Shay Snyder

Honors 1260 – Spring 2020

Credit Card List Design Document

Problem: Spring 2020 Honors 1260 Project 3: Credit Card List

### List of Inputs, Outputs, and Processing Required

INPUTS

* Allow the user to add a CreditCard object to an ArrayList of CreditCard objects
* Use a menu system to afford the user a visually appealing interface that makes interfacing with the program and its various methods much easier.

OUTPUTS

* A CreditCard object from a given index **n**
* A list of all CreditCard objects with a particular card holder
* A list of all CreditCard objects that have not expired at the time the program has run
* For the Find and Retrieve methods, display the results to verify they are correct
* Display all credit card information about every credit card in the list

PROCESSING

* Retrieve the CreditCard currently in position **n** in the list
* Remove a CreditCard from the list
* Find a CreditCard with a particular number in the list
* Find all CreditCard objects that have not expired at the time the program is run
* Sort the CreditCard objects in the list by credit card number
* Sort the CreditCard objects in the list by card holder’s name
* All methods should function properly even if the list is empty or the requested item is not in the list

### Identification of Classes and Their Responsibilities

**Class name: CreditCardListDriver**

Responsibilities:

**Class name:** **CreditCardList**

Responsibilities:

* Contain a private attribute representing an ArrayList of CreditCard objects
* Add a CreditCard to the list
* Retrieve the CreditCard currently in position **n** in the list
* Remove a CreditCard from the list
* Find all CreditCard Objects with a particular number from the list
* Find all CreditCard objects that hae not expired at the time the program is run
* Sort the CreditCard objects in the list by credit card number
* Sort the CreditCard objects in the list by the card holder’s number

### UML Class Diagram

A screenshot of a cell phone

Description automatically generated

### Algorithms

**Class:** CreditCardListDriver

**Method:** main(args[] : String) : void

**Desciption:** The main() method in the driver will act as the basis of functionality or project 4, a.k.a every method call will stem from this point

**Class:** CreditCardListDriver

**Method:** intro() : void

**Description:** The intro() method will use JOptionPane to display a promt to the user that introduces them to the program and details its functionality

**Class:** CreditCardListDriver

**Method:** core() : void

**Description:** The core() method will be used to interact with the CreditCardList class. Thus, allowing the user to create a custom list of credit cards and manipulate the list as they wish. A JOptionPane Dialog Box will be used to allow the user the opportunity to decide how the program will behave. This will be housed in a while loop that will run until the user decided to end the program.

Decision = show JOptionPane dialog box with input, remove, retrieve, and sort options

If (decision = input)

Use JOptionPane to allow the user to enter Cardholder’s name, card number, and expiration date

Call the CreditCardList addCard() method to add the credit card to the ArrayList

Else if (decision = remove)

Use JOptionPane to allow the user to enter an int that represents the index of the card they would like to remove

Call the CreditCardList removeCard() method with the aforementions integer

Else if (decision = retrieve)

Call the CreditCardListDriver getCards() method

Else if (decision = sort)

Call the CreditCardListDriver sortCards() method

**Class:** CreditCardListDriver

**Method:** exit() : void

**Description:** The exit() method will use JOptionPane to display a prompt to the user that shows our appreictaion for their using of the program.

**Class:** CreditCardListDriver

**Method:** getCards() : void

**Description:** The getCards() will be called when the user selects to retrieve cards in the core() method. A JOptionPane dialog box will be used to give them the opportunity to get cards by expiration date, index, cardholder name, card number, or all of them.

Decision = show JOptionPane dialog box with expiration, index, name, number, or all options

If (decision = expiration)

Call the CreditCardList getCardsByExpir() method

Show the returned ArrayList of CreditCard objects

Else if (decision = index)

Use JOptionPane to allow the user to enter an integer that represents the CreditCard object thet would like to retrieve

Call the CreditCardList getCardByIndex() method with the appropriate integer argument

Show the returned CreditCard object

Else if (decision = name)

Use JOptionPane to allow the user to enter an String variable that represents the cardholder’s name

Call the CreditCardList getCardByName() method with the appropriate String argument

Show the returned ArrayList of CreditCard objects

Else if (decision = number)

Use JOptionPane to allow the user to enter a String variable that represents the desired credit card number

Call the CreditCardList getCardByNumber() method with the appropriate String argument

Show the returned CreditCard object

Else if (decision = number)

Call the CreditCardList getAllCards() method

Show the returned ArrayList of CreditCard objects

**Class:** CreditCardListDriver

**Method:** sortCards() : void

**Description:** The sortCards() method will be called from the core() method in the driver.

Decision = JOptionPane dialog box (sort by number or sort by name)

If (decision = sort by number)

Call the CreditCardList sortCardsByNumber() method

Else if (decision = sort by name)

Call the CreditCardList sortCardsByName() method

**Class:** CreditCardList

**Method:** CreditCardList()

**Description:** Default constructor for the CreditCardList class. An empty ArrayList of CreditCard objects is created

**Class:** CreditCardList

**Method:** addCard(cardHolderName : String, cardNumber : String, expirationDate : String) : void

**Description:** use the arguments to create a new CreditCard object and put it in the array list

CreditCard card = new CreditCard()

Card.setCardHolderName(cardHolderName)

Card.setCardNumber(cardNumber)

Card.setExpirationDate(expirationDate)

Cards.add(card)

**Class:** CreditCardList

**Method:** removeCard(index : int) : boolean

**Description:** remove the card object at index n

Try Card.remove(n)

If successful, return true

If unsuccessful, return false

**Class:** CreditCardList

**Method:** getCardsByExpir() : ArrayList<CreditCard>

**Description:** Create a new ArrayList<CreditCard> object to temporily hold the original dataset

Create a new array list to hold te desired cards

For each item in the copied array list:

Use the isCardExpired() method

If card is expired:

Add the CreditCard object to the newly created list

Return the new array list

**Class:** CreditCardList

**Method:** getCardsByIndex(index : int) : CreditCard

**Description:** return the Card object at a given index n within the list

return cards.get(n)

**Class:** CreditCardList

**Method:** getCardsByName(name : String) : ArrayList<CreditCard>

**Description:** use the argumented String and determine if any of the CreditCard objects in the cards list have the same cardholder name, return those CreditCard objects whose cardholder’s name matches in the form of a new ArrayList.

Create a new array list to hold the desired cards

For each item in the copied array list:

If (card holder name = desired name):

Add the reditCard object to the newly created list

Return the new array list

**Class:** CreditCardList

**Method:** getCardsByNumber(number : String) : CreditCard

**Description:** use the argumented String and determine if any of the CreditCard objects in the cards list have the same card number, return those CreditCard objects whose card number matches in the form of a new ArrayList

Create a new array list to hold the desired cards

For each item in the copied array list:

If (card number = desired number):

Add the reditCard object to the newly created list

Return the new array list

**Class:** CreditCardList

**Method:** getAllCards() : ArrayList<CreditCard>

**Description:** return the entire ArrayList

return cards

**Class:** CreditCardList

**Method:** sortCardsByNumber() : void

**Description:** use the sequential search algorithm to loop through the entire array amd sort by card number

Create a new array list to hold a copy of the original

Empty the original ArrayList

Until every item has been removed:

Use sequential search algorithm to find the lowest card number left in the copied array

Add the lowest card to the original ArrayList

Remove the lowest card from copied list

Repeat

Return the sorted ArrayList

**Class:** CreditCardList

**Method:** sortCardsByName() : void

**Description:** use the sequential search algorithm to loop through the entire array amd sort by cardholder name

Create a new array list to hold a copy of the original

Empty the original ArrayList

Until every item has been removed:

Use sequential search algorithm to find the lowest cardholdler name left in the copied array

Add the lowest card to the original ArrayList

Remove the lowest card from copied list

Repeat

Return the sorted ArrayList

### Test Cases

This information will be initially entered into the program before . Methods and expected outputs will be listed below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| John Doe | 378282246310005 | 01/2020 | Expired |
| John Cena | 371449635398431 | 02/2020 | Expired |
|  |  |
| Katherine Booher | 378734493671000 | 04/2019 | Expired |
| Justin Timberlake | 9610591081018250 | 12/2021 | Invalid |
|  |  |
| Krikor Faddey | 6011111111111117 | 04/2023 | Valid |
| Dmittrii Sasha | 6011000990139424 | 05/2022 | Valid |
| Leonty Natali | 5425233430109903 | 04/2023 | Valid |
| Polina Lazar | 4917484589897107 | 05/2025 | Valid |

**Test Case #1 (Testing the addCard() method)**

Input the following card using the addCard() method

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| Bob the Builder | 2222 4053 4324 8877 | 04/2023 | Valid |

Expected output when calling the getAllCards() method:

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| John Doe | 378282246310005 | 01/2020 | Expired |
| John Cena | 371449635398431 | 02/2020 | Expired |
|  |  |
| Katherine Booher | 378734493671000 | 04/2019 | Expired |
| Justin Timberlake | 9610591081018250 | 12/2021 | Invalid |
|  |  |
| Krikor Faddey | 6011111111111117 | 04/2023 | Valid |
| Dmittrii Sasha | 6011000990139424 | 05/2022 | Valid |
| Leonty Natali | 5425233430109903 | 04/2023 | Valid |
| Polina Lazar | 4917484589897107 | 05/2025 | Valid |
| Bob the Builder | 2222 4053 4324 8877 | 04/2023 | Valid |

**Test Case #2 (Test the removeCard() method)**

Expected output of Cards.remove(0):

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| John Cena | 371449635398431 | 02/2020 | Expired |
|  |  |
| Katherine Booher | 378734493671000 | 04/2019 | Expired |
| Justin Timberlake | 9610591081018250 | 12/2021 | Invalid |
|  |  |
| Krikor Faddey | 6011111111111117 | 04/2023 | Valid |
| Dmittrii Sasha | 6011000990139424 | 05/2022 | Valid |
| Leonty Natali | 5425233430109903 | 04/2023 | Valid |
| Polina Lazar | 4917484589897107 | 05/2025 | Valid |

**Test Case #3 (Test the getCardsByExpir() method)**

Expected result of Cards.getCardsByExpir()

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| John Doe | 378282246310005 | 01/2020 | Expired |
| John Cena | 371449635398431 | 02/2020 | Expired |
|  |  |
| Katherine Booher | 378734493671000 | 04/2019 | Expired |

**Test Case #4 (Test getCardByIndex)**

Expected result of Cards.getCardByIndex(0):

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| John Doe | 378282246310005 | 01/2020 | Expired |

**Test Case #5 (Test getCardsByName() method)**

Expected result of Cards.getCardsByName(“johndoe”)

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| John Doe | 378282246310005 | 01/2020 | Expired |

**Test Case #6 (Test getCardsByNumber() method)**

Expected result of Cards.getCardByNumber(“6011000990139424”)

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| Dmittrii Sasha | 6011000990139424 | 05/2022 | Valid |

**Test Case #7 (Test getAllCards() method)**

Expected results of Cards.getAllCards():

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| John Doe | 378282246310005 | 01/2020 | Expired |
| John Cena | 371449635398431 | 02/2020 | Expired |
|  |  |
| Katherine Booher | 378734493671000 | 04/2019 | Expired |
| Justin Timberlake | 9610591081018250 | 12/2021 | Invalid |
|  |  |
| Krikor Faddey | 6011111111111117 | 04/2023 | Valid |
| Dmittrii Sasha | 6011000990139424 | 05/2022 | Valid |
| Leonty Natali | 5425233430109903 | 04/2023 | Valid |
| Polina Lazar | 4917484589897107 | 05/2025 | Valid |

**Test Case #8 (Test sortCardsByNumber() method)**

Input: Cards.sortCardsByNumber()

Expected results:

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| John Cena | 371449635398431 | 02/2020 | Expired |
| John Doe | 378282246310005 | 01/2020 | Expired |
| Katherine Booher | 378734493671000 | 04/2019 | Expired |
| Polina Lazar | 4917484589897107 | 05/2025 | Valid |
| Leonty Natali | 5425233430109903 | 04/2023 | Valid |
| Dmittrii Sasha | 6011000990139424 | 05/2022 | Valid |
| Krikor Faddey | 6011111111111117 | 04/2023 | Valid |
| Justin Timberlake | 9610591081018250 | 12/2021 | Invalid |

**Test case #9 (Test sortCardsByName() method)**

Input: Cards.sortCardsByName()

Expected Results:

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| Dmittrii Sasha | 6011000990139424 | 05/2022 | Valid |
| John Cena | 371449635398431 | 02/2020 | Expired |
| John Doe | 378282246310005 | 01/2020 | Expired |
| Justin Timberlake | 9610591081018250 | 12/2021 | Invalid |
| Katherine Booher | 378734493671000 | 04/2019 | Expired |
| Krikor Faddey | 6011111111111117 | 04/2023 | Valid |
| Leonty Natali | 5425233430109903 | 04/2023 | Valid |
| Polina Lazar | 4917484589897107 | 05/2025 | Valid |

**Test Case #10**

Attempt to add a credit card via the addCreditCard() method but input null values.

Expected results: No error should be thrown, just don’t add the card

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder’s Name** | **Card Number** | **Expiration Date** | **Expected Output** |
| John Doe | 378282246310005 | 01/2020 | Expired |
| John Cena | 371449635398431 | 02/2020 | Expired |
|  |  |
| Katherine Booher | 378734493671000 | 04/2019 | Expired |
| Justin Timberlake | 9610591081018250 | 12/2021 | Invalid |
|  |  |
| Krikor Faddey | 6011111111111117 | 04/2023 | Valid |
| Dmittrii Sasha | 6011000990139424 | 05/2022 | Valid |
| Leonty Natali | 5425233430109903 | 04/2023 | Valid |
| Polina Lazar | 4917484589897107 | 05/2025 | Valid |

**Test Case #11**

Attempt to remove a credit card that isn’t in the list using removeCard() method

Expected results: No error should be thrown, just don’t remove anything and display a prompt

**Test Case #12**

Attempt to retrieve a card from an index that isn’t in the list

Expected results: No error should be thrown, just don’t retrieve a card and display a prompt saying that no card was able to be accessed

**Test Case #13**

Attempt to retrieve a card by name that isn’t in the list

Expected results: No error should be thrown, just don’t retrieve a card and display a prompt saying that no card was able to be accessed

**Test Case #14**

Attempt to retrieve a card by number that isn’t in the list

Expected results: No error should be thrown, just don’t retrieve a card and display a prompt saying that no card was able to be accessed

**Test Case #15**

When no cards are in the list and the user tries to getAllCards(), sortCardsByNumber(), or sortCardsByName(), No error should be thrown, just don’t retrieve a card and display a prompt saying that no card was able to be accessed