|  |  |
| --- | --- |
| **ELECTRONIC ASSIGNMENT COVERSHEET** | Murdoch_land_RGB |

|  |  |
| --- | --- |
| **Student Number:** | 35517678 |
| Surname: | New |
| Given name: | Wei Nern |
| Email: | 35517678@student.murdoch.edu.au |
|  |  |
| **Unit Code:** | ICT373 |
| Unit name: | Software Architecture |
| Enrolment mode: | Internal |
| Date: | 9/6/2025 |
| Assignment number: |  |
| Assignment name: |  |
| Tutor: | Andy Lee |

|  |
| --- |
| **Student’s Declaration:**   * Except where indicated, the work I am submitting in this assignment is my own work and has not been submitted for assessment in another unit. * This submission complies with Murdoch University's academic integrity commitments. I am aware that information about plagiarism and associated penalties can be found at <http://our.murdoch.edu.au/Educational-technologies/Academic-integrity/>. If I have any doubts or queries about this, I am further aware that I can contact my Unit Coordinator prior to submitting the assignment. * I acknowledge that the assessor of this assignment may, for the purpose of assessing this assignment:   + reproduce this assignment and provide a copy to another academic staff member; and/or   + submit a copy of this assignment to a plagiarism-checking service. This web-based service may retain a copy of this work for the sole purpose of subsequent plagiarism checking, but has a legal agreement with the University that it will not share or reproduce it in any form. * I have retained a copy of this assignment. * I will retain a copy of the notification of receipt of this assignment. If you have not received a receipt within three days, please check with your Unit Coordinator. |

|  |
| --- |
| I am aware that I am making this declaration by submitting this document electronically and by using my Murdoch ID and password it is deemed equivalent to executing this declaration with my written signature. |
| **Optional Comments to Tutor:**  *e.g. If this is a group assignment, list group members here* |

*If you can, please insert this completed form into the body of* ***each*** *assignment you submit. Follow the instructions in the Unit Information and Learning Guide about how to submit your file(s) and how to name them, so the Unit Coordinator knows whose work it is.*

***Start your assignment on the next page.***

Magazine

Subscription Management

System

*Simplify removing and adding of subscription and customer, while at the same time providing an easy to navigate interface.*

**Author:** New Wei Nern

**Murdoch Student ID**: 35517678

**Date:** 4/6/2025

**Student Note:** This is a GitHub [repository](https://github.com/nwnisworking/ICT373-Assignment-1/tree/V2) that contains the code.

**Files:** Please refer to page 3 for the list of files.

Contents

[List of Files 5](#_Toc200890477)

[User Guide 6](#_Toc200890478)

[Requirement / Specification 8](#_Toc200890479)

[Input Conditions 9](#_Toc200890480)

[Output Conditions 9](#_Toc200890481)

[Structure / Design 11](#_Toc200890482)

[Class Diagram 11](#_Toc200890483)

[Structure Chart 12](#_Toc200890484)

[Design Considerations 13](#_Toc200890485)

[Limitations 14](#_Toc200890486)

[Assumptions 14](#_Toc200890487)

[Pseudocode 14](#_Toc200890488)

[Main Method 15](#_Toc200890489)

[WeeklyEmail Method 16](#_Toc200890490)

[InvoiceCustomer Method 17](#_Toc200890491)

[ViewCustomers Method 18](#_Toc200890492)

[AddCustomer Method 19](#_Toc200890493)

[DeleteCustomer Method 20](#_Toc200890494)

[SetPayer Method 21](#_Toc200890495)

[RemovePayer Method 22](#_Toc200890496)

[AddSubscription Method 23](#_Toc200890497)

[RemoveSubscription Method 24](#_Toc200890498)

[CreateMagazineSupplement Method 25](#_Toc200890499)

[DeleteMagazineSupplement Method 26](#_Toc200890500)

[Testing 27](#_Toc200890501)

# List of Files

|  |  |
| --- | --- |
| Files | Summary |
| customers/AssociateCustomer.java | Represents an associate customer. |
| customers/PayingCustomer.java | Represents a paying customer. The paying customer is allowed to pay for a subscription for someone and itself. |
| customers/Customer.java | A base class for all customers, with common attributes and behaviours. |
| magazines/Magazine.java | Represents a magazine. |
| magazines/Subscription.java | A base class for all subscriptions, with common attributes and behaviours. |
| magazines/Supplement.java | Represents a supplement. |
| payment/CreditCard.java | Represents a credit card. The credit card class contains customer ID, card number, and expiry date. |
| payment/DirectDebit.java | Represents a direct debit. The direct debit class contains customer ID, bank name, and account number. |
| payment/Method.java | An interface to represent different payment methods. |
| utils/IO.java | Provides utility methods for receiving user input and displaying string in the console. |
| utils/ANSI.java | An ANSI command for removing and setting cursor back in place |
| Main.java | The main program that handles user’s input and process instructions. |
| Page.java | A representation of each page based on the choice from Main.java. |
| pom.xml | An XML file with information about the project and configuration details. |

# User Guide

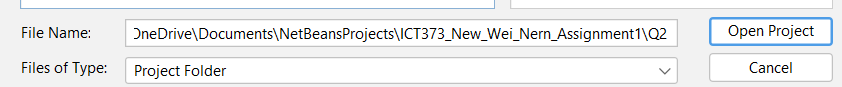
Before running ICT373\_New\_Wei\_Nern\_Assignment-1 project, ensure you have the following programs installed on your system:

1. Java Development Kit (JDK) 23.0.1
   * Download the JDK from the official [Oracle Archive](https://www.oracle.com/java/technologies/javase/jdk23-archive-downloads.html) website
   * Install the JDK and ensure Java is set up by running java -version in the CMD.
2. Apache NetBeans IDE 24
   * Download Apache NetBeans from the official [Apache NetBeans](https://netbeans.apache.org/front/main/download/) website
   * Install NetBeans and ensure it detects the installed JDK

To run the project:

1. Extract the zip file “ICT373\_New\_Wei\_Nern\_Assignment1.zip” to NetBeansProjects folder A screenshot of a computer

   AI-generated content may be incorrect.
2. Open Apache NetBeans IDE 24
3. Click on File, Open Project
4. Open “ICT373\_New\_Wei\_Nern\_Assignment1” folder and click on “Q2” A screenshot of a computer

   AI-generated content may be incorrect.
5. Click on Open Project
6. Click the Run button in the toolbar or press F6 to run the project

# Requirement / Specification

The Magazine Subscription Management System enables administrators to create, remove, and manage both associate and paying customers with ease. It also provides functionality to add and remove subscriptions, making it highly efficient in managing customers’ subscription lists.

Customers can be classified into two types: **Paying** and **Associate**. While paying customers are required to provide a payment method, associates do not need to; instead, each associate must be linked to a paying customer who covers their subscription costs.

With that consideration, administrators can choose to remove any associates from a paying customer, thereby deactivating the associate’s subscription until a new payer is assigned. The same applies if a paying customer is deleted, which triggers the deactivation of all associated customers’ subscriptions. Associate customers who do not have an assigned payer will be filtered so that a new payer can be designated.

In addition to managing customers, the system allows administrators to easily set or remove subscriptions, including the main magazines and its related supplements. Supplements cannot be subscribed independently; they are only made available once the main magazine subscription is active. Once subscribed to the magazine, customers are entitled to subscribe to its related supplements multiple times.

The current system does not send actual emails, nor does it utilize Java Date class. Instead, it simulates email output in the console. Weekly emails are generated to all customers with active subscriptions, informing them that their content is ready for viewing. At the end of the four-week cycle, paying customers receive a monthly billing email summarizing the total amount owed for both their own and their associates’ subscriptions.

## Input Conditions

* **Valid Customer Type** - Each customer must be either a paying customer or an associate customer.
* **Magazine Subscription** - Customers must subscribe to the main magazine to subscribe to its supplements.
* **Subscription Duplication** - Customers can subscribe to the same magazines and supplements multiple times.
* **One Payer per Associate** - Associates must be assigned to one paying customer.
* **Payment Details** - Only paying customers are required to input their payment details.
* **Set Paying Customer** - Once the associate customer is created, the system will ask the administrator to set a paying customer to the associate customer.
* **Paying Customer Deletion** - If a paying customer is removed, all associate customers related to paying customer will become inactive.

## Output Conditions

* **Console Only Output** - All data, including customers, subscriptions, weekly and monthly emails are displayed via console.
* **Customer Detail Viewing** - A list of customers is presented when “View Customers” is selected. The administrator may select a specific customer to view their information, payment details (paying customer), and currently subscribed supplements.
* **Weekly Email Simulation** - Customers with a magazine subscription receive a simulated weekly email showing their name and subscribed supplements.
* **Monthly Billing Email** - At the end of every four weeks, a billing summary is generated for each paying customer, including the cost of their own subscriptions as well as those of their associated customers.
* **Inactive Associates** - Associate customers who do not have a designated paying customer will still receive weekly emails, informing them their subscription is inactive until a new payer is found.

# Structure / Design

## Class Diagram

A screenshot of a computer screen

AI-generated content may be incorrect.

## Structure Chart

A screenshot of a computer screen

AI-generated content may be incorrect.

The system adheres to design principles which allow each component to function independently. All classes are designed with the Open/Closed Principle in mind to ensure the system can be extended without altering existing logic. An example of this can be seen in the **Customer** and **Subscription** classes, which contain all the base methods, with their respective subclasses providing specific behaviour and additional properties. The Liskov Substitution Principle is applied using polymorphism, allowing instances of **AssociateCustomer** and **PayingCustomer** to be treated as Customer objects. This is exemplified by their storage in a **TreeMap** using the Customer type as a reference.

In addition to inheritance-based polymorphism, the system also follows the Dependency Inversion Principle, allowing **PayingCustomer** to use different types of payment methods such as **CreditCard** and **DirectDebit** without being tightly coupled to either.

By designing each class to adhere to the Single Responsibility Principle, maintainability is greatly improved, as each class is responsible for one aspect of the system’s behaviour. A good example is the **IO** class, which handles all user interaction, while the **Page** class is dedicated to rendering various pages.

## Design Considerations

These are the design considerations that I faced while developing the system.

1. **Map vs ArrayList**

The use of an **ArrayList** is not ideal when searching for elements by key. Since much of the system’s input requires locating a customer or subscription by ID, a Map is preferred.

1. **TreeMap over HashMap**

Although both **TreeMap** and **HashMap** offer efficient key-based lookups, **TreeMap** was chosen because it maintains its keys in sorted order. This is especially essential as it determines the next unique ID for a new customer without accidental risk of duplicating ID.

## Limitations

**No Advanced Input Validation** – The system does not perform checks for duplicate names or email addresses when registering new customers. It is assumed that the administrator will avoid conflicts manually.

**Unvalidated Payment Details** – Input for credit card and direct debit details is accepted without format validation. No regular expressions or pattern-matching logic are applied to enforce official card number lengths, prefixes, or structure.

**Lack of Persistent Storage** – In line with assignment brief, all data exists in memory only and is lost once the program terminates.

**No Date Tracking** – The system does not utilize Java’s **Date** or **LocalDate** classes to track the current week / month. Instead, the events are simulated without reference to real calendar dates.

## Assumptions

**Administrator view** – The design is intended primarily for administrator use, with the layout mainly focused on data entry and management.

**Inactive Associates** – Deleting a paying customer does not necessarily result in the deletion of their associate customers. However, when a payer is removed, all linked associates will have their subscriptions marked as inactive until a new payer is assigned.

**Duplicate subscriptions** – The system allows a customer to subscribe to the same supplement and magazine multiple times.

**Payment Details** – Payment details are assumed to be valid as entered. The system does not perform checking on these details.

## Pseudocode

### Main Method

Function Main()

try

  options = -1

     while options not equal 0

      ClearScreen()

HomeMenu()

       options = GetInt(“Select one of the following options: “)

       PrintLn()

       Switch options

        1 => WeeklyEmail(customers)

         2 => InvoiceCustomer(customers)

         3 => ViewCustomers(customers)

         4 => AddCustomer(customers)

         5 => Deletecustomer(customers)

         6 => SetPayer(customers)

         7 => RemovePayer(customers)

         8 => ViewMagazineSupplement(subscriptions)

         9 => AddSubscription(subscriptions, customers)

         10 => RemoveSubscription(subscriptions, customers)

         11 => options = 0

         default =>

          options = -1

           Println("Selection unavailable")

  catch(Exception e)

  Println(e.msg)

  Println("Thank you for using Magazine Subscription System!")

  Close()

### WeeklyEmail Method

Function WeeklyEmail(customers)

customers\_arr = customers.all

  for i = 0, size = customer.size; i less than size; i plus 1

  customer = customers\_arr[i]

    if customer.subscriptions.size equal 0

    continue

    Clear()

    HomePosition()

    if customer matches AssociateCustomer and customer.payer equal null

    printText(`

      To: {customer.email}

        Subject: Your weekly subscription is inactive :(

        Dear {customer.name},

        A payer is required to keep your subscription active.

        If you have enquiries, feel free to reach out to us.

        Warm regards,

        Magazine Daily

      `)

    else

    printText(`

      To: {customer.email}

        Subject: Your weekly subscription is ready!

        Dear {customer.name},

        Your magazine is ready for viewing!

        Here is what you currently subscribed to:

      `)

      for subscription in customer.subscriptions

      Println(`- {subscription.name}`)

        PrintText(`

        If you have any enquiries, feel free to reach out to us.

          Warm regards,

          Magazine Daily

        `)

    Prompt(i less than size - 1 then "press enter to continue the next customer" else null)

### InvoiceCustomer Method

Function InvoiceCustomer(customers)

paying\_customers = customers.filter(e=>e matches PayingCustomer)

total = 0

  for i = 0, size = paying\_customers.size; i less than size; i plus 1

  customer = paying\_customers[i]

    Clear()

    HomePosition()

    PrintText(`

      To: {customer.email}

      Subject: Invoice for Paying Customer {customer.name}

      Dear: {customer.name},

      Thank you for being our valued customer!

      Here is the summary for this month:

    `)

    if customer.subscriptions.size equal 0

      Println("No available subscription")

    else

      DisplayTable(Subscription, customer.subscriptions)

      PrintText(`Total cost: {customer.subscriptions.cost}`)

total = customer.subscriptions.cost

if customer.associates.size equal 0

PrintText(“No available associate customers”, “”)

else

for associate in customer.associates

PrintText(“”, `Associate Customer: {associate.name}`)

if associate.subscriptions.size equal 0

PrintText(“”, “Associate customer does not have any subscription yet.”)

else

DisplayTable(Subscription, associate.subscriptions)

PrintText(“”, `Total cost for {associate.name}’s subscription: {associate.subscriptions.cost}`)

total+= associate.subscriptions.cost

PrintText(`

Here is the total cost of all the subscriptions: {total}

Warm regards,

Magazine Daily

`)

Prompt(i less than size – 1 then “Press enter to continue to the next customer” else null)

### ViewCustomers Method

Function ViewCustomers(customers)

Clear()

HomePosition()

PrintText(“View Customer”, “”)

DisplayTable(Customer, customers.all)

Println(“”)

customer = customers.get(GetInt(“Enter customer ID to view its information: “))

Println(“”)

if customer equal null

Println(“Customer cannot be found.”)

else

Clear()

HomePosition()

PrintText(`

Name: {customer.name}

Email: {customer.email}

Type: {customer matches PayingCustomer ? “Paying Customer” : “Associate Customer”}

`)

if customer matches PayingCustomer

PrintText(“”, “Payment Details”)

if customer.payment\_method matches CreditCard

credit = customer.payment\_method

PrintText(`

Type: Credit Card

Card Number: {credit.card\_number}

Card Expiry: {credit.expiry\_date}

`)

else

debit = customer.payment\_method

PrintText(`

Type: Direct Debit

Card Number: {debit.account\_number}

Bank Name: {credit.bank\_name}

`)

Println(“”)

if customer.subscriptions.size equal 0

Println(“Customer does not have subscription”)

else

DisplayTable(Subscription, customer.subscriptions)

Prompt(null)

### AddCustomer Method

Function AddCustomer(customers)

Clear()

HomePosition()

PrintText(“Add Customer”, “”)

method = null

customer = null

do

try

customer = Customer.getType(GetInt(“Is this a paying customer? (1 for Yes, 0 for No): “))

catch(Exception e)

while customer equal null

customer.id = customers.empty() then 1 else customers.lastId + 1

customer.name = GetString(“Enter customer name: “)

customer.email = GetString(“Enter customer email: “)

if customer matches PayingCustomer)

PrintText(“”, “Payment Method”)

do

try

method = Method.getType(GetInt(“Select customer’s payment method (1 for Credit Card, 2 for Direct Debit”))

catch(Exception e)

while method equal null

if method matches DirectDebit

method.account\_number = GetString(“Enter account number: “)

method.bank\_name = GetString(“Enter bank name: “)

else

method.card\_number = GetString(“Enter credit card number: “)

method.expiry\_date = GetString(“Enter bank name: “)

customer.payment\_method = method

else

payers = customers.filter(e=>e matches PayingCustomer)

PrintText(“”, “List of paying customers”, “”)

DisplayTable(Customer, payers)

paying\_customer = customers.get(GetInt(“Select payer: “))

if paying\_customer equal null

Println(“Unable to find payer by ID.”)

Prompt(null)

return

else if paying\_customer matches AssociateCustomer

Println(“Customer is not a paying customer”)

Prompt(null)

return

else

paying\_customer.associates[] = customer

customers.put(customer.id, customer)

PrintText(“”, “{customer.name} added successfully”)

Prompt(null)

### DeleteCustomer Method

Function DeleteCustomer(customers)

Clear()

HomePosition()

PrintText(“Delete Customer”, “”)

DisplayTable(Customer, customers.all)

customer = customers.get(GetInt(“Select one of the customer: “))

if customer equal null

PrintText(“”, “Unable to find customer”)

else

if customer matches PayingCustomer

customer.associates = null

else

if customer.payer not equal null

customer.payer.removeAssociate(customer)

customers.remove(customer)

PrintText(“”, `{customer.name} deleted successfully.`)

Prompt(null)

### SetPayer Method

Function SetPayer(customers)

Clear()

HomePosition()

PrintText(“Set Payer”, “”)

associate\_customers = []

paying\_customers = []

associate\_customer = null

paying\_customer = null

for customer in customers.all

if customer matches PayingCustomer

paying\_customers[] = customer

else

if customer.payer equal null

associate\_customers[] = customer

if associate\_customers.empty()

Println(“All the associates have a payer”)

Prompt(null)

return

Println(“Associate Customer”)

DisplayTable(Customer, associate\_customers)

associate\_customer = customers.get(GetInt(“Enter associate ID: “))

if not associate\_customers.contains(associate\_customer)

PrintText(“”, “The ID is not part of the list.”)

Prompt(null)

return

Clear()

HomePosition()

PrintText(“”, “Paying Customer”)

DisplayTable(Customer, paying\_customers)

Println(“”)

payer\_customer = customers.get(GetInt(“Enter paying customer ID: “))

if not paying\_customers.contains(paying\_customer)

PrintText(“”, “The ID is not part of the list.”)

Prompt(null)

return

paying\_customer.associate[] = associate\_customer

PrintText(“”, `Payer {paying\_customer.name} set to associate {associate\_customer.name} successfully`)

Prompt(null)

### RemovePayer Method

Function RemovePayer(customers)

Clear()

HomePosition()

PrintText(“Remove Payer”, “”)

associate\_customers = []

paying\_customers = customers.filter(e=>e matches PayingCustomer)

paying\_customer = null

associate\_customer = null

Println(“Paying Customer”)

DisplayTable(Customer, paying\_customers)

Println(“”)

paying\_customer = customers.get(GetInt(“Enter ID of the paying customer: “))

Println(“”)

if paying\_customer equal null or not paying\_customers.contains(paying\_customer)

Println(“ID is not part of the list”)

Prompt(null)

return

associate\_customers = paying\_customer.associates

if associate\_customers.empty()

Println(`Paying customer {paying\_customer.name} does not have an associate.`)

Prompt(null)

return

Clear()

HomePosition()

PrintText(“Remove Payer”, “”, “Associates”)

DisplayTable(Customer, associate\_customers)

Println(“”)

associate\_customer = customers.get(GetInt(“Enter ID of the associate customer: “))

Println(“”)

if associate\_customer equal null or not associate\_customers.contains(associate\_customer)

Println(“ID is not part of the list”)

Prompt(null)

return

paying\_customer.removeAssociate(associate\_customer)

Println(`Paying customer {paying\_customer.name} removed {associate\_customer.name} from associate list`)

Prompt(null)

### AddSubscription Method

Function AddSubscription(subscriptions, customers)

customer = null

subscription = null

Clear()

HomePosition()

PrintText(“Add Subscription”, “”)

DisplayTable(Customer, customer.all)

customer = customers.get(GetInt(“Seletc customer from the list: “))

if customer equal null

PrintText(“”, “Unable to set subscription from a non-existent customer”)

Prompt(null)

return

Clear()

HomePosition()

PrintText(“Add Subscription”, “”)

DisplayTable(Subscription, subscriptions.all)

subscription = subscriptions.get(GetInt(“Select subscription from the list: “))

if subscription equal null

PrintText(“”, “Subscription is invalid”)

Prompt(null)

return

if subscription.isSupplement() and not customer.hasMagazine(subscription.magazine)

PrintText(“”, “Customer do not have the required magazine to read the subscription”)

Prompt(null)

return

customer.subscriptions[] = subscription

PrintText(“”, `Subscription {subscription.name} added to {customer.name}`)

Prompt(null)

### RemoveSubscription Method

Function RemoveSubscription(subscriptions, customers)

Clear()

HomePosition()

PrintText(“Remove Subscription”, “”)

DisplayTable(Customer, customers.all)

customer = customers.get(GetInt(“Select customer from the list: “))

if customer equal null

PrintText(“”, “Unable to remove subscription from a non-existent customer”)

Prompt(null)

return

if customer.subscriptions.size equal 0

PrintText(“”, “Nothing to remove from customer’s subscription”)

Prompt(null)

return

Clear()

HomePosition()

PrintText(“Remove Subscription”, “”)

DisplayTable(Subscription, customer.subscriptions)

subscription = subscriptions.get(GetInt(“Select subscription from the list: “))

if subscription equal null

PrintText(“”, “Subscription is invalid”)

Prompt(null)

return

else if not customer.subscriptions.contains(subscription)

PrintText(“”, `Customer {customer.name} does not have a subscription {subscription.name}`)

Prompt(null)

return

customer.removeSubscription(subscription)

PrintText(“”, `Subscription {subscription.name} removed from {customer.name}`)

Prompt(null)

### CreateMagazineSupplement Method

Function CreateMagazineSupplement(subscriptions, customers)

Clear()

HomePosition()

PrintText(“CREATE MAGAZINE / SUPPLEMENT”, “”)

subscription = null

type = null

do

try

subscription = Subscription.getType(GetInt(“Is this a magazine? (1 for yes, 0 for no):”))

type = subscription.isMagazine() then “Magazine” else “Supplement”

catch(Exception)

while(subscription equals null)

subscription.id = subscriptions.isEmpty() then 1 else subscription.lastKey + 1

subscription.name = GetString(“Enter “ + type + “ name: “)

subscription.cost = GetDouble(“Enter “ + type + “ cost; “)

if subscription matches Supplement

magazines = subscriptions.filter(e=>e.isMagazine())

DisplayTable(Magazine, subscriptions)

magazine = subscriptions.get(GetInt(“Select one of the magazine: “))

if magazine equals null

PrintText(“Magazines does not exist”)

Prompt(null)

Return

else if not magazines.contains(magazine)

PrintText(magazine.name + “ is not a magazine”)

Prompt(null)

Return

else

supplement.setMagazine(magazine)

subscriptions.put(subscription.id, subscription)

PrintText(type + “ “ + subscription.name + “ created”)

Prompt(null)

### DeleteMagazineSupplement Method

Function DeleteMagazineSupplement(subscriptions, customers)

Clear()

HomePosition()

PrintText(“DELETE MAGAZINE / SUPPLEMENT”, “”)

DisplayTable(Subscription, subscriptions)

subscription = subscriptions.get(GetInt(“Select one of the magazine / supplement: “))

if subscription equals null)

PrintText(“Magazine / Supplement does not exist”)

Prompt(null)

Return

else if subscription matches Magazine

subscriptions.values().remove(e=>e.isSupplement() then e.magazine.equals(magazine) else e.equals(magazine)

else

subscriptions.remove(subscription.id)

PrintText(“Supplement “ + subscription.name + “ deleted successfully”)

for customer in customers

if subscription.isSupplement()

customer.removeSubscription(subscription)

else

for c\_subscription in customer.subscriptions

if c\_subscription.isSupplement() and c\_subscription.magazine.equals(subscription)or c\_subscription.equals(subscription)

customer.removeSubscription()

Prompt(null)

# Testing

The testing is done using a test matrix to demonstrate the program is systematically tested across different scenarios. Tables generated from the code are truncated in the test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Feature** | **Description** | **Expected outcome** | **Outcome** |
| VW\_1 | View weekly email | View weekly email from an associate called David Smith without a paying customer. | An email informing the customer their subscriptions are currently inactive. | Passed |
| VI\_1 | View monthly invoice email | View John Leroy’s invoice email. | An email displaying the subscription costs for him and his associates. | Passed |
| VI\_2 | View monthly invoice email after removing one of the associates. | View John Leroy’s invoice email after the removal of an associate customer: Alice Smith | An email displaying the subscription costs for him and his associates without Alice Smith’s subscription. | Passed |
| VI\_3 | View monthly invoice email that does not contain any subscriptions. | View Ashton with no subscription from himself and his associates. | An email informing the paying customer that no subscriptions were made by him and his associate. | Passed |
| VC\_1 | View paying customer data without subscription. | View New Wei Nern’s (Paying Customer) data. | Displays paying customer data with payment method credentials. | Passed |
| VC\_2 | View associate customer data without subscription. | View Adam West’s (Associate Customer) data | Displays associate customer data. | Passed |
| VC\_3 | View associate customer data with subscription. | View David Smith’s (Associate customer) with subscriptions. | Displays associate customer data alongside the subscriptions. | Passed |
| AC\_1 | Add a new paying customer. | Add a new paying customer called Adele. | Customer is added and appears in the list. | Passed |
| AC\_2 | Add a new associate customer. | Add a new associate customer called Dennis and assign a valid paying customer. | Creates an associate customer with an assigned paying customer. | Passed |
| AC\_3 | Attempt to add a new associate customer with non-existent paying customer. | Add a new customer called Jerry and assign a non-existent paying customer. | Associate is not added; error message “Unable to find payer by ID.” Is shown. | Passed |
| AC\_4 | Attempt to add a new associate customer with another associate customer. | Add a new customer called Jerry and assign an associate customer. | Cancels adding of new customer after displaying “Customer is not a paying customer” | Passed |
| DC\_1 | Delete a paying customer. | Delete John Leroy (Paying Customer) from the system. | Removes John Leroy from the system. | Passed |
| DC\_1\_1 | Get the associated customer status linked to the deleted payer via weekly email. | Display weekly email from David Smith (Associate Customer) and Alice Smith (Associate Customer) after the deletion of John Leroy (Paying Customer) | Alice Smith and David Smith subscription should be inactive. | Passed |
| DC\_2 | Delete an associate customer. | Delete Alice Smith (Associate Customer) from the system. | Removes Alice Smith from the system. | Passed |
| DC\_2\_1 | Get the paying customer | Display invoice email from John Leroy (Paying Customer) | Alice Smith subscription should not show in John Leroy’s invoice. | Passed |
| DC\_3 | Delete an unknown customer. | Attempt to delete an unknown customer ID. | Cancels deletion of customer and display “Unable to find customer” | Passed |
| SP\_1 | Assign a paying customer to an associate customer while all the associates contain a paying customer. | Attempt to set a paying customer to an associate customer when all the associate customers already have a paying customer. | The system prevents reassignment and displays: “All the associates have a payer.” | Passed |
| SP\_2 | Select an invalid associate customer ID. | Attempt to assign a paying customer that is not part of the list. | The system blocks the operation and displays: “The ID is not part of the list.” | Passed |
| SP\_3 | Select an invalid paying customer ID | Attempt to assign an associate customer that is not part of the list. | The system blocks the operation and displays: “The ID is not part of the list.” | Passed |
| SP\_4 | Assign a paying customer to an associate customer. | Add a paying customer (New Wei Nern) to an associate (Adam West) | Assigns a paying customer to an associate customer. | Passed |
| RP\_1 | Select an invalid paying customer ID. | Attempt to select a paying customer that is not part of the list. | The system blocks the operation and displays: “The ID is not part of the list.” | Passed |
| RP\_2 | Select a paying customer that does not have an associate. | Select a paying customer (New Wei Nern) that does not contain an associate customer. | Display “Paying customer New Wei Nern does not have an associate” | Passed |
| RP\_3 | Select an invalid associate customer ID. | Attempt to select an associate customer that is not part of the list. | The system rejects the selection and displays: “The ID is not part of the list.” | Passed |
| RP\_4 | Delete an associate customer from a paying customer. | Delete associate (David Smith) from paying customer (John Leroy) | Deletes David Smith from John Leroy’s associate. | Passed |
| AS\_1 | Select an invalid customer. | Attempt to select an invalid customer. | The system blocks the operation and displays: “Unable to set subscription from a non-existing customer.” | Passed |
| AS\_2 | Select an invalid subscription. | Attempt to select an invalid subscription for a paying customer (New Wei Nern) | The system blocks the subscription and displays: “Subscription is invalid.” | Passed |
| AS\_3 | Select a supplement that the customer does not have a magazine subscription to. | Select a supplement subscription (MTV English) that the customer (New Wei Nern) does not own a magazine to. | The system blocks the action and displays: “Customer does not have the required magazine to read the subscription.” | Passed |
| AS\_4 | Add a magazine subscription to a customer | Add a magazine subscription (Daily News) to an associate customer (David Smith) | The system successfully adds Daily News to David Smith’s subscriptions. | Passed |
| RS\_1 | Select an invalid customer. | Attempt to select an invalid customer. | The system blocks the operation and displays: “Unable to remove subscription from a non-existing customer.” | Passed |
| RS\_2 | Remove subscription from a customer that does not have active subscription. | Attempt to remove a subscription from a customer (New Wei Nern) that does not have an active subscription. | Display “Nothing to remove from customer’s subscription. | Passed |
| RS\_3 | Select an invalid subscription. | Attempt to select an invalid subscription. | The system blocks the action and displays: “Subscription is invalid.” | Passed |
| RS\_4 | Remove a valid subscription that does not exist in customer’s subscription. | Attempt to remove a valid magazine (Daily News) that the customer (John Leroy) did not purchase. | Display “Customer John Leroy does not have a subscription Daily News” | Passed |
| RS\_5 | Remove a magazine from a customer | Remove magazine (MTV) from a customer (John Leroy) | Removes MTV from John Leroy’s subscription. | Passed |
| RS\_5\_1 | Display the list of subscriptions after the removal of the magazine. | View customer’s (John Leroy) subscription after the deletion of the magazine (MTV) | An empty list of subscriptions from John Leroy. | Passed |
| CMS\_1 | Select invalid magazine during creation of supplement | Attempt to select an invalid magazine | Display “Magazine does not exist” | Passed |
| CMS\_2 | Select a supplement during creation of supplement | Select supplement (MTV English) while attempting to create a supplement | Display “MTV English is not a magazine” | Passed |
| CMS\_3 | Create a supplement and assign it to a magazine | Create a supplement (MTV Japan) and assign it to a magazine (MTV) | Creates a new supplement called MTV Japan which is assigned to Magazine. | Passed |
| DMS\_1 | Select an invalid subscription | Attempt to select an invalid subscription | Display “Subscription does not exist” | Passed |
| DMS\_2 | Delete a magazine | Delete magazine (MTV) along with its supplements (MTV English) and (MTV Chinese) | Remove MTV, MTV English, MTV Chinese from customer’s subscription and subscription list. | Passed |
| DMS\_2\_1 | View all magazines and supplements | Display the magazines and supplements after the deletion of a magazine (MTV) | Views all magazines and supplements except for MTV and its subscriptions | Passed |
| DMS\_2\_2 | Checks customer’s subscription after deletion of the magazine | View customer’s (John Leroy) subscription after the magazine (MTV) is deleted. | View John Leroy’s subscription that does not contain MTV subscriptions. | Passed |
| DMS\_3 | Delete a supplement | Delete a supplement (MTV English) | Deletes MTV English from subscription list and customer’s subscription. | Passed |

**Test ID**: VW\_1

**Description**: View weekly from an associate called David Smith without a paying customer.   
**Prerequisite**: Remove David Smith as an associate of John Leroy or delete John Leroy as a customer.

**Data**:

Select one of the following options: 5  
Select one of the customer: 2

Select one of the following options: 1

**Expected Output**:

To: david\_smith2342@yahoo.com

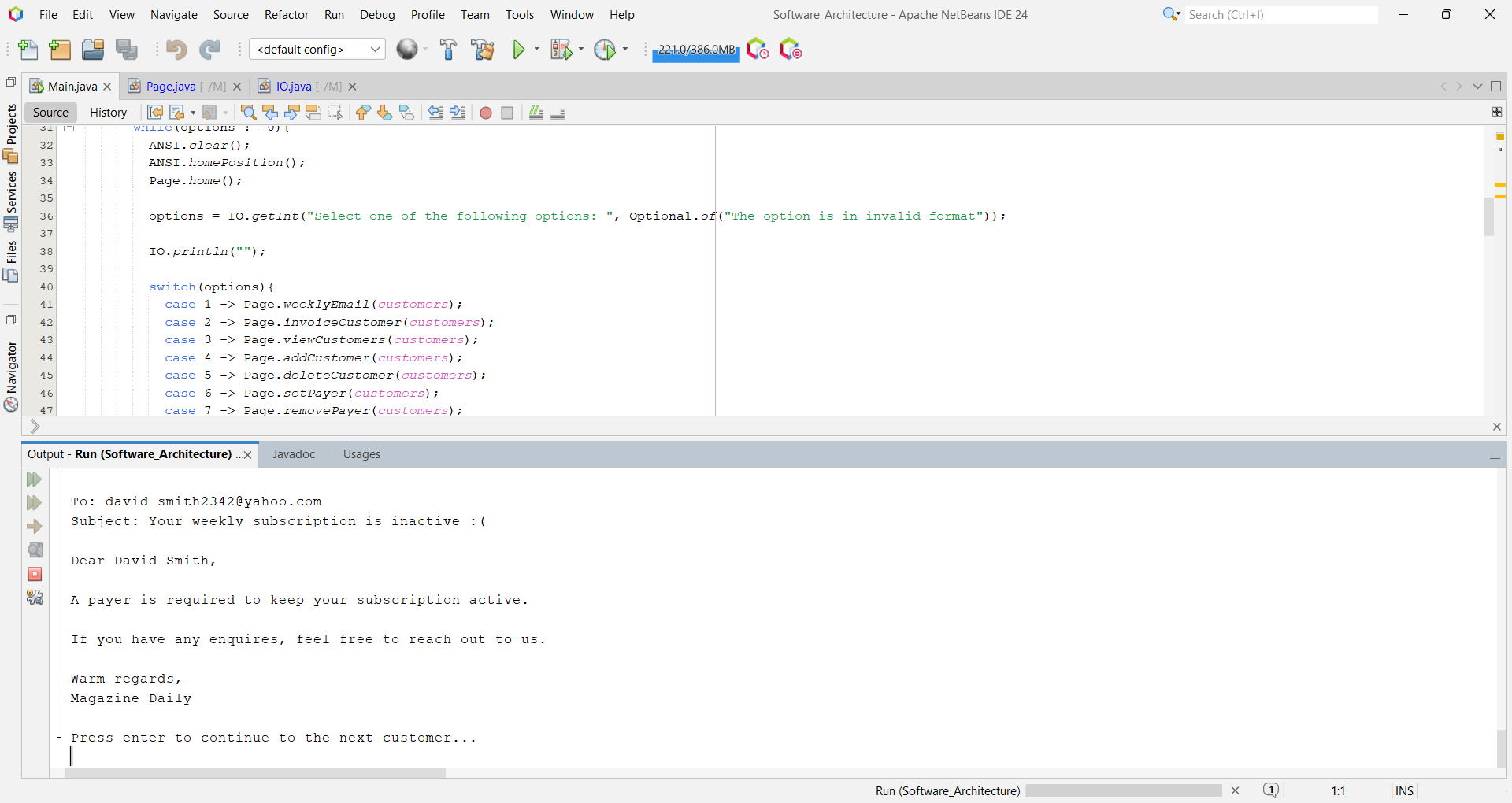
Subject: Your weekly subscription is inactive :(

Dear David Smith,

A payer is required to keep your subscription active.

If you have any enquires, feel free to reach out to us.

Warm regards,

Magazine Daily

**Test ID**: VI\_1

**Description**: View John Leroy’s invoice email.

**Data**:

Select one of the following options: 2  
**Expected Output**:

To: j\_leroy@gmail.com

Subject: Invoice for Paying Customer John Leroy

Dear John Leroy,

Thank you for being our valued customer!

Here is the summary for this month:

ID | Name | Cost | Type

1 | MTV | 5.9 | Magazine

2 | MTV Chinese | 3.5 | Supplement

3 | MTV English | 3.5 | Supplement

Total cost: 12.90

Associate Customer: David Smith

ID | Name | Cost | Type

4 | Daily News | 2.5 | Magazine

5 | Daily Sports | 2.8 | Supplement

Total cost for David Smith's subscription: $5.30

Associate Customer: Alice Smith

ID | Name | Cost | Type

1 | MTV | 5.9 | Magazine

1 | MTV | 5.9 | Magazine

2 | MTV Chinese | 3.5 | Supplement

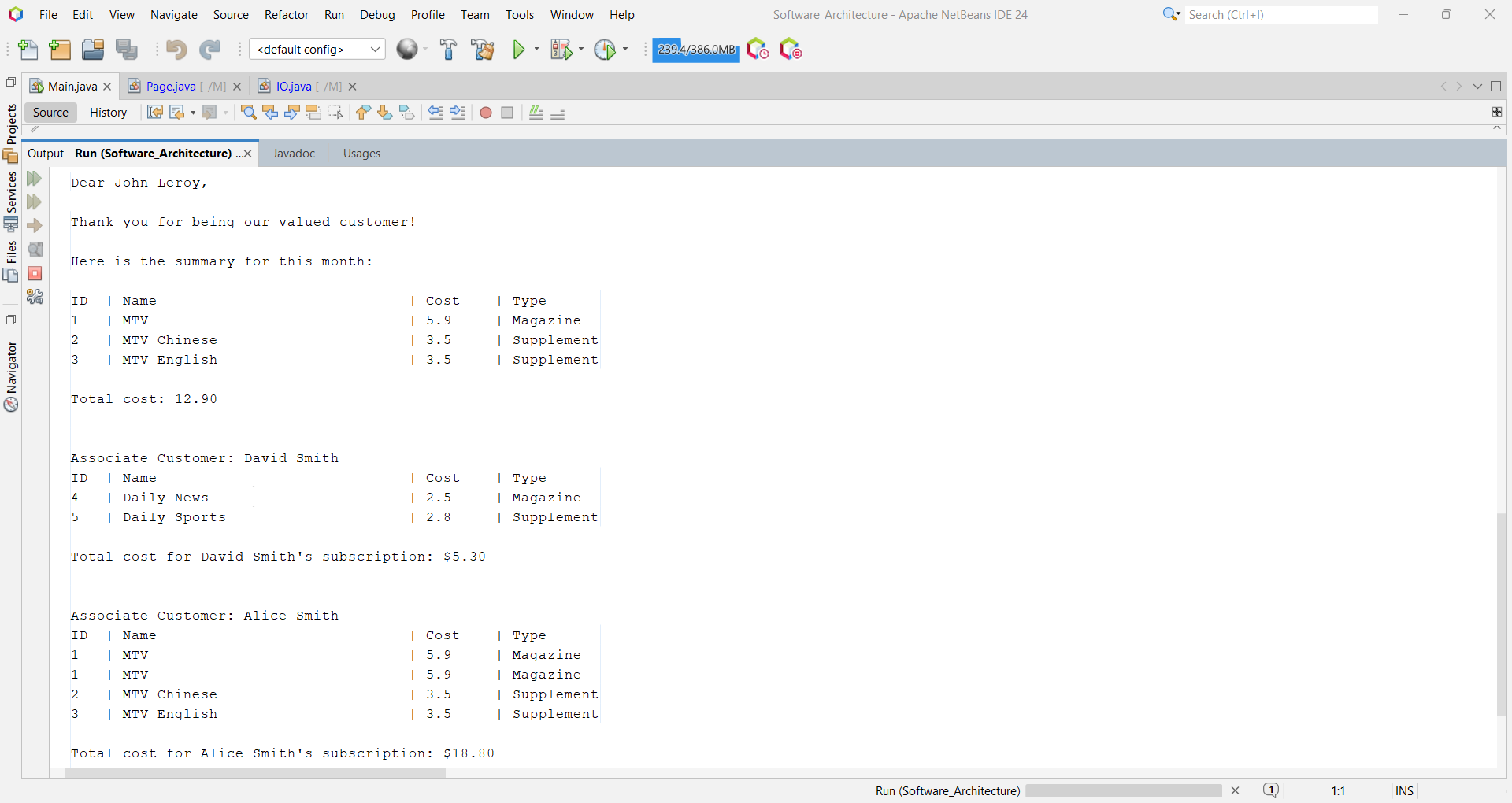
3 | MTV English | 3.5 | Supplement

Total cost for Alice Smith's subscription: $18.80

Here is the total cost of all the subscriptions: $37.00

Warm regards,

Magazine Daily



**Test ID**: VI\_2

**Description**: View John Leroy’s invoice email after the removal of an associate customer: Alice Smith.

**Prerequisite**: Remove Alice Smith as an associate of John Leroy or delete Alice Smith as a customer.

**Data**:

Select one of the following options: 2  
**Expected Output**:

To: j\_leroy@gmail.com

Subject: Invoice for Paying Customer John Leroy

Dear John Leroy,

Thank you for being our valued customer!

Here is the summary for this month:

ID | Name | Cost | Type

1 | MTV | 5.9 | Magazine

2 | MTV Chinese | 3.5 | Supplement

3 | MTV English | 3.5 | Supplement

Total cost: 12.90

Associate Customer: David Smith

ID | Name | Cost | Type

4 | Daily News | 2.5 | Magazine

5 | Daily Sports | 2.8 | Supplement

Total cost for David Smith's subscription: $5.30

Here is the total cost of all the subscriptions: $18.20

Warm regards,

Magazine Daily

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: VI\_3

**Description**: View Ashton with no subscription from himself and his associates.

**Prerequisite**: Ashton and Dave Luka must not subscribe to any magazine / supplement. Ashton must be a paying customer for Dave Luka.

**Data**:

Select one of the following options: 2  
**Expected Output**:

To: ash.ton@gmail.com

Subject: Invoice for Paying Customer Ashton

Dear Ashton,

Thank you for being our valued customer!

Here is the summary for this month:

No available subscription.

Associate Customer: Dave Luka

Associate customer does not have any subscription yet.

Total cost for Dave Luka's subscription: $5.30

Here is the total cost of all the subscriptions: $0.00

Warm regards,

Magazine Daily

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: VC\_1

**Description**: View New Wei Nern’s (Paying Customer) data.

**Data**:

Select one of the following options: 3

Enter customer ID to view its information: 1

**Expected Output**:

Name: New Wei Nern

Email: nwnisworking@gmail.com

Type: Paying Customer

Payment Details

Type: Credit Card

Card Number: 1234-1234-4321-4321

Card Expiry: 05/29

Customer does not have subscription

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: VC\_2

**Description**: View Adam West’s (Associate Customer) data.

**Data**:

Select one of the following options: 3

Enter customer ID to view its information: 4

**Expected Output**:

Name: Adam West

Email: ada\_wew@ty.com

Type: Associate Customer

Customer does not have subscription

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: VC\_3

**Description**: View David Smith’s (Associate customer) with subscriptions.

**Data**:

Select one of the following options: 3

Enter customer ID to view its information: 3

**Expected Output**:

Name: David Smith

Email: david\_smith2342@yahoo.com

Type: Associate Customer

ID | Name | Cost | Type

4 | Daily News | 2.5 | Magazine

5 | Daily Sports | 2.8 | Supplement

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: AC \_1

**Description**: Add a new paying customer called Adele.

**Data**:

Select one of the following options: 4

Is this a paying customer? (1 for Yes, 0 for No): 1

Enter customer name: Adele

Enter customer email: adele@yahoo.com.uk

PAYMENT METHOD

Select customer's payment method (1 for Credit Card, 2 for Direct Debit): 1

Enter credit card number: 1234-4532-1324-5123

Enter bank name: Bank

**Expected Output**:

Adele added successfully

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: AC \_2

**Description**: Add a new associate customer called Dennis and assign a valid paying customer.

**Data**:

Select one of the following options: 4

ADD CUSTOMER

Is this a paying customer? (1 for Yes, 0 for No): 0

Enter customer name: Dennis

Enter customer email: dennis@yahoo.com.sg

Select payer: 1

**Expected Output**:

Dennis added successfully

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: AC\_3

**Description**: Add a new customer called Jerry and assign a non-existent paying customer.

**Data**:

Select one of the following options: 4

ADD CUSTOMER

Is this a paying customer? (1 for Yes, 0 for No): 0

Enter customer name: Jerry

Enter customer email:jerry@gmail.com

Select payer: 3

**Expected Output**:

Customer is not a paying customer.A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: AC\_4

**Description**: Add a new customer called Jerry and assign an associate customer.

**Data**:

Select one of the following options: 4

ADD CUSTOMER

Is this a paying customer? (1 for Yes, 0 for No): 0

Enter customer name: Jerry

Enter customer email: jerry@gmail.com

LIST OF PAYING CUSTOMERS

Select payer: 3

**Expected Output**:

Customer is not a paying customer.A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: DC \_1

**Description**: Delete John Leroy (Paying Customer) from the system.

**Data**:

Select one of the following options: 5

DELETE CUSTOMER

Select one of the customer: 2

**Expected Output**:

New Wei Nern deleted successfully.

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: DC \_1\_1

**Description**: Display weekly email from David Smith (Associate Customer) and Alice Smith (Associate Customer) after the deletion of John Leroy (Paying Customer).

**Data**:

Select one of the following options: 1

**Expected Output**:

To: david\_smith2342@yahoo.com

Subject: Your weekly subscription is inactive :(

Dear David Smith,

A payer is required to keep your subscription active.

If you have any enquires, feel free to reach out to us.

Warm regards,

Magazine Daily

Press enter to continue to the next customer...

To: alicesmith@yahoo.com

Subject: Your weekly subscription is inactive :(

Dear Alice Smith,

A payer is required to keep your subscription active.

If you have any enquires, feel free to reach out to us.

Warm regards,

Magazine DailyA screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: DC\_2

**Description**: Delete Alice Smith (Associate Customer) from the system.

**Data**:

Select one of the following options: 5

DELETE CUSTOMER

Select one of the customer: 7

**Expected Output**:

Alice Smith deleted successfully.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Test ID**: DC\_2\_1

**Description**: Display invoice email from John Leroy (Paying Customer).

**Data**:

Select one of the following options: 2

**Expected Output**:

To: j\_leroy@gmail.com

Subject: Invoice for Paying Customer John Leroy

Dear John Leroy,

Thank you for being our valued customer!

Here is the summary for this month:

ID | Name | Cost | Type

1 | MTV | 5.9 | Magazine

2 | MTV Chinese | 3.5 | Supplement

3 | MTV English | 3.5 | Supplement

Total cost: 12.90

Associate Customer: David Smith

ID | Name | Cost | Type

4 | Daily News | 2.5 | Magazine

5 | Daily Sports | 2.8 | Supplement

Total cost for David Smith's subscription: $5.30

Here is the total cost of all the subscriptions: $18.20

Warm regards,

Magazine Daily

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: DC\_3

**Description**: Attempt to delete an unknown customer ID.

**Data**:

Select one of the following options: 5

Select one of the customer: 8

**Expected Output**:

Unable to find customer

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: SP\_1

**Description**: Attempt to set a paying customer to an associate customer when all the associate customers already have a paying customer.

**Prerequisite:** Adam West already has a paying customer assigned.

**Data**:

Select one of the following options: 6

**Expected Output**:

All the associates have a payer

**A screenshot of a computer

AI-generated content may be incorrect.**

**Test ID**: SP\_2

**Description**: Attempt to assign an associate customer that is not part of the list.

**Data**:

Select one of the following options: 6

Enter associate ID: 1

**Expected Output**:

The ID is not part of the list A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: SP\_3

**Description**: Attempt to assign a paying customer that is not part of the list.

**Data**:

Select one of the following options: 6

Enter associate ID: 4

Enter paying customer ID: 3

**Expected Output**:

The ID is not part of the list **A screenshot of a computer

AI-generated content may be incorrect.**

**Test ID**: SP\_4

**Description**: Add a paying customer (New Wei Nern) to an associate (Adam West)

**Data**:

Select one of the following options: 6

SET PAYER

Enter associate ID: 4

Enter paying customer ID: 1

**Expected Output**:

Payer New Wei Nern set to associate Adam West successfully

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: RP\_1

**Description**: Attempt to select a paying customer that is not part of the list.

**Data**:

Select one of the following options: 7

REMOVE PAYER

Enter ID of the paying customer: 8

**Expected Output**:

ID is not part of the list

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: RP \_2

**Description**: Select a paying customer (New Wei Nern) that does not contain an associate customer.

**Data**:

Select one of the following options: 7

REMOVE PAYER

Enter ID of the paying customer: 1

**Expected Output**:

Paying customer New Wei Nern does not have an associate.

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: RP\_3

**Description**: Attempt to select an associate customer that is not part of the list.

**Data**:

Select one of the following options: 7

REMOVE PAYER

Enter ID of the paying customer: 2

Enter ID of the associate customer: 4

**Expected Output**:

ID is not part of the list

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: RP\_4

**Description**: Delete associate (David Smith) from a paying customer (John Leroy)

**Data**:

Select one of the following options: 7

REMOVE PAYER

Enter ID of the paying customer: 2

Enter ID of the associate customer: 3

**Expected Output**:

Paying customer John Leroy removed David Smith from associate list

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: AS\_1

**Description**: Select a non-existent customer when adding a subscription.

**Data**:

Select one of the following options: 9

ADD SUBSCRIPTION

Select customer from the list: 8

**Expected Output**:

Unable to set subscription from a non-existing customer

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: AS \_2

**Description**: Attempt to select an invalid subscription for a paying customer (New Wei Nern)

**Data**:

Select one of the following options: 9

ADD SUBSCRIPTION

Select customer from the list: 1

Select subscription from the list: 8

**Expected Output**:

Subscription is invalid.

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: AS \_3

**Description**: Select a supplement subscription (MTV English) that the customer (New Wei Nern) does not own a magazine to.

**Data**:

Select one of the following options: 9

ADD SUBSCRIPTION

Select customer from the list: 1

Select subscription from the list: 3

**Expected Output**:

Customer do not have the required magazine to read the subscription.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Test ID**: AS \_4

**Description**: Add a subscription to an associate customer (David Smith)

**Data**:

Select one of the following options: 9

ADD SUBSCRIPTION

Select customer from the list: 3

Select subscription from the list: 4

**Expected Output**:

Subscription Daily News added to David Smith

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: RS\_1

**Description**: Attempt to select an invalid customer.

**Data**:

Select one of the following options: 10

REMOVE SUBSCRIPTION

Select customer from the list: 8

**Expected Output**:

Unable to remove subscription from a non-existing customer

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: RS\_2

**Description**: Attempt to remove a subscription from a customer (New Wei Nern) that does not have an active subscription.

**Data**:

Select one of the following options: 10

REMOVE SUBSCRIPTION

Select customer from the list: 1

**Expected Output**:

Nothing to remove from customer's subscription

**A screenshot of a computer

AI-generated content may be incorrect.**

**Test ID**: RS\_3

**Description**: Attempt to select an invalid subscription.

**Data**:

Select one of the following options: 10

REMOVE SUBSCRIPTION

Select customer from the list: 2

Select subscription from the list: 10

**Expected Output**:

Subscription is invalid

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: RS\_4

**Description**: Attempt to remove a valid magazine (Daily News) that the customer (John Leroy) did not purchase.

**Data**:

Select one of the following options: 10

REMOVE SUBSCRIPTION

Select customer from the list: 2

Select subscription from the list: 4

**Expected Output**:

Customer John Leroy does not have a subscription Daily NewsA screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: RS\_5

**Description**: Remove magazine (MTV) from a customer (John Leroy)

**Data**:

Select one of the following options: 10

REMOVE SUBSCRIPTION

Select customer from the list: 2

Select subscription from the list: 1

**Expected Output**:

Subscription MTV removed from John Leroy

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: RS\_5\_1

**Description**: View customer’s (John Leroy) subscription after the deletion of the magazine (MTV)

**Data**:

Select one of the following options: 3

VIEW CUSTOMER

Enter customer ID to view its information: 2

**Expected Output**:

Customer does not have subscription.

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: CMS\_1

**Description**: Attempt to select an invalid magazine

**Data**:

Select one of the following options: 11

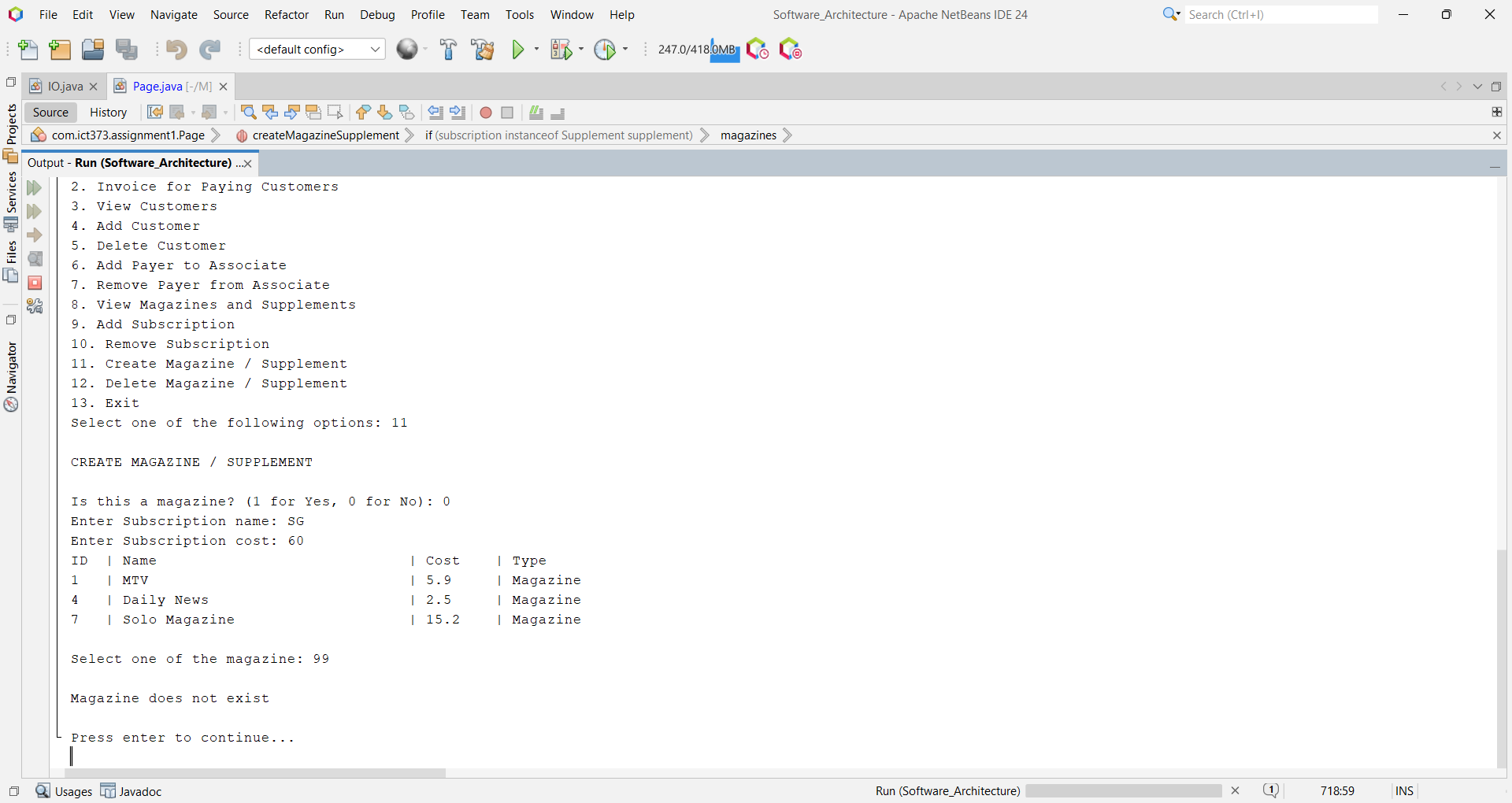
Is this a magazine? (1 for Yes, 0 for No): 0

Enter Supplement name: SG

Enter Supplement cost: 60

Select one of the magazine: 99

**Expected Output**:

Magazine does not exist

**Test ID**: CMS\_2

**Description**: Select a supplement during creation of supplement

**Data**:

Select one of the following options: 11

Is this a magazine? (1 for Yes, 0 for No): 0

Enter Supplement name: MTV Japan

Enter Supplement cost: 11

Select one of the magazine: 3

**Expected Output**:

MTV English is not a magazineA screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: CMS\_3

**Description**: Create a supplement and assign it to a magazine

**Data**:

Select one of the following options: 11

Is this a magazine? (1 for Yes, 0 for No): 0

Enter Supplement name: MTV Japan

Enter Supplement cost: 11

Select one of the magazine: 1

**Expected Output**:

Supplement MTV Japan createdA screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: DMS\_1

**Description**: Select an invalid subscription

**Data**:

Select one of the following options: 12

Select one of the magazine / supplement : 9

**Expected Output**:

Magazine / Supplement does not exist

A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: DMS\_2

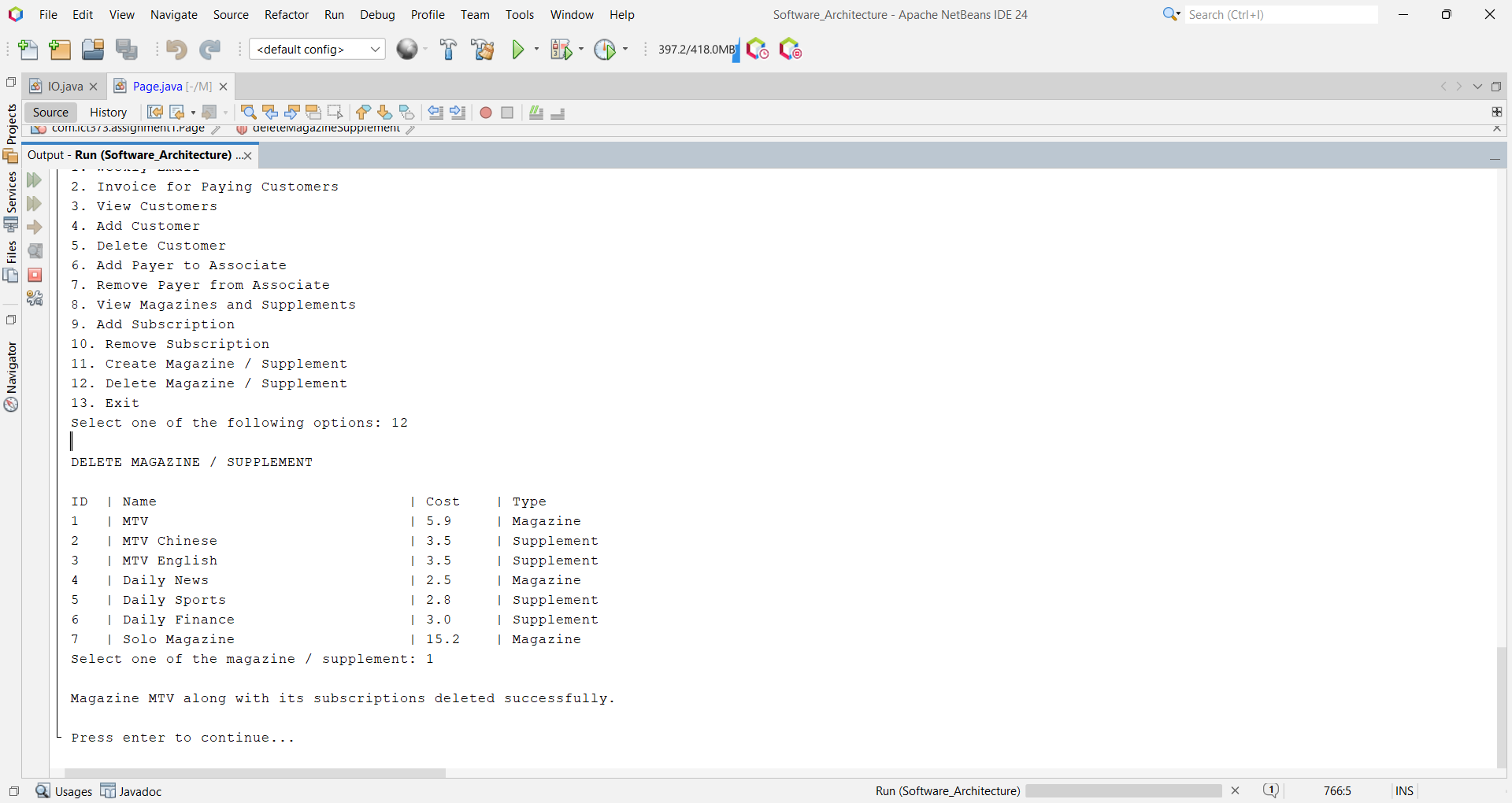
**Description**: Delete magazine (MTV) along with its supplements (MTV English) and (MTV Chinese)

**Data**:

Select one of the following options: 12

Select one of the magazine / supplement : 1

**Expected Output**:

Magazine MTV along with its subscriptions deleted successfully. 

**Test ID**: DMS\_2\_1

**Description**: Delete magazine (MTV) along with its supplements (MTV English) and (MTV Chinese)

**Data**:

Select one of the following options: 8

**Expected Output**:

ID | Name | Cost | Type

4 | Daily News | 2.5 | Magazine

5 | Daily Sports | 2.8 | Supplement

6 | Daily Finance | 3.0 | Supplement

7 | Solo Magazine | 15.2 | Magazine A screenshot of a computer

AI-generated content may be incorrect.

**Test ID**: DMS\_2\_2

**Description**: View customer’s (John Leroy) subscription after the magazine (MTV) is deleted.

**Data**:

Select one of the following options: 3

Enter customer ID to view its information: 2

**Expected Output**:

Name: John Leroy

Email: j\_leroy@gmail.com

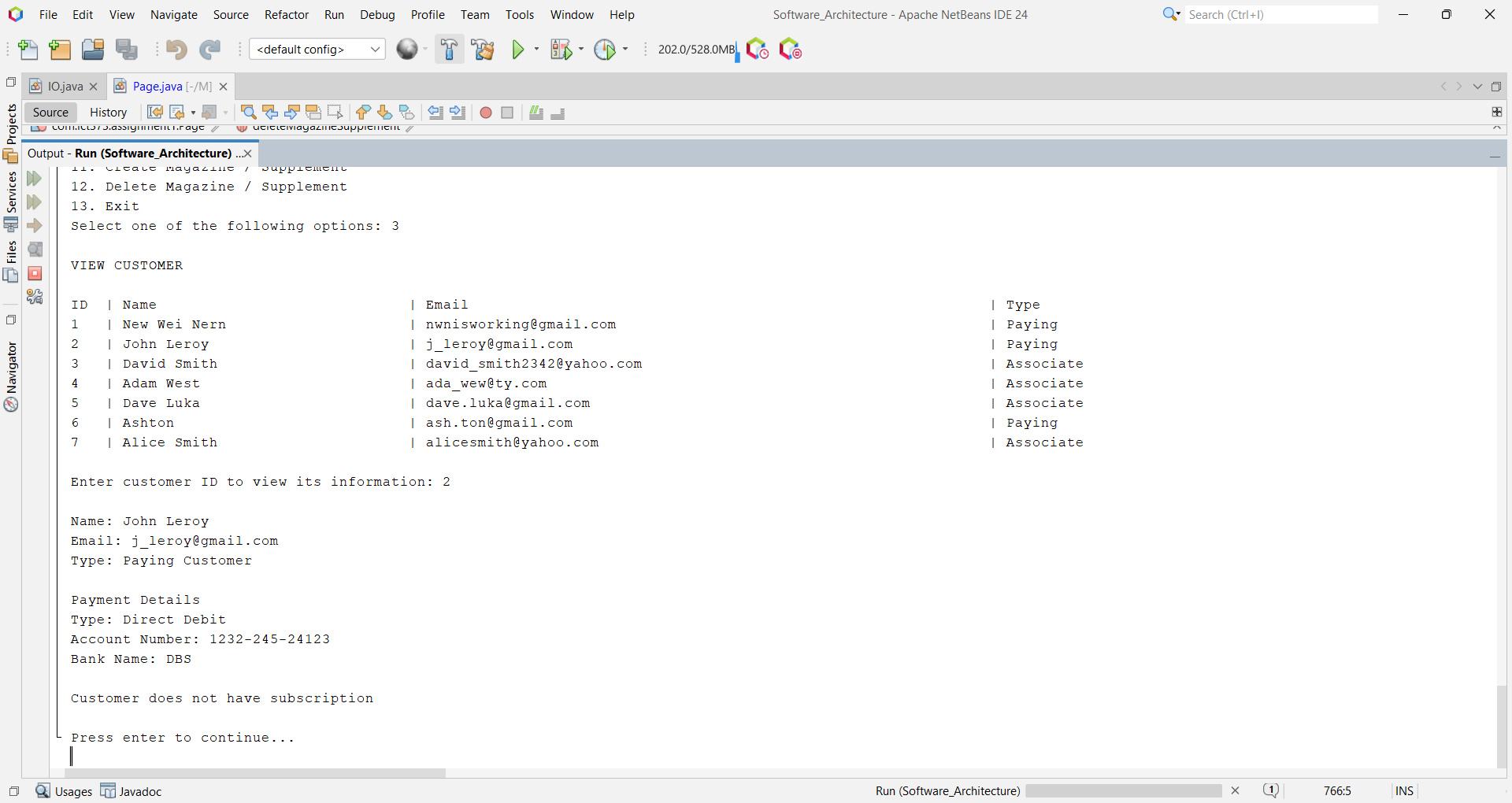
Type: Paying Customer

Payment Details

Type: Direct Debit

Account Number: 1232-245-24123

Bank Name: DBS

Customer does not have subscription 

**Test ID**: DMS\_3

**Description**: Delete a supplement (MTV English)

**Data**:

Select one of the following options: 12

Select one of the magazine / supplement: 3

**Expected Output**:

Supplement MTV English deleted successfully

