ICT167 Principles of Computer Science

Assignment 1

Jin Cherng Chong

33170193

Murdoch University

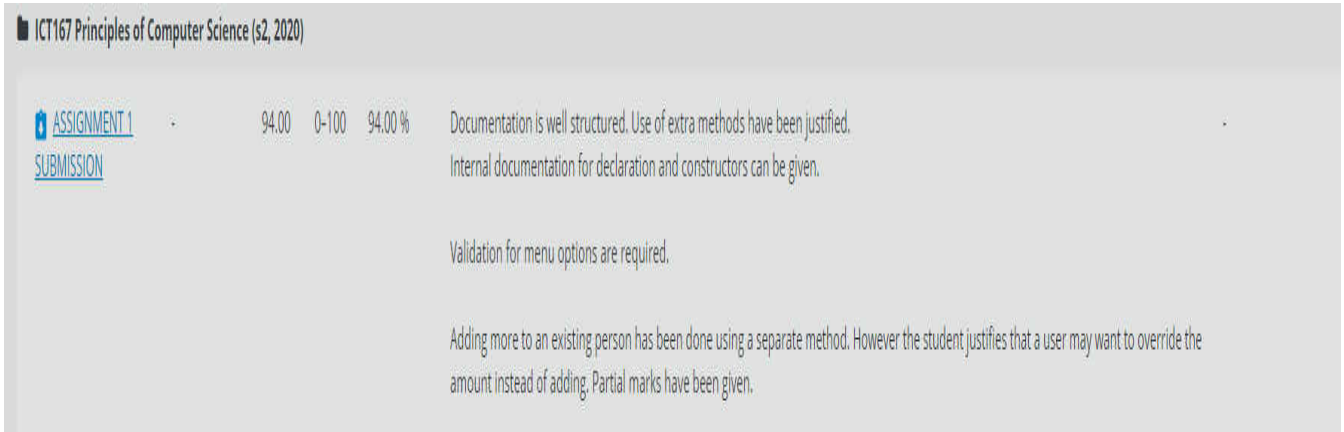


Table of content

Introduction

This documentation represents the external documentation for Jin Cherng Chong ICT167 Assignment 1 program. The document outlines the planning, constructing, and testing that went into the development of the program. The files that are referenced throughout the documentation include- Client.java and Change.java

Change.java contains the change class while client.java contains the client program. This documentation is for version: 0.01 which is the latest version as of 4/09/2020. This program is a money changer program where money given by a person (referred to as the change amount) is exchanged for the equivalent coin amount. The program is called change machine.

Requirements/Specifications

This money changer program records the amount of money a person has given and returns the same amount of money but in coins. The attribute “amountOfChange” represents the change amount a person has given. While the attributes- “numOfFiftyCents”, “numOfTwentyCents”, “numOfTenCents” and “numOfFiveCents” represent the different possible coin values the amountOfChange can be divided up into. A systematic approach is utilised when the amount of money given by a person is exchanged for coins. The larger valued coins will be given out first to reduce the total amount of coins given. So for example if a person gives 100 cents the program will return the change consisting of 2x 50 cent coins and not 10x 10 cent coins. Every person will have a unique identifiable name. So if a repeated name is entered then the associated amountOfChange is added to the existing entered amountOfChange.

The assumptions for the program include-

- Assume the amount of money a person gives (amountOfChange) will be in cents only
- Assume user will input first name of a person only and it is only a one string first name. This entry excludes the person's middle name and last name
- Assume user will input data of the correct data type
- Assume the user doesn't name a person: "None"
- Assume every person has a unique identifiable name. Therefore, no two people will have the same name
- Assume the currency is AUD
- Assume that there is no GST involved
- Assume the user will have no preference in what coin he wants. For example- the client doesn't demand change be In 50 cent coins only
- Assume the question "Do you have more person to enter" is only outputted when a new person is entered. So the question will not show when a user enters an existing person again

Class	Attributes	Responsibilities	Operations
Change	name amountOfChange numOfFiftyCents numOfTwentyCents numOfTenCents numOfFiveCents	Receive name of user Validate name of user	+ SetName()
		Receive change amount Validate change amount	+ SetAmountOfChange()
		Receive change amount and add to existing total change amount for an existing person	+ AddAmountOfChange()
		Receive number of coins for different coin value types	+ SetNumOfFiftyCents() + SetNumOfTwentyCents() + SetNumOfTenCents() + SetNumOfFiveCents()
		Return name of user to client	+ GetName()
		Return to client the number of fifty cent coins given to a user	+ GetNumOfFiftyCents()
		Return to client the number of twenty cent coins given to a user	+ GetNumOfTwentyCents()
		Return to client the number of ten cent coins given to a user	+ GetNumOfTenCents()
		Return to client the number of five cent coins given to a user	+ GetNumOfFiveCents()
		Returns to the client the initial (unset) instance variables of the object	+ WriteInitialRecord()

Class	Responsibilities	Operations
Client	Input name of user Input change amount of user	+ Main()

	Calculate the denominations of coin change amount for a user Check whether another person should be added Check same name Display Menu Process menu options Display total coins given for each denomination	+ CalculateDenominations() + ValidateAddAnotherPerson() + ChkIsNewName() + DspMenu() + SelectOption1() + SelectOption2() + SelectOption3() + SelectOption4() +DenominationsBreakdown() + DspDenominations()

User Guide

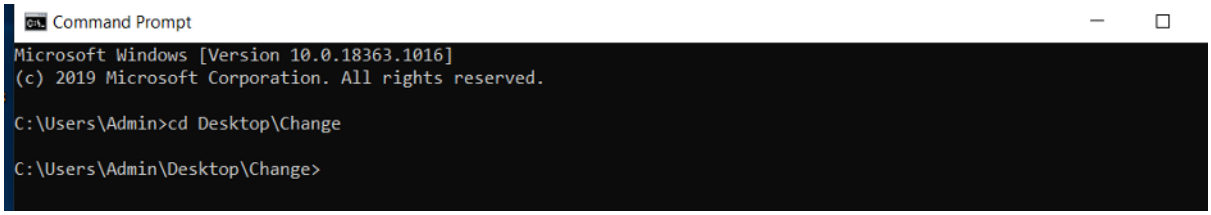
<u>Option 1- Run with jar</u>

Step 1:

- Extract the Change folder to desktop

Step 2:

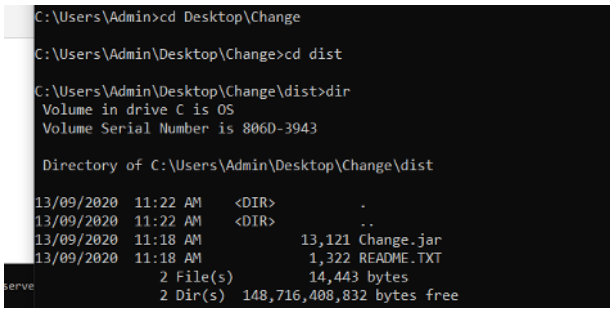
- Open up command prompt
- Go to change directory
    - Command: Cd [Change folder]



Step 3:

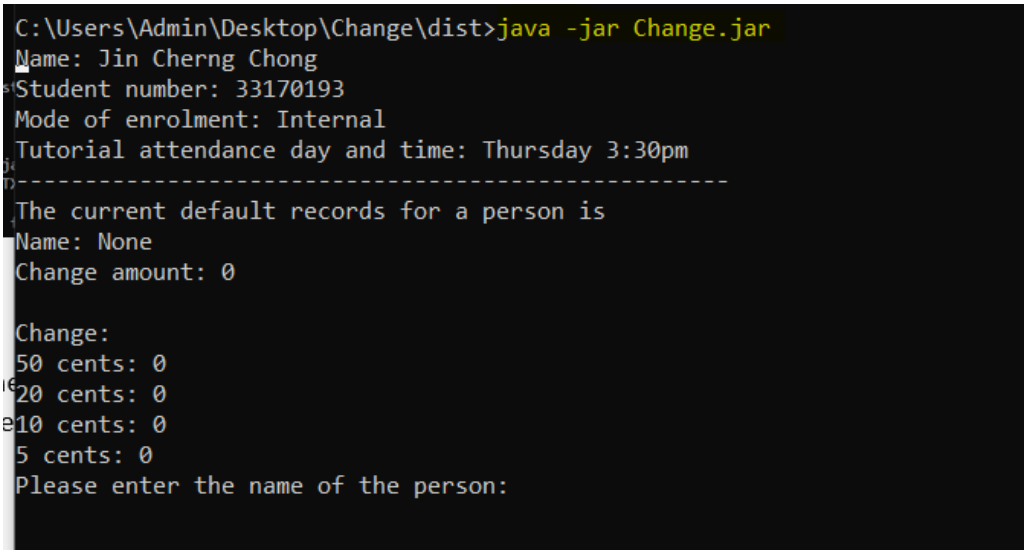
- Go to dist directory

- Command: Cd [dist folder]



Step 4:

- Once in: Change/dist → Execute the Change.jar
  - Command: java -jar Change.jar



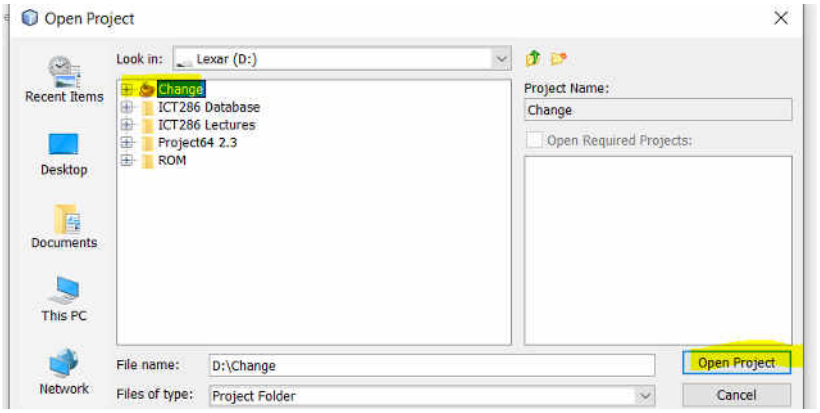
sss

Step 5: Well done! You can now type away in the command prompt

<u>Option 2- Running with Netbean</u>

Step 1:

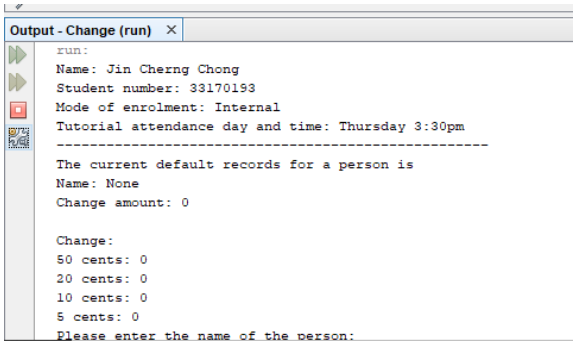
- Open project through netbean
    - File → Open project → Click on project → Open project



Step 2:

- Run project
    - Click run → Click Run Project (Change)

Step 3: Well done! You can now type away in the console in netbean



Structure/Design/Algorithm

Additional method for Change class-

Methods	Justification
WriteInitialOutput()	This method is called upon by the client program in order to get the initial variable values of the objects. This displayed at the start of the program in order to inform the client about the default values. These

	default values can be used as part of input validation
AddAmountOfChange(newAmountOfChange)	This method takes in an integer parameter of newAmountOfChange. The method is used to add the current total change amount requested by a person with the newAmountOfChange. This result would provide the person with an updated AmountOfChange total. Needed for objects that have already set their initial amountOfChange and we want add some more change. Using a setAmountOfChange() method would not be correct since sometimes we want to have the total change amount outside the acceptable range

Low level algorithm for client program-

```
**********************************************************************************

Change Machine Client Program Algorithm

By: Jin Cherng Chong

This program illustrates the process in which money given by a person (refered to as the
change amount) is exchanged for the equivialent coin amount

Assumptions for the input and output are-

Assume the amount of money a user gives (amountOfChange) will be in cents only

Assume the user will input the first name of a person only as a single string. This entry
excludes the person's middle name and last name

Assume user will input data of the correct data type

Assume that there is no GST attached to the exchange

Assume every person has a unique identifable name. Therefore, no two people will have same
name

Assume the user doesn't name a person: "None"

**********************************************************************************


Procedure void Main()


        String userName

        String currentUserName

        Integer changeQuantity = 0

        Integer currentQuantityOfChange = 0
```

```
Character addAnotherPerson = 'Y'
Boolean invalidInput = true

Change[] person = new Change[15]

for Integer i = 0 To (i < 15) Do
        Person[i] = new Person()
EndFor

Output "The current default records for a person is"
WriteInitialRecord(Person[0])

for Integer i = 0 To ((i < 15) AND (addAnotherPerson == 'Y')) Do

        do
                Output "Please enter the name of the person: "
                Input userName

                Output "Please enter the coin value for the person (range 5 to 95,
multiple of 5): "
                Input changeQuantity

                newName = ChkIsNewName(userName, changeQuantity, person)

                if(newName) then
                        SetName(Person[i], userName)
                        SetAmountOfChange(Person[i], changeQuantity)
                EndIf

                currentUserName = GetName(Person[i])
                currentQuantityOfChange = GetAmountOfChange(Person[i])

        while((currentUserName == 'None') OR (currentQuantityOfChange == 0))



        do
                Output "Do you have more person to enter (Y/N): "
                Input addAnotherPerson
                invalidInput = true
```

```
                ValidateAddAnotherPerson(addAnotherPerson)

                if(ToUpperCase(addAnotherPerson) == 'Y') then
                        invalidInput = false
                        addAnotherPerson = 'Y'
                EndIf

                if(ToUpperCase(addAnotherPerson) == 'N') then
                        invalidInput = false
                        addAnotherPerson = 'Y'
                EndIf


        while(invalidInput)

    EndFor

    CalculateDenominations(person)
    //Method to call hardcoded data here
    DspMenu(person)

EndProcedure


Procedure Boolean ChkIsNewName(String chkUserName, Integer changeAmount, Person)

    String personName = 'None'

    for Integer i = 0 To (i < 15) Do

        personName = GetName(Person[i])

        if(personName equals chkUserName) then
            AddAmountOfChange(Person[i], changeAmount)
            Output "Updates existing person"
            return false
        EndIf
```

```
        EndFor

        return true

EndProcedure


Procedure void ValidateAddAnotherPerson(Character chkAddAnotherPerson)

        if((chkAddAnotherPerson !== Y) OR (chkAddAnotherPerson !== N)) then
                return
                        else
                                Output "Error: Invalid input"
        EndIf

EndProcedure


Procedure void CalculateDenominations(person)

        for Integer i = 0 To (i < 15) Do
                Integer remainingAmountOfChange = 0
                Integer numOfFiftyCents = 0
                Integer numOfTwentyCents = 0
                Integer numOfTenCents = 0
                Integer numOfFiveCents = 0
                remainingAmountOfChange = GetAmountOfChange(Person[i])

                while(remainingAmountOfChange >= 50) then
                        numOfFiftyCents += 1
                        remainingAmountOfChange -= 50
                EndWhile

                while(remainingAmountOfChange >= 20) then
                        numOfTwentyCents += 1
                        remainingAmountOfChange -= 20
                EndWhile

                while(remainingAmountOfChange >= 10) then
```

```
                numOfTenCents += 1
                remainingAmountOfChange -= 10
        EndWhile

        while(remainingAmountOfChange >= 5) then
                numOfFiveCents += 1
                remainingAmountOfChange -= 5
        EndWhile

        SetNumOfFiftyCents(Person[i], numOfFiftyCents)
        SetNumOfTwentyCents(Person[i], numOfTwentyCents)
        SetNumOfTenCents(Person[i], numOfTenCents)
        SetNumOfFiveCents(Person[i], numOfFiveCents)

    EndFor

EndProcedure


Procedure void DspMenu(person)

    Integer option = 0

    while(option != 5) then

        Output "1. Enter a name and display change to be given for each denomination"
        Output "2. Find the name(s) with the smallest amount and display change to be
given for each denomination"
        Output "3. Find the name(s) with the largest amount and display change to be
given for each denomination"
        Output "4. Calculate and display the total number of coins for each
denomination, and the sum of these totals"
        Output "5. Exit"

        Output "Enter an option: "
        Input option

        Switch(option)
                Case 1:
                        SelectOption1(person)
                Case 2:
```

```
                        SelectOption2(person)
                Case 3:
                        SelectOption3(person)
                Case 4:
                        SelectOption4(person)
                Case 5:
                        Output "Farewell! Exiting menu"
                default:
                        Output "Invalid option!"
        EndCase

    EndWhile

EndProcedure


Procedure void SelectOption1(person)

    String searchName
    String currentName
    Boolean nameNotFound = True

    Output "-----------------------------------------------------"

    do
        Output "Enter a name"
        Input searchName
    while(searchName is Empty AND searchName != null)

    for Integer i = 0 To ((i < 15) AND (nameNotFound)) Do

        currentName = GetName(Person[i])

        if(searchName equals currentName) then
            DenominationsBreakdown(i, person)
            nameNotFound = false
        EndIf

    EndFor
```

```
        if(nameNotFound) then
                Output "Name: " + searchName
                Output "Not found"
                Output "-----------------------------------------------------"
        EndIf

EndProcedure


Procedure void SelectOption2(person)

        Integer smallestNumLocaton = 0
        Integer smallestChangeAmount = GetAmountOfChange(Person[0])
        Integer changeAmount = 0

        for Integer i = 0 To (i < 15) Do
                changeAmount = GetAmountOfChange(Person[i])

                if((changeAmount < smallestChangeAmount) AND (changeAmount !== 0)) then
                        smallestChangeAmount = changeAmount

                EndIf

        EndFor

        Output "-----------------------------------------------------"
        Output "The people(s) with the smallest change amount is"

        for Integer j = 0 To (j < 15) Do
                changeAmount = GetAmountOfChange(Person[j])

                if(smallestChangeAmount == changeAmount) then
                        smallestNumLocation = j
                        DenominationsBreakdown(smallestNumLocation, person)
                EndIf

        EndDo

EndProcedure
```

```
Procedure void SelectOption3(person)

        Integer largestNumLocaton = 0
        Integer largestChangeAmount = GetAmountOfChange(Person[0])
        Integer changeAmount = 0

        for Integer i = 0 To (i < 15) Do
                changeAmount = GetAmountOfChange(Person[i])

                if(changeAmount > largestChangeAmount) then
                        largestChangeAmount = changeAmount
                EndIf

        EndFor

        Output "-----------------------------------------------------"
        Output "The people with the largest change amount requested is"

        for Integer j = 0 To (j < 15) Do
                changeAmount = GetAmountOfChange(Person[j])

                if(largestChangeAmount == changeAmount) then
                        largestNumLocation = j
                        DenominationsBreakdown(largestNumLocation, person)
                EndIf

        EndDo

EndProcedure


Procedure void SelectOption4(person)

        Integer TotalFiftyCents = 0
        Integer TotalTwentyCents = 0
        Integer TotalTenCents = 0
        Integer TotalFiveCents = 0
```

```
        Integer TotalAmountOfChange = 0

        for Integer i = 0 To (i < 15) Do

                Integer fiftyCents = GetNumOfFiftyCents(Person[i])
                Integer twentyCents = GetNumOfTwentyCents(Person[i])
                Integer tenCents = GetNumOfTenCents(Person[i])
                Integer fiveCents = GetNumOfFiveCents(Person[i])
                Integer amountOfChange = GetAmountOfChange(Person[i])

                TotalFiftyCents += fiftyCents
                TotalTwentyCents += twentyCents
                TotalTenCents += tenCents
                TotalFiveCents += fiveCents

                TotalAmountOfChange += amountOfChange

        EndFor

        Output "-----------------------------------------------------"
        Output "Total number of coins and sum of these total"
        Output "Total amount of change: " + TotalAmountOfChange
        dspDenominations(TotalFiftyCents, TotalTwentyCents, TotalTenCents, TotalFiveCents)


EndProcedure


Procedure void DenominationsBreakdown(Integer location, person)

        String dspName
        Integer numOfFiftyCents = 0
        Integer numOfTwentyCents = 0
        Integer numOfTenCents = 0
        Integer numOfFiveCents = 0
        Integer dspChangeAmount = 0

        numOfFiftyCents = GetNumOfFiftyCents(Person[location])
        numOfTwentyCents = GetNumOfTwentyCents(Person[location])
```

```
        numOfTenCents = GetNumOfTenCents(Person[location])
        numOfFiveCents = GetNumOfFiveCents(Person[location])
        dspChangeAmount = GetAmountOfChange(Person[location])
        dspName = GetName(Person[location]) //Retrieves the name of the person

        Output "Customer: "
        Output dspName + dspChangeAmount + "cents"

        dspDenominations(numOfFiftyCents, numOfTwentyCents, numOfTenCents, numOfFiveCents)

EndProcedure


Procedure void DspDenominations(Integer dspNumOfFiftyCents, Integer dspNumOfTwentyCents,
Integer dspNumOfTenCents, Integer dspNumOfFiveCents)

        Output "Change: "


        if(dspNumOfFiftyCents !== 0) then
                Output "50 cents: " + dspNumOfFiftyCents
        EndIf

        if(dspNumOfTwentyCents !== 0) then
                Output "20 cents: " + dspNumOfTwentyCents
        EndIf

        if(dspNumOfTenCents !== 0) then
                Output "10 cents: " + dspNumOfTenCents
        EndIf

        if(dspNumOfFiveCents !== 0) then
                Output "5 cents: " + dspNumOfFiveCents
        EndIf

        Output "--------------------------------------------------"

EndProcedure
```

Low level algorithm for change class-

```
private String name
private Integer amountOfChange
private Integer numOfFiftyCents
private Integer numOfTwentyCents
private Integer numOfTenCents
private Integer numOfFiveCents
```

```
Procedure Change()

        name = "None"
        amountOfChange = 0
        numOfFiftyCents = 0
        numOfTwentyCents = 0
        numOfTenCents = 0
        numOfFiveCents = 0
```

```
EndProcedure

Procedure Change(String initialName, Integer initialAmountOfChange)

        name = InitialName
        amountOfChange = initialAmountOfChange
        numOfFiftyCents = 0
        numOfTwentyCents = 0
        numOfTenCents = 0
        numOfFiveCents = 0

EndProcedure




Procedure Change(string initialName)

        SetName(initialName)
        amountOfChange = 0
        numOfFiftyCents = 0
        numOfTwentyCents = 0
        numOfTenCents = 0
        numOfFiveCents = 0

EndProcedure


Procedure Change(Integer initialAmountOfChange)

        name = "None"
        amountOfChange = initialAmountOfChange
        numOfFiftyCents = 0
        numOfTwentyCents = 0
        numOfTenCents = 0
        numOfFiveCents = 0
```

```
EndProcedure

Procedure void SetChange(String newName, Integer newAmountOfChange)

        SetName(newName)
        SetAmountOfChange(newAmountOfChange)
        numOfFiftyCents = 0
        numOfTwentyCents = 0
        numOfTenCents = 0
        numOfFiveCents = 0

EndProcedure


Procedure void SetName(String newName)

        if(!newName is Empty AND newName != null) then
                name = newName
                else
                        Output "Error: Invalid name"
        EndIf

EndProcedure


Procedure void SetAmountOfChange(Integer newAmountOfChange)

        Boolean inRange = ((newAmountOfChange >= 5 AND newAmountOfChange <= 95)
        Boolean multipleOfFive = (newAmountOfChange is MultipleOf 5)

        if(inRange And multiplefFive) then
                amountOfChange = newAmountOfChange
                        else
                                Output "Incorrect coin value. Must be in the range 5 to 95, and
multiple of 5."
                        EndIf
                EndIf
        EndIf

EndProcedure
```

```
Procedure void AddAmountOfChange(Integer newAmountOfChange)

        Boolean inRange = ((newAmountOfChange >= 5 AND newAmountOfChange <= 95)
        Boolean multipleOfFive = (newAmountOfChange is MultipleOf 5)

        if(inRange And multiplefFive) then
                amountOfChange += newAmountOfChange
                        else
                                Output "Incorrect coin value. Must be in the range 5 to 95, and
multiple of 5."
                        EndIf
                EndIf
        EndIf

EndProcedure


Procedure void SetNumOfFiftyCents(Integer newNumOfFiftyCents)

        numOfFiftyCents = newNumOfFiftyCents

EndProcedure

Procedure void SetNumOfTwentyCents(Integer newNumOfTwentyCents)

        numOfTwentyCents = newNumOfTwentyCents

EndProcedure

Procedure void SetNumOfTenCents(Integer newNumOfTenCents)

        numOfTenCents = newNumOfTenCents

EndProcedure

Procedure void SetNumOfFiveCents(Integer newNumOfFiveCents)

        numOfFiveCents = newNumOfFiveCents
```

```
EndProcedure



Procedure void WriteInitialRecord()

        Output "Name: " + name
        Output "Change amount: " + amountOfChange

        Output "Change: "
        Output "50 cents: " + numOfFiftyCents
        Output "20 cents: " + numOfTwentyCents
        Output "10 cents: " + numOfTenCents
        Output "5 cents: " + numOfFiveCents

EndProcedure



Procedure String GetName()
        return name
EndProcedure

Procedure Integer GetAmountOfChange()
        return amountOfChange
EndProcedure

Procedure Integer GetNumOfFiftyCents()
        return numOfFiftyCents
EndProcedure

Procedure Integer GetNumOfTwentyCents()
        return numOfTwentyCents
EndProcedure

Procedure Integer GetNumOfTenCents()
        return numOfTenCents
EndProcedure
```

```
Procedure Integer GetNumOfFiveCents()
        return numOfFiveCents
EndProcedure
```

High level algorithm for client program-

Start

Input the name of person

Validate the name of person

Input the coin value for the person

Validate the coin value

Check whether another person needs to be entered

Calculate denominations for all people

Display menu

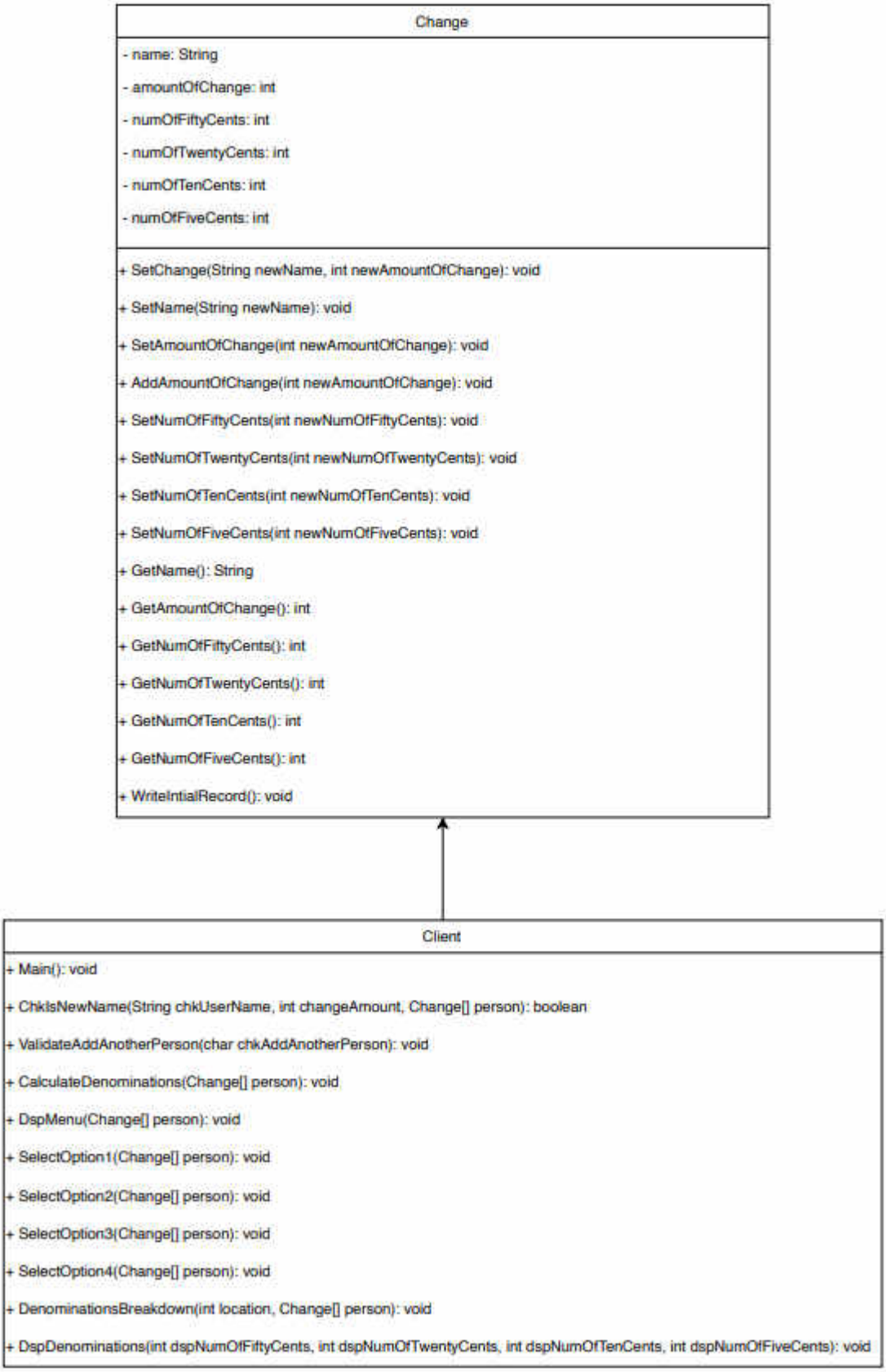
Input option for menu

Process option

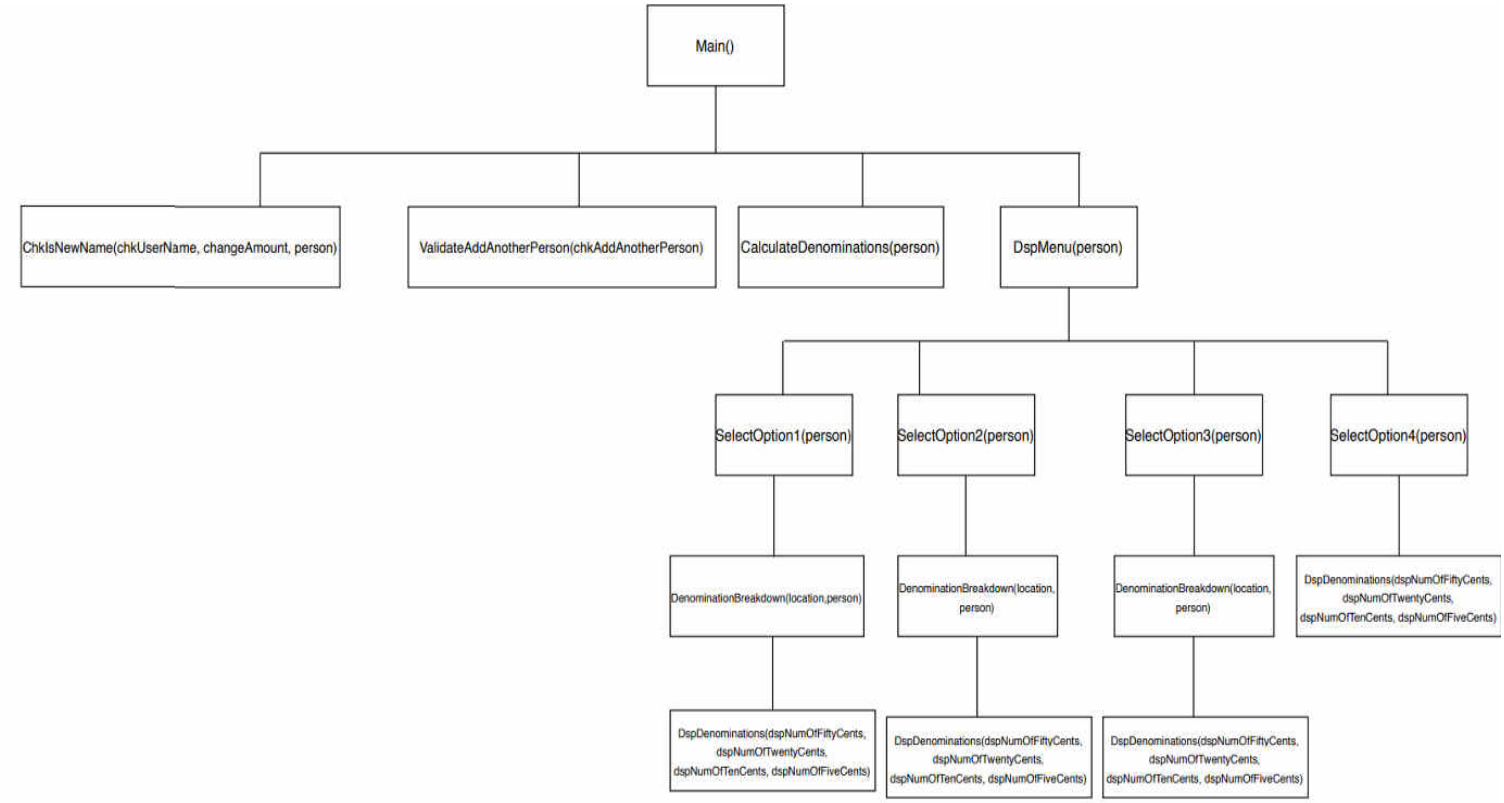
Check whether another option needs processed

End

UML diagram for Change class and client program-


Change
- name: String
- amountOfChange: int
- numOfFiftyCents: int
- numOfTwentyCents: int
- numOfTenCents: int
- numOfFiveCents: int
+ SetChange(String newName, int newAmountOfChange): void
+ SetName(String newName): void
+ SetAmountOfChange(int newAmountOfChange): void
+ AddAmountOfChange(int newAmountOfChange): void
+ SetNumOfFiftyCents(int newNumOfFiftyCents): void
+ SetNumOfTwentyCents(int newNumOfTwentyCents): void
+ SetNumOfTenCents(int newNumOfTenCents): void
+ SetNumOfFiveCents(int newNumOfFiveCents): void
+ GetName(): String
+ GetAmountOfChange(): int
+ GetNumOfFiftyCents(): int
+ GetNumOfTwentyCents(): int
+ GetNumOfTenCents(): int
+ GetNumOfFiveCents(): int
+ WriteIntialRecord(): void
Client
+ Main(): void
+ ChkIsNewName(String chkUserName, int changeAmount, Change[] person): boolean
+ ValidateAddAnotherPerson(char chkAddAnotherPerson): void
+ CalculateDenominations(Change[] person): void
+ DspMenu(Change[] person): void
+ SelectOption1(Change[] person): void
+ SelectOption2(Change[] person): void
+ SelectOption3(Change[] person): void
+ SelectOption4(Change[] person): void
+ DenominationsBreakdown(int location, Change[] person): void
+ DspDenominations(int dspNumOfFiftyCents, int dspNumOfTwentyCents, int dspNumOfTenCents, int dspNumOfFiveCents): void

Structure chart for client program- (zoom to view clearer)



Limitations

One of the issues with my program was that I had to have an additional method in the change class to add more change amount to an existing person (AddChangeAmount). So instead of having one method (SetAmountOfChange) to handle the change amount for a person I needed an additional method (AddChangeAmount). I tried to implement the addChangeAmount method to the setAmountOfChange but there are instances where a user would like to just override the current change amount for a person and set a new change amount for the person. So it would be difficult for the program to differentiate adding more change to a person and setting a brand new change for a person. Thus, two methods were required.

Another issue with my program is that an assumption had to be made for the name of a person. The program assumed that the user would enter the name of a person as one string. If a user accidentally entered a name of a person as one string but with a space at the end (Lucas) the program would still accept and store the input. If the client typed (Lucas) without a space to search the name the program would return not found. Thus, the assumption that the names have to be one string was made to combat this.

The third issue with my program is that a strings starting with the letter Y or N entered in response to the question “Do you have more person to enter (Y/N)” is always treated as valid response. The program should really only accept the character Y or N and not a string with the first letter Y or N. This failure has to do with the way in which the program accepts the character input; keyboard.next().charAt(0) is used to obtain the character input. As a result of this method a string will always be accepted but only the first character of that string. Unfortunately I was unable to identify another method to obtain a single character input

Testing

Testing was divided in several parts. These parts include- option 1 menu, option 2 menu, option 3 menu, option 4 menu, option 5 menu, option menu itself, and user inputting details of a person. Test table: user inputs person details contains the only failures that arose during the testing. In that test table, TestCase 13-15 were all failed. Refer to the limitations section for an explanation.

Test Table: Option 1 (Enter a name and display change to be given for each denomination)

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Enters a name wrong	josh 95 y tim 45 y rock 5 n 1 joshi	Name: joshi Not Found	Success
2	Enters a correct name.	josh 95 y tim 45 y rock 5 n 1 tim	Customer: tim 45 cents Change: 20 cents: 2 5 cents: 1	Success
3	Enters a correct name but in different cases	josh 95 y tim 45 y	Customer: rock 5 cents Change: 5 cents: 1	Success

		rock 5 n 1 rOcK		
4	Enter an empty name	(same input as above but with the addition of empty name instead if rOcK)	Enter a name: Enter a name:	Success
5	Enters a correct name but the input is in different cases	jOsh 20 y tiM 30 y josh	Customer: jOsh 20 cents Change: 20 cents: 1	Success
6	Enters a correct name but the input contains same person	kelly 45 Y Thomas 75 y kelly 95 Josh 20 1 kelly	Customer: Kelly 140 cents Change: 50 cents: 2 20 cents: 2	Success

Result of programing testing

TestCase 1:

Enter a name:

joshi

Name: joshi

Not Found

TestCase 2:

Enter a name:

tim

Customer:

tim 45 cents

Change:

20 cents: 2

5 cents: 1

TestCase 3:

Enter a name:

rOcK

Customer:

rock 5 cents

Change:

5 cents: 1

TestCase 4:

Enter a name:

Enter a name:

TestCase 5:

Enter a name:

josh

Customer:

jOsh 20 cents

Change:

20 cents: 1

TestCase 6:

Enter a name:

Kelly

Customer:

kelly 140 cents

Change:

50 cents: 2

20 cents: 2

Test Table: Option 2 (Find the name(s) with the smallest change amount and display change given for each denomination)

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	User enters multiple people with the smallest change amount	Tony 35 Y Shaw 35 Y Rioli 55 N 2	Customer: tony 35 cents Change: 20 cents: 1 10 cents: 1 5 cents: 1 Customer: shaw 35 cents Change: 20 cents: 1 10 cents: 1 5 cents: 1	Success
2	User enters person with the smallest change amount. But enters change amount incorrect for one of the people	Bob 10 Y Tom 5 Y Lewis 7 Lewis 15 N 2	Customer: Tom 5 cents Change: 5 cents: 1	Success

3	User enters person with the smallest change amount incorrectly the first time. But correctly the subsequent time	Lewie 6 George 15 Lewie 10 N 2	Customer: Tom 10 cents Change: 10 cents: 1	Success
4	User enters person with smallest change amount by entering the person twice	Lucas 20 Y Lucas 20 Lewie 60 N 2	Customer: Lucas 40 cents Change: 10 cents: 1	Success
5	User enters person with smallest change amount by entering the person twice but the names in different cases	LuCas 20 Y Lucas 40 Ron 70 N 2	Customer: LuCas 60 cents Change: 50 cents: 1 10 cents: 1	Success

Result of programing testing

TestCase 1:

The people(s) with the smallest change amount is

Customer:

tony 35 cents

Change:

20 cents: 1

10 cents: 1

5 cents: 1

Customer:

shaw 35 cents

Change:

20 cents: 1

10 cents: 1

5 cents: 1

TestCase 2:

The people(s) with the smallest change amount is

Customer:

Tom 5 cents

Change:

5 cents: 1

TestCase 3:

The people(s) with the smallest change amount is

Customer:

Lewie 10 cents

Change:

10 cents: 1

TestCase 4:

The people(s) with the smallest change amount is

Customer:

Lucas 40 cents

Change:

20 cents: 2

TestCase 5:

The people(s) with the smallest change amount is

Customer:

LuCas 60 cents

Change:

50 cents: 1

10 cents: 1

Test Table: Option 3 (Find the name(s) with the largest change amount and display change given for each denomination)

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	User enters person with largest change amount by entering the person twice	Matt 40 Y Matt 95 George 95 N 3	Customer: Matt 135 cents Change: 50 cents: 2 20 cents: 1 10 cents: 1 5 cents: 1	Success
2	User enters multiple people with the largest change amount	Matt 95 Y George 95 N	Customer: Matt 95 cents Change: 50 cents: 1 20 cents: 2 10 cents: 1	Success

		3	Customer: George 95 cents Change: 50 cents: 1 20 cents: 2 10 cents: 1	

Result of programing testing

TestCase 1:

The people(s) with the largest change amount is

Customer:

Matt 135 cents

Change:

50 cents: 2

20 cents: 1

10 cents: 1

5 cents: 1

TestCase 2:

The people(s) with the largest change amount is

Customer:

Matt 95 cents

Change:

50 cents: 1

20 cents: 2

5 cents: 1

Customer:

George 95 cents

Change:

50 cents: 1

20 cents: 2

5 cents: 1

Test Table: Option 4 (Calculate and display the total number of coins for each denomination, and the sum of these totals)

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	User enters people with the same change amount	Arc 10 Y Bob 10 N 4	Total amount of change: 20 Change: 10 cents: 2	Success
2	Illustrates all denominations are working for option 4	Floyd 5 Y Rob 10 Y Dunk 20 Y Leroy 50 N 4	Total amount of change: 85 Change: 50 cents: 1 20 cents: 1 10 cents: 1 5 cents: 1	Success

Result of programing testing

TestCase 1:

Total number of coins and sum of these total

Total amount of change: 20

Change:

10 cents: 2

TestCase 2:

Total number of coins and sum of these total

Total amount of change: 85

Change:

50 cents: 1

20 cents: 1

10 cents: 1

5 cents: 1

Test Table: Option 5 (Exit)

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	User select option 5	Ron 5 Y 5	Farewell! Exiting menu	Success

Result of programing testing

TestCase 1:

Farewell! Exiting menu

Test Table: Option menu input

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	User enters non-valid entry for option menu	bob 5 n 1234	Invalid option!	Success

Result of programing testing

TestCase 1:

Enter an option:

1234

Invalid option!

Test Table: User inputs person details

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	User inputs the lowest valid change amount for a person	Bob 5 N 1 Bob	Customer: Bob 5 cents Change: 5 cents: 1	Success
2	User inputs non-multiple of 5 change amount for a person	Bob 6	Incorrect coin value. Must be in range between 5 to 95 and multiple of 5	Success
3	User inputs the highest valid change amount for a person	Tom 95 N 1 Tom	Customer: Tom 95 cents Change: 50 cents: 1 20 cents: 2 5 cents: 1	Success
4	User inputs a multiple of 5 change amount for a person but outside the maximum valid change amount range	Jin 100	Incorrect coin value. Must be in range between 5 to 95 and multiple of 5	Success
5	User inputs a multiple of 5 change amount for a person but outside the minimum valid change amount range	Jin -5	Incorrect coin value. Must be in range between 5 to 95 and multiple of 5	Success
6	User inputs change amount for a person as negative	Lucas -20	Incorrect coin value. Must be in range between 5 to 95 and	Success

			multiple of 5	
7	User inputs zero change amount for a person	Felix 0	Incorrect coin value. Must be in range between 5 to 95 and multiple of 5	Success
8	User input empty string for name of person and invalid change amount	100	Error: Invalid name Incorrect coin value. Must be in range between 5 to 95 and multiple of 5	Success
9	User enters lowercase and uppercase Y for whether another person should be added	Bob 5 y Tim 5 Y	Please enter the name of the person:	Success
10	User enters lowercase N for whether another person should be added	Bob 5 n	Enter an option:	Success
11	User enters uppercase N for whether another person should be added	Tim 5 N	Enter an option:	Success
12	User enters a non Y or N input for whether another person should be added	Bob 5 asdf	Error: invalid input	Success
13	User enters a non Y or N input for whether another person should be added but input starts with letter Y uppercase or lowercase	Bob 5 yess Tom 5 Yep	Error: invalid input	Failure
14	User enters a non Y or N input for whether another person should be added but input starts with letter N uppercase	Bob 5 NewPerson	Error: invalid input	Failure

15	User enters a non Y or N input for whether another person should be added but input starts with letter n lowercase	mitch 5 no	Error: invalid input	Failure

Result of programing testing

TestCase 1:

Enter a name:

Bob

Customer:

Bob 5 cents

Change:

5 cents: 1

TestCase 2:

Please enter the name of the person:

Bob

Please enter the coin value for the person (range 5 to 95, multiple of 5:

6

Incorrect coin value. Must be in range between 5 to 95 and multiple of 5

TestCase 3:

Enter a name:

Tom

Customer:

Tom 95 cents

Change:

50 cents: 1

20 cents: 2

5 cents: 1

TestCase 4:

Please enter the name of the person:

Jin

Please enter the coin value for the person (range 5 to 95, multiple of 5:

100

Incorrect coin value. Must be in range between 5 to 95 and multiple of 5

TestCase 5:

Please enter the name of the person:

Jin

Please enter the coin value for the person (range 5 to 95, multiple of 5:

-5

Incorrect coin value. Must be in range between 5 to 95 and multiple of 5

TestCase 6:

Please enter the name of the person:

Lucas

Please enter the coin value for the person (range 5 to 95, multiple of 5:

-20

Incorrect coin value. Must be in range between 5 to 95 and multiple of 5

TestCase 7:

Please enter the name of the person:

Felix

Please enter the coin value for the person (range 5 to 95, multiple of 5:

0

Incorrect coin value. Must be in range between 5 to 95 and multiple of 5

TestCase 8:

Please enter the name of the person:

Please enter the coin value for the person (range 5 to 95, multiple of 5:

100

Error: Invalid name

Incorrect coin value. Must be in range between 5 to 95 and multiple of 5

TestCase 9:

Please enter the name of the person:

Bob

Please enter the coin value for the person (range 5 to 95, multiple of 5:

5

Do you have more person to enter (Y/N):

y

Please enter the name of the person:

Tim

Please enter the coin value for the person (range 5 to 95, multiple of 5:

5

Do you have more person to enter (Y/N):

Y

Please enter the name of the person:

TestCase 10:

Please enter the name of the person:

Bob

Please enter the coin value for the person (range 5 to 95, multiple of 5:

5

Do you have more person to enter (Y/N):

n

1. Enter a name and display change to be given for each denomination

2. Find the name(s) with the smallest amount and display change to be given for each denomination

3. Find the name(s) with the largest amount and display change to be given for each denomination

4. Calculate and display the total number of coins for each denomination, and the sum of these totals

5. Exit

Enter an option:

TestCase 11:

Please enter the name of the person:

Tim

Please enter the coin value for the person (range 5 to 95, multiple of 5:

5

Do you have more person to enter (Y/N):

N

1. Enter a name and display change to be given for each denomination

2. Find the name(s) with the smallest amount and display change to be given for each denomination

3. Find the name(s) with the largest amount and display change to be given for each denomination

4. Calculate and display the total number of coins for each denomination, and the sum of these totals

5. Exit

Enter an option:

TestCase 12:

Please enter the name of the person:

Bob

Please enter the coin value for the person (range 5 to 95, multiple of 5:

5

Do you have more person to enter (Y/N):

asdf

Error: invalid input

TestCase 13:

Please enter the name of the person:

Bob

Please enter the coin value for the person (range 5 to 95, multiple of 5:

5

Do you have more person to enter (Y/N):

yess

Please enter the name of the person:

Tom

Please enter the coin value for the person (range 5 to 95, multiple of 5:

5

Do you have more person to enter (Y/N):

Yep

Please enter the name of the person:

TestCase 14:

Please enter the name of the person:

Bob

Please enter the coin value for the person (range 5 to 95, multiple of 5:

5

Do you have more person to enter (Y/N):

NewPerson

1. Enter a name and display change to be given for each denomination

2. Find the name(s) with the smallest amount and display change to be given for each denomination

3. Find the name(s) with the largest amount and display change to be given for each denomination

4. Calculate and display the total number of coins for each denomination, and the sum of these totals

5. Exit

Enter an option:

TestCase 15:

Please enter the name of the person:

mitch

Please enter the coin value for the person (range 5 to 95, multiple of 5:

5

Do you have more person to enter (Y/N):

no

1. Enter a name and display change to be given for each denomination

2. Find the name(s) with the smallest amount and display change to be given for each denomination

3. Find the name(s) with the largest amount and display change to be given for each denomination

4. Calculate and display the total number of coins for each denomination, and the sum of these totals

5. Exit

Enter an option:

Source program listing

Java source code for client program (Client.java)-

```
/*
 * Change Machine
 * By: Jin Cherng Chong
 * 4/09/2020
 * Files: Client.java and Change.java (class name)
 * This program illustrates the process in which money given by a person (referred to as the change amount) is exchanged for the equivalent coin amount
 * Assumptions for the inputs and outputs are-
 * Assume the amount of money a person gives (change amount) will be in cents only
 * Assume the user will input the first name of a person only as a single string. This entry excludes the person's middle name and last name
 * Assume the user will input data of the correct data type
 * Assume there is no GST involved
 * Assume every person has a unique identifiable name. Therefore, no two people will have the same name
 * Assunme the user doesn't name a person: "None"
 */
package change;

import java.util.Scanner;

public class Client {

    public static void main(String[] args) {

        String userName;
        String currentUserName;
        int changeQuantity = 0;
        int currentQuantityOfChange = 0;
        char addAnotherPerson = 'Y';
        boolean invalidInput = true;
```

```
        Scanner keyboard = new Scanner(System.in);


        Change[] person = new Change[15]; //Creates an array of new objects. Each object can
be thought of as a unique person


        for (int i = 0; i < 15; i++) { //Instantiates the new objects

            person[i] = new Change();


        }


        StudentInfo();




        System.out.println("The current default records for a person is");

        person[0].WriteInitialRecord(); //Displays curent instance variables for the object

        //HardcodeData();         //Remove the front // to enable hard coded data

        for (int i = 0; (i < 15) && (addAnotherPerson == 'Y'); i++) //Loops through the whole
array of persons (object) and makes sure addAnotherPerson is 'Y' which indicates: Yes, we need
to add another person

        {


            do {


                System.out.println("Please enter the name of the person:");

                userName = keyboard.nextLine(); //User inputs the name of person


                System.out.println("Please enter the coin value for the person (range 5 to 95,
multiple of 5:");

                changeQuantity = keyboard.nextInt(); //User inputs the amount of money to be
given (called change amount because it will be exchanged for coins)


                boolean newName = ChkIsNewName(userName, changeQuantity, person); //Check
whether name is unique thus indicating a new person (object)


                if (newName) { // This is true when a user enters a unique name which
indicates a new person (object)

                    person[i].SetName(userName); //Set a unique/new name for the person
(object)

                    person[i].SetAmountOfChange(changeQuantity); //Set the change amount for
the person

                }


                currentUserName = person[i].GetName();
```

```
                currentQuantityOfChange = person[i].GetAmountOfChange();


                keyboard.nextLine(); //This fixes the nextLine common issue where the /n is
part of buffer


            } while ((currentUserName.equals("None")) || (currentQuantityOfChange == 0));
//Validates whether or not the name of the person and change amount for the person is actually
entered correctly by the user


            do {

                System.out.println("Do you have more person to enter (Y/N):");

                addAnotherPerson = keyboard.next().charAt(0);

                char addAnotherPersonUpper = Character.toUpperCase(addAnotherPerson);

                invalidInput = true; //Resets the input


                ValidateAddAnotherPerson(addAnotherPersonUpper); //Validates whether or not
the user response to the question "...more person to enter" is a valid response


                if (Character.toUpperCase(addAnotherPerson) == 'Y') { //Checks whether input
is one of the valid options

                    invalidInput = false; //Assigns false to invalidInput indicating that user
input is valid

                    addAnotherPerson = 'Y'; // Converts lower case y to Upper case Y


                }


                if (Character.toUpperCase(addAnotherPerson) == 'N') { //Checks whether input
is one of the valid options

                    invalidInput = false; //Assigns false to invalidInput indicating that user
input is valid

                    addAnotherPerson = 'N'; //Converts lower case n to Upper case N


                }


                keyboard.nextLine(); //This fixes the nextLine common issue where the /n is
part of buffer


            } while (invalidInput); //Keep looping when input is invalid (false)


        }


        CalculateDenominations(person);

        DspMenu(person);
```

```
    }

    public static boolean ChkIsNewName(String chkUserName, int changeAmount, Change[] person)
{ //Method that validates whether or not name entered by the user is unique (true) or not
unique (false)

        String personName = "None";

        for (int i = 0; i < 15; i++) { //Loops through array containing many persons (object)
and checks whether name entered by user refers to an exisiting person

            personName = person[i].GetName();

            if (personName.equalsIgnoreCase(chkUserName)) { //The user-entered name of person
(chkUserName) refers to an exisiting name of person (personName). Ignores case sensitivity

                person[i].AddAmountOfChange(changeAmount); //Adds the additional change amount
to the existing person

                System.out.println("Update exisiting person");

                return false; //Assign false to the boolean method to indicate that the name
of the person is not unique/new

            }

        }

        return true;

    }

    public static void ValidateAddAnotherPerson(char chkAddAnotherPerson) { //Method validates
whether or not another person is to be added

        if ((chkAddAnotherPerson == 'Y') || (chkAddAnotherPerson == 'N')) { //

            return;

        } else {
```

```
            System.out.println("Error: invalid input"); //Displays error message when the
input is neither Y or N
        }

    }
```

```
    public static void CalculateDenominations(Change[] person) { //Method that calculates the
denominations for each person in respect to the person's total change amount

        for (int i = 0; i < 15; i++) { //Loop's through array containing many persons (object)
and calculate denominations for each person (object)
            int remainingAmountOfChange = 0;
            int numOfFiftyCents = 0;
            int numOfTwentyCents = 0;
            int numOfTenCents = 0;
            int numOfFiveCents = 0;
            remainingAmountOfChange = person[i].GetAmountOfChange();

            while (remainingAmountOfChange >= 50) {
                numOfFiftyCents += 1; //Give 1 more fifty cent coin to the person
                remainingAmountOfChange -= 50;
            }

            while (remainingAmountOfChange >= 20) {
                numOfTwentyCents += 1; //Give 1 more twenty cent coin to the person
                remainingAmountOfChange -= 20;
            }

            while (remainingAmountOfChange >= 10) {
                numOfTenCents += 1; //Give 1 more ten cent coin to the person
                remainingAmountOfChange -= 10;
            }

            while (remainingAmountOfChange >= 5) {
                numOfFiveCents += 1; //Give 1 more five cent coin to the person
                remainingAmountOfChange -= 5;
            }
```

```
            person[i].SetNumOfFiftyCents(numOfFiftyCents); //Sets the number of coins given to
the person for each denomination

            person[i].SetNumOfTwentyCents(numOfTwentyCents);

            person[i].SetNumOfTenCents(numOfTenCents);

            person[i].SetNumOfFiveCents(numOfFiveCents);


        }

    }







    public static void DspMenu(Change[] person) { //Method that displays a menu to the client


        int option = 0;


        Scanner keyboard = new Scanner(System.in);


        while (option != 5) { //Stops displaying the menu when the option entered is 5


            System.out.println();

            System.out.println("1. Enter a name and display change to be given for each
denomination");

            System.out.println("2. Find the name(s) with the smallest amount and display
change to be given for each denomination");

            System.out.println("3. Find the name(s) with the largest amount and display change
to be given for each denomination");

            System.out.println("4. Calculate and display the total number of coins for each
denomination, and the sum of these totals");

            System.out.println("5. Exit");


            System.out.println("Enter an option: ");

            option = keyboard.nextInt();


            switch (option) {

                case 1:

                    SelectOption1(person);

                    break;

                case 2:
```

```
                    SelectOption2(person);
                    break;
                case 3:
                    SelectOption3(person);
                    break;
                case 4:
                    SelectOption4(person);
                    break;
                case 5:
                    System.out.println("Farewell! Exiting menu");
                    break;
                default:
                    System.out.println("Invalid option!");
            }

        }

    }
```

```
    public static void SelectOption1(Change[] person) { //Method that gets the client to enter a name and display corresponding denominations for the person

        String searchName;
        String currentName;
        Boolean nameNotFound = true;

        Scanner keyboard = new Scanner(System.in);

        System.out.println("-----------------------------------------------------");

        do {
              System.out.println("Enter a name: ");
              searchName = keyboard.nextLine();
        } while(searchName.isEmpty() && searchName != null); //Error checks for empty string or null
```

```
        for (int i = 0; (i < 15) && (nameNotFound); i++) { //Keep looping through array of persons (object) and until the last object in the array or until a name is found

            currentName = person[i].GetName();

            if (searchName.equalsIgnoreCase(currentName)) { //Checks whether the entered name is found in the array of persons (object). Ignores case sensitivity
                DenominationsBreakdown(i, person);
                nameNotFound = false; //Assigns false to the boolean to indicate that the name is found
            }
        }

        if (nameNotFound) { //When the entered name is NOT found in the array of persons (object) an error message is displayed to the client
            System.out.printf("Name: %s", searchName);
            System.out.println();
            System.out.println("Not Found");
            System.out.println("-----------------------------------------------------");
        }

    }

    public static void SelectOption2(Change[] person) { //Method that finds the name(s) with the smallest change amount. This method will then display the corresponding denomination for each name

        int smallestNumLocation = 0;
        int smallestChangeAmount = person[0].GetAmountOfChange(); //Assign an initial smallest amount of change to compare with other change amounts
        int changeAmount = 0;

        for (int i = 0; i < 15; i++) { //Loop identifies the smallest change amount
            changeAmount = person[i].GetAmountOfChange();

            if ((changeAmount < smallestChangeAmount) && (changeAmount != 0)) { //This checks whether the current smallestChangeAmount is still the smallest. The addition of change amount not equal 0 is validation checking
```

```
                smallestChangeAmount = changeAmount;                            // for a
default unset person (object). A default object is a person whom has not yet been
intilialised/set variables for. It must be ignored

            }

        }


        System.out.println("-----------------------------------------------------");

        System.out.println("The people(s) with the smallest change amount is");


        for (int j = 0; j < 15; j++) { //This loop deals with multiple people having the
smallest change amount. Displays the resulting people with the smallest amount of change

            changeAmount = person[j].GetAmountOfChange();


            if (smallestChangeAmount == changeAmount) { //Displays the person(s) that has the
smallest change amount

                smallestNumLocation = j;

                DenominationsBreakdown(smallestNumLocation, person);

            }


        }


    }









    public static void SelectOption3(Change[] person) { //Method that finds the name(s) with
the largest change amount. This method will then display the corresponding denomination for
each name


        int largestNumLocation = 0;

        int largestChangeAmount = person[0].GetAmountOfChange(); //Assign an initial largest
amount of change to compare with other change amounts

        int changeAmount = 0;


        for (int i = 0; i < 15; i++) { //Loop identifies the largest change amount

            changeAmount = person[i].GetAmountOfChange();


            if (changeAmount > largestChangeAmount) { //This checks whether the current
largest ChangeAmount is still the largest.

                largestChangeAmount = changeAmount; //Update the largest changeAmount when the
current one is no longer the largest
```

```
            }
        }

        System.out.println("-----------------------------------------------------");
        System.out.println("The people(s) with the largest change amount is");

        for (int j = 0; j < 15; j++) { //This loop deals with multiple people having the
largest change amount. Displays the resulting people with the largest amount of change
            changeAmount = person[j].GetAmountOfChange();

            if (largestChangeAmount == changeAmount) { //Displays the person(s) that has the
largest change amount
                largestNumLocation = j;
                DenominationsBreakdown(largestNumLocation, person);
            }

        }

    }

    public static void SelectOption4(Change[] person) { //Method finds the total change amount
for all the people and total number of coins given for each denomination

        int TotalFiftyCents = 0;
        int TotalTwentyCents = 0;
        int TotalTenCents = 0;
        int TotalFiveCents = 0;
        int TotalAmountOfChange = 0;

        for (int i = 0; i < 15; i++) { //Loop works out the total number of coins given for
each denomination and the total change amount for all the people

            int numOfFiftyCents = person[i].GetNumOfFiftyCents(); //Retrieve the denominations
for the person
            int numOfTwentyCents = person[i].GetNumOfTwentyCents();
            int numOfTenCents = person[i].GetNumOfTenCents();
            int numOfFiveCents = person[i].GetNumOfFiveCents();
```

```
            int amountOfChange = person[i].GetAmountOfChange(); //Retrieve the change amount
for the person


            TotalFiftyCents += numOfFiftyCents; //Keeps a running total of the number of coins
given for each denominations for all people
            TotalTwentyCents += numOfTwentyCents;
            TotalTenCents += numOfTenCents;
            TotalFiveCents += numOfFiveCents;


            TotalAmountOfChange += amountOfChange;
        }


        System.out.println("-----------------------------------------------------");
        System.out.println("Total number of coins and sum of these total");
        System.out.printf("Total amount of change: %d", TotalAmountOfChange);
        System.out.println();
        DspDenominations(TotalFiftyCents, TotalTwentyCents, TotalTenCents, TotalFiveCents);
//Displays the total number of coins given for all people




    }







    public static void DenominationsBreakdown(int location, Change[] person) { //Method that
breakdowns the denominations for the person


        String dspName;
        int numOfFiftyCents = 0;
        int numOfTwentyCents = 0;
        int numOfTenCents = 0;
        int numOfFiveCents = 0;
        int dspChangeAmount = 0;


        numOfFiftyCents = person[location].GetNumOfFiftyCents(); //Retrieve the denominations
for the person
        numOfTwentyCents = person[location].GetNumOfTwentyCents();
        numOfTenCents = person[location].GetNumOfTenCents();
        numOfFiveCents = person[location].GetNumOfFiveCents();
```

```
        dspChangeAmount = person[location].GetAmountOfChange(); //Retrieve the name of the
person


        dspName = person[location].GetName();


        System.out.println();

        System.out.println("Customer: "); //Displays name of the person and their change
amount

        System.out.printf("%s %d cents", dspName, dspChangeAmount);

        System.out.println();


        DspDenominations(numOfFiftyCents, numOfTwentyCents, numOfTenCents, numOfFiveCents);
//Displays the total number of coins given for each denomination for the person


    }









    public static void DspDenominations(int dspNumOfFiftyCents, int dspNumOfTwentyCents, int
dspNumOfTenCents, int dspNumOfFiveCents) { //Method displays the total number of coins given
for each denomination


        System.out.println();

        System.out.print("Change: ");


        if (dspNumOfFiftyCents != 0) {

            System.out.println();

            System.out.printf("50 cents: %d", dspNumOfFiftyCents); //Does not display the
following when there are 0 fifty cent coins given to people


        }


        if (dspNumOfTwentyCents != 0) {

            System.out.println();

            System.out.printf("20 cents: %d", dspNumOfTwentyCents); //Does not display the
following when there are 0 fifty cent coins given to people


        }


        if (dspNumOfTenCents != 0) {
```

```
            System.out.println();

            System.out.printf("10 cents: %d", dspNumOfTenCents); //Does not display the
following when there are 0 fifty cent coins given to people


        }


        if (dspNumOfFiveCents != 0) {

            System.out.println();

            System.out.printf("5 cents: %d", dspNumOfFiveCents); //Does not display the
following when there are 0 fifty cent coins given to people


        }

        System.out.println();

        System.out.println("-----------------------------------------------------");

    }




    public static void StudentInfo() {


        System.out.println("Name: Jin Cherng Chong ");

        System.out.println("Student number: 33170193 ");

        System.out.println("Mode of enrolment: Internal ");

        System.out.println("Tutorial attendance day and time: Thursday 3:30pm");

        System.out.println("-----------------------------------------------------");

    }



    public static void HardcodeData() {


        Change[] hardcodePerson = new Change[15]; //Creates an array of new objects. Each
object can be thought of as a unique person


        hardcodePerson[0] = new Change("Jane", 30); //NOTE: The new Change() creates a brand
new object.

        hardcodePerson[1] = new Change("John", 50); //NOTE: The hard coded data treats every
new object as a new person. Thus creating two object named Jane will result in two people
named Jane

        hardcodePerson[2] = new Change("Fred", 95);

        hardcodePerson[3] = new Change("Tom", 25);

        hardcodePerson[4] = new Change("Blitz", 45);

        hardcodePerson[5] = new Change("Luke", 30);
```

```
        hardcodePerson[6] = new Change("Fair", 30);
        hardcodePerson[7] = new Change("Jane", 30);
        hardcodePerson[8] = new Change("george", 10);
        hardcodePerson[9] = new Change("Lucas", 75);
        hardcodePerson[10] = new Change("Tim", 5);
        hardcodePerson[11] = new Change("Felix", 86);
        hardcodePerson[12] = new Change("Racheal", 5);
        hardcodePerson[13] = new Change("Ryan", 10);
        hardcodePerson[14] = new Change("May", 40);


        CalculateDenominations(hardcodePerson);
        DspMenu(hardcodePerson);
    }

}
```

Java source code for Change class(Change.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 change;

import java.util.Scanner;

public class Change {

    private String name;
    private int amountOfChange;
    private int numOfFiftyCents;
    private int numOfTwentyCents;
    private int numOfTenCents;
    private int numOfFiveCents;



    public Change() {

        name = "None";
        amountOfChange = 0;
        numOfFiftyCents = 0;
        numOfTwentyCents = 0;
        numOfTenCents = 0;
        numOfFiveCents = 0;
    }
```

```
    public Change(String initialName, int initialAmountOfChange) {

        name = initialName;
        amountOfChange = initialAmountOfChange;
        numOfFiftyCents = 0;
        numOfTwentyCents = 0;
        numOfTenCents = 0;
        numOfFiveCents = 0;
    }

     public Change(String initialName) {

        name = initialName;;
        amountOfChange = 0;
        numOfFiftyCents = 0;
        numOfTwentyCents = 0;
        numOfTenCents = 0;
        numOfFiveCents = 0;
    }

     public Change(int initialAmountOfChange) {

        name = "None";
        amountOfChange = initialAmountOfChange;
        numOfFiftyCents = 0;
        numOfTwentyCents = 0;
        numOfTenCents = 0;
        numOfFiveCents = 0;
    }




    /**
     * Pre-condition: name is a string that is neither non null or empty. Also
     * newAmountOfChange is an integer that represents the amount of money given by the person
(called change amount because it will be exchanged for coins). It must be in range and
multiple of 5
     *
```

```
     * Post-condition: assigns the newName parameter string to the current name instance
variable or displays an error message
     *
     */

   public void SetChange(String newName, int newAmountOfChange) {
       SetName(newName);
       SetAmountOfChange(newAmountOfChange);
       numOfFiftyCents = 0;
       numOfTwentyCents = 0;
       numOfTenCents = 0;
       numOfFiveCents = 0;
   }



    /**
     * Pre-condition: name is a string that is neither non null or empty
     * Post-condition: assigns the newName parameter string to the current name instance
variable or displays an error message
     *
     */

    public void SetName(String newName) {

        if(!newName.isEmpty() && newName != null) { //Validates whether the name set by the
user is empty or null. An error will be outputed when either of those are true
            name = newName;
        } else {
            System.out.println("Error: Invalid name"); //Output error message when string in
the parameter is empty or null
        }
    }



    /**
     * Pre-condition: newAmountOfChange is an integer that represents the amount of money
given by the person (called change amount because it will be exchanged for coins). It must be
in range and multiple of 5
```

```
     * Post-condition: assigns the newAmountOfChange parameter integer to the current
amountOfChange instance variable or displays an error message

     *

     */



    public void SetAmountOfChange(int newAmountOfChange) {



        Boolean inRange = ((newAmountOfChange >= 5) && (newAmountOfChange <= 95)); //Stores a
boolean variable containing the in range condition check

        Boolean multipleOfFive = (newAmountOfChange % 5 == 0);



        if((inRange) && (multipleOfFive)) { //Checks whether the new AmountOfChange inputed by
the user is in rage and a multiple of five

            amountOfChange = newAmountOfChange;

        } else {

            System.out.println("Incorrect coin value. Must be in range between 5 to 95 and
multiple of 5");

        }



    }









    /**

     * Pre-condition: newAmountOfChange is an integer that represents the amount of money
given by the person (called change amount because it will be exchanged for coins). It must be
in range and multiple of 5

     * Post-condition: adds the newAmountOfChange integer value to the current existing
amountOfChange instance variable or displays an error message. Used when a user wants to add
more change to a persons set change amount

     *

     */



    public void AddAmountOfChange(int newAmountOfChange) {



        Boolean inRange = ((newAmountOfChange >= 5) && (newAmountOfChange <= 95)); //Stores a
boolean variable containing the in range condition check

        Boolean multipleOfFive = (newAmountOfChange % 5 == 0);
```

```
       if((inRange) && (multipleOfFive)) { //Checks whether the new AmountOfChange inputed by
the user is in rage and a multiple of five

            amountOfChange += newAmountOfChange; //Add the newAmountOfChange to the
existing/set amountOfChange

        } else {

            System.out.println("Incorrect coin value. Must be in range between 5 to 95 and
multiple of 5");

        }




    }









    /**

     * Pre-condition: newNumOfFiftyCents is an integer that represents the number of fifty
cents a person has

     * Post-condition: assigns newNumOfFiftyCents integer value to the current numOfFiftyCents
instance variable

     *

     */



    public void SetNumOfFiftyCents(int newNumOfFiftyCents) {

        numOfFiftyCents = newNumOfFiftyCents;

    }



     /**

     * Pre-condition: newNumOfTwentyCents is an integer that represents the number of twenty
cents a person has

     * Post-condition: assigns newNumOfTwentyCents integer value to the current
numOfTwentyCents instance variable

     *

     */



    public void SetNumOfTwentyCents(int newNumOfTwentyCents) {

        numOfTwentyCents = newNumOfTwentyCents;

    }



    /**

     * Pre-condition: newNumOfTenCents is an integer that represents the number of ten cents a
person has
```

```
     * Post-condition: assigns newNumOfTenCents integer value to the current numOfTenCents
instance variable
     *
     */

    public void SetNumOfTenCents(int newNumOfTenCents) {
        numOfTenCents = newNumOfTenCents;
    }

    /**
     * Pre-condition: newNumOfFiveCents is an integer that represents the number of five cents
a person has
     * Post-condition: assigns newNumOfFiveCents integer value to the current numOfFiveCents
instance variable
     *
     */

    public void SetNumOfFiveCents(int newNumOfFiveCents) {
        numOfFiveCents = newNumOfFiveCents;
    }

     /**
     * Post-condition: returns the instance variable name as a string
     */
    public String GetName() {

        return name;
    }
    /**
     * Post-condition: returns the instance variable amountOfChange as an integer
     */
    public int GetAmountOfChange() {

        return amountOfChange;
    }
    /**
     * Post-condition: returns the instance variable numOfFiftyCents as an integer
     */
```

```
public int GetNumOfFiftyCents() {
    return numOfFiftyCents;
}
/**
 * Post-condition: returns the instance variable numOfTwentyCents as an integer
 */
public int GetNumOfTwentyCents() {
    return numOfTwentyCents;
}
/**
 * Post-condition: returns the instance variable numOfTenCents as an integer
 */
public int GetNumOfTenCents() {
    return numOfTenCents;
}
/**
 * Post-condition: returns the instance variable numOfFiveCents as an integer
 */
public int GetNumOfFiveCents() {
    return numOfFiveCents;
}
```

```
/**
 * Pre-condition: object must be instantiated
 * Post-condition: displays the initial (unset) instance variables of the object
 *
 */

public void WriteInitialRecord() {

    System.out.printf("Name: %s \n", name);
    System.out.printf("Change amount: %d \n \n", amountOfChange);

    System.out.println("Change: ");
    System.out.printf("50 cents: %d \n", numOfFiftyCents);
```

```
        System.out.printf("20 cents: %d \n", numOfTwentyCents);
        System.out.printf("10 cents: %d \n", numOfTenCents);
        System.out.printf("5 cents: %d \n", numOfFiveCents);
    }


}
```