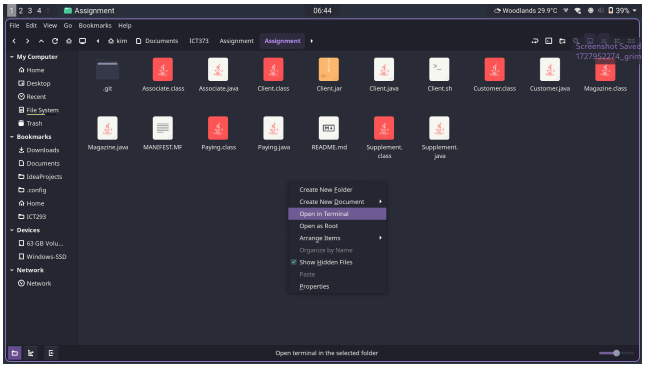
Assignment 1 – Weekly Magazine Management System. A java program made by Kim Andrew Dela Cruz that aims to efficiently manage customer and magazine issues/supplements information. The program consists of 6 java source codes; Customer.java, Paying.java, Associate.java, Magazine.java, Supplement.java, Client.java. This program and documentation is developed and submitted before October 6, 2024.

Requirements/Specification

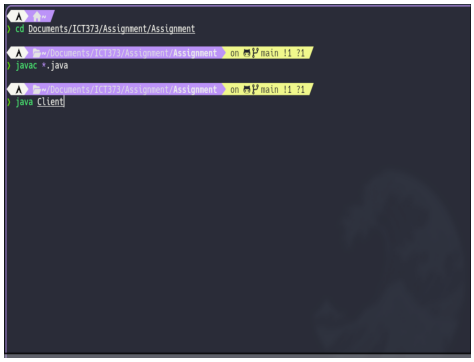
The java program stores Customer information such as email addresses, list of supplements that they are interested in, total cost of all the magazine issue and supplements that they have acquired, payment method, and associated customers which they pay for. The program also stores information about each weekly magazine issue and supplements like the name and cost. All the information is stored within the program in ArrayList data structures. The program allows the creation of new customers and magazine supplements by the user and removal of existing customers. The program can also print out all customer emails and print out the end of month invoices for all paying customers.

User Guide

To compile the program you must first have a java development kit installed; I have openJDK 17 installed in my machine. After checking or installing, you can compile the program by navigating or opening the project folder directory in your terminal.



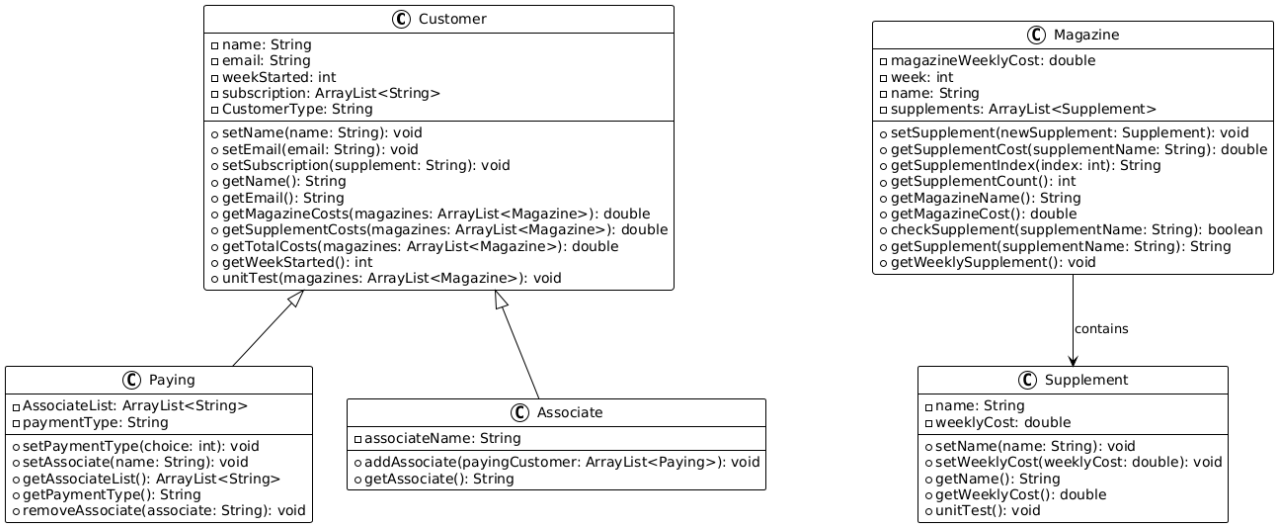
Once you are in the directory, you can compile by using the command “javac *.java”. After compiling you can run the program with the command “java Client”.



Once you run the program you can follow the on-screen instructions to use the program. Please check the names loaded customers and supplements by using selecting option 9 in the main menu. One more thing to note is that all names are not case sensitve in this program.

Structure/Design – the approach I used is an object oriented approach. The reason being is, by using object oriented approach I can modularize and reuse many parts of my program. Modularity, and encapsulation also helps with the maintainability and readability of the program.

UML Class Diagram



Psuedocode

Client Class:

function createSupplement(tempSupplements, userInput)
```
  name = get input for supplement name
  weeklyCost = get input for weekly cost
  print "Enter Supplement Name:"
  name = read input
  print "Enter Weekly Cost:"
  weeklyCost = read input
  create new Supplement object with name and weeklyCost
  add Supplement object to tempSupplements list
  print "Supplement Created"
  read input to consume newline
```

---

function createCustomer(payingCustomer, associateCustomer, currentWeek, userInput)
```
  name = get input for customer name
  email = get input for customer email
  choice = get input for customer type (1 for paying, 2 for associate)
  print "Enter Customer Name:"
  name = read input
  print "Enter Email:"
  email = read input
  print "Type 1 for paying, 2 for associate (1|2):"
  choice = read input
  switch choice
    case "1"
      print "Enter payment type, type 1 for Credit Card, 2 for Debit Card (1|2):"
```

```
        choice = read input
        if choice is "1"
          print "Paying Customer added"
          add new Paying customer to payingCustomer list with name, email, currentWeek,
and "Credit Card" payment type
        else if choice is "2"
          print "Paying Customer added"
          add new Paying customer to payingCustomer list with name, email, currentWeek,
and "Debit Card" payment type
        else
          print "Invalid Response. Try Again."
      case "2"
        print "Enter paying customer:"
        payingCustomerName = read input
        for each payingCustomer in payingCustomer
          if payingCustomerName equals payingCustomer.getName()
            add new Associate customer to associateCustomer list with name, email,
currentWeek, payingCustomerName, and payingCustomer
            userExist = true
            print "Associate Customer Added."
            break

        if not userExist
          print "Paying User does not exist. Try Again."
```

```
function addSupplement(tempSupplements, magazines, currentWeek, userInput)
  if tempSupplements is empty
    print "There is no available supplements to add"
  else
    print "Available Supplements to add to magazine:"
    for each supplement in tempSupplements
      print index + ". " + supplement.getName() + " - " + supplement.getWeeklyCost()
    print "Select which supplement to add (Type the number)"
    choice = read input
    choice = choice - 1
    if choice is out of bounds
      print "Choice out of bounds. Try Again"
    else
      tempName = tempSupplements.get(choice).getName()
      tempCost = tempSupplements.get(choice).getWeeklyCost()
      remove tempSupplements.get(choice) from tempSupplements
```

```
        magazines.get(currentWeek).setSupplement(new Supplement(tempName,
tempCost))
        read input to consume newline
        print "Successfully Added"
```

```
function addCustomer(payingCustomer, associateCustomer, magazines, currentWeek,
userInput)
  userExist = false
  paying = false
  index = 0
  print "Enter Customer Name:"
  name = read input
  for each payingCustomer in payingCustomer
    if payingCustomer.getName() equals name
      userExist = true
      paying = true
      index = current index
      break
  for each associateCustomer in associateCustomer
    if associateCustomer.getName() equals name
      userExist = true
      index = current index
      break
  if userExist
    print "Select which supplement to subscribe to(Type the index): "
    choice = read input
    choice = choice - 1
    if choice is out of bounds
      print "Choice out of bounds. Try Again."
    else
      supplementName = magazines.get(currentWeek).getSupplementIndex(choice)
```

```
        if paying
            payingCustomer.get(index).setSubscription(supplementName)
        else
            associateCustomer.get(index).setSubscription(supplementName)
        print "Successfully added customer"
    else
        print "User does not Exist. Try Again."
    read input to consume newline
```

```
function printAllEmails(payingCustomer, associateCustomer)
    print "Paying Customer:"
    for each payingCustomer in payingCustomer
        print payingCustomer.getEmail()
    print "Associate Customer:"
    for each associateCustomer in associateCustomer
        print associateCustomer.getEmail()
```

```
function printInvoice(payingCustomer, associateCustomer, magazines)
    print "End of Month Invoice:"
    for each payingCustomer in payingCustomer
        totalCost = 0
        associateList = payingCustomer.getAssociateList()
        print "Name:", payingCustomer.getName()
        print "Email:", payingCustomer.getEmail()
        print "Subscribed since Week:", payingCustomer.getWeekStarted()
        print "Payment Type:", payingCustomer.getPaymentType()
        print "Payment Breakdown:"
        print "\tBase Magazine subscription -",
payingCustomer.getMagazineCosts(magazines), "$"
        totalCost = totalCost + payingCustomer.getMagazineCosts(magazines)
        print "\tSupplement subscription -",
payingCustomer.getSupplementCosts(magazines), "$"
        totalCost = totalCost + payingCustomer.getSupplementCosts(magazines)
        print "\tAssociate Cost Breakdown:"
```

```
    for each associateName in associateList
      for each associateCustomer in associateCustomer
        if associateCustomer.getName() equals associateName
          print "\t\tAssociate's Email Address:", associateCustomer.getEmail()
          print "\t\tAssociate's Total cost:", associateCustomer.getTotalCosts(magazines),
"$"
          totalCost = totalCost + associateCustomer.getTotalCosts(magazines)
    print "Total:", totalCost, "$"
```

```
function removeCustomer(payingCustomer, associateCustomer, userInput)
  userExist = false
  paying = false
  index = 0
  print "Enter Customer Name:"
  name = read input
  for each payingCustomer in payingCustomer
    if payingCustomer.getName() equals name
      userExist = true
      paying = true
      index = current index
      break
  for each associateCustomer in associateCustomer
    if associateCustomer.getName() equals name
      userExist = true
      index = current index
      break
  if userExist
    if paying
      remove payingCustomer.get(index) from payingCustomer
      print "Customer Successfully removed"
    else
      for each payingCustomer in payingCustomer
        if associateCustomer.get(index).getAssociate() equals payingCustomer.getName()
```

```
payingCustomer.get(i).removeAssociate(associateCustomer.get(index).getAssociate())
        print "Customer Successfully removed"
        remove associateCustomer.get(index) from associateCustomer
    else
      print "Customer not found. Try Again."
```

Customer Class:

```
function getTotalCosts(magazines)
    totalCost = 0
    for each magazine in magazines
      totalCost = totalCost + magazine.getMagazineCost()

    for each magazine in magazines
      for each subscription in subscription
        if magazine.checkSupplement(subscription)
          totalCost = totalCost + magazine.getSupplementCost(subscription)
    return totalCost
```

Magazine Class:

```
function getSupplementCost(supplementName)
    weeklyCosts = 0
    for each supplement in supplements
      if supplement.getName() equals supplementName
        weeklyCosts = weeklyCosts + supplement.getWeeklyCost()
    return weeklyCosts
```

```
function checkSupplement(supplementName)
    for each supplement in supplements
      if supplement.getName() equals supplementName
```

```
        return true
    return false
```

---

function getWeeklySupplement()

```
    print "Available Supplements:"
    for each supplement in supplements
        print index + ". " + supplement.getName() + " - " + supplement.getWeeklyCost(), "$"
```

Associate Class (Sub-Class of Customer):

function addAssociate(payingCustomer)

```
    for each payingCustomer in payingCustomer
        if payingCustomer.getName() equals associateName
            payingCustomer.setAssociate(this.getName())
```

Paying Class (Sub-Class of Customer):

function removeAssociate(associate)

```
    for each associateName in AssociateList
        if associateName equals associate
            remove associateName from AssociateList
```

Limitation – the program is not able to :

- automatically change the current week
- give a more detailed payment breakdown
- add magazine issues in the program
- remove magazine issues and supplements
- edit name, cost, email, and more from existing customers and supplements

Testing:

TestID	Test Description	Input	Expected Result	Actual Result	Pass/Fail
T01	Test option 1. Create a new supplement	Supplement Name: test_1 Supplement Cost: 7.99	In option 9 (Unit Test). The created supplement should be shown.	Refer to figure. T01 Actual Result	Pass
T02	Test option 2. Create a new paying customer	Customer Name: paying_customer Email: customer@example.com Type: 1 Payment Type: 1	In option 9 (Unit Test). The created customer should be shown.	Refer to figure. T02 Actual Result	Pass
T03	Test option 2. Create a new associate customer	Customer Name: associate_customer Email: associate@example.com Type: 2 Paying Customer: Emma Walker	In option 9 (Unit Test). The created customer should be shown.	Refer to figure. T03 Actual Result	Pass
T04	Test option 3. Add new supplement.	Assumption: test_01 (cost: 7.99) is already created. Enter: 1	After entering . The program should clear screen and Prompt “Successfully Added”	Refer to figure. T04 Actual Result	Pass
T05	Test option 4. Add new customer.	Enter Customer Name: Emma	After entering . The program should clear	Refer to figure. T05 Actual Result	Pass

		Walker. Available supplement – 1. Culinary delights	screen and Prompt “Successfully Added Customer”		
T06	Test option 5. Print all exsiting customer’s email.	Enter option 5 from main menu.	Screen Prints all existing customer’s email	Refer to figure. T06 Actual Result	Pass
T07	Test option 6. Print end of month invoice for paying customer.	Enter option 6 from main menu.	Screen Prints the end of month invoice for paying customer.	Refer to figure. T07 Actual Result	Pass
T08	Test option 7. Remove existing customer.	Customer Name: Emma Walker	Screen Prompt’s “Successfully Removed Customer”	Refer to figure. T08 Actual Result	Pass
T09	Test option 8. Exit program.	Enter option 8 from main menu.	Screen Promp’s “Program Exited” and exits the program.	Refer to figure. T09 Actual Result	Pass

T01 Actual Result:

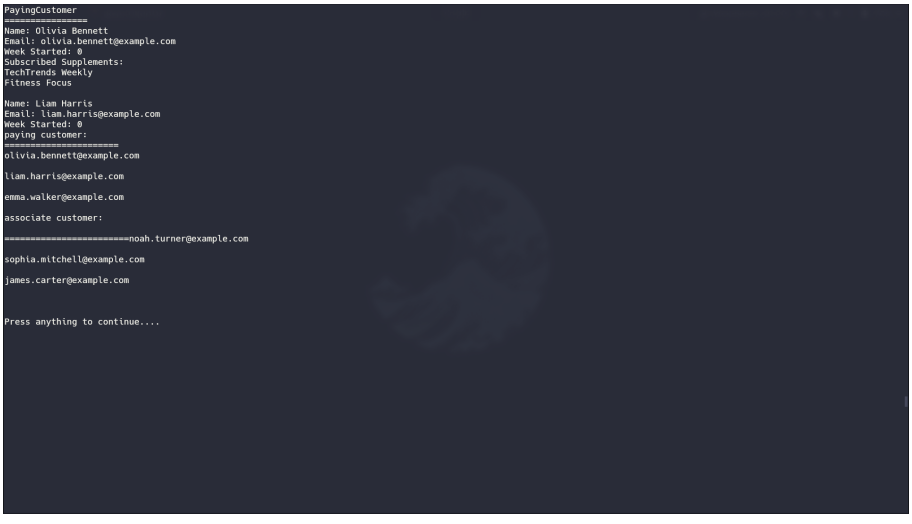
T02 Actual Result:

T03 Actual Result:

T04 Actual Result:

T05 Actual Result

T06 Actual Result



T07 Actual Result

T08 Actual Result

T09 Actual Result