Sadiki Brown

Homework 2

COP4331 001

Number 1:

Delete from Specific Number

1. The mailbox owner carries out Log In.
2. The mailbox owner selects the “retrieve your messages” menu option.
3. The voicemail system plays the message menu:
    - *Enter 1 to listen to the current message.*
    - *Enter 2 to save the current message.*
    - *Enter 3 to delete the current message.*
    - *Enter 4 to return to the mailbox menu.*
    - *Enter 5 to search for specific caller.*
4. The mailbox owner selects the “Search” menu option.
5. The voicemail system speaks a prompt.
    - *Enter caller phone number followed by #.*
6. The user types in the phone number of the caller.
7. The voicemail system plays specific caller message menu
    - *Listen current*
    - *Listen all*
    - *Save current*
    - *Save all*
    - *Delete current*
    - *Delete all*
    - *Return to mailbox menu*
8. The mailbox owner selects delete all
9. Continue with step 7.

Variation #I. Enter incorrect number

1.1.Start at step 5

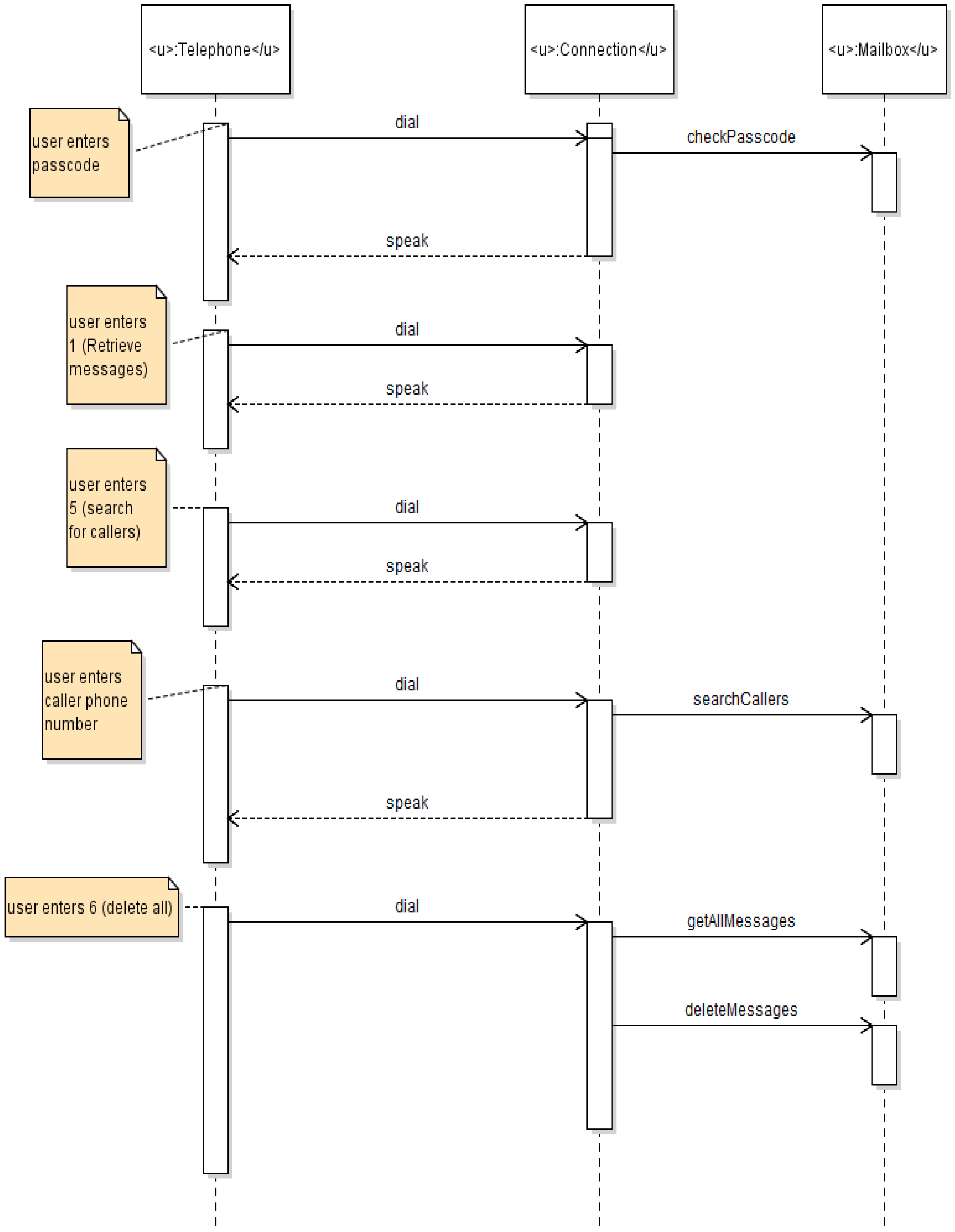
1.2.The user types in the phone number of the caller incorrectly.

1.3.The voicemail system speaks.

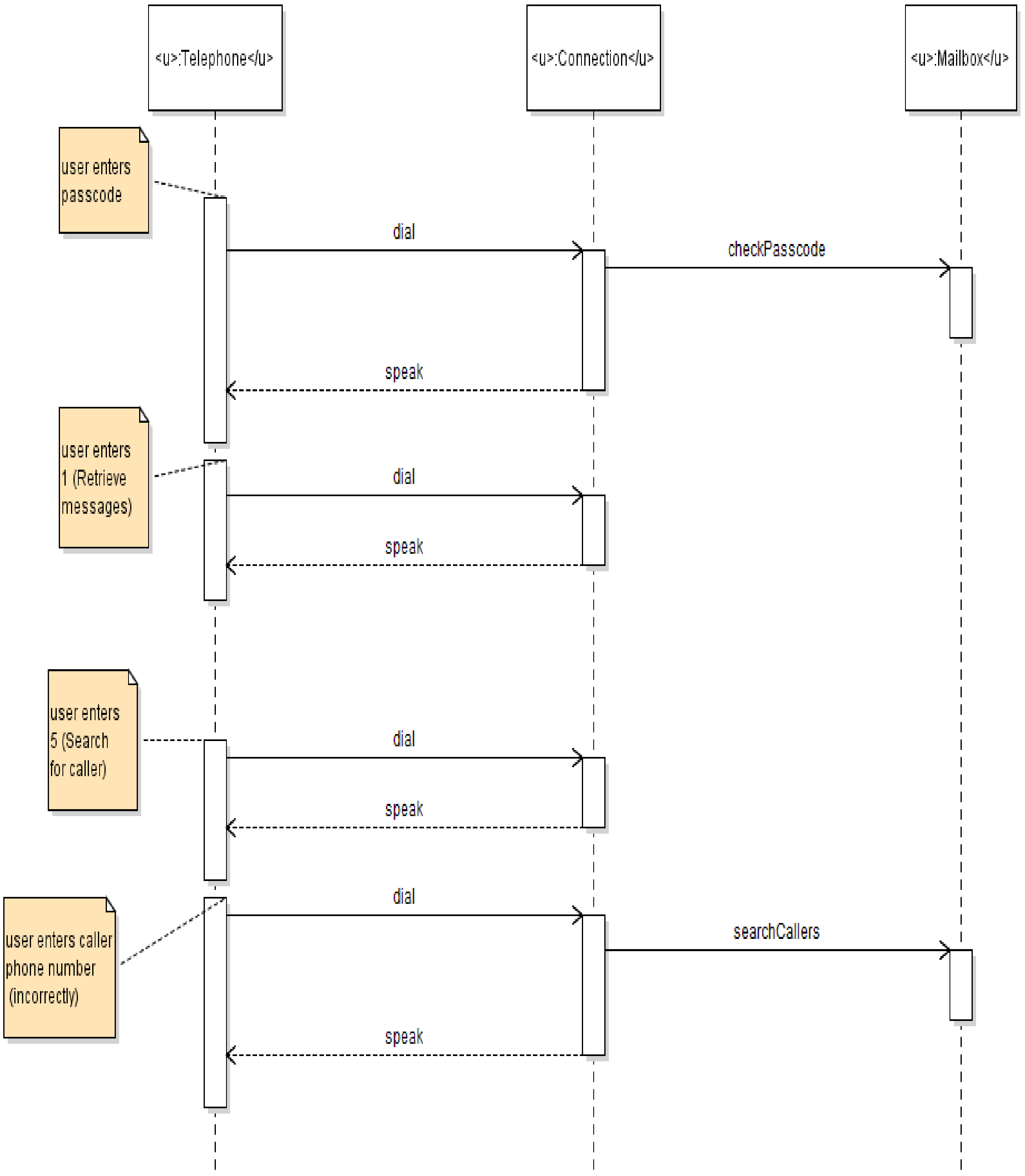
- *No messages from that caller found*

1.4.Continue with step 3.

1b - Sequence diagram:

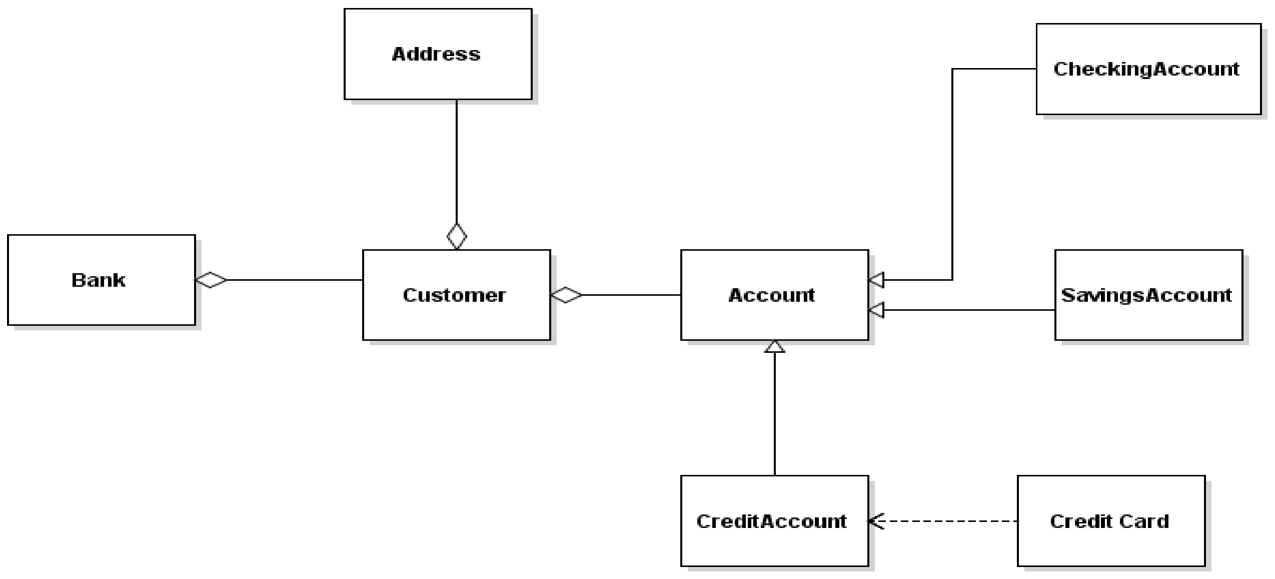


1b - Sequence w/Variation:



Number 2:

UML Class diagram:

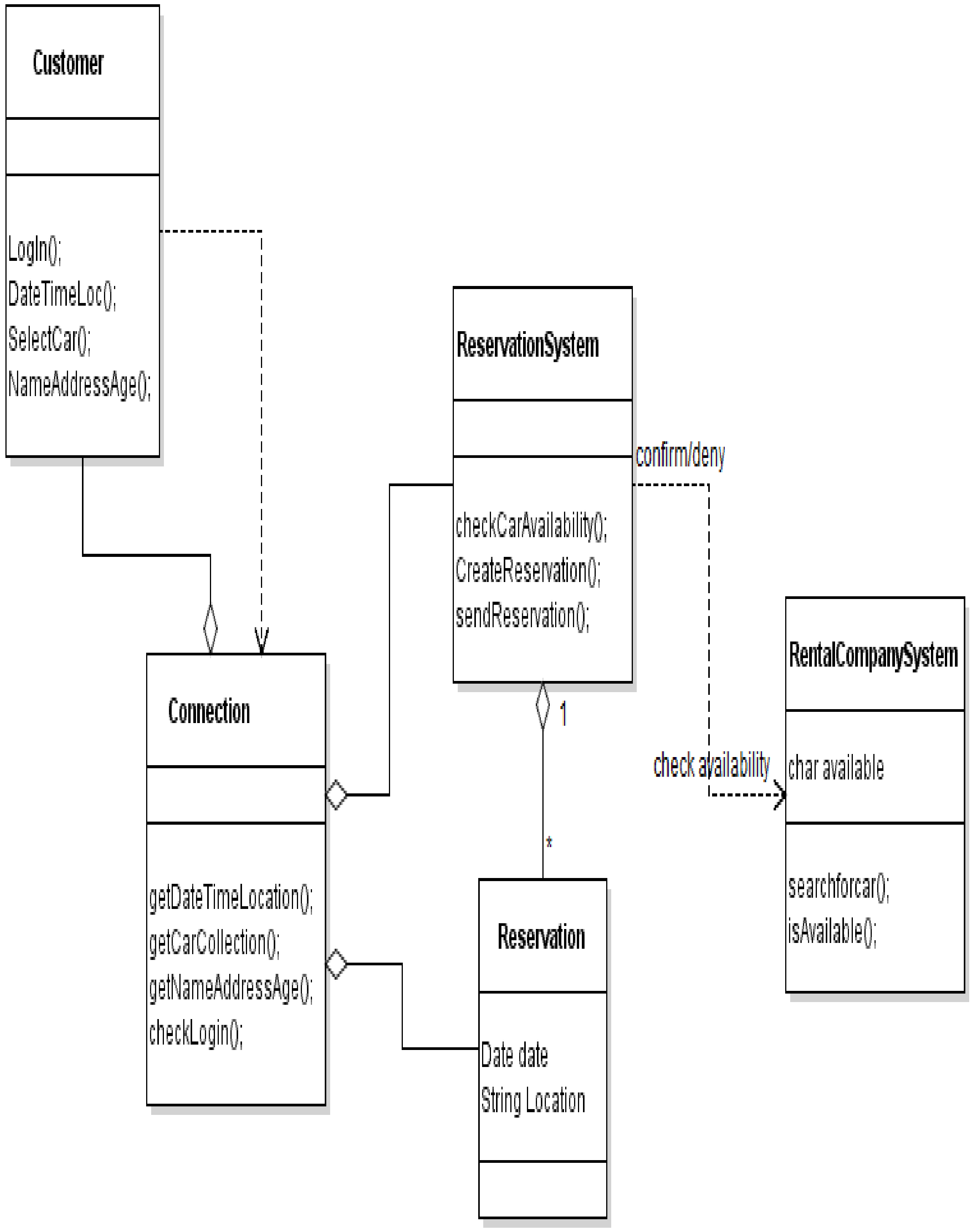


Number 3:

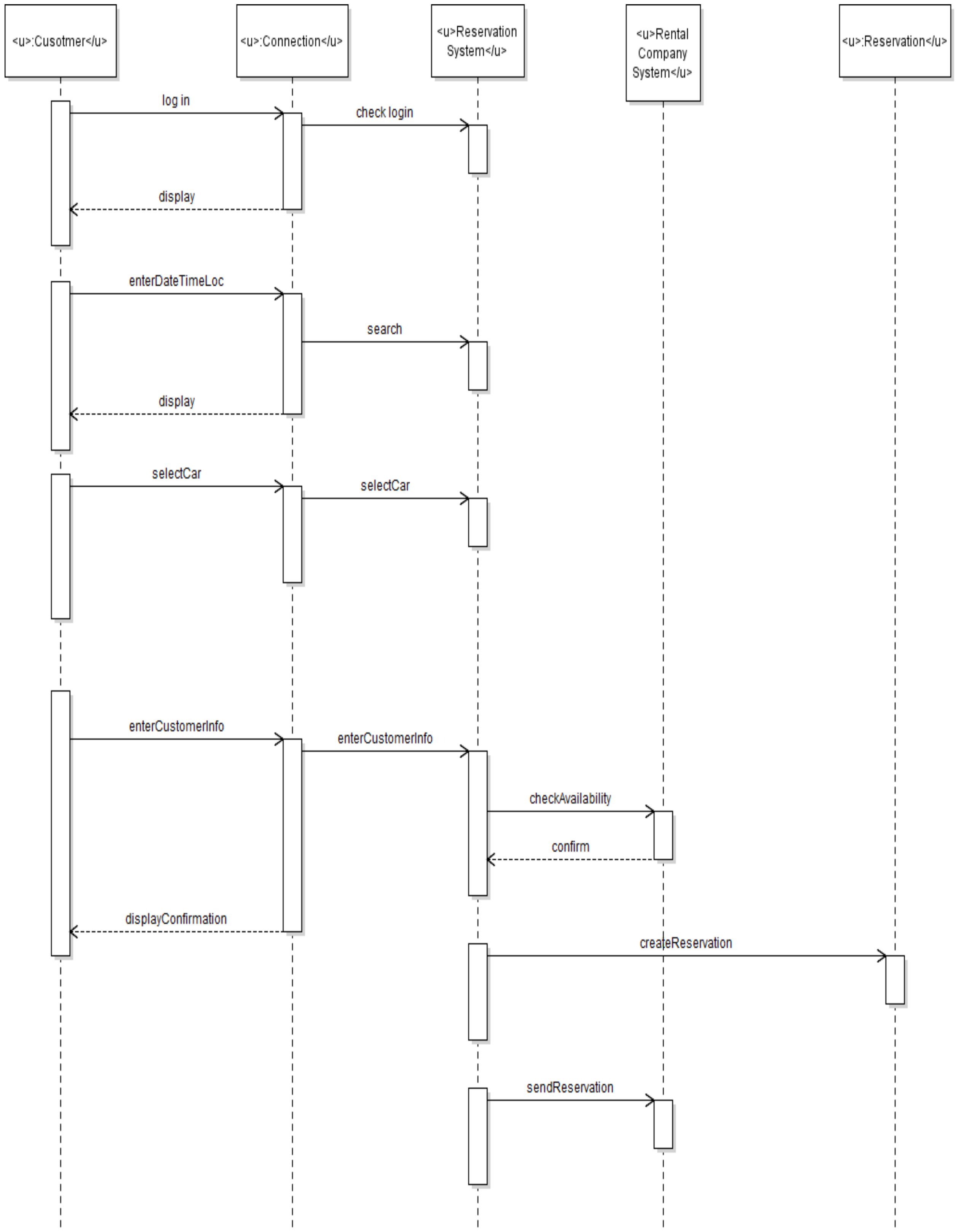
3a: CRC cards

- Customer
  - ➢ Enter prompted information
    - o Connection();
- Connection
  - ➢ Get user input
    - o Customer();
  - ➢ Execute/apply user commands/input
    - o ReservationSystem();
- ReservationSystem
  - ➢ Process user commands/input
    - o Connection();
  - ➢ Check car availability/send reservations
    - o RentalCompanySystem();
  - ➢ Create reservations
    - o Reservation();
- RentalCompanySystem
  - ➢ Confirm/deny car availability
    - o ReservationSystem();
- Reservation
  - ➢ Manage reservation information

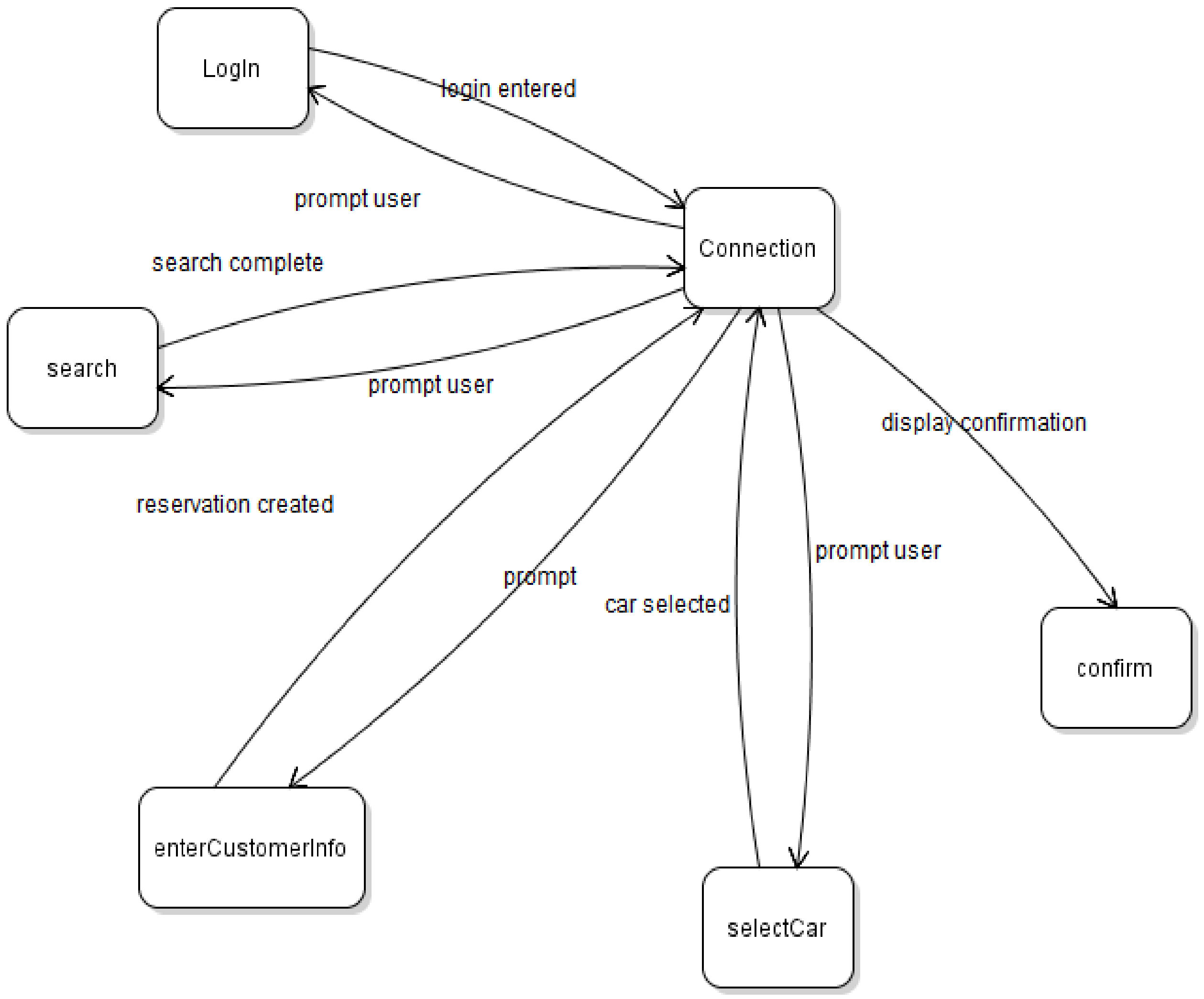
3b: UML class diagram:



3c: Sequence Diagram:



3d: State diagram:

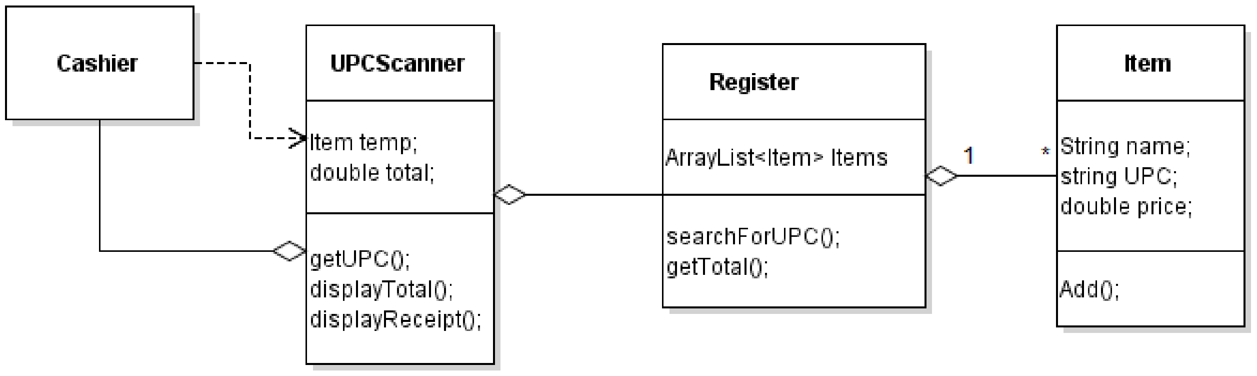


Number 4:

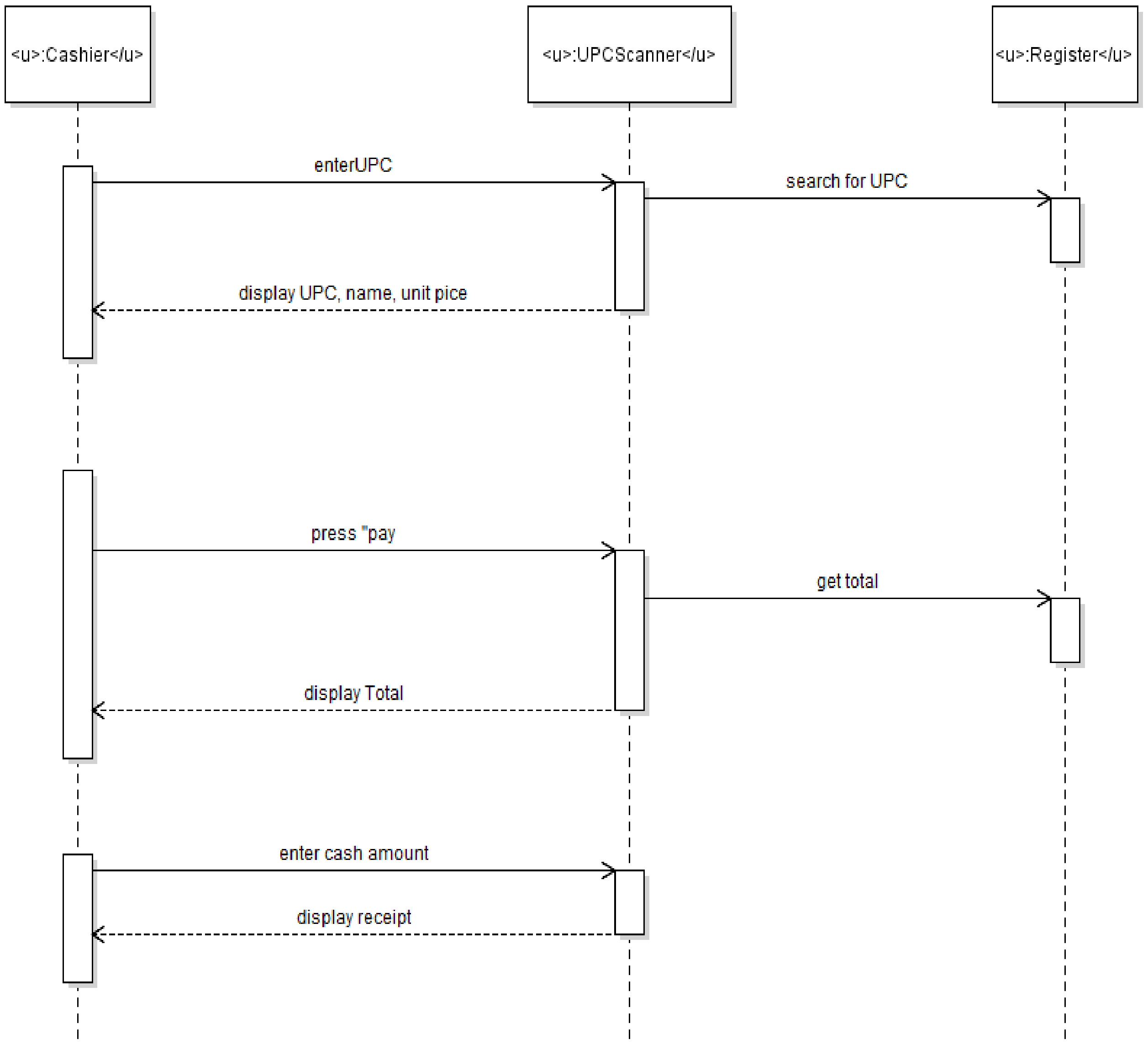
CRC cards:

- Cashier
  - ➢ Enter prompted information
    - o UPCScanner();
- UPCScanner
  - ➢ Take input from cashier
    - o Cashier();
- Register
  - ➢ Manage item information
    - o Item();
- Item
  - ➢ Hold item structure

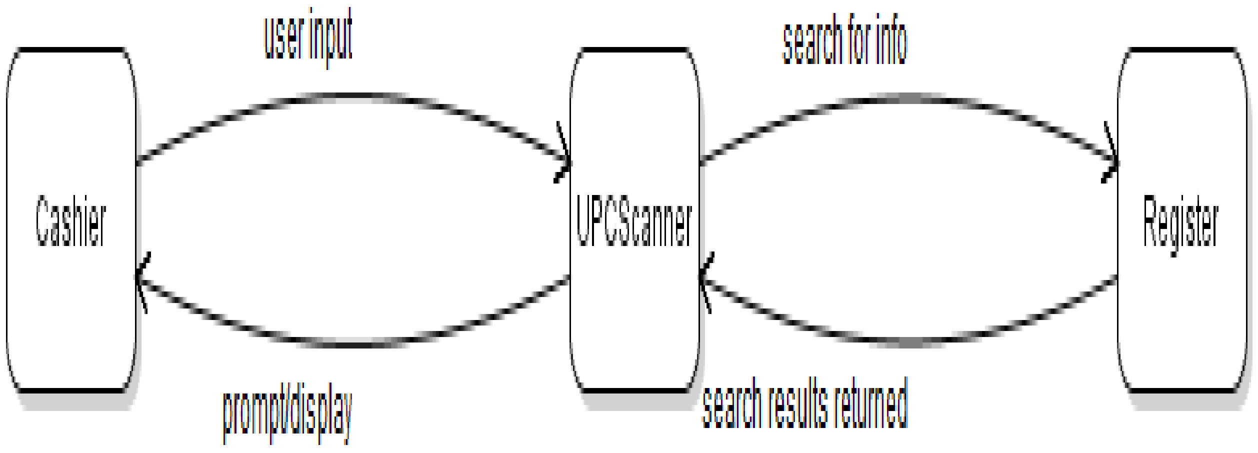
UML class diagrams:



Sequence Diagram:



State diagram:



Item.java:

```
package q4;

/**Class to define item*/
public class Item {

	/**Universal Product Number variable*/
	public String UPC;

	/**Constructor - Initialize class fields with input from parameters.*/
	public Item(String UPC, String name, double unitprice)
	{
		this.UPC = UPC;
		this.name = name;
		this.unitprice = unitprice;
		this.quantity = 0;
	}

	/**Add method to use to add another item to the list of items.*/
	public void add()
	{
		/**Increase quantity of of list*/
		this.quantity = this.quantity + 1;
	}

	/**Item name*/
	public String name;

	/**Item unit price*/
	public double unitprice;

	/**Quanitity of items in list of items*/
	private int quantity;
}
```

Register.java:

```
package q4;
import java.util.ArrayList;

/**Manage item information*/
public class Register {

      /**Constructor - Initialize register with hardcoded values.*/
      public Register() {
        this.items.add(new Item("101", "Snickers bar ", 1.00));
        this.items.add(new Item("102", "Skittles     ", 1.50));
        this.items.add(new Item("103", "Twix         ", 1.75));
        this.items.add(new Item("104", "Gum          ", 2.00));
        this.items.add(new Item("105", "Hershey bar  ", 2.50));
        this.items.add(new Item("106", "Kit Kat      ", 1.50));
        this.items.add(new Item("107", "Nerds        ", 1.00));
        this.items.add(new Item("108", "Butterschotch", 1.50));
        this.items.add(new Item("109", "Starburst    ", 2.00));
        this.items.add(new Item("110", "York Patty   ", 1.75));
    }

      /**Search for given UPC and displays info on object if found.*/
      public void SearchandDisplay(String UPC)
      {
            /**Iterate through all items*/
            for (int i = 0; i < items.size(); i++)
            {
                  /**Give "current" the current item in the list of items.*/
                  current = items.get(i);

                  /**If statement - If the given UPC equals the current item's UPC
enter*/
                  if(UPC.equals(current.UPC))
                  {
                        /**create namepass to hold the current item's name and
then print it*/
                        String namepass = current.name;
                        System.out.println("Item name: " + current.name);

                        /**Create UPCpass to hold current item's UPC and then
print it*/
                        String UPCpass = current.UPC;
                        System.out.println("Item UPC: " + current.UPC);

                        /**Create pricepass to hold current item's price and print
it.*/
                        double pricepass = current.unitprice;
                        System.out.println("Item unit price: $" +
current.unitprice);

                        /**Add item to list of selected items.*/
                        selections.add(new Item(UPCpass, namepass, pricepass));
                        return;
                  }
        }

        /**Notify cashier that UPC was not found.*/
        System.out.println("UPC not found");
        return;

    }

    /**Create array list to hold selected items.*/
    public ArrayList<Item> selections = new ArrayList<Item>();

    /**Create array list to hold all items in register*/
    public ArrayList<Item> items = new ArrayList<Item>();

    /**Create dummy item object*/
    Item current = new Item("junk", "junk", 0);
}
```

UPCScanner.java:

```
package q4;

public class UPCScanner {

    /**Constructor*/
    public UPCScanner()
    {

    }

    /**Calls getUPC.*/
    public void displayItemInfo()
    {
        user.getUPC();
    }

    /**Calls calculateTotal and prints the total.*/
    public void displayTotal()
    {
        /**Call calculateTotal and store in variable totalprice.*/
        totalprice = user.calculateTotal();

        /**Print totalprice.*/
        System.out.println("Total: $" + totalprice);
    }

    /**Displays receipt - Call printSelections and print the total.*/
    public void displayReceipt()
    {
        /**Print array list of selections*/
        user.PrintSelections();

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

        /**Print total price.*/
```

```java
		System.out.println("Total: $" + totalprice);
	}

	/**Variable to hold the total price once calculated.*/
	public double totalprice;

	/**Create new cashier object called user.*/
	private Cashier user = new Cashier();

	/**Create item object called current to hold current item when traversing list
of items*/
	Item current = new Item("junk", "junk", 0);

	public static void main(String[] args) {
		/**Create UPCScanner object*/
		UPCScanner buffer = new UPCScanner();

		/**Call displayItemInfo*/
		buffer.displayItemInfo();

		/**Call displayTotal*/
		buffer.displayTotal();

		/**Call displayReceipt*/
		buffer.displayReceipt();
	}

}
```

Cashier.java:

```java
package q4;

import java.util.Scanner;

/**Cashier class*/
public class Cashier {

	/**Constructor*/
	public Cashier() {}

	/**Prompts cashier for UPC until the enter N/n indicating they want to exit.*/
	public void getUPC()

	{
		/**Create scanner object.*/
		Scanner scan = new Scanner(System.in);

		/**Loop control variable initialization.*/
		char pay = 'N';

		/**While loop that prompts cashier for UPC and calls SearchandDisplay to
find that item.*/
		while (pay == 'N')
		{
```

```
                        /**Prompt user for UPC and scan.*/
                        System.out.println("Please enter the UPC: ");
                        UPC = scan.nextLine();

                        /**Call Search and Display to find the item and display its 
information.*/
                        register.SearchandDisplay(UPC);

                        /**Prompt user to update status on loop control variable*/
                        System.out.println("Are you ready to pay? (y/n): ");
                        pay = scan.next().charAt(0);
                        pay = Character.toUpperCase(pay);
                        scan.nextLine();
                }

                /**Close scanner*/
                scan.close();
        }

        /**Calculates total price of selected items.*/
        public double calculateTotal()
        {
                /**Create variable to hold total price and itinitialize to zero.*/
                double total = 0.00;

                /**For loop to get total price from each item in list of selections.*/
                for (int i = 0; i < register.selections.size(); i++)
                {
                        /**Temp holds current item in list of selections.*/
                        temp = register.selections.get(i);

                        /**Update total with included item price.*/
                        total += temp.unitprice;
                }

                /**Return total price value*/
                return total;
        }

        /**Print list of selections*/
        public void PrintSelections()
        {
                /**Indicate start of receipt.*/
                System.out.println("\n------------RECEIPT------------");

                /**Loop to go through all items in list of selections.*/
                for (int i = 0; i < register.selections.size(); i++)
                {
                        /**temp holds current item in list of selections.*/
                        temp = register.selections.get(i);

                        /**Print info on all items in list of selections.*/
                        System.out.println(temp.name + "\t" + temp.UPC + "\t$" + 
temp.unitprice);
                }
        }

        /**Holds the UPC from the cashier.*/
        public String UPC;

        /**Temp to hold current item in list of items.*/
        Item temp = new Item("junk", "junk", 0);

        /**Create register object*/
        Register register = new Register();
}
```

Number 5:

Read Recent Posts

1. X enters their login information
2. Menu displayed
    - *Edit Profile*
    - *Write Post*
    - *Read Recent Posts*
3. X selects Read Recent Posts
4. ChainedIn retrieves recent posts made by people in X's social network from ChainStream and displays it to X.
5. X returns to 2.

Write Post

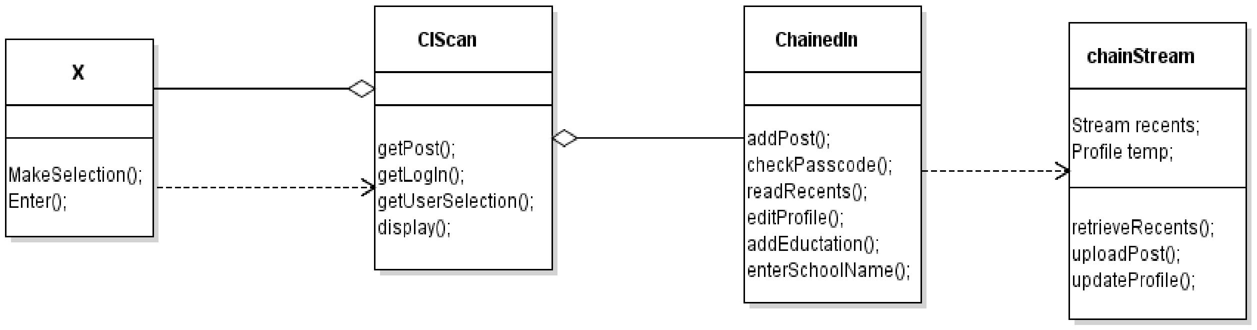
1. X enters their login information
2. Menu displayed
    - *Edit Profile*
    - *Write Post*
    - *Read Recent Posts*
3. X selects Write Post
4. ChainedIn prompts X for post.
5. X writes and sends post
6. ChainedIn adds posts to ChainStream
7. Return to step 2.

Edit Profile

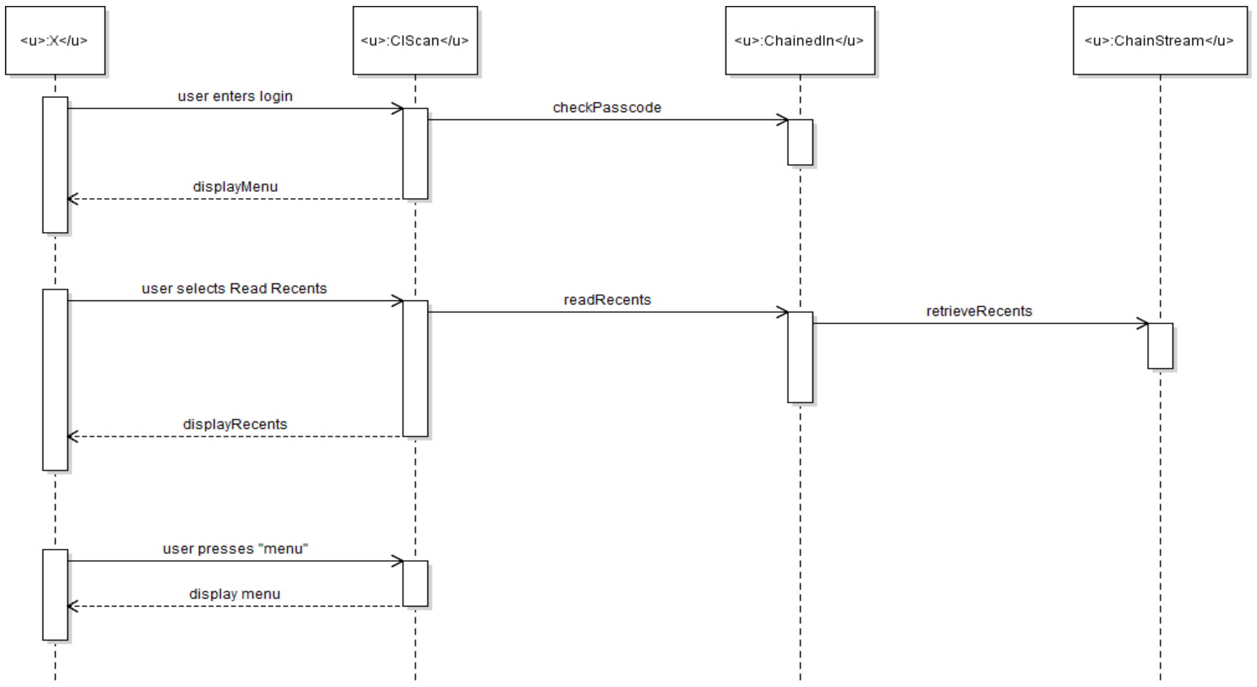
1. X enters their login information
2. Menu displayed
    - *Edit Profile*
    - *Write Post*
    - *Read Recent Posts*
3. X selects Edit Profile

4. ChainedIn displays Edit Menu
    - *Name*
    - *Education*
    - *Work History*
5. X selects Education
6. ChainedIn displays Edit Menu
    - *Add*
    - *Remove*
7. X selects add
8. ChainedIn prompts user for new school name
9. X enters school name
10. ChainedIn updates X's profile
11. Return to step 2

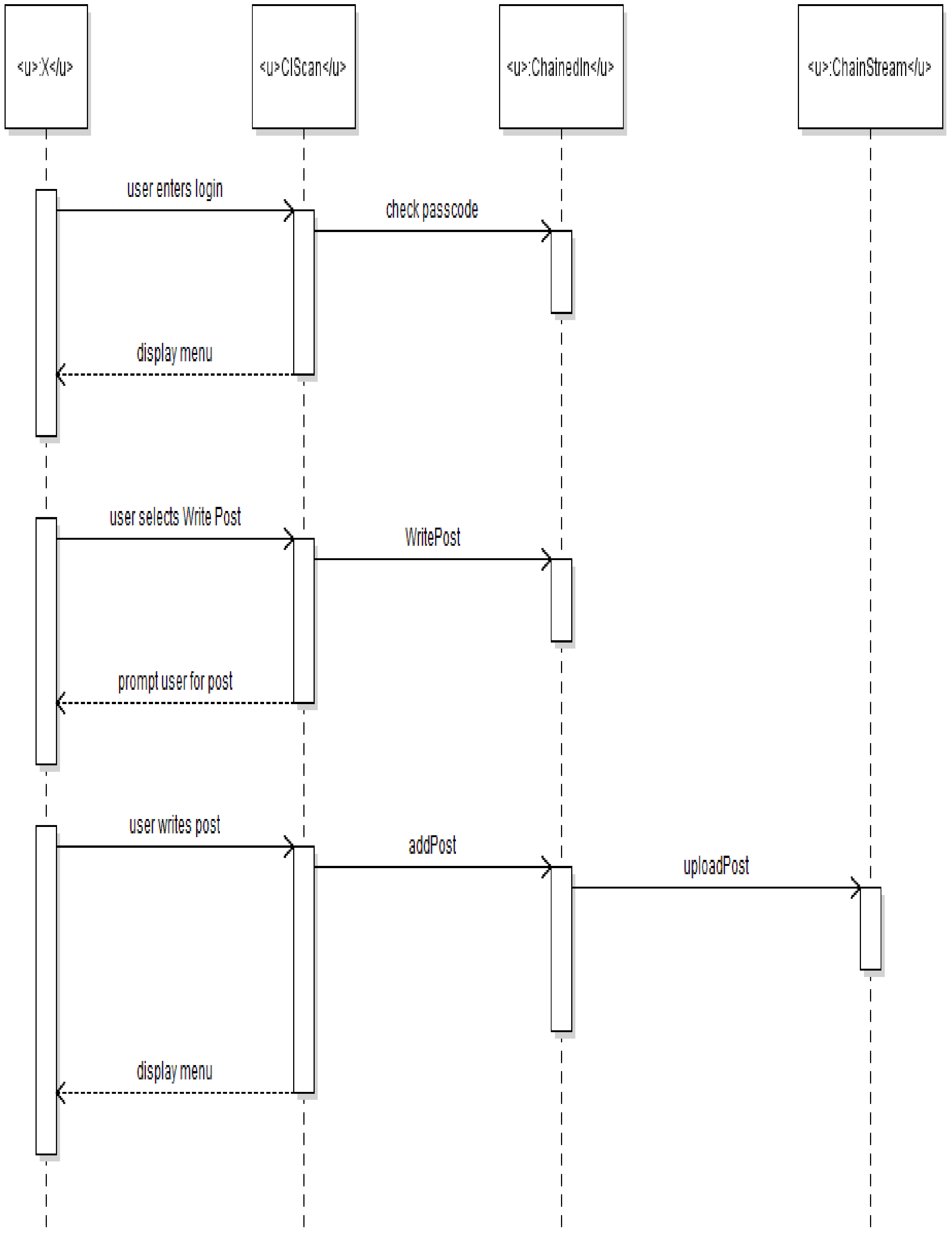
UML class diagram:



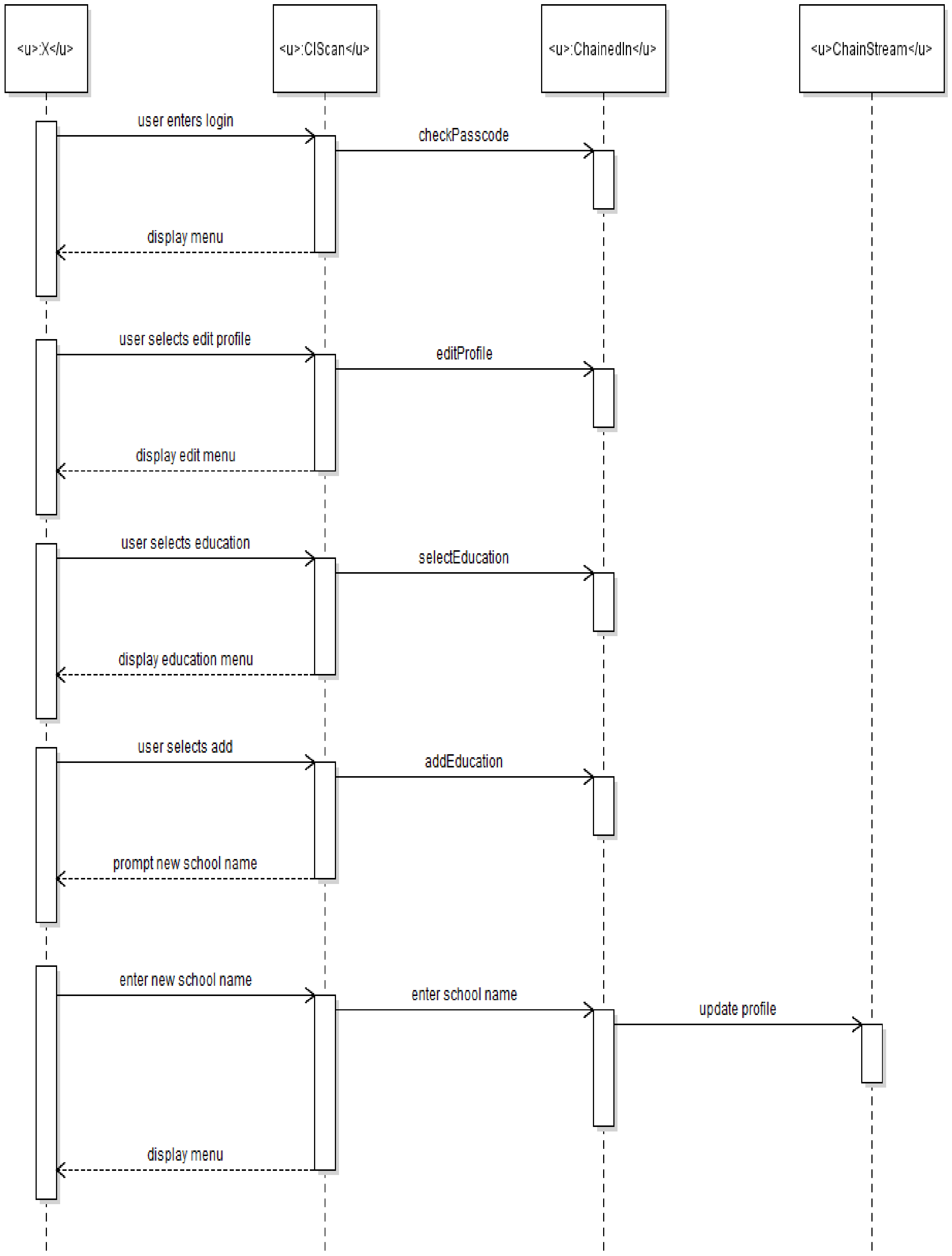
Read Recent Posts Sequence Diagram:



Write Post Sequence Diagram:



Edit Profile Sequence Diagram:



<u>:X</u>
<u>:CIScan</u>
<u>:ChainedIn</u>
<u>ChainStream</u>
user enters login
checkPasscode
display menu
user selects edit profile
editProfile
display edit menu
user selects education
selectEducation
display education menu
user selects add
addEducation
prompt new school name
enter new school name
enter school name
update profile
display menu