1a)

My assignment in relation to Question 1 is a web application that allows users to input details about themselves and display the details via http://www.it.murdoch.edu.au/cgi-bin/reply1.pl. The application front-end utilises a HTML form page to allow for user details input. JavaScript in the front end is responsible for HTML form page validation and processing. The HTML form input needs to be validated to ensure the form is filled out correctly and thus displayed correctly and the server with user information is not put at risk of being compromised. In the future, if the server needs to actually permanently store all the user details this application should be extended and back end or server-side validation should be implemented. Bypassing the front-end validation web applications rather simple to do. Given the current specifications requires the server to collect, display, and discard each HTML form submission server-side validation is not necessary.   Both runtime and batch validation has been implemented

Assumptions:

- Allow first name to have spaces. Evident my first name is Jin Cherng
- Smallest year starts at 1 because we follow Gregorian calendar
- Assume phone number prefix is based on international calling code
- Assume we use English numerals so Chinese numbers is not accepted

- Assume Australian two decimal place money do not need to be rounded as per law
    - https://www.commerce.wa.gov.au/consumer-protection/rounding
    - https://www.justweb.com.au/law-articles/rounding-up.html

1b)

Run time validation has been implemented. Runtime validation is where individual events provide feedback. So, client is given feedback as he fills in the form. The disadvantage is that it can easily be bypassed as it is just warnings. Runtime validation is seen in file: addUserFormValidator.js

Batch validation has been implemented. Batch validation is where the entire form is checked for any errors. This is triggered at the end when the user presses the submit button. This validation will stop the form submission from going to the server side to be processed. The benefit is the error can be displayed without the for a page to be reloaded. Batch validation is seen in file: addUserFormValidator.js ValidateHomeForm(…) function. The advantage of batch validation is that errors can be displayed without the need to reload or re-render the page unlike server side validation.

<u>Adding user details:</u>

- AddUserFormHome.html
  - Provides the client the ability to submit user a person's details through POST
  - Responsible for batch validating the client's submission via clicking submission button
    - Calls ValidateHomeForm(….) from addUserFormValidator.js
  - Responsible for providing real time validation via HTML textbox onchange attribute
    - Calls ValidateBirthDate(…), ValidatePrefixCode(…), ValidatePhoneNumber(…), ValidateFavPastTime(…), ValidFirstName(…), or ValidLastName(..)
- addUserFormValidator.js
  - Responsible for carrying out the HTML form validation of the input
  - Contains ValidateHomeForm(…) and this function initiates the entire form being validated. Importance is that that it handles POST batch validation
    - Contains ValidateBirthDate(..) which is responsible for initiating the date of birth textbox input being validated. This is run time validation
      - Calls IsValidDateFormat(…) and IsValidDate(…) from birthDateValidator.js
    - Contains ValidatePrefixCode(..) which is responsible for initiating the prefix code textbox input validation. This is run time validation
      - Calls IsValidPrefixFormat(…) from phoneNumberValidator.js
    - Contains ValidatePhoneNumber (..) which is responsible for initiating textbox input phone number being validated. This is run time validation
      - Calls IsValidPhoneNumber(…) from phoneNumberValidator.js
    - Contains ValidateFavPastTime (..) which is responsible for initiating drop down list input being validated. This is run time validation
    - Contains ValidateFullName(..) responsible for initiating both full name and last name input being validated.
  - Responsible for displaying form messages functions (alert) providing client direct form submission alerts

- birthDateValidator.js
  - Responds by carrying out the processing of user's date of birth input
  - The creation of this file and abstracting the birth date logic to a separate file has several benefits
    - Improves readability through information hiding of the birth day implementation logic
    - Encourages efficient code reuse by allowing other programs to import this file and are not exposed to other irrelevant logic
    - Promotes open closed principle because when a potential change is made to the date of birth logic the risk of accidentally harming the phone number logic is minimised due the separate file encapsulation of date of birth logic
- phoneNumberValidator.js
  - Responds by carrying out the processing of the user's phone number or user's prefix code input
  - Contains IsValidPhoneNumber(…) that handles the processing of user's phone number

- Contains IsValidPrefixFormat(…) that handles the processing of the user's prefix code
- The value of separating phone number validation logic in its own file is to improve readability in that when the code base of the application gets larger and modification is required with the phone number logic can easily be identified

1d)

List of Source Code for Question 1

addUserFormValidator.js

```
"use strict";


//Function for validating entire adding user form
//
//

function ValidateHomeForm(e) { //SELF NOTE: Why not seperate each test into it's own if? Putting all of them in one AND condition ensures return at the first falsy encounter

    var addFormValid = false;

    if(ValidateFullName() && ValidateBirthDate() && ValidatePrefixCode() && ValidatePhoneNumber() && ValidateFavPastTime()) {
        addFormValid = true;
    } else {
        e.preventDefault(); //SELF NOTE: Why not only just return false? Don't want form details to be reset this keeps current form state
        addFormValid = false;
    }

    return addFormValid;
}

//Functions for validating individual components of adding user form
//
//

function ValidateFullName() {

    var successName = false; //SELF NOTE: Why assign default false value? Better to be safe then sorry in ensuring return is always going to be boolean

    if(ValidFirstName() && ValidLastName()) {
        successName = true;
    } else {
        successName = false;
    }

    return successName;
}

function ValidFirstName() {

    var isValid = false; //SELF NOTE: Why assign default false value? Better to be safe then sorry in ensuring return is always going to be boolean
    var tempFirstName = document.getElementById("frmFirstName").value; //SELF NOTE: Why not put this line in ValidateFullName()? Because this is a form entry point gets called in onchange="ValidFirstName()" html

    if(ValidName(tempFirstName)) {
        isValid = true
    } else {
        DspFirstNameError(tempFirstName);
        isValid = false;
    }

    return isValid;
}
```

```
function ValidLastName() {

    var isValid = false; //SELF NOTE: Why assign default false value? Better to be safe then sorry in
ensuring return is always going to be boolean
    var tempLastName = document.getElementById("frmLastName").value; //SELF NOTE: Why not put this line in
ValidateFullName()? Because this is a form entry point gets called in onchange="ValidLastName()" html

    if(ValidName(tempLastName)) {
        isValid = true
    } else {
        DspLastNameError(tempLastName);
        isValid = false;
    }

    return isValid;
}



function ValidateBirthDate() {

    var tempFullDate = document.getElementById("frmBirthDate").value;

    if(!IsValidDateFormat(tempFullDate)) { //SELF NOTE: Why not follow single return policy? Because we
only want one error displayed and we don't want to follow through with bad input
        DspBirthDateFormatError();
        return false;
    }

    if(!IsValidDate(tempFullDate)) {
        DspBirthDateValueError();
        return false;
    }


    return true; //SELF NOTE: Why return true rather then boolean assigned true? Because follows the guard
clause pattern
}


function ValidatePrefixCode() {

    var tempPrefixCode = document.getElementById("frmPhonePrefix").value;
    var validPrefix = false;
    var tempFullPrefix = tempPrefixCode.trimEnd(); //SELF NOTE: Why not put this logic in
IsValidPrefixFormat? Because this is an application specific action

    if(IsValidPrefixFormat(tempFullPrefix)) {
        validPrefix = true;
    } else {
        DspPrefixCodeError(); //SELF NOTE: Why not put in IsValidPrefixFormat logic? Because
IsValidPrefixFormat is logically concerned with whether prefix is valid or not. Not concerned with output
in response to validity
        validPrefix = false;
    }

    return validPrefix;

}

function ValidatePhoneNumber() {

    var tempPhoneNum = document.getElementById("frmPhoneNumber").value;
    var tempFullPhoneNum = tempPhoneNum.trimEnd(); //Removes likely space at end of input box
    var validNum = false;

    if(IsValidPhoneNumber(tempFullPhoneNum)) {
        validNum = true;
    } else {
        DspPhoneNumberError(); //SELF NOTE: Why not put in IsValidPhoneNumber? Because IsValidPhoneNumber
should be concerned with whether it is valid or not. Not concerned with output in response to validity
        validNum = false;
    }

    return validNum;
}

function ValidateFavPastTime() {

    var pastTimeList = document.getElementById("frmActivityList");
    var pastTimeSelectedIndex = pastTimeList.selectedIndex;
    var isValidOption = false;

    if(pastTimeSelectedIndex === 0) { //SELF NOTE: Why not == OR !pastTimeSelectedIndex? Because that means
null is accepted and undefined
        DspActivitySelectionError();
        isValidOption = false;
    } else {
        isValidOption = true;
    }

    return isValidOption;
}


//Functions for displaying form messages
//
//

function DspFirstNameError(tempFirstName) { //SELF NOTE: Why seperate output in it's own function? Because
output may change ensures open/close principle abide by could output through console.log

    PrintNameError(tempFirstName);
    document.getElementById("frmFirstName").focus();
    document.getElementById("frmFirstName").select();
}

function DspLastNameError(tempLastName) { //SELF NOTE: Why seperate output in it's own function? Because
output may change ensures open/close principle abide by could output through console.log

    PrintNameError(tempLastName);
    document.getElementById("frmLastName").focus();
    document.getElementById("frmLastName").select();
}

function DspBirthDateFormatError() {  //SELF NOTE: Why seperate output in it's own function? Because output
may change ensures open/close principle abide by could output through console.log

    alert("Please enter Date of Birth in the correct format");
    document.getElementById("frmBirthDate").focus();
    document.getElementById("frmBirthDate").select();
}

function DspBirthDateValueError() { //SELF NOTE: Why seperate output in it's own function? Because output
may change ensures open/close principle abide by could output through console.log
    alert("Please enter a valid day, month and year");
    document.getElementById("frmBirthDate").focus();
    document.getElementById("frmBirthDate").select();
}

function DspPrefixCodeError() { //SELF NOTE: Why seperate output in it's own function? Because output may
change ensures open/close principle abide by could output through console.log

    alert("Please enter a valid prefix code");
    document.getElementById("frmPhonePrefix").focus();
    document.getElementById("frmPhonePrefix").select();
}

function DspPhoneNumberError() { //SELF NOTE: Why seperate output in it's own function? Because output may
change ensures open/close principle abide by could output through console.log

    alert("Please enter a valid phone number");
    document.getElementById("frmPhoneNumber").focus();
    document.getElementById("frmPhoneNumber").select();
}

function DspActivitySelectionError() { //SELF NOTE: Why seperate output in it's own function? Because
output may change ensures open/close principle abide by could output through console.log

    alert("Please select favourite past time");
    document.getElementById("frmActivityList").focus();
}

function PrintNameError(tempName) {

    if(IsEmpty(tempName)) {
        alert("Please enter your name");
        return;
    }

    if(!IsAllLetters(tempName)) {
        alert("Please use only letters (a-z) for names");
        return;
    }
}


//General Helper Functions
//Contains functions that can be used in different context not application specific
//

function ValidName(tempName) {
    var isValid = false; //SELF NOTE: Why assign default false value? Better to be safe then sorry in 
ensuring return is always going to be boolean
    var tempName = tempName.trimEnd();//SELF NOTE: Why not put this logic in ValidLastName/ValidFirstName? 
Regardless removing white space at end has no side effects in this function

    if(!IsEmpty(tempName) && IsAllLetters(tempName)) {
        isValid = true; //SELF NOTE: Why not also send output message? Because breaks single responsibility 
in that ValidName should only return true/false NOT also handle error message logic
    } else {
        isValid = false; //SELF NOTE: Isn't this redundant? Yes, but we want to rely on condition assigning 
false rather then default false
    }

    return isValid;
}


function IsEmpty(tempField) { //SELF NOTE: Why not have document.getElementById(...) in here? Because this 
function can be reused in many different context we don't want to tie it down to a single field or app 
logic

    var emptyField = false; //SELF NOTE: Why assign default false value? Better to be safe then sorry in 
ensuring return is always going to be boolean

    if(tempField === null || tempField === "") {
        emptyField = true;
    } else {
        emptyField = false;
    }

    return emptyField;

}

function IsAllLetters(tempField) { //SELF NOTE: Why not have document.getElementById(...) in here? Because 
this function can be reused in many different context we don't want to tie it down to a single field or app 
logic

    var letterField = false; //SELF NOTE: Why assign default false value? Better to be safe then sorry in 
ensuring return is always going to be boolean
    var onlyLetters = /^[A-Za-z]+$/; //https://stackoverflow.com/questions/23476532/check-if-string-
contains-only-letters-in-javascript

    if(tempField.match(onlyLetters)) {
        letterField = true;
    } else {
        letterField = false;
    }

    return letterField;
}

function IsAllNumbers(tempField) { //SELF NOTE: Why not have document.getElementById(...) in here? Because 
this function can be reused in many different context we don't want to tie it down to a single field or app 
logic

    var numberField = false;
    var onlyNumbers = /^\d+$/;

    if(tempField.match(onlyNumbers)) {
        numberField = true;
    } else {
        numberField = false;
    }

    return numberField;
}
```

birthDateValidator.js

```
"use strict";


//Public Interface Functions for ValidateBirthDate function (date format validation)
//Contains the entire date of birth validation logic
//

function IsValidDateFormat(tempFullDate) {

    var dateFormat = /^\d{1,2}\/\d{1,2}\/[1-9]\d*$/;
    var validDateFormat = false;

    var tempFullDate = tempFullDate.trimEnd(); //Removes trailing ending zeros
    if(tempFullDate.match(dateFormat)) {
        validDateFormat = true;
    } else {
        validDateFormat = false;
    }

    return validDateFormat;
}




//Private Implementation Functions for ValidateBirthDate function (date value validation)
//Contains the entire date of birth validation logic
//

function IsValidDate(tempFullDate) { //SELF NOTE: Why isn't this considered a public interface function?
Because this functions error checking is dependent on IsValidDateFormat() being called before hand

    var tempFullDate = document.getElementById("frmBirthDate").value;
    var dateArray = tempFullDate.split("/").map(Number); //https://www.geeksforgeeks.org/how-to-convert-
array-of-strings-to-array-of-numbers-in-javascript/

    if(!IsValidMonth(dateArray[1]) || !IsValidYear(dateArray[2])) {  //SELF NOTE: Why aren't we also
validating day? Because pointless since it depends on month
        return false;
    }

    if(IsValidDateCombination(dateArray[0], dateArray[1], dateArray[2])) {//SELF NOTE: Why not combine with
the above if? Thats short circuit evaluation and logically this function should not execute if MM/YYYY is
invalid
        return true;
    } else {
        return false;
    }
}


function IsValidDateCombination(tempDay, tempMonth, tempYear) {

    var validDate = false;
    const START_DAY = 1;


    switch(tempMonth) {
        case 1:
                validDate = WithinRange(tempDay, START_DAY, 31);
                break;
        case 2:
                var endDay = NumDaysInFeb(tempYear); //Handles leap year
                validDate = WithinRange(tempDay, START_DAY, endDay);
                break;
        case 3:
                validDate = WithinRange(tempDay, START_DAY, 31);
                break;
        case 4:
                validDate = WithinRange(tempDay, START_DAY, 30);
                break;
        case 5:
                validDate = WithinRange(tempDay, START_DAY, 31);
                break;
        case 6:
                validDate = WithinRange(tempDay, START_DAY, 30);
                break;
        case 7:
                validDate = WithinRange(tempDay, START_DAY, 31);
                break;
        case 8:
                validDate = WithinRange(tempDay, START_DAY, 31);
                break;
        case 9:
                validDate = WithinRange(tempDay, START_DAY, 30);
                break;
        case 10:
                validDate = WithinRange(tempDay, START_DAY, 31);
                break;
        case 11:
                validDate = WithinRange(tempDay, START_DAY, 30);
                break;
        case 12:
                validDate = WithinRange(tempDay, START_DAY, 31);
                break;
        default:
            validDate = false; //Default is vital because it does our error checking just in case of null
    }

    return validDate;
```

```
}


function NumDaysInFeb(tempYear) { //SELF NOTE: Why not validate incorrect year input like null? Because
this is helper internal function and entry point has been already validated

    var days = 28;
    if(IsLeapYear(tempYear)) {
        days = 29
    } else {
        days = 28;
    }

    return days;
}


function IsLeapYear(tempYear) { //SELF NOTE: Why not early return? Because guard clauses best used for non-
function logic such as null checks. Checking divisible by 4 is part of function logic and violates single
exit point

    var validLeapYear = false;

    if(((tempYear % 4 == 0) && (tempYear % 100 != 0)) || (tempYear % 400 == 0)) {
        validLeapYear = true;
    } else {
        validLeapYear = false
    }

    return validLeapYear;
}

function IsValidYear(tempYear) {  //SELF NOTE: Why not validate incorrect year input like null? Because
this is helper internal function and entry point has been already validated

    return tempYear >= 1; //SELF NOTE: Why not use if statement then return? Because it's nothing else is
done but this simple check
}


function IsValidMonth(tempMonth) { //SELF NOTE: Why not validate incorrect MONTH input like null? Because
this is helper internal function and entry point has been already validated

    return ((tempMonth >= 1) && (tempMonth <= 12)) //SELF NOTE: Why not use if statement then return?
Because it's nothing else is done but this simple check
}

function WithinRange(tempNumCheck, tempStartNum, tempEndNum) { //SELF NOTE: Why not validate input? Because
this is helper internal function and entry point has been already validated

    return ((tempNumCheck >= tempStartNum) && (tempNumCheck <= tempEndNum))
}
```

phoneNumberValidator.js

```
"use strict";


//Private Implementation Functions for ValidatePhoneNumber function (number validation)
//Contains the entire phone number validation logic
//

function IsValidPhoneNumber(tempPhoneNumber) {

    const MAX_DIGITS = 12;
    var currentPhoneNumDigits = 0;
    if(!IsAllNumbers(tempPhoneNumber)) {
        return false; //SELF NOTE: Why not follow through with single return? Because we shouldn't proceed
if the input is not numbers whether number or not influences the below logic
    }

    currentPhoneNumDigits = tempPhoneNumber.length;

    if(currentPhoneNumDigits <= 12) { //SELF NOTE: Isn't this risky since null would be true? Yes, but
IsAllNumbers logic is strict in stopping this
        return true;
    }

    return false; //SELF NOTE: Why return true rather then boolean assigned true? Because follows the guard
clause pattern
}


//Private Implementation Functions for ValidatePhoneNumber function (prefix number validation)
//Contains the entire phone number validation logic
//

function IsValidPrefixFormat(tempFullPrefix) { //SELF NOTE: Why not put logic inside ValidatePrefixCode()?
Because this promotes abstraction, also open/closed principle and function likely to be extended with more
prefix formats

    var currentPrefixFormat = /^[+]{1}[1-9]\d{0,5}$/;
    var validPrefixFormat = false;

    if(tempFullPrefix.match(currentPrefixFormat)) {
        validPrefixFormat = true;
    } else {
        validPrefixFormat = false;
    }

    return validPrefixFormat;
}
```

AddUserFormHome.html

```html
<!DOCTYPE html>
<html lang="en">
        <head>
                <title>User Details Form</title>
                <script src="addUserFormValidator.js"></script>
                <script src="phoneNumberValidator.js"></script>
                <script src="birthDateValidator.js"></script>
        <head>
        <body>
                <h1><u>Add User Details:</u></h1>
                <p></p>
                <form action="http://www.it.murdoch.edu.au/cgi-bin/reply1.pl" method="post"
onsubmit="return ValidateHomeForm(event)">
                        <label for="frmFirstName">First Name: </label><br />
                        <input type="text" name="frmFirstName" id="frmFirstName"
onchange="ValidFirstName()" /><br />
                        <p></p>
                        <label for="frmLastName">Last Name: </label><br />
                        <input type="text" name="frmLastName" id="frmLastName" onchange="ValidLastName()"
/>
                        <p></p>
                        <label for="frmBirthDate">Date of birth:</label><br />
                        <input type="text" id="frmBirthDate" name="frmBirthDate" placeholder="DD/MM/YYYY"
onchange="ValidateBirthDate()" />

                        <fieldset style="border:none;padding: 0px;margin:0px">
                                <p>Phone number:</p>
                                <input type="text" id="frmPhonePrefix" name="frmFullPhoneNum"
placeholder="+61" size="2.5" onchange="ValidatePrefixCode()" maxlength="6" />
                                <input type="text" id="frmPhoneNumber" name="frmFullPhoneNum"
placeholder="0123456789" onchange="ValidatePhoneNumber()" />
                        </fieldset>

                        <p>Favourite Past Time:</p>
                        <select name="activities" id="frmActivityList" onchange="ValidateFavPastTime()">
                                <option value="">--Please choose an option--</option>
                                <option value="Surfing the Web">Surfing the Web</option>
                                <option value="Playing sport">Playing sport</option>
                                <option value="Listening to Music">Listening to Music</option>
                                <option value="Watching TV">Watching TV</option>
                                <option value="Playing Games">Playing Games</option>
                                <option value="Community Service">Community Service</option>
                                <option value="Daydreaming">Daydreaming</option>
                                <option value="Reading">Reading</option>
                                <option value="Meditation">Meditation</option>
                        </select>

                        <p><input type="submit" value="Submit"/></p>
                </form>
        </body>
</html>
```

ICT373– Assignment 1 | Semester 2, 2023

Magazine Service Program Document

Date of Submission: 7/4/2023
Author: Jin Cherng Chong

Java Files: AssociateCustomer, AssociateDatabase, Customer, Magazine, PayingCustomer, PaymentMethod, Supplement, SupplementDatabase, Client, MagazineService, EmailPrinter, IPayForAssociates

Question 2

Requirements/Specifications

Assumptions:

- End of month email includes cost of magazine for each customer that PayingCustomer is paying
- Money is in AUD
- Money had been rounded to two decimal places. However, further rounding i.e 7.97 or 9.99 has not been done since legally electronic transaction does not require them to be rounded. Online banking applications will show 7.98 and be able to deduct 0.77 cents. Refer-
  - https://www.commerce.wa.gov.au/consumer-protection/rounding
  - https://www.justweb.com.au/law-articles/rounding-up.html
- End of month is always 4 weeks i.e x 4
- In order for two supplement objects to be the same both the name and cost must be the same
  - For example, one supplement object has name- Times and cost: 8.00 while another supplement object has name TiMeS and cost 7.99. While both supplements have the same name the cost is different in that 8.00 is not 7.99 thus will be treated as not equal
- A paying customer will never equal an associate customer regardless if they have the same name, email address and list of supplements interested

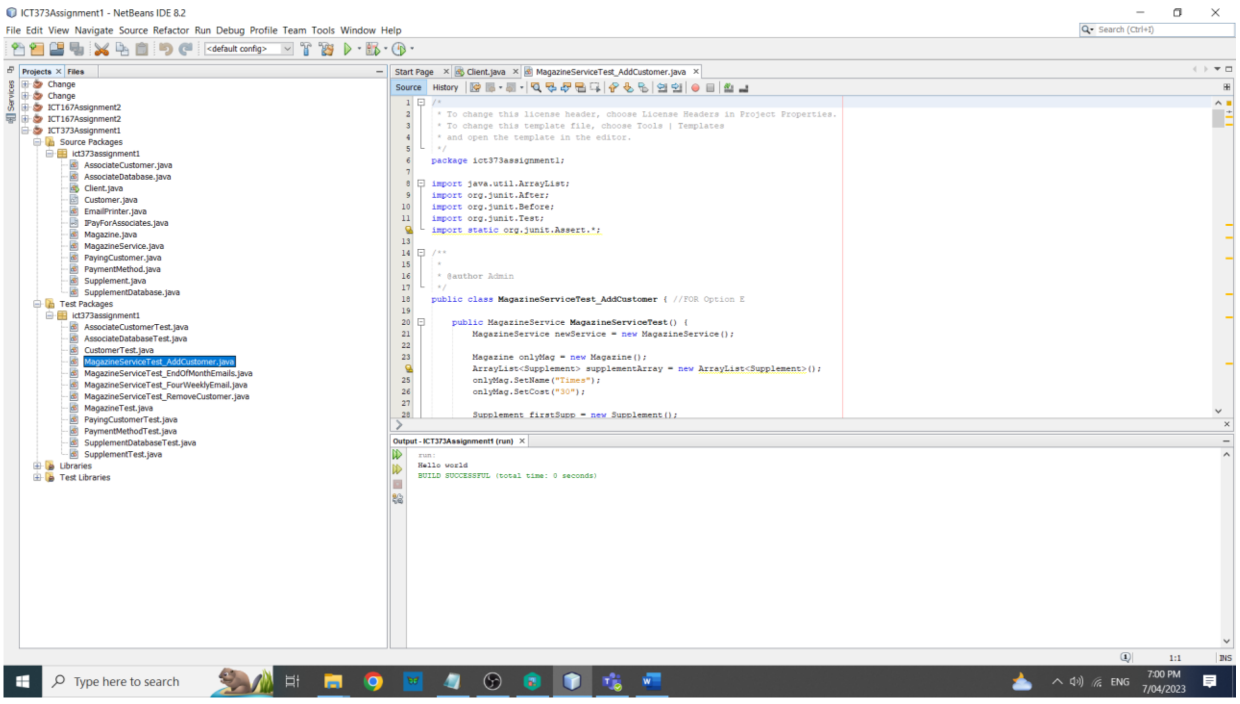
The program seeks to make it easier to manage a personalised magazine service. This program maintains a list of customers paying into the magazine service and paying for supplements offered and subscribed to. A customer in the magazine service must be paid by some form. The customer can pay for the services themselves this is classified as a paying customer. Or the customer must have another paying customer pay for them. The program provides the option of generating weekly emails, generating end of month email, adding a new customer, or removing existing customer.

User guide

To test out the magazine service program. Four files have been provided illustrate the programs functionality-

1. MagazineServiceTest_AddCustomer.java
2. MagazineServiceTest_RemoveCustomer.java,
3. MagazineServiceTest_EndOfMonthEmails.java
4. MagazineServiceTest_FourWeeklyEmail.java

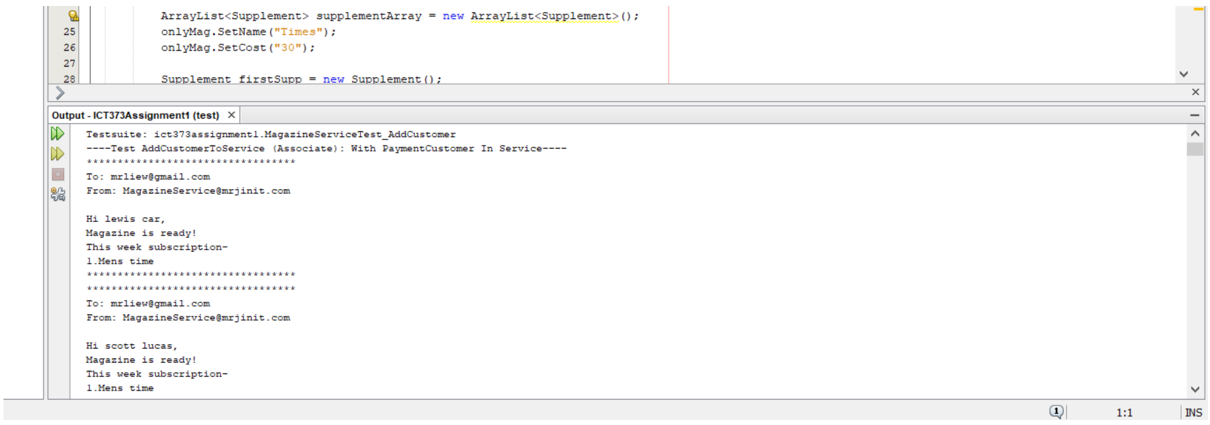
Double click (left click) on one of the files to open as illustrated



```java
/*
 * 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.ArrayList;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author Admin
 */
public class MagazineServiceTest_AddCustomer { //FOR Option E

    public MagazineService MagazineServiceTest() {
        MagazineService newService = new MagazineService();

        Magazine onlyMag = new Magazine();
        ArrayList<Supplement> supplementArray = new ArrayList<Supplement>();
        onlyMag.SetName("Times");
        onlyMag.SetCost("30");

        Supplement firstSupp = new Supplement();
```

```
run:
Hello world
BUILD SUCCESSFUL (total time: 0 seconds)
```

Then in the same location you double clicked, right click on the file and click run file. That will show the console output outlining the program running



```
        ArrayList<Supplement> supplementArray = new ArrayList<Supplement>();
        onlyMag.SetName("Times");
        onlyMag.SetCost("30");

        Supplement firstSupp = new Supplement();
```

Output - ICT373Assignment1 (test)

```
Testsuite: ict373assignment1.MagazineServiceTest_AddCustomer
----Test AddCustomerToService (Associate): With PaymentCustomer In Service----
*********************************
To: mrliew@gmail.com
From: MagazineService@mrjinit.com

Hi lewis car,
Magazine is ready!
This week subscription-
1.Mens time
*********************************
*********************************
To: mrliew@gmail.com
From: MagazineService@mrjinit.com

Hi scott lucas,
Magazine is ready!
This week subscription-
1.Mens time
```

1:1 INS

Structure/Design

Design description: [10]

There were several design decisions that were made when developing the application. In all decisions I've tried to ensure SOLID principles were adhered to.

Keeping Customer Information

In my implementation of the "manage an online weekly personalised magazine service" program a customer has to be paying for the service. However, a customer (paying customer) can pay for the service themselves or have another customer pay for them. A customer who doesn't pay for the service in any form serves no purpose in my program. Any information in relation to them is not kept. There are several reasons for this decision.

Firstly, this decision was made to ensure the program is as efficient as possible. Let's say the program has 2 million customers and only two customers are paying for the service. Performing operations for a specific customer will require searching through large data set. Hence, removing customers not paying in any form will make data set minimal as possible and program more efficient ensuring program future proofing.

Secondly, the decision of not keeping customer's details for customers not paying in any form was made with data security being top of mind. In the event, where all the customer information is leaked or breached only a minimal number of customers would be affected. This is supported by the decision of not keeping customer's details of customers not paying in any form.

It could be argued that keeping their information stops the need to add the customer information all over again when the customer decides to use the service again in the future. Although valid, data security, program efficiency and reducing memory usage were prioritised.

Customer, Associate Customer, Paying Customer

In this assignment there were several roles that was described. The assignment indicated a need for a customer, associate customer, and paying customer. All three had unique attributes and behaviours. For example, a paying customer had a payment method, customer had a name and a paying customer could pay for associate customer. Hence, given customer, associate customer and paying customer had behaviours and fields a class can be used to encapsulate each role. This is to adhere to object orientation design.

The next decision was whether there was a relationship between the three potential classes. From the assignment information it is stated that an associate customer "is a" customer. The phrase "is a" is distinct because it commonly describes an inheritance relationship. An inheritance relationship is described as one class having all the characteristics of another class but one class has additional unique characteristics. In this case, it makes logical sense for associate customer to have the characteristics of the customer class such as associate customer having a name, email, and supplements interested. Similarly, even though paying customer is not described as 'is a' customer it makes logical sense for paying customer to have all the characteristics of a customer such as having a name, email, and supplements interested. Furthermore, both paying customer and associate customer has additional unique characteristics such as paying customer having a payment method and associate customer is not paying for their own subscription. So, inheritance relationship so far seems logical.

In order, for an inheritance relationship between customer and paying customer and also customer and associate customer to make the most sense Liskov substitution principle i.e the L in SOLID must not be violated. The principal outlines three rules for ensuring inheritance relationship is the best choice. Firstly, a child class must use all the data members of the parent class. Secondly, a child class must require all the methods of the parent class. Thirdly, the 'is a' must make logical behaviour sense. Associate customer and paying customer is a customer makes logical English sense. Secondly, both associate customer and paying customer use all the customer data members (name, email etc.). Thirdly, both require all the customer classes method. Hence, Inheritance adhering to Liskov substitution principle further supports inheritance relationship.

The next decision made was whether customer class itself is an abstract class or a concrete superclass. Customer class could not be an interface because an inheritance relationship was needed. Also, customer required instance variables such as name which an interface can not support. One main reason why an abstract class is needed is that it allows for a method without implementation (abstract method). In this context, there is no abstract methods required in the customer class. Therefore, customer class being an abstract class does seem not suitable. However, another key reason for a class to be an abstract class is when the class makes no logical sense to be instantiated. For example- an animal class with dog class subclass and fish subclass. The animal class is an abstract concept and it doesn’t make logical sense for an animal to be running around or instantiated. For assignment 1, the class customer does seem like an abstract concept. There is no use case for a customer needing to be instantiated given the assignment specification. Especially given my program does not store any information of customers not paying for the service in any form as explained above. Also, not having to deal with customer object reduces the difficulty of developing the program. The run time logic of converting a customer object to an associate customer object or paying customer object is now of no concern. Thus, the customer class is an abstract class because there is no logical reasons for a base customer to be instantiated

Façade design pattern

Façade design pattern was used and implemented in several classes in my program. MagazineService.java, SupplementDatabase.java, and AssociateDatabase.java adopts a façade design pattern. A façade design pattern provides a simple interface for complex subsystem. The MagazineService class needs to deal with many different logical complexities. For example-MagazineService needs to ensure an associate customer has a valid paying customer and allow for the addition and deletion of customers in the MagazineService. A façade pattern in this case has several benefits.

Firstly, by nature facade promotes abstraction and code reuse. Abstraction is when implementation details is hidden from the client. MagazineService provides an AddCustomer and DeleteCustomer method. The logic of determining whether a customer should be added to the service involves searching the existing customers data set, seeing if it’s a duplicate, validating each attribute of customer, and inserting to data structure. This entire logic is hidden from client and client only needs to be concerned with what customer to add or remove. Also, since the addition and deletion logic are encapsulated in their respective methods code reuse is promoted.

Secondly, it promotes single responsibility principle adherence. Single responsibility principle states how a class should have only one responsibility. In the assignment 1 specification it mentions how a paying customer will have a list of associate customers whom they pay for. A paying customer should be responsible for their name, address, and email. However, the paying customer also having to deal with the logic associated with maintaining their list of associate customers violates single responsibility principle. Since, to maintain a valid list of associate customers it involves many operations such as ensuring duplicate associate customer are not added, removing associate customers and inserting and deleting from the data structure. Thus, façade encapsulates the logic for maintaining the associate customer list for each paying customer leading to adherence of single responsibility principle

In the future MagazineService.java may be more suitable to adopting singleton pattern design. A singleton ensures a class has only one instance. MagazineService in the future would likely only allow one instance of the class.

Associate customer & Paying customer

Associate customer and Paying customer relationship have two key characteristics.

1. An associate customer is being paid for by an existing paying customer i.e paying customer currently in customerList (magazineService)
2. A paying customer can only pay for existing associate customer i.e associate customer currently in customerList (magazineService)

There are several characteristics and operations that support this relationship-

Firstly, in order to create an instance of an associate customer the associate customer constructor requires a paying customer parameter. This is to ensure an associate customer will always have a paying customer.

Secondly, MagazineService encapsulates a list of customers in the service. When an associate customer is added the class will determine whether associate customer to be added is being paid for by an existing paying customer i.e paying customer currently in customerList (magazineService). If associate customer is being paid by an existing paying customer the associate customer will be added to the magazineService customerList. When a new paying customer is added to the MagazineService all associate customers which the new paying customer supposedly is paying for is all removed.

In essence, the addition of an associate customer to the magazine service will validate whether associate customer to be added is being paid for by an existing paying customer and add associate customer to the paying customer list of associates. Thereby, ensuring paying customer is paying for existing associate customers.

Similarly, when an associate customer is removed from the MagazineService the MagazineService also removes the associate customer from the paying customer list of associates. When a paying customer is removed from the MagazineService all the associate customers being paid for by the removed paying customer is also removed.

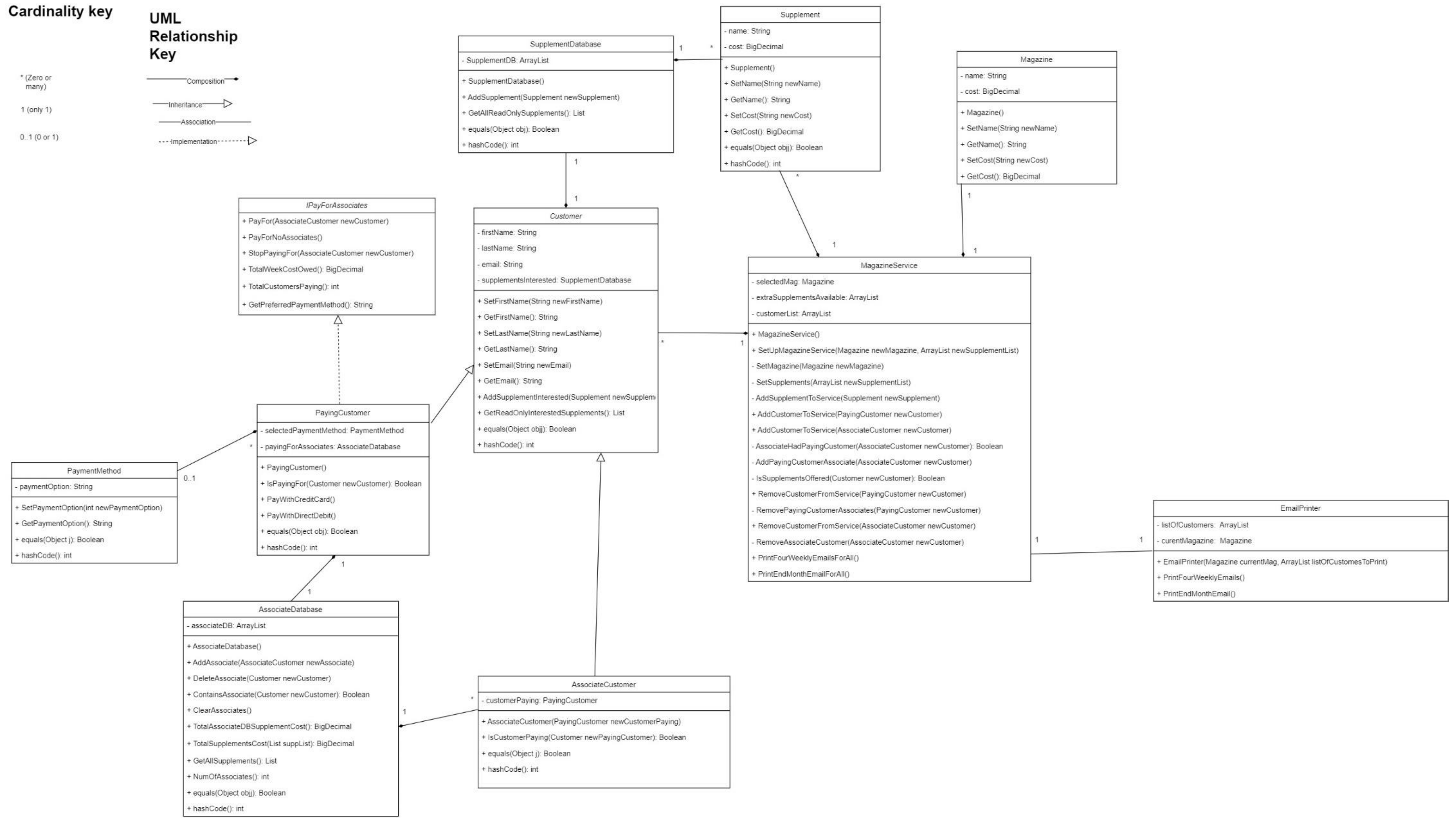
Adding/Removing Customer

The low-level implementation of the method adding customer and deleting customer was difficult. Initially I wanted a single method for addition of any customers and a single method for removing any customer. The initial thought was to reduce code duplication. However, during implementation there was a pitfall that made me change approach. When a client adds a customer to the service it required run time type identification via instance of. That’s because with the addition of associate customers it required the class to also validate whether there is an existing paying customer in the service that can pay for the associate customer to be added. Run time type identification (instance of) is code smells. It violates open-closed principle. If an additional type of customer was created in the future it would require the code to be with instance of to be edited to handle the additional type. The approach of having overloaded AddCustomer and DeleteCustomer was selected to negate the need to identify type at run time via instance of. Common operations were extracted out to their own methods so it could be reused in each overloaded add customer and delete customer method. Additionally overloaded methods is polymorphism utilisation.

1. Abstract class
2. Downcasting
3. AddCustomer(….,). The single biggest issue is that it would require instanceOf. This is code smells. Thus I split them up on their own individual functions

4. Logic- Delete customer etc
5. Splitting AssociateDatabase/CustomerDatabase
6. Lack of respondes/error checking
7. Override equals

UML: [10]


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)
+ GetAllReadOnlySupplements(): List
+ equals(Object obj): Boolean
+ hashCode(): int
Supplement
- name: String
- cost: BigDecimal
+ Supplement()
+ SetName(String newName)
+ GetName(): String
+ SetCost(String newCost)
+ GetCost(): BigDecimal
+ equals(Object obj): Boolean
+ hashCode(): int
Magazine
- name: String
- cost: BigDecimal
+ Magazine()
+ SetName(String newName)
+ GetName(): String
+ SetCost(String newCost)
+ GetCost(): BigDecimal
IPayForAssociates
+ PayFor(AssociateCustomer newCustomer)
+ PayForNoAssociates()
+ StopPayingFor(AssociateCustomer newCustomer)
+ TotalWeekCostOwed(): BigDecimal
+ TotalCustomersPaying(): int
+ GetPreferredPaymentMethod(): String
Customer
- firstName: String
- lastName: String
- email: String
- supplementsInterested: SupplementDatabase
+ SetFirstName(String newFirstName)
+ GetFirstName(): String
+ SetLastName(String newLastName)
+ GetLastName(): String
+ SetEmail(String newEmail)
+ GetEmail(): String
+ AddSupplementInterested(Supplement newSupplem
+ GetReadOnlyInterestedSupplements(): List
+ equals(Object obj): Boolean
+ hashCode(): int
MagazineService
- selectedMag: Magazine
- extraSupplementsAvailable: ArrayList
- customerList: ArrayList
+ MagazineService()
+ SetUpMagazineService(Magazine newMagazine, ArrayList newSupplementList)
- SetMagazine(Magazine newMagazine)
- SetSupplements(ArrayList newSupplementList)
- AddSupplementToService(Supplement newSupplement)
+ AddCustomerToService(PayingCustomer newCustomer)
+ AddCustomerToService(AssociateCustomer newCustomer)
- AssociateHadPayingCustomer(AssociateCustomer newCustomer): Boolean
- AddPayingCustomerAssociate(AssociateCustomer newCustomer)
- IsSupplementsOffered(Customer newCustomer): Boolean
+ RemoveCustomerFromService(PayingCustomer newCustomer)
- RemovePayingCustomerAssociates(PayingCustomer newCustomer)
+ RemoveCustomerFromService(AssociateCustomer newCustomer)
- RemoveAssociateCustomer(AssociateCustomer newCustomer)
+ PrintFourWeeklyEmailsForAll()
+ PrintEndMonthEmailForAll()
PayingCustomer
- selectedPaymentMethod: PaymentMethod
- payingForAssociates: AssociateDatabase
+ PayingCustomer()
+ IsPayingFor(Customer newCustomer): Boolean
+ PayWithCreditCard()
+ PayWithDirectDebit()
+ equals(Object obj): Boolean
+ hashCode(): int
PaymentMethod
- paymentOption: String
+ SetPaymentOption(int newPaymentOption)
+ GetPaymentOption(): String
+ equals(Object j): Boolean
+ hashCode(): int
EmailPrinter
- listOfCustomers: ArrayList
- curentMagazine: Magazine
+ EmailPrinter(Magazine currentMag, ArrayList listOfCustomesToPrint)
+ PrintFourWeeklyEmails()
+ PrintEndMonthEmail()
AssociateDatabase
- associateDB: ArrayList
+ AssociateDatabase()
+ AddAssociate(AssociateCustomer newAssociate)
+ DeleteAssociate(Customer newCustomer)
+ ContainsAssociate(Customer newCustomer): Boolean
+ ClearAssociates()
+ TotalAssociateDBSupplementCost(): BigDecimal
+ TotalSupplementsCost(List supplList): BigDecimal
+ GetAllSupplements(): List
+ NumOfAssociates(): int
+ equals(Object obj): Boolean
+ hashCode(): int
AssociateCustomer
- customerPaying: PayingCustomer
+ AssociateCustomer(PayingCustomer newCustomerPaying)
+ IsCustomerPaying(Customer newPayingCustomer): Boolean
+ equals(Object j): Boolean
+ hashCode(): int
1
*
1
1
*
1
1
1
*
1
1
1
0..1
*
1
1
1
*

Testing: [20]

Feature testing: Add a new customer to magazine service

Test File: MagazineServiceTest_AddCustomer.java

Evidence: https://www.youtube.com/playlist?list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC

Test #	Test objective(s)	Test step(s)	Expected results	Pass /Fail	Evidence
1	Add PayingCustomer with their supplement of interest not offered by MagazineService	• Create MagazineService • Create PayingCustomer • AddSupplementInterested for PayingCustomer • Add PayingCustomer to MagazineService • Print any of PayingCustomer email to verify	PayingCustomer is not added to MagazineService. He will have no emails displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1
2	Add PayingCustomer with their supplement(s) of interest being offered by MagazineService	• Create MagazineService • Add an offered supplement to MagazineService • Create PayingCustomer • AddSupplementInterested for PayingCustomer • Add PayingCustomer to MagazineService • Print any of PayingCustomer email to verify	PayingCustomer is added to MagazineService. He will have an email displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1
3	Add AssociateCustomer with their supplement of interest not offered by MagazineService	• Create MagazineService • Create AssociateCustomer • AddSupplementInterested for AssociateCustomer • Add AssociateCustomer to MagazineService • Print any of AssociateCustomer email to verify	AssociateCustomer is not added to MagazineService. He will have no emails displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1
4	Add AssociateCustomer with their supplement(s) of interest being offered by MagazineService	• Create MagazineService • Add an offered supplement to MagazineService • Create AssociateCustomer • AddSupplementInterested for AssociateCustomer • Add AssociateCustomer to MagazineService • Print any of AssociateCustomer email to verify	AssociateCustomer is added to MagazineService. He will have an email displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1
5	Add AssociateCustomer without his PayingCustomer in MagazineService	• Create MagazineService • Create AssociateCustomer with identified PayingCustomer • Add AssociateCustomer to MagazineService • Print any of AssociateCustomer email to verify	AssociateCustomer is not added to MagazineService. He will have no emails displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1

6	Add AssociateCustomer with his PayingCustomer in MagazineService	• Create MagazineService • Create PayingCustomer • Create AssociateCustomer with identified PayingCustomer • Add PayingCustomer to MagazineService • Add AssociateCustomer to MagazineService • Print any of AssociateCustomer email to verify	AssociateCustomer is added to MagazineService. He will have an email displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1
7	Add Customer without full name	• Create MagaineService • Create AssociateCustomer • Create AssociateCustomer with firstName only • Add AssociateCustomer to MagazineService • Print any of AssociateCustomer email to verify	Customer is not added to MagazineService. He will have no emails displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1
8	Add Customer without an email	• Create MagazineService • Create PayingCustomer • Create PayingCustomer without email • Add PayingCustomer to MagazineService • Print any of PayingCustomer email to verify	Customer is not added to MagazineService. He will have no emails displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1
9	Add the same AssociateCustomer more than once	• Create MagazineService • Create AssociateCustomer • Add AssociateCustomer to MagazineService • Add AssociateCustomer to MagazineService • Print any of AssociateCustomer email to verify	AssociateCustomer is not added to MagazineService. Duplicate AssociateCustomer is not stored in service.	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1
10	Add more than one valid AssociateCustomer	• Create MagazineService • Create AssociateCustomer • Create AssociateCustomer • Add AssociateCustomer to MagazineService • Add AssociateCustomer to MagazineService with all the same details as previous added AssociateCustomer except different firstName • Print any of AssociateCustomer email to verify	All valid AssociateCustomer is added to the MagazineService. All valid AssociateCustomer added will have an email (of theirs) displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1
11	Add more than one valid PayingCustomer	• Create MagazineService • Create PayingCustomer • Create PayingCustomer • Add PayingCustomer to MagazineService • Add PayingCustomer to MagazineService • Print any of PayingCustomer email to verify	All valid PayingCustomer is added to the MagazineService. All valid PayingCustomer added will have an email (of theirs) displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1
12	Add more than one valid customer of different types	• Create MagazineService • Create PayingCustomer • Create AssociateCustomer • Create AssociateCustomer • Add AssociateCustomer to MagazineService • Add AssociateCustomer to MagazineService	All valid customers (PayingCustomer and AssociateCustomer types) are added to the MagazineService. All valid customers added will have an email (of theirs) displayed	Pass	https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1

		• Add PayingCustomer to MagazineService • Print any of Customer email to verify			
13	Add PayingCustomer with their supplement of interest having the same name but different price compared to the one offered by MagazineService	• Create MagazineService • Create PayingCustomer • AddSupplementInterested for PayingCustomer ○ Price $9.99 offered $10 in magazineService • Add PayingCustomer to MagazineService • Print any of PayingCustomer email to verify	PayingCustomer is not added to MagazineService. He will have no emails displayed		https://www.youtube.com/watch?v=3_Xy_JZa9tA&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=1

Feature testing: Remove an existing customer from the magazine service

Test File: MagazineServiceTest_RemoveCustomer.java

Evidence: https://www.youtube.com/playlist?list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC

Test #	Test objective(s)	Test step(s)	Expected results	Pass /Fail	Evidence
1	Remove PayingCustomer that has no associates	• Create MagazineService • Create PayingCustomer • Add PayingCustomer to MagazineService • Remove PayingCustomer from MagazineService • Print any of PayingCustomer email to verify	PayingCustomer is removed from MagazineService. He will have no emails displayed	Pass	https://www.youtube.com/watch?v=vtz4X3o5OHQ&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=2
2	Remove a non-existing PayingCustomer	• Create MagazineService • Create PayingCustomer • Add PayingCustomer to MagazineService • Remove non-existing PayingCustomer from MagazineService • Print any of PayingCustomer email to verify	PayingCustomer is not removed from MagazineService	Pass	https://www.youtube.com/watch?v=vtz4X3o5OHQ&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=2
3	Remove PayingCustomer that has one associate	• Create MagazineService • Create PayingCustomer • Create AssociateCustomer belonging to PayingCustomer • Add PayingCustomer to MagazineService • Add AssociateCustomer to MagazineService • Print any of Customer email to verify • Remove PayingCustomer from MagazineService • Print any of Customer email to verify	The first print (Before removal) should display (of theirs) all valid customers (PayingCustomer and AssociateCustomer types) email The second print will display no emails	Pass	https://www.youtube.com/watch?v=vtz4X3o5OHQ&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=2

4	Remove PayingCustomer that has more than one associate	• Create MagazineService • Create PayingCustomer • Create AssociateCustomer belonging to PayingCustomer • Create AssociateCustomer belonging to PayingCustomer • Add PayingCustomer to MagazineService • Add AssociateCustomer to MagazineService • Add AssociateCustomer to MagazineService • Print any of Customer email to verify • Remove PayingCustomer from MagazineService • Print any of Customer email to verify	The first print (Before removal) should display (of theirs) all valid customers (PayingCustomer and AssociateCustomer types) email The second print will display no emails		https://www.youtube.com/watch?v=vtz4X3o5OHQ&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=2
5	Remove AssociateCustomer	• Create MagazineService • Create AssociateCustomer • Add AssociateCustomer to MagazineService • Remove AssociateCustomer from MagazineService • Print any of Customer email to verify	AssociateCustomer is removed from MagazineService. He will have no emails displayed	Pass	https://www.youtube.com/watch?v=vtz4X3o5OHQ&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=2

Feature testing: Print out the text of all the emails for all customers for four weeks of magazine

Test File: MagazineServiceTest_FourWeeklyEmail.java

Evidence: https://www.youtube.com/playlist?list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC

Test #	Test objective(s)	Test step(s)	Expected results	Pass /Fail	Evidence
1	Print out the text of all the emails for all customers for four weeks of magazine given: 1. PayingCustomer has two associates 2. All three customers have one supplement each	• Create MagazineService • Create PayingCustomer • Create AssociateCustomer belonging to PayingCustomer • Create AssociateCustomer belonging to PayingCustomer • Add PayingCustomer to MagazineService • Add AssociateCustomer to MagazineService • Add AssociateCustomer to MagazineService • Print Four Weekly Email for all	4x same weekly email associated with PayingCustomer details only 4x same weekly email associated with AssociateCustomer details only 4x same weekly email associated with second AssociateCustomer details only	Pass	https://www.youtube.com/watch?v=hfkW7QVIYuE&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=3
2	Print out the text of all the emails for all customers for four weeks of magazine given: 1. Three customers	• Create MagazineService • Create PayingCustomer • Create AssociateCustomer belonging to PayingCustomer • Create AssociateCustomer belonging to PayingCustomer	4x same weekly email associated with PayingCustomer details only 4x same weekly email associated with	Pass	https://www.youtube.com/watch?v=hfkW7QVIYuE&list=PLUktbenWQI5gDoRzDnhuNc

	2. All three customers have one supplement each 3. The three supplements are not the same	• Set one of the customers with a different supplement of interest then another customer • Add PayingCustomer to MagazineService • Add AssociateCustomer to MagazineService • Add AssociateCustomer to MagazineService • Print Four Weekly Email for all	AssociateCustomer details only 4x same weekly email associated with second AssociateCustomer details only 1 customer will have a different supplement displayed for the entire four-week period		Yt3JbxL5elC&index=3
3	Print out the text of all the emails for all customers for four weeks of magazine given: 1. Three customers 2. Some customers have more than 1 supplement of interest	• Create MagazineService • Create PayingCustomer • Create AssociateCustomer belonging to PayingCustomer • Create AssociateCustomer belonging to PayingCustomer • Assign PayingCustomer and one AssociateCustomer with two supplements • Assign the other AssociateCustomer with one supplement • Add PayingCustomer to MagazineService • Add AssociateCustomer to MagazineService • Add AssociateCustomer to MagazineService • Print Four Weekly Email for all	4x same weekly email associated with PayingCustomer details only. Emails display two supplements subscribed to 4x same weekly email associated with AssociateCustomer details only. Emails display two supplements subscribed to 4x same weekly email associated with second AssociateCustomer details only	Pass	https://www.youtube.com/watch?v=hfkW7QVIYuE&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=3
4	Print out the text of all the emails for all customers for four weeks of magazine given: 1. Three customers 2. Customer has three supplements	• Create MagazineService • Create 3x Customers • Assign Customer(s) with three supplements • Add all customers to MagazineService • Print Four Weekly Email for all	4x same weekly email associated with each specific customer details. One customer's emails will display three supplements subscribed to	Pass	https://www.youtube.com/watch?v=hfkW7QVIYuE&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=3
5	Print out the text of all the emails for all customers for four weeks of magazine given: 1. Three customers 2. Remove AssociateCustomer	• Create MagazineService • Create 3x Customers • Add all customers to MagazineService • Print Four Weekly Email for all • Remove AssociateCustomer from MagazineService • Print Four Weekly Email for all	The first print (Before removal) should display in total 12 emails. Each customer should have 4x same email relevant to their customer details The second print (after removal) will display total 8 emails. The 8 emails does not include any emails from removed AssociateCustomer	Pass	https://www.youtube.com/watch?v=hfkW7QVIYuE&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=3
6	Print out the text of all the emails for all customers for four weeks of magazine given: 1. Three customers consisting of one PayingCustomer has two associates 2. Remove PayingCustomer	• Create MagazineService • Create PayingCustomer • Create AssociateCustomer belonging to PayingCustomer • Create AssociateCustomer belonging to PayingCustomer • Add all customers to MagazineService • Print Four Weekly Email for all • Remove AssociateCustomer from MagazineService	The first print (Before removal) should display in total 12 emails. Each customer should have 4x same email relevant to their customer details The second print (after PayingCustomer removal) will display no emails	Pass	https://www.youtube.com/watch?v=hfkW7QVIYuE&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=3

		• Print Four Weekly Email for all			

Feature testing: Print out the text for the end of month emails for paying customers

Test File: MagazineServiceTest_EndOfMonthEmails.java

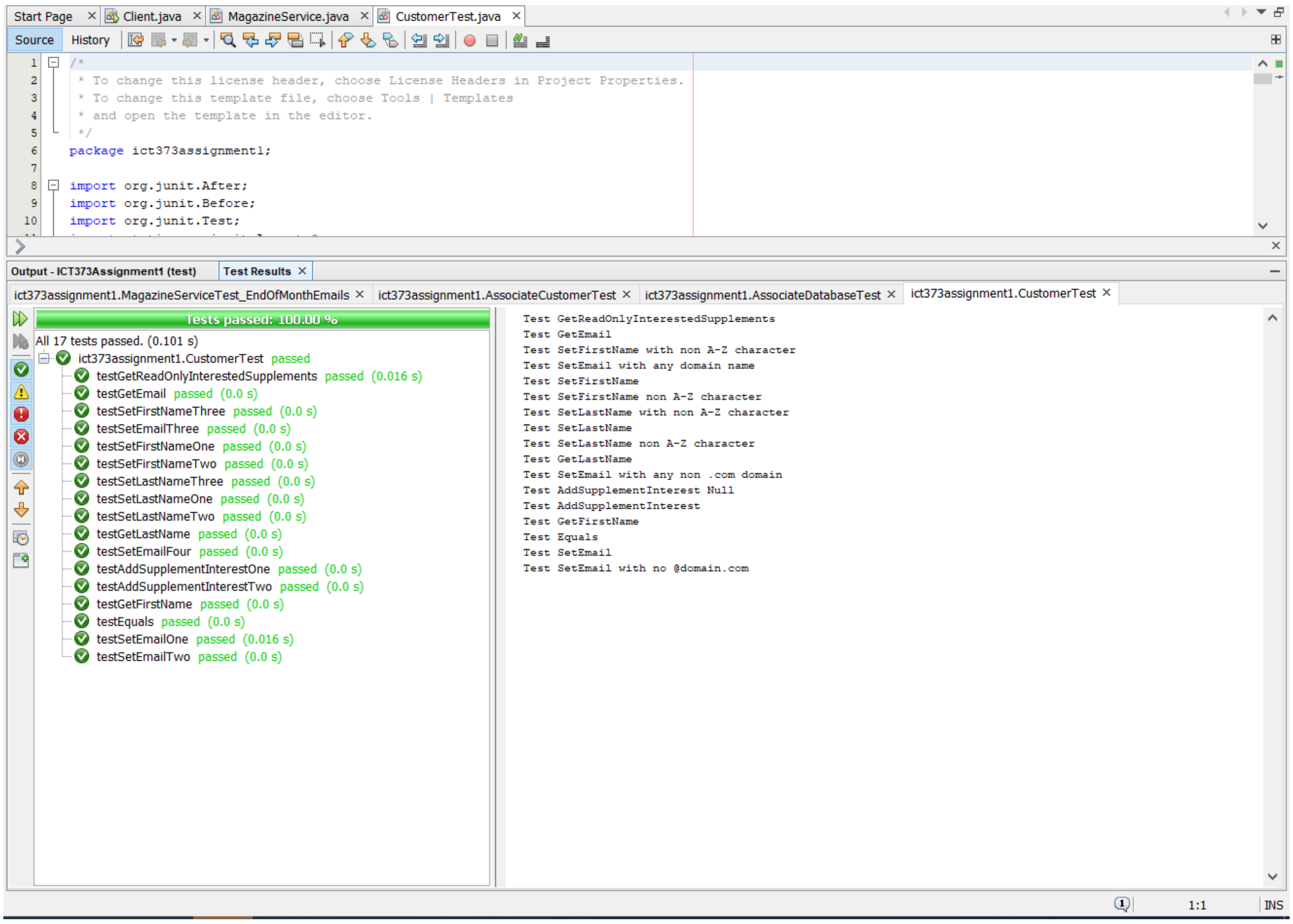
Evidence: https://www.youtube.com/playlist?list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC

Test #	Test objective(s)	Test step(s)	Expected results	Pass /Fail	Evidence
1	Print out the text for the end of month emails for paying customers given: 1. One PayingCustomer has many associates	• Create MagazineService • Set magazine (MagazineService)- cost $30 and called Times • Create PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Men time • Pay by credit card • Create AssociateCustomer belonging to PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Men time • Create AssociateCustomer belonging to PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Men time • Add all customers to MagazineService • Print End Month Email	1x email end of month email associated with PayingCustomer. Containing atleast • 3x Supplements.with name displayed Men time. Each supplement will end up costing (7 x 4) $28. Cost for each supplement is displayed • Taken from credit card displayed • Total cost of magazine- ($30 x 4 weeks) x 3 customers = 360 • Total charge- (28 + 28 + 28 + 360) $444	Pass	https://www.youtube.com/watch?v=hmSimdnFcnE&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=4
2	Print out the text for the end of month emails for paying customers given: 1. One PayingCustomer without associates 2. One PayingCustomer with associate	• Create MagazineService • Set magazine (MagazineService)- cost $30 and called Times • Create PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Mens time • Pay by direct debit • Create AssociateCustomer belonging to PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Mens time • Create PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Mens time • Pay by direct debit • Add all customers to MagazineService • Print End Month Email	1x email end of month associated with PayingCustomer that has associate. Containing email has atleast- • 2x Supplements called Mens time. Each supplement cost (7 x 4) $28 • Taken from direct debit • Total magazine cost- ($30 x 4) x 2 = $240 • Total charge - $240 + (28 + 28) = 296 1x email end of month associated with PayingCustomer without	Pass	https://www.youtube.com/watch?v=hmSimdnFcnE&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=4

			associate. Containing email has atleast- • 1x Supplements called Mens time for $28 • Taken from direct debit • Total magazine cost- $120 • Total charge- 148		
3	Print out the text for the end of month emails for paying customers given: 1. One PayingCustomer without associates 2. One PayingCustomer with associate 3. Remove PayingCustomer with associate	• Set magazine (MagazineService)- cost $30 and called Times • Create PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Mens time • Pay by direct debit • Create AssociateCustomer belonging to PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Mens time • Create PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Mens time • Pay by direct debit • Add all customers to MagazineService • Print End Month Email • Remove PayingCustomer from MagazineService • Print End Month Email	The first print (Before removal) should display 2x payingCustomer end of month email The second print (After removal) should display PayingCustomer without associate. Containing email has atleast- • 1x Supplements called Mens time for $28 • Taken from direct debit • Total magazine cost- $120 • Total charge- 148	Pass	https://www.youtube.com/watch?v=hmSimdnFcnE&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=4
4	Print out the text for the end of month emails for paying customers given: 1. One PayingCustomer without associates 2. One PayingCustomer with associate 3. Remove AssociateCustomer	• Set magazine (MagazineService)- cost $30 and called Times • Create PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Mens time • Pay by direct debit • Create AssociateCustomer belonging to PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Mens time • Create PayingCustomer • Add offered Supplement to PayingCustomer- Cost 7 called Mens time • Pay by direct debit • Add all customers to MagazineService • Print End Month Email	The first print (Before removal) should display 1x email end of month associated with PayingCustomer that has associate. Containing email has atleast- • 2x Supplements called Mens time. Each supplement cost (7 x 4) $28 • Taken from direct debit • Total magazine cost- ($30 x 4) x 2 = $240 • Total charge - $240 + (28 + 28) = 296	Pass	https://www.youtube.com/watch?v=hmSimdnFcnE&list=PLUktbenWQI5gDoRzDnhuNcYt3JbxL5elC&index=4

		• Remove AssociateCustomer from MagazineService • Print End Month Email	1x email end of month associated with PayingCustomer without associate. Containing email has atleast- • 1x Supplements called Mens time for $28 • Taken from direct debit • Total magazine cost- $120 • Total charge- 148 The second print (After removal) should display 2x PayingCustomer without associate. Containing email has atleast- • 1x Supplements called Mens time for $28 • Taken from direct debit • Total magazine cost- $120 • Total charge- 148		

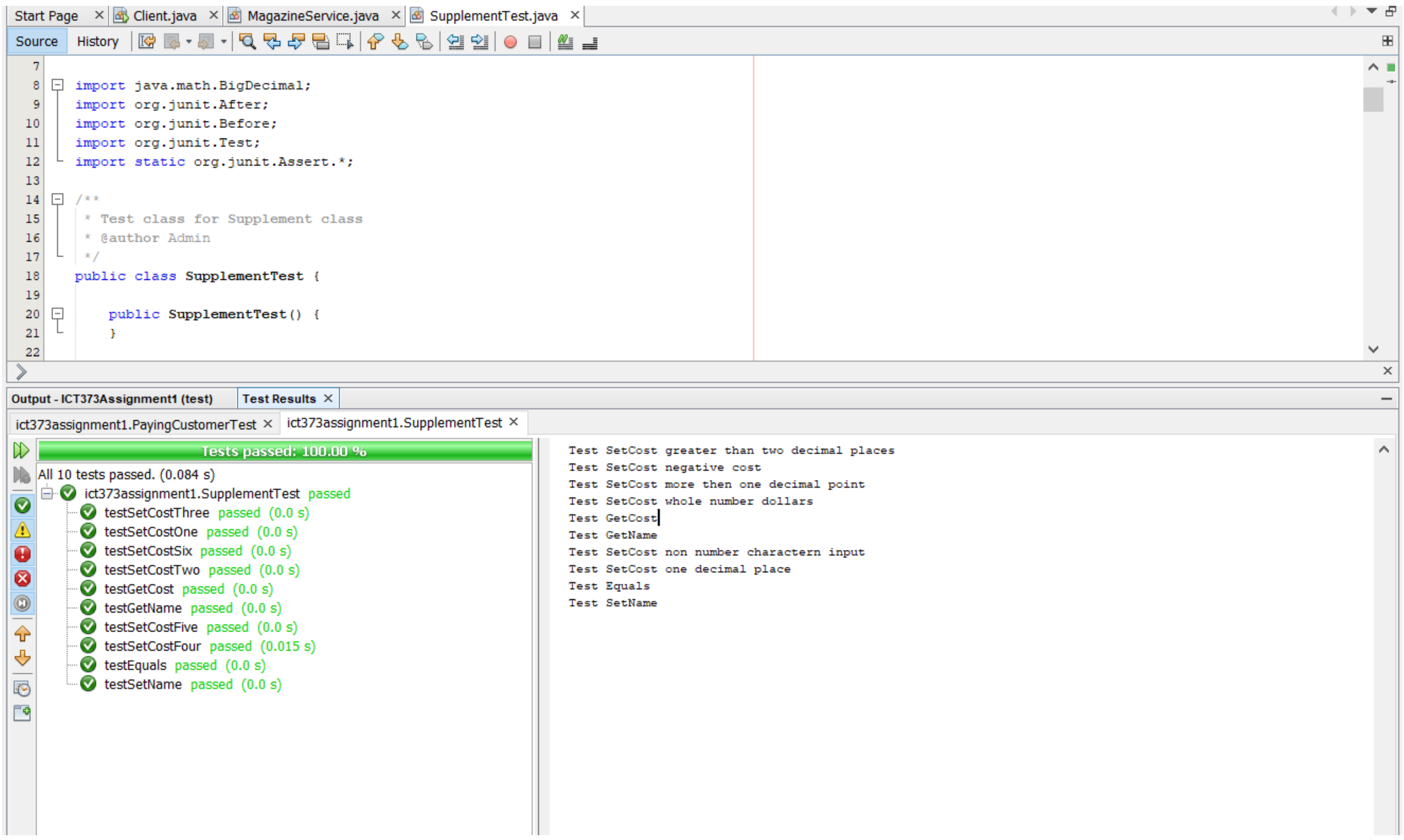
Unit testing has been completed for the following classes- AssociateCustomer, AssociateDatabase, Customer, Magazine, PayingCustomer, PaymentMethod, Supplement, SupplementDatabase.



```
/*
 * 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;
```

```
Tests passed: 100.00 %
All 17 tests passed. (0.101 s)
ict373assignment1.CustomerTest passed
    testGetReadOnlyInterestedSupplements passed (0.016 s)
    testGetEmail passed (0.0 s)
    testSetFirstNameThree passed (0.0 s)
    testSetEmailThree passed (0.0 s)
    testSetFirstNameOne passed (0.0 s)
    testSetFirstNameTwo passed (0.0 s)
    testSetLastNameThree passed (0.0 s)
    testSetLastNameOne passed (0.0 s)
    testSetLastNameTwo passed (0.0 s)
    testGetLastName passed (0.0 s)
    testSetEmailFour passed (0.0 s)
    testAddSupplementInterestOne passed (0.0 s)
    testAddSupplementInterestTwo passed (0.0 s)
    testGetFirstName passed (0.0 s)
    testEquals passed (0.0 s)
    testSetEmailOne passed (0.016 s)
    testSetEmailTwo passed (0.0 s)
```

```
Test GetReadOnlyInterestedSupplements
Test GetEmail
Test SetFirstName with non A-Z character
Test SetEmail with any domain name
Test SetFirstName
Test SetFirstName non A-Z character
Test SetLastName with non A-Z character
Test SetLastName
Test SetLastName non A-Z character
Test GetLastName
Test SetEmail with any non .com domain
Test AddSupplementInterest Null
Test AddSupplementInterest
Test GetFirstName
Test Equals
Test SetEmail
Test SetEmail with no @domain.com
```

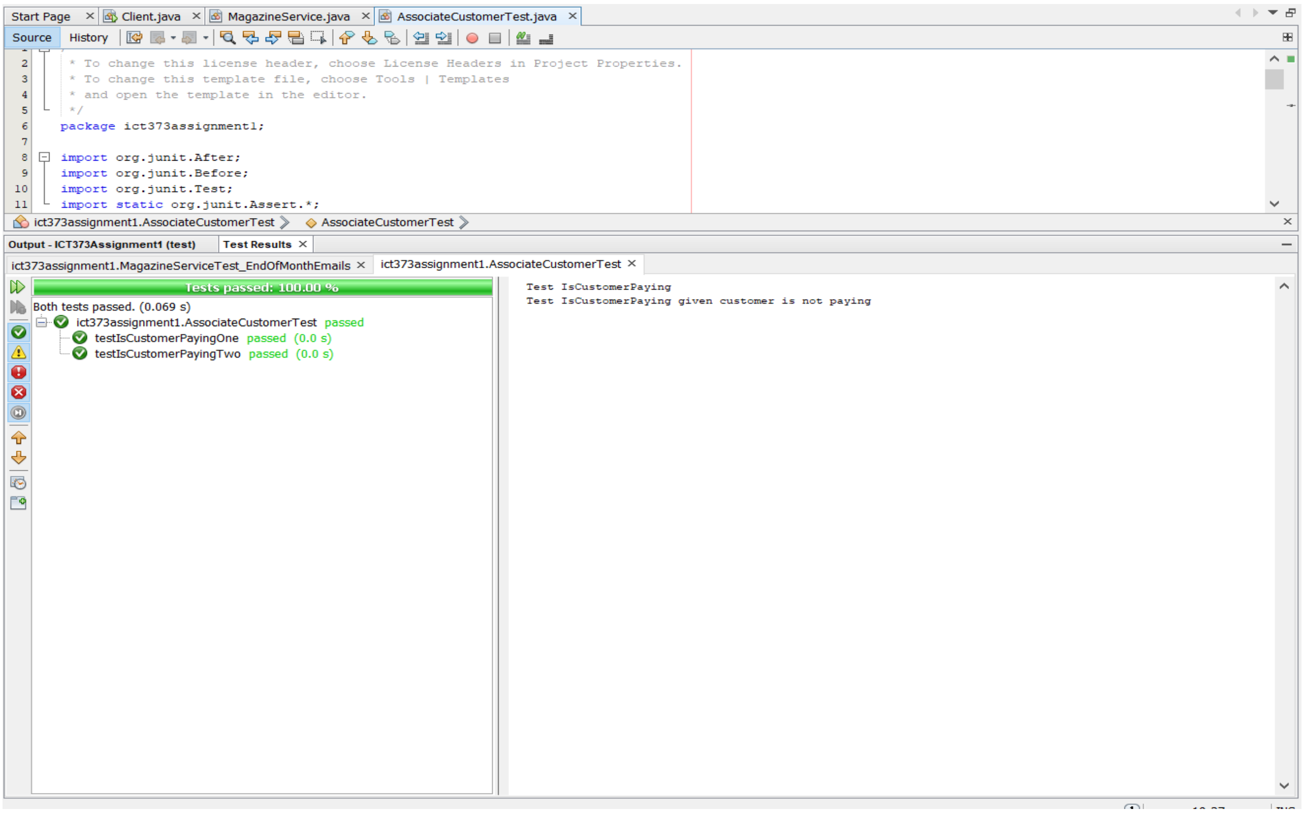


```java
import java.math.BigDecimal;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 * Test class for Supplement class
 * @author Admin
 */
public class SupplementTest {

    public SupplementTest() {
    }

```

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
```

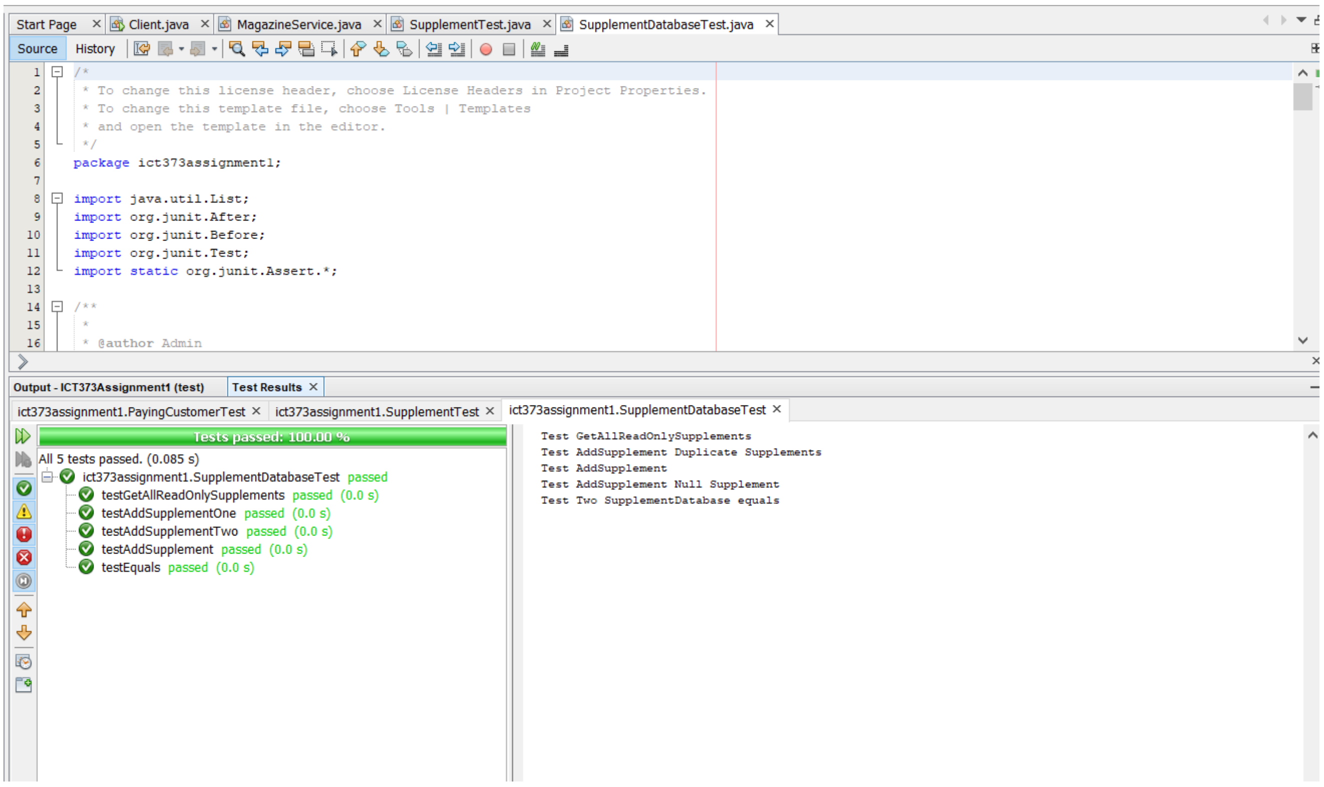

Start Page
Client.java
MagazineService.java
AssociateCustomerTest.java
Source
History

```java
 * 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

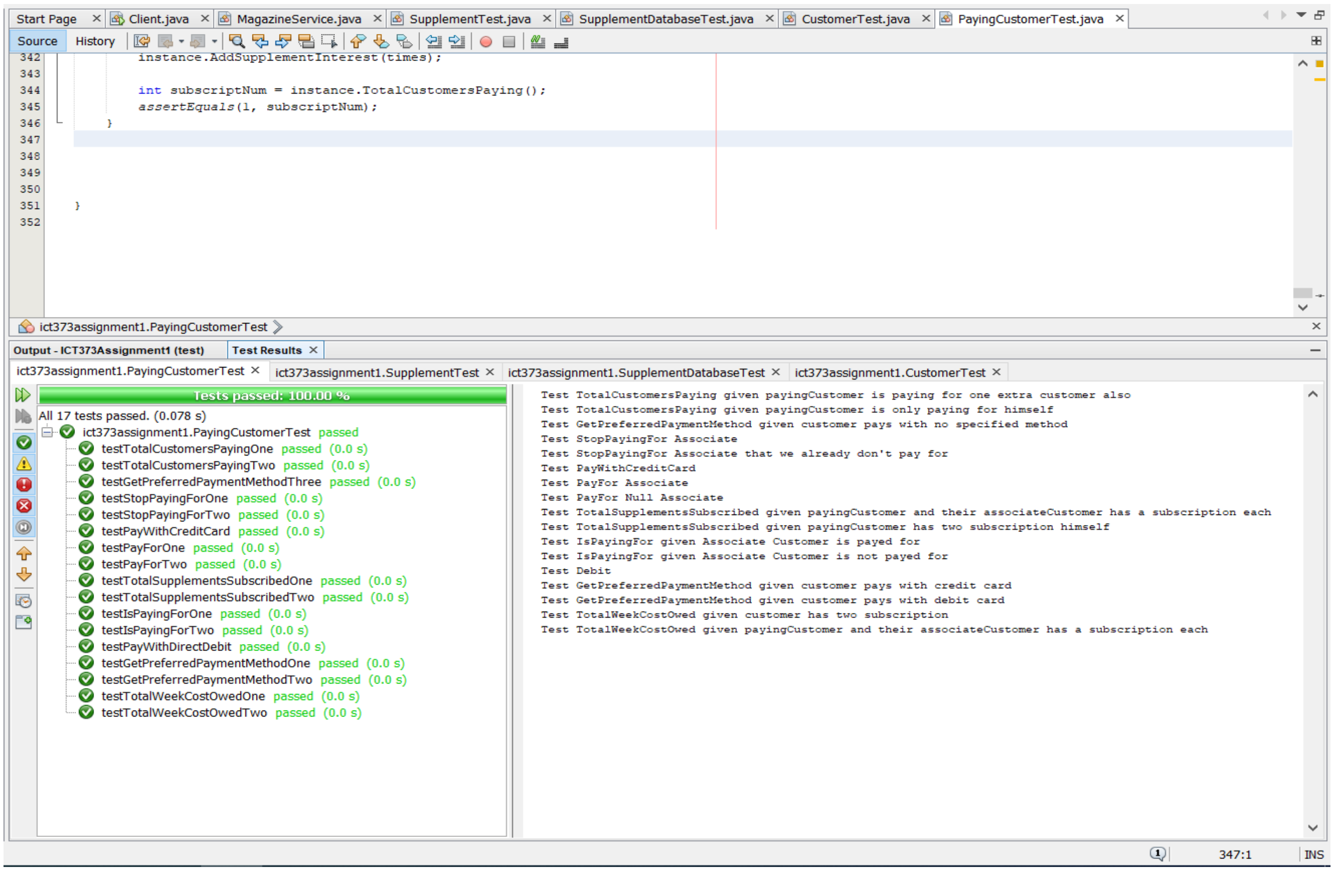
```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ict373assignment1;

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 | SupplementTest.java | SupplementDatabaseTest.java | CustomerTest.java | PayingCustomerTest.java

```java
        instance.AddSupplementInterest(times);

        int subscriptNum = instance.TotalCustomersPaying();
        assertEquals(1, subscriptNum);
    }




}
```

ict373assignment1.PayingCustomerTest

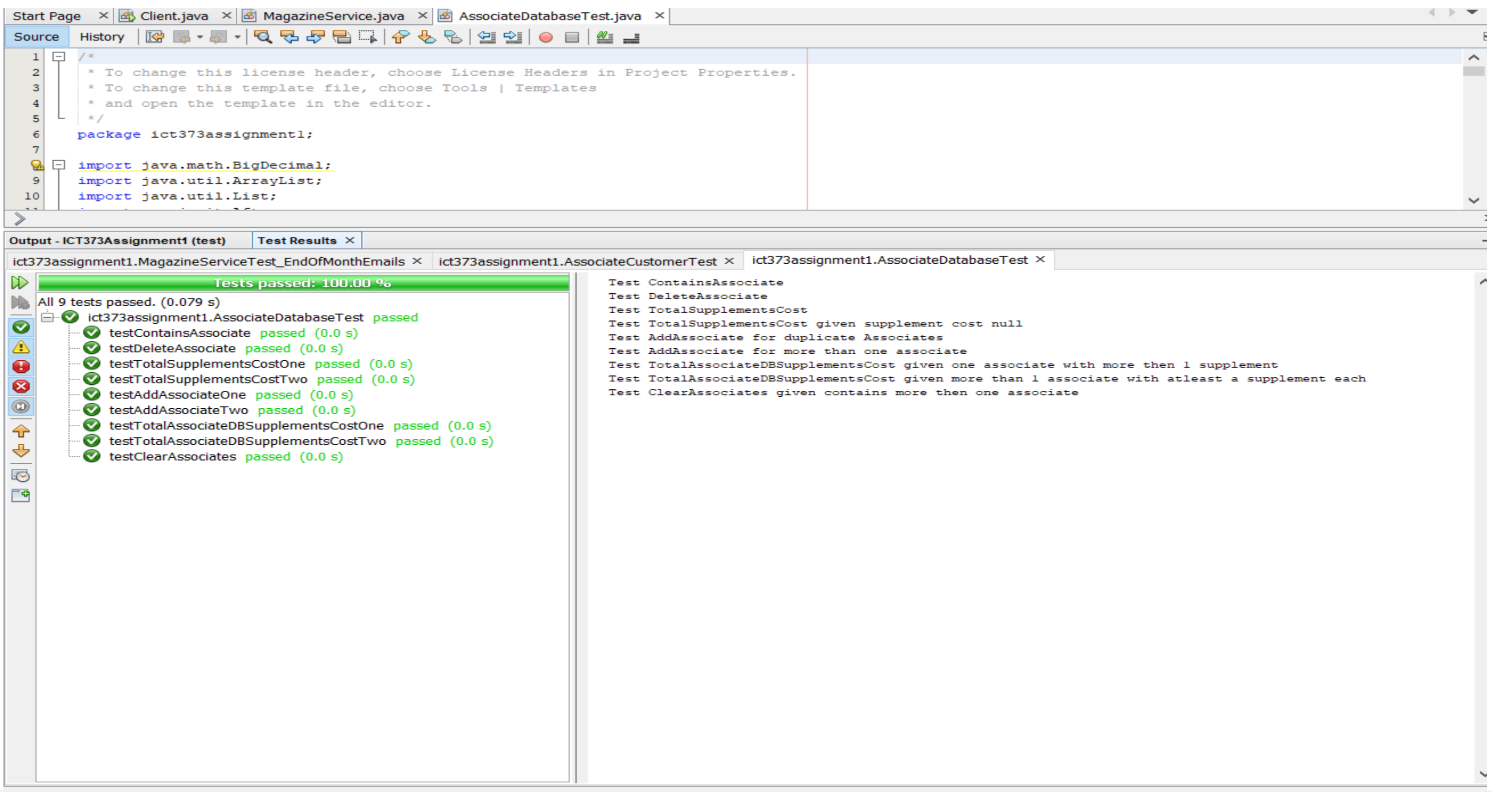
Output - ICT373Assignment1 (test) | Test Results

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

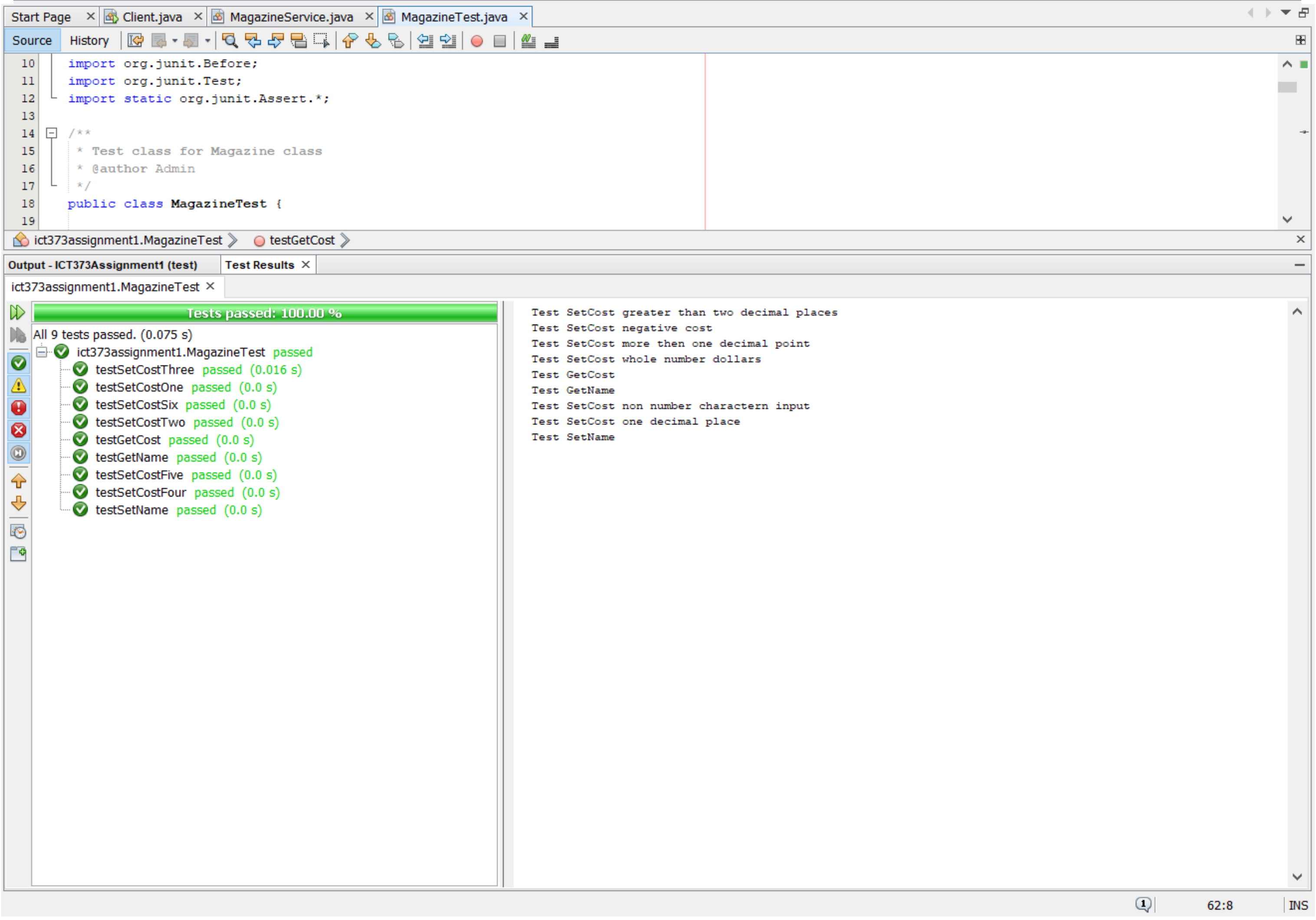
```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ict373assignment1;

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
```



Start Page | Client.java | MagazineService.java | MagazineTest.java

Source | History

```java
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 * Test class for Magazine class
 * @author Admin
 */
public class MagazineTest {
```

ict373assignment1.MagazineTest > testGetCost >

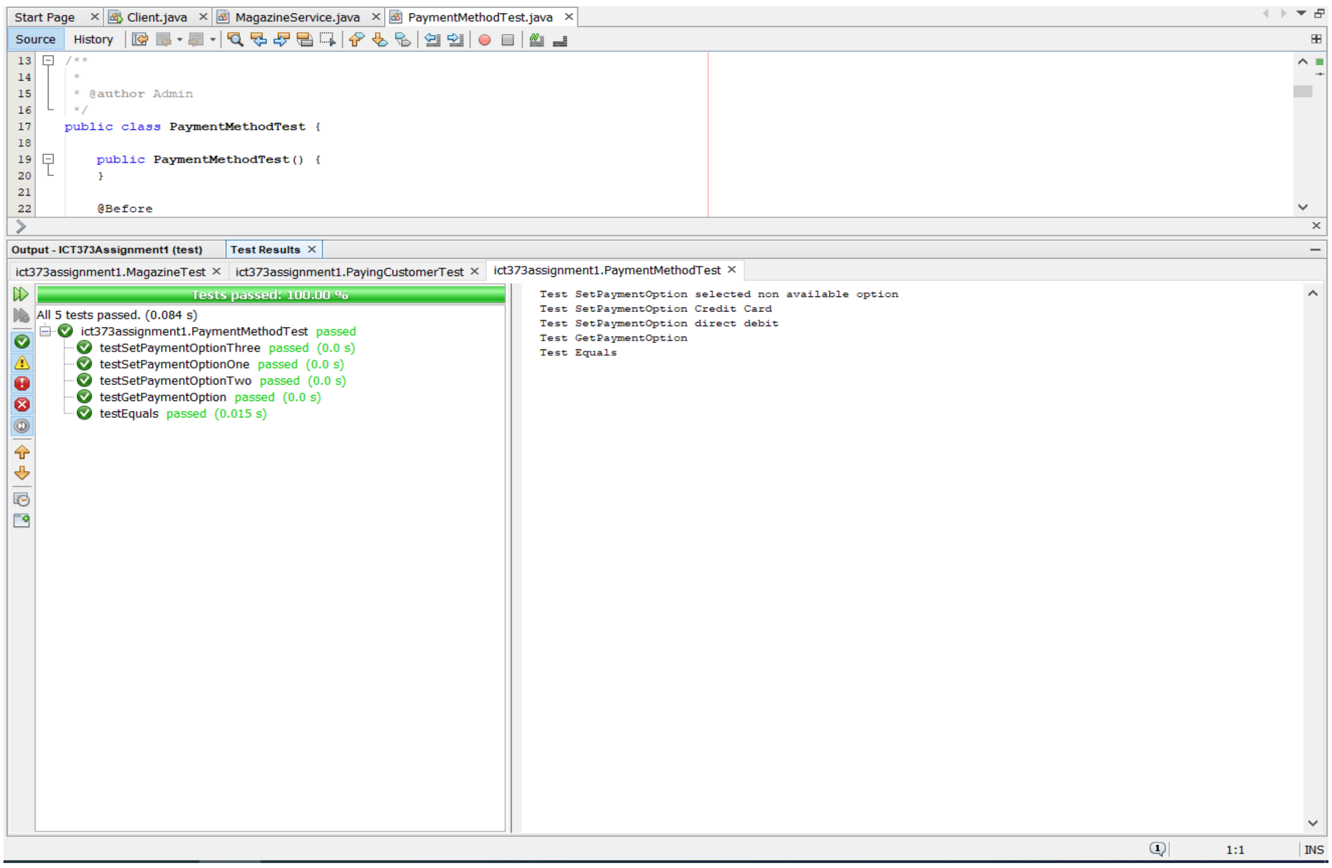
Output - ICT373Assignment1 (test) | Test Results

ict373assignment1.MagazineTest

```
Tests passed: 100.00 %
All 9 tests passed. (0.075 s)
ict373assignment1.MagazineTest passed
    testSetCostThree passed (0.016 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 (0.0 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 SetName
```

62:8 INS


Start Page
Client.java
MagazineService.java
PaymentMethodTest.java
Source
History
Output - ICT373Assignment1 (test)
Test Results
ict373assignment1.MagazineTest
ict373assignment1.PayingCustomerTest
ict373assignment1.PaymentMethodTest
1:1
INS

```
/**
 *
 * @author Admin
 */
public class PaymentMethodTest {

    public PaymentMethodTest() {
    }

    @Before
```

```
Tests passed: 100.00 %
All 5 tests passed. (0.084 s)
ict373assignment1.PaymentMethodTest passed
    testSetPaymentOptionThree passed (0.0 s)
    testSetPaymentOptionOne passed (0.0 s)
    testSetPaymentOptionTwo passed (0.0 s)
    testGetPaymentOption passed (0.0 s)
    testEquals passed (0.015 s)
```

```
Test SetPaymentOption selected non available option
Test SetPaymentOption Credit Card
Test SetPaymentOption direct debit
Test GetPaymentOption
Test Equals
```

Limitations

The program has several limitations. Firstly, the program is not interactive. The basis of the program is that it can be executed running the test files. There is no GUI interface for the client to add and delete customer in real time. Another limitation is end of month is calculated as being only 4 weeks. A month may vary on the number of weeks. March and June are 5 weeks not 4 weeks. But the program will treat all the months as 4 weeks. Thirdly, there is a lack of console error messages. For example- if a client adds a customer whom is interested in a supplement that is offered by the magazine service but at a different price then an error message will not be displayed. While, the supplement name is the same as the one offered the price is different. The customer will not be added to magazine service but an error message is not displayed to client. This was done deliberately. Classes should not be responsible for producing output since there may be various ways to output class information. For example, in the future the output may be to a form not the console. Thus, should never couple a class with output