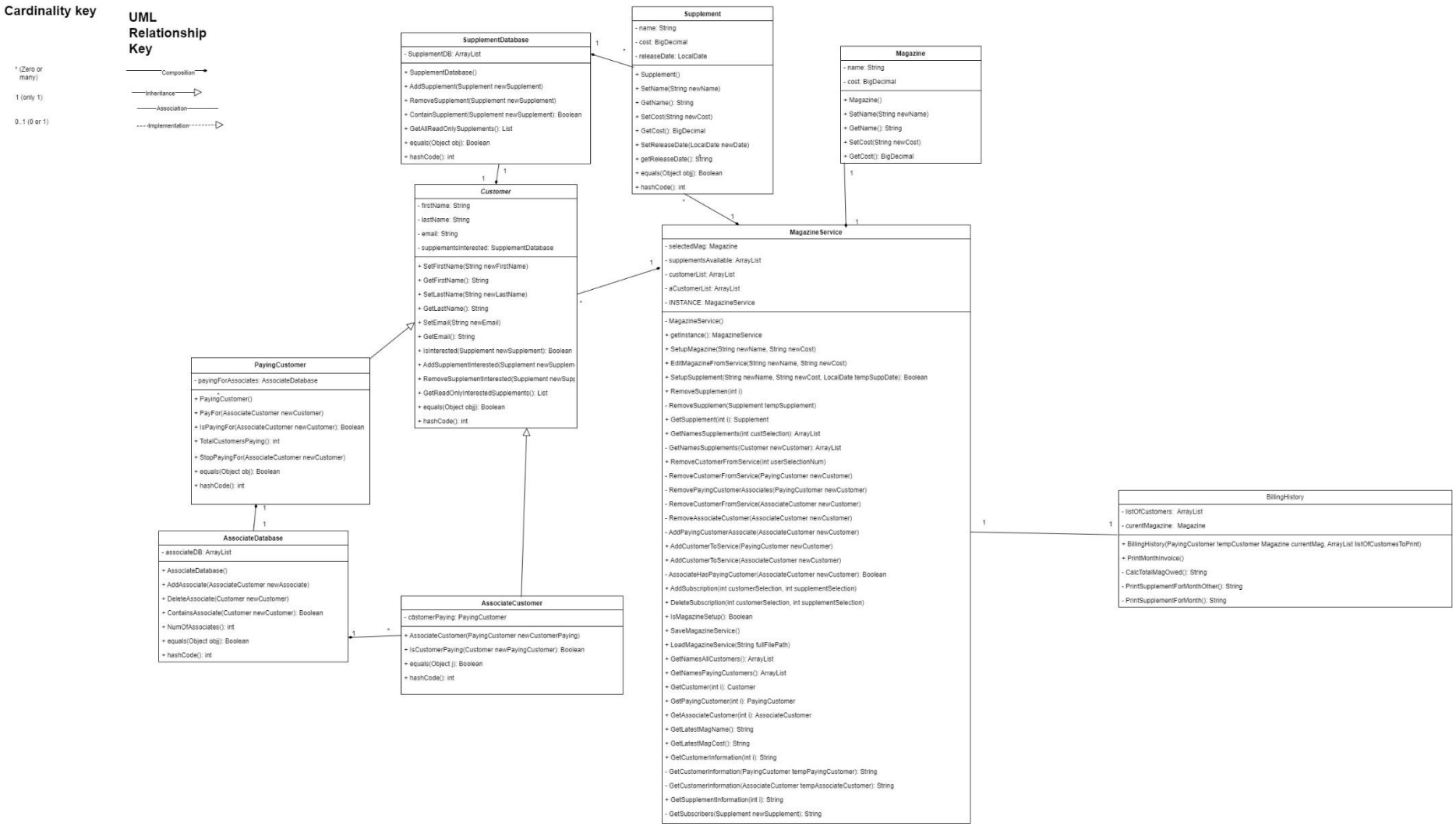
Promotion of abstraction

The MagazineService class is designed to promote abstraction and information hiding. To achieve this, I have intentionally ensured that, whenever possible, the controller interacts with primitive data types instead of objects and their creation. For example, the RemoveSupplement() method in the MagazineService class has a public interface that accepts a primitive data type, specifically an integer parameter (i). The controller will pass a primitive integer (i) representing the location of the supplement to be removed.

The advantage of using primitive data types in this manner is that it separates the business logic from the user interface (UI). Consequently, any modifications made to the supplement object will not impact the front-end UI. For instance, if we decide to introduce a new class called SupplementVersionTwo and replace the Supplement object with the Supplement object created from the SupplementVersionTwo class, the overall UI will remain unaffected. The client will still be able to pass a primitive integer (i) as a parameter, and the backend will handle the necessary object changes, allowing the continued use of the method.

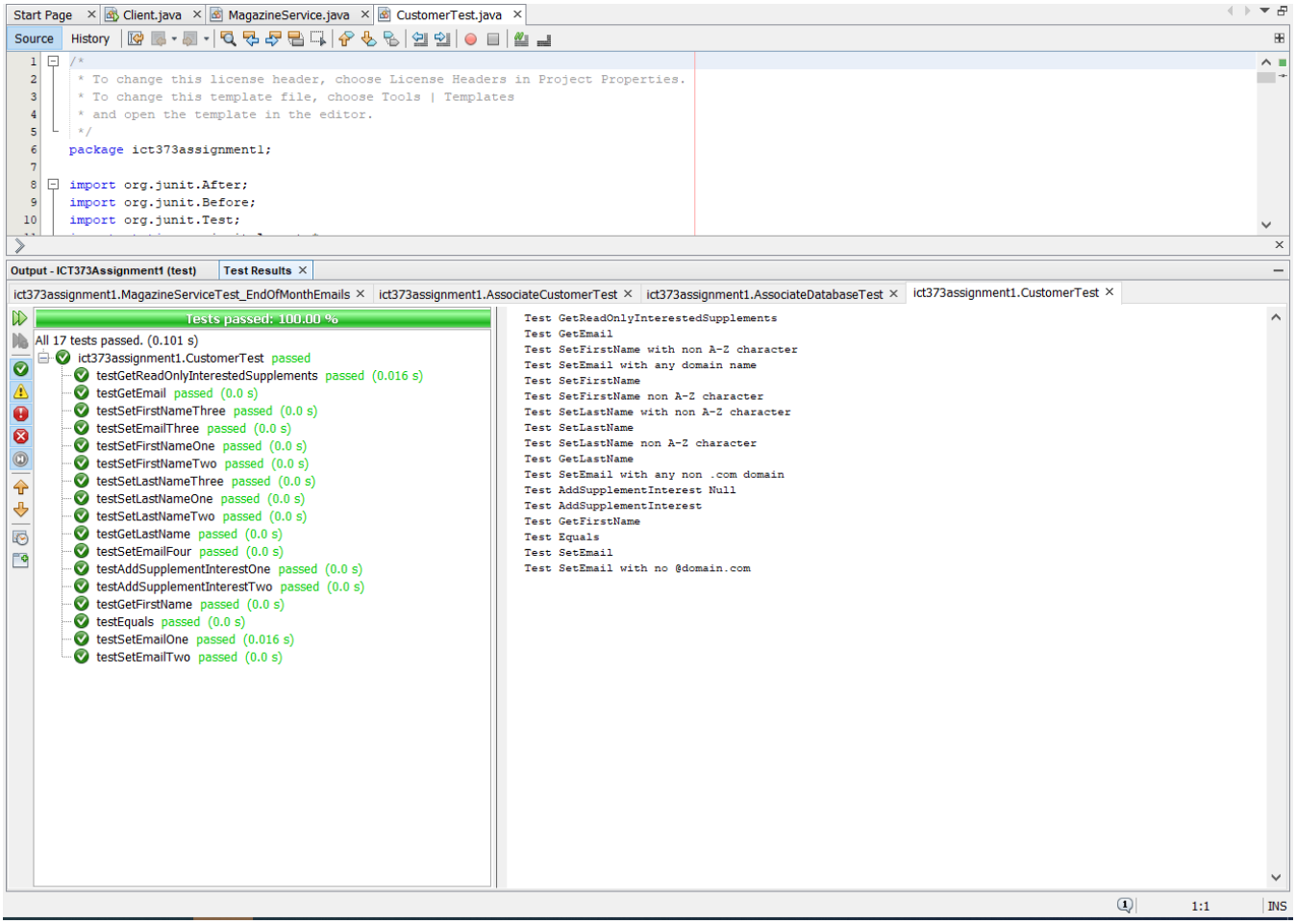
This approach enhances the decoupling of the UI and the underlying implementation, promoting flexibility and maintainability in the system.

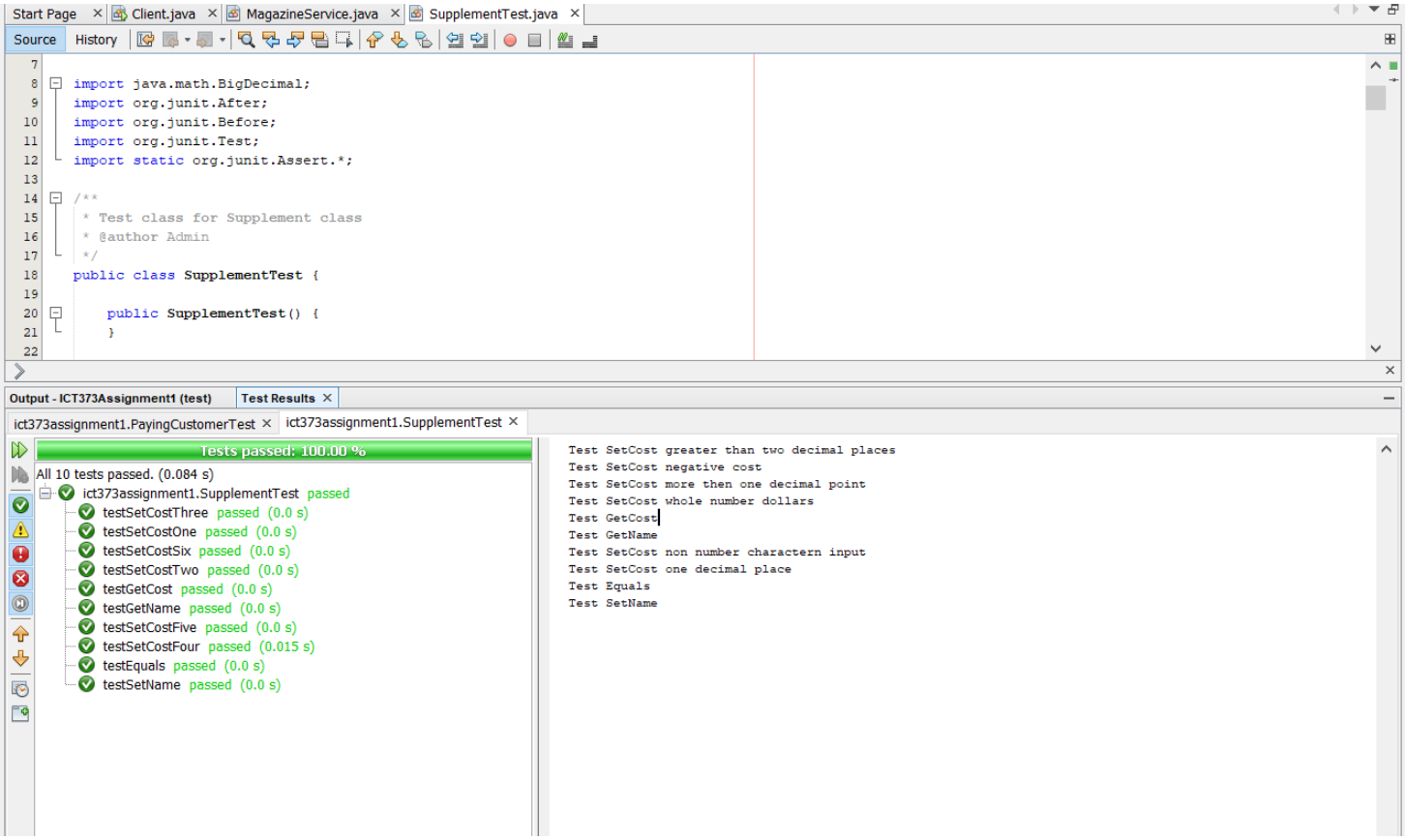
UML

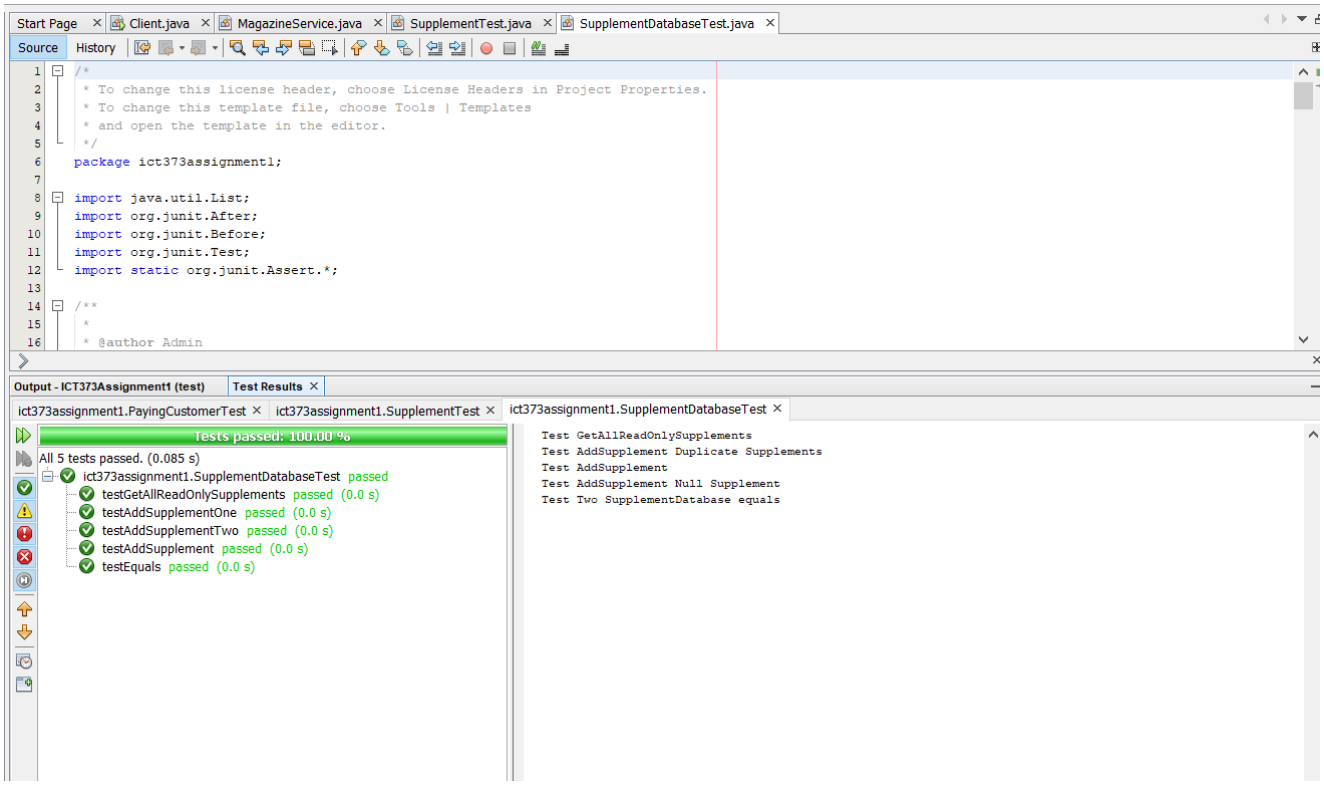

Cardinality key
* (Zero or many)
1 (only 1)
0..1 (0 or 1)
UML Relationship Key
Composition
Inheritance
Association
Implementation
SupplementDatabase
- SupplementDB: ArrayList
+ SupplementDatabase()
+ AddSupplement(Supplement newSupplement)
+ RemoveSupplement(Supplement newSupplement)
+ ContainSupplement(Supplement newSupplement): Boolean
+ GetAllReadOnlySupplements(): List
+ equals(Object obj): Boolean
+ hashCode(): int
Supplement
- name: String
- cost: BigDecimal
- releaseDate: LocalDate
+ Supplement()
+ SetName(String newName)
+ GetName(): String
+ SetCost(String newCost)
+ GetCost(): BigDecimal
+ SetReleaseDate(LocalDate newDate)
+ getReleaseDate(): String
+ equals(Object obj): Boolean
+ hashCode(): int
Magazine
- name: String
- cost: BigDecimal
+ Magazine()
+ SetName(String newName)
+ GetName(): String
+ SetCost(String newCost)
+ GetCost(): BigDecimal
Customer
- firstName: String
- lastName: String
- email: String
- supplementsInterested: SupplementDatabase
+ SetFirstName(String newFirstName)
+ GetFirstName(): String
+ SetLastName(String newLastName)
+ GetLastName(): String
+ SetEmail(String newEmail)
+ GetEmail(): String
+ IsInterested(Supplement newSupplement): Boolean
+ AddSupplementInterested(Supplement newSupplem
+ RemoveSupplementInterested(Supplement newSupp
+ GetReadOnlyInterestedSupplements(): List
+ equals(Object obj): Boolean
+ hashCode(): int
MagazineService
- selectedMag: Magazine
- supplementsAvailable: ArrayList
- customerList: ArrayList
- aCustomerList: ArrayList
- INSTANCE: MagazineService
- MagazineService()
+ getInstance(): MagazineService
+ SetupMagazine(String newName, String newCost)
+ EditMagazineFromService(String newName, String newCost)
+ SetupSupplement(String newName, String newCost, LocalDate tempSuppDate): Boolean
+ RemoveSupplemen(int i)
- RemoveSupplemen(Supplement tempSupplement)
+ GetSupplement(int i): Supplement
+ GetNamesSupplements(int custSelection): ArrayList
- GetNamesSupplements(Customer newCustomer): ArrayList
+ RemoveCustomerFromService(int userSelectionNum)
- RemoveCustomerFromService(PayingCustomer newCustomer)
- RemovePayingCustomerAssociates(PayingCustomer newCustomer)
- RemoveCustomerFromService(AssociateCustomer newCustomer)
- RemoveAssociateCustomer(AssociateCustomer newCustomer)
- AddPayingCustomerAssociate(AssociateCustomer newCustomer)
+ AddCustomerToService(PayingCustomer newCustomer)
+ AddCustomerToService(AssociateCustomer newCustomer)
- AssociateHasPayingCustomer(AssociateCustomer newCustomer): Boolean
+ AddSubscription(int customerSelection, int supplementSelection)
+ DeleteSubscription(int customerSelection, int supplementSelection)
+ IsMagazineSetup(): Boolean
+ SaveMagazineService()
+ LoadMagazineService(String fullFilePath)
+ GetNamesAllCustomers(): ArrayList
+ GetNamesPayingCustomers(): ArrayList
+ GetCustomer(int i): Customer
+ GetPayingCustomer(int i): PayingCustomer
+ GetAssociateCustomer(int i): AssociateCustomer
+ GetLatestMagName(): String
+ GetLatestMagCost(): String
+ GetCustomerInformation(int i): String
- GetCustomerInformation(PayingCustomer tempPayingCustomer): String
- GetCustomerInformation(AssociateCustomer tempAssociateCustomer): String
+ GetSupplementInformation(int i): String
- GetSubscribers(Supplement newSupplement): String
PayingCustomer
- payingForAssociates: AssociateDatabase
+ PayingCustomer()
+ PayFor(AssociateCustomer newCustomer)
+ IsPayingFor(AssociateCustomer newCustomer): Boolean
+ TotalCustomersPaying(): int
+ StopPayingFor(AssociateCustomer newCustomer)
+ equals(Object obj): Boolean
+ hashCode(): int
AssociateDatabase
- associateDB: ArrayList
+ AssociateDatabase()
+ AddAssociate(AssociateCustomer newAssociate)
+ DeleteAssociate(Customer newCustomer)
+ ContainsAssociate(Customer newCustomer): Boolean
+ NumOfAssociates(): int
+ equals(Object obj): Boolean
+ hashCode(): int
AssociateCustomer
- cstomerPaying: PayingCustomer
+ AssociateCustomer(PayingCustomer newCustomerPaying)
+ IsCustomerPaying(Customer newPayingCustomer): Boolean
+ equals(Object j): Boolean
+ hashCode(): int
BillingHistory
- listOfCustomers: ArrayList
- curentMagazine: Magazine
+ BillingHistory(PayingCustomer tempCustomer Magazine currentMag, ArrayList listOfCustomesToPrint)
+ PrintMonthInvoice()
- CalcTotalMagOwed(): String
- PrintSupplementForMonthOther(): String
- PrintSupplementForMonth(): String

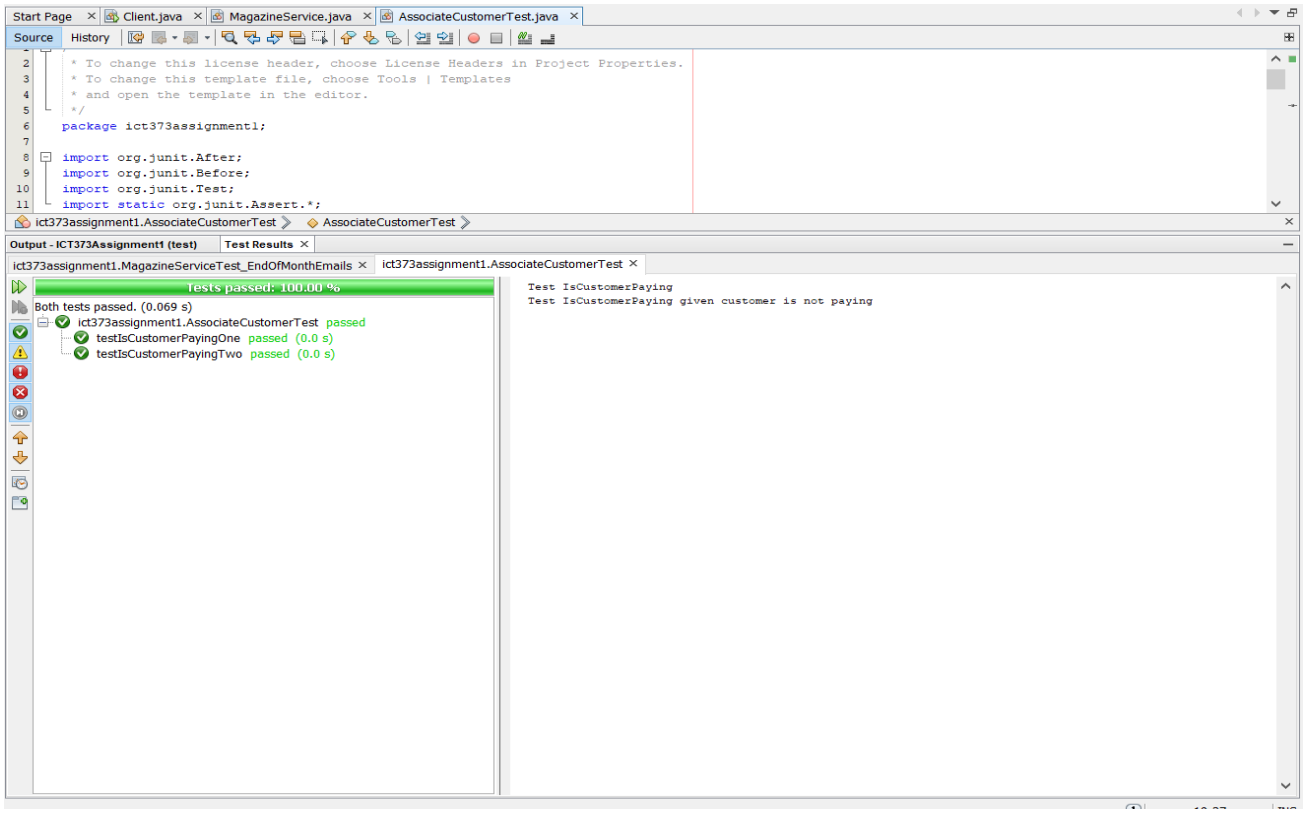
Testing:

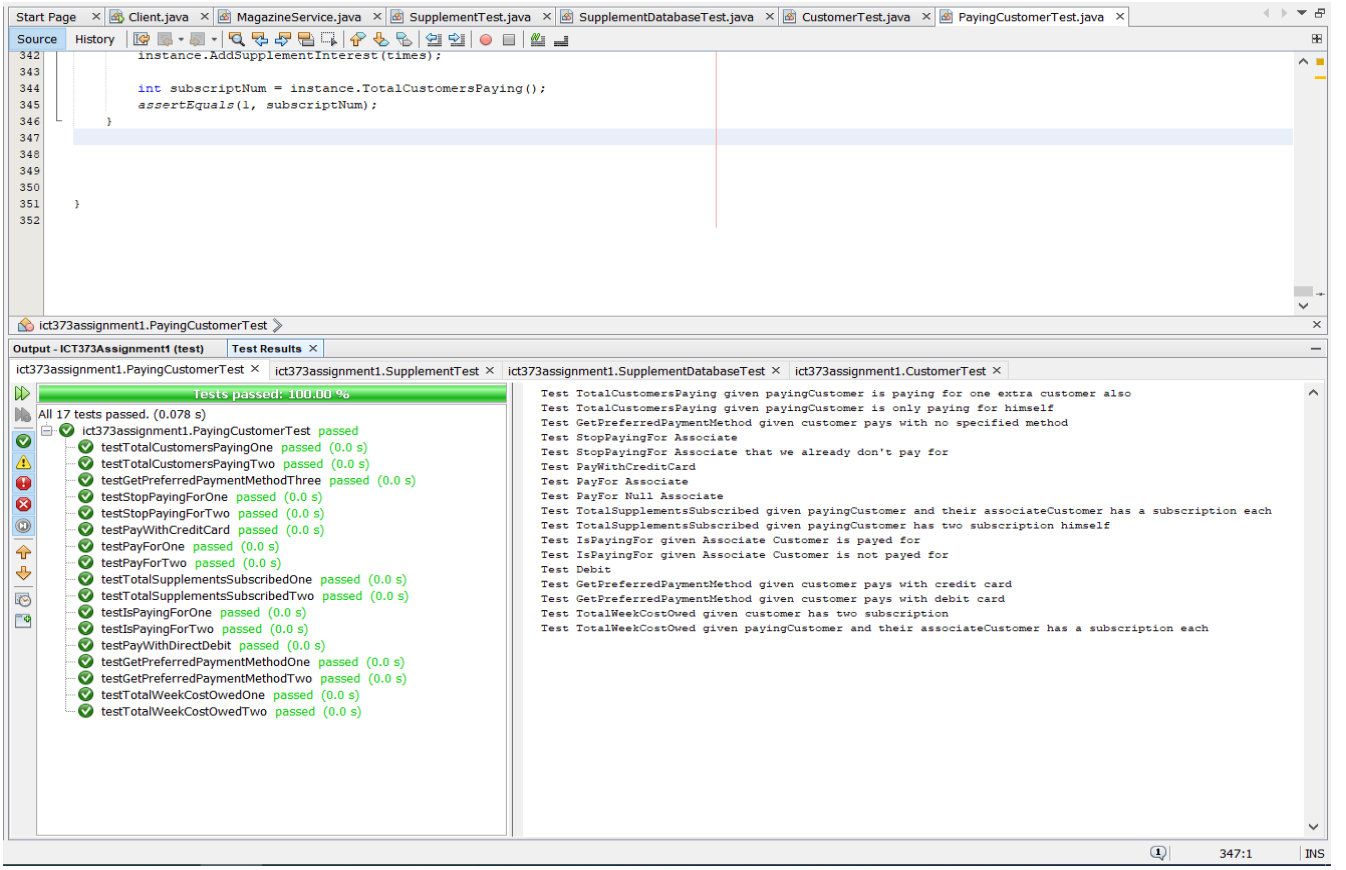
Unit testing had been completed for the following classes- AssociateCustomer, AssociateDatabase, Customer, Magazine, Supplement, SupplementDatabase

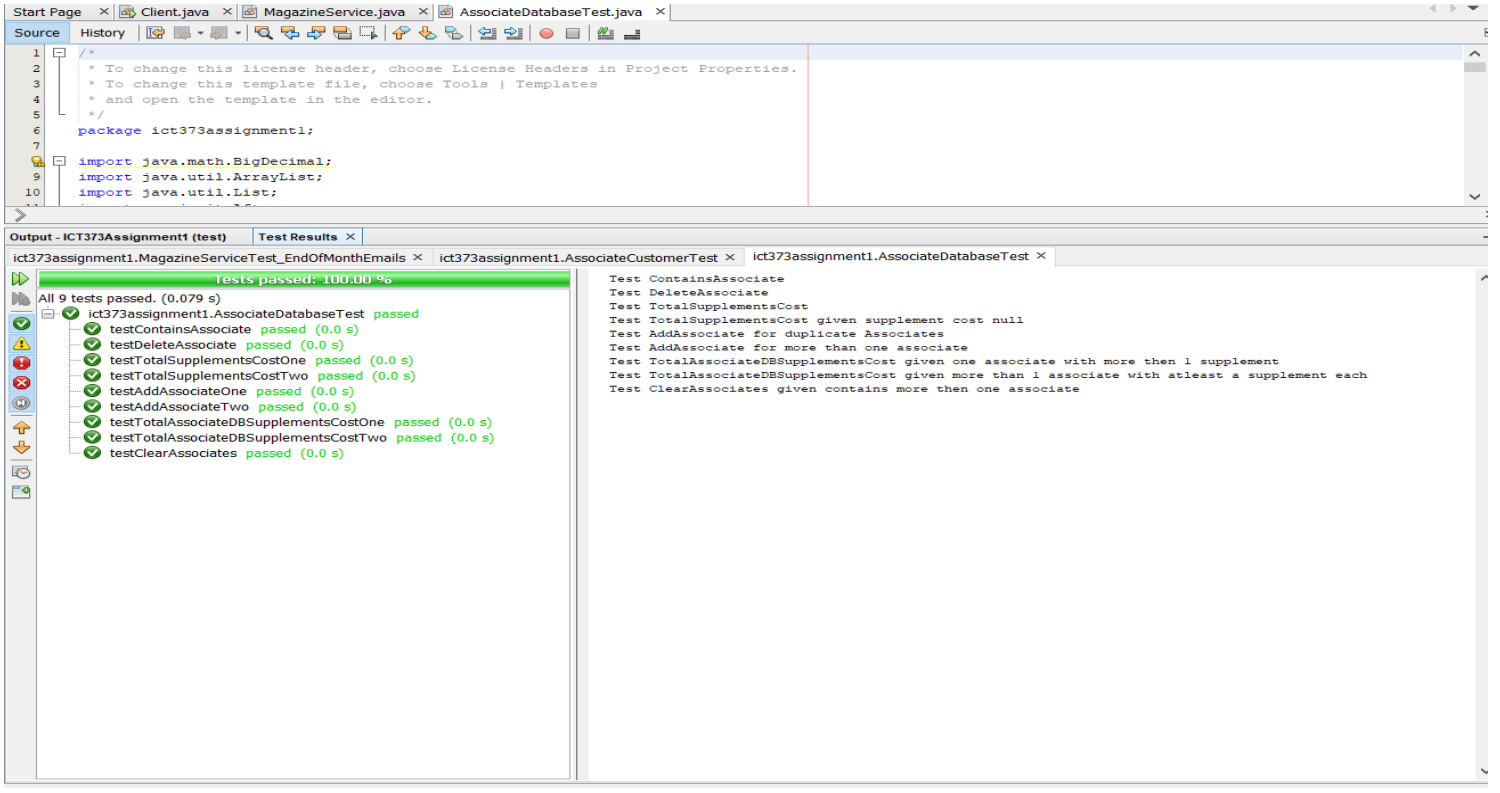



Start Page
Client.java
MagazineService.java
SupplementTest.java
Source
History
7
8 import java.math.BigDecimal;
9 import org.junit.After;
10 import org.junit.Before;
11 import org.junit.Test;
12 import static org.junit.Assert.*;
13
14 /**
15 * Test class for Supplement class
16 * @author Admin
17 */
18 public class SupplementTest {
19
20 public SupplementTest() {
21 }
22
Output - ICT373Assignment1 (test)
Test Results
ict373assignment1.PayingCustomerTest
ict373assignment1.SupplementTest
Tests passed: 100.00 %
All 10 tests passed. (0.084 s)
ict373assignment1.SupplementTest passed
testSetCostThree passed (0.0 s)
testSetCostOne passed (0.0 s)
testSetCostSix passed (0.0 s)
testSetCostTwo passed (0.0 s)
testGetCost passed (0.0 s)
testGetName passed (0.0 s)
testSetCostFive passed (0.0 s)
testSetCostFour passed (0.015 s)
testEquals passed (0.0 s)
testSetName passed (0.0 s)
Test SetCost greater than two decimal places
Test SetCost negative cost
Test SetCost more then one decimal point
Test SetCost whole number dollars
Test GetCost
Test GetName
Test SetCost non number charactern input
Test SetCost one decimal place
Test Equals
Test SetName


Start Page
Client.java
MagazineService.java
SupplementTest.java
SupplementDatabaseTest.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 ict373assignment1;
import java.util.List;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
/**
*
* @author Admin
Output - ICT373Assignment1 (test)
Test Results
ict373assignment1.PayingCustomerTest
ict373assignment1.SupplementTest
ict373assignment1.SupplementDatabaseTest
Tests passed: 100.00 %
All 5 tests passed. (0.085 s)
ict373assignment1.SupplementDatabaseTest passed
testGetAllReadOnlySupplements passed (0.0 s)
testAddSupplementOne passed (0.0 s)
testAddSupplementTwo passed (0.0 s)
testAddSupplement passed (0.0 s)
testEquals passed (0.0 s)
Test GetAllReadOnlySupplements
Test AddSupplement Duplicate Supplements
Test AddSupplement
Test AddSupplement Null Supplement
Test Two SupplementDatabase equals


Start Page
Client.java
MagazineService.java
AssociateCustomerTest.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 ict373assignment1;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
ict373assignment1.AssociateCustomerTest
AssociateCustomerTest
Output - ICT373Assignment1 (test)
Test Results
ict373assignment1.MagazineServiceTest_EndOfMonthEmails
ict373assignment1.AssociateCustomerTest
Tests passed: 100.00 %
Both tests passed. (0.069 s)
ict373assignment1.AssociateCustomerTest passed
testIsCustomerPayingOne passed (0.0 s)
testIsCustomerPayingTwo passed (0.0 s)
Test IsCustomerPaying
Test IsCustomerPaying given customer is not paying


Start Page
Client.java
MagazineService.java
SupplementTest.java
SupplementDatabaseTest.java
CustomerTest.java
PayingCustomerTest.java
Source
History
instance.AddSupplementInterest(times);
int subscriptNum = instance.TotalCustomersPaying();
assertEquals(1, subscriptNum);
}
}
ict373assignment1.PayingCustomerTest
Output - ICT373Assignment1 (test)
Test Results
ict373assignment1.PayingCustomerTest
ict373assignment1.SupplementTest
ict373assignment1.SupplementDatabaseTest
ict373assignment1.CustomerTest
Tests passed: 100.00 %
All 17 tests passed. (0.078 s)
ict373assignment1.PayingCustomerTest passed
testTotalCustomersPayingOne passed (0.0 s)
testTotalCustomersPayingTwo passed (0.0 s)
testGetPreferredPaymentMethodThree passed (0.0 s)
testStopPayingForOne passed (0.0 s)
testStopPayingForTwo passed (0.0 s)
testPayWithCreditCard passed (0.0 s)
testPayForOne passed (0.0 s)
testPayForTwo passed (0.0 s)
testTotalSupplementsSubscribedOne passed (0.0 s)
testTotalSupplementsSubscribedTwo passed (0.0 s)
testIsPayingForOne passed (0.0 s)
testIsPayingForTwo passed (0.0 s)
testPayWithDirectDebit passed (0.0 s)
testGetPreferredPaymentMethodOne passed (0.0 s)
testGetPreferredPaymentMethodTwo passed (0.0 s)
testTotalWeekCostOwedOne passed (0.0 s)
testTotalWeekCostOwedTwo passed (0.0 s)
Test TotalCustomersPaying given payingCustomer is paying for one extra customer also
Test TotalCustomersPaying given payingCustomer is only paying for himself
Test GetPreferredPaymentMethod given customer pays with no specified method
Test StopPayingFor Associate
Test StopPayingFor Associate that we already don't pay for
Test PayWithCreditCard
Test PayFor Associate
Test PayFor Null Associate
Test TotalSupplementsSubscribed given payingCustomer and their associateCustomer has a subscription each
Test TotalSupplementsSubscribed given payingCustomer has two subscription himself
Test IsPayingFor given Associate Customer is payed for
Test IsPayingFor given Associate Customer is not payed for
Test Debit
Test GetPreferredPaymentMethod given customer pays with credit card
Test GetPreferredPaymentMethod given customer pays with debit card
Test TotalWeekCostOwed given customer has two subscription
Test TotalWeekCostOwed given payingCustomer and their associateCustomer has a subscription each
347:1
INS


Start Page
Client.java
MagazineService.java
AssociateDatabaseTest.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 ict373assignment1;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;
Output - ICT373Assignment1 (test)
Test Results
ict373assignment1.MagazineServiceTest_EndOfMonthEmails
ict373assignment1.AssociateCustomerTest
ict373assignment1.AssociateDatabaseTest
Tests passed: 100.00 %
All 9 tests passed. (0.079 s)
ict373assignment1.AssociateDatabaseTest passed
testContainsAssociate passed (0.0 s)
testDeleteAssociate passed (0.0 s)
testTotalSupplementsCostOne passed (0.0 s)
testTotalSupplementsCostTwo passed (0.0 s)
testAddAssociateOne passed (0.0 s)
testAddAssociateTwo passed (0.0 s)
testTotalAssociateDBSupplementsCostOne passed (0.0 s)
testTotalAssociateDBSupplementsCostTwo passed (0.0 s)
testClearAssociates passed (0.0 s)
Test ContainsAssociate
Test DeleteAssociate
Test TotalSupplementsCost
Test TotalSupplementsCost given supplement cost null
Test AddAssociate for duplicate Associates
Test AddAssociate for more than one associate
Test TotalAssociateDBSupplementsCost given one associate with more then 1 supplement
Test TotalAssociateDBSupplementsCost given more than 1 associate with atleast a supplement each
Test ClearAssociates given contains more then one associate

Feature testing:

Test #	Test objective(s)	Test step(s)	Expected results	Pass /Fail
1	Load existing magazine service	• Click the “Create Magazine Service (via File Upload)” • Submit and Click “Load File” • Select saveFile.Dat	File successfully loaded	Pass
2	Create Associate without PayingCustomer	• Click Create Customer • Fill out details and select role as associate • Select N/A for nominated paying customer • Click submit	Error: mentioning require payingCustomerr	Pass
3	Create PayingCustomer with nominated paying customer	• Click Create Customer • Fill out details and select role as paying • Select another paying customer for payee • Click submit	Error: Mentioning paying customer cannot have another payee	Pass

Limitations

The program has several limitations:

- Spam clicking the create a customer form or any form creation may cause runtime error. Only one form can be created for that one form at a time
- Billing only provides running total billing for that month and join data/subscribed data does not influence the billing
- Customers does not have address