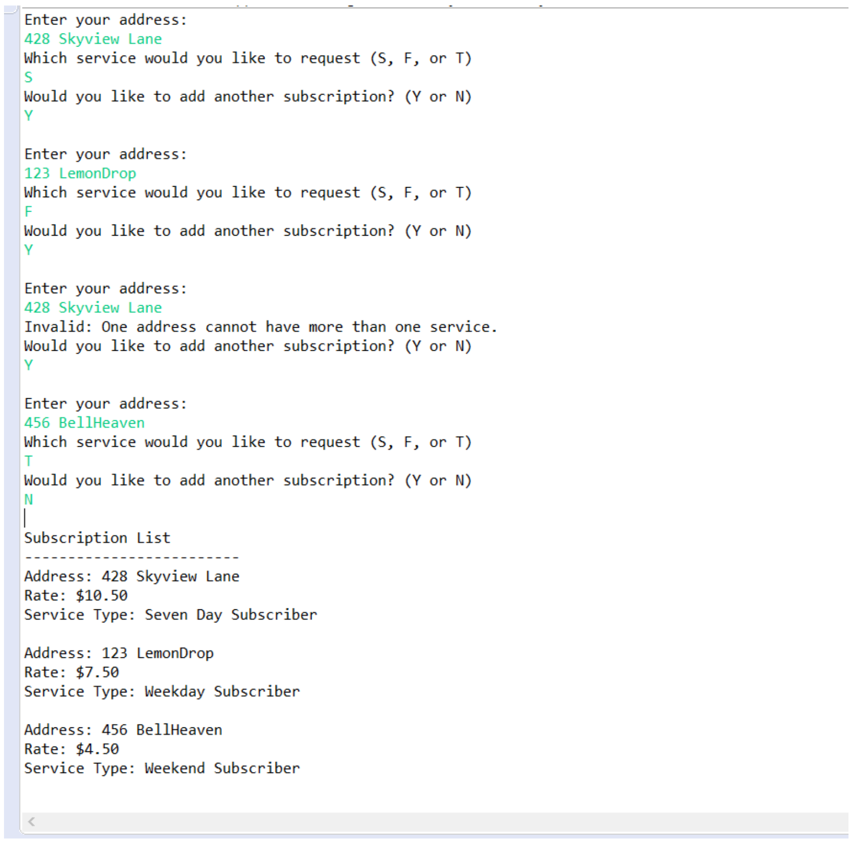
Jeremy Boyd
Assignment 05
7/19/2018

Problem 1

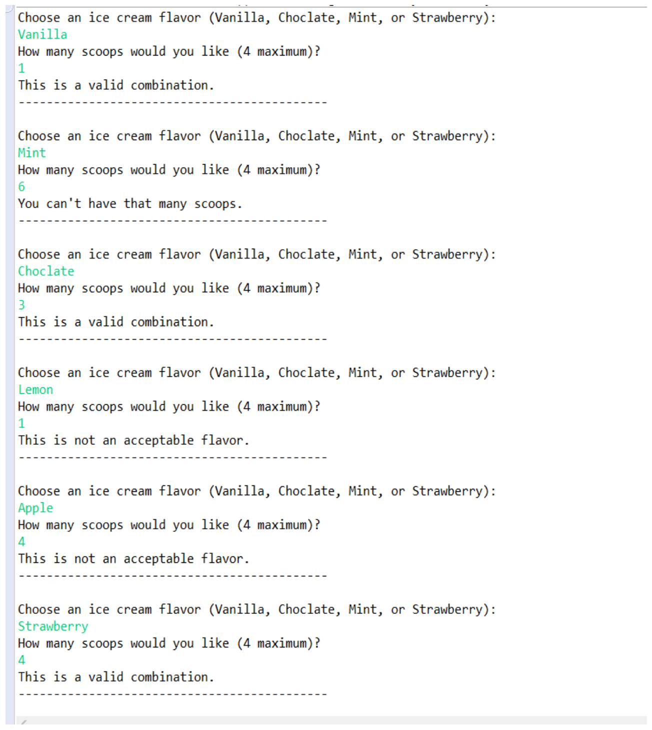
This problem had a few challenges but overall wasn't too difficult to get through. I started off by creating the *NewsPaperSubscriber* class and made it abstract. I then created two instance variables for the address and subscription rate. Next I created the constructor for the class which would accept an address. I then created my getter and setter methods for my instance variables and specifically made the *setRate()* method abstract so each class would have to have their own definition for it. Then I created an *equals()* method that will test if two addresses are the same and using a Boolean variable return true or false. Next, I created the three child classes called *SevenDaySubscriber*, *WeekdaySubscriber*, and *WeekendSubscriber* each class had a constructor that accepted an address and used the keyword super to reference the parent classes constructor. Also, each class defined the *setRate()* method according to how much it should cost for each subscription. And each class included a *toString()* method that displayed the information about the subscription in a nice manner. Next, I created a client class called *Subscribers* and imported the ArrayList and Scanner class. Then I created a scanner object and an array list of type *NewsPaperSubscriber* as well as various other variables that would be used to collect information. After this I created some output and input statements that would get information from the user and based off what the user inputs for their subscription choice a switch statement would insert the variables into the array list under the proper subscription type. Then the program would ask the user whether or not they want to add another subscription if the user says yes, the program enters a while loop and the user is prompted to enter more information. Within the while loop the *equals()* method is used to make sure the user isn't using the same address twice. If they are, an error message is displayed and the user once again is asked if they want to add another subscription. Once the user says no, the program enters a for loop that displays the list of subscribers.



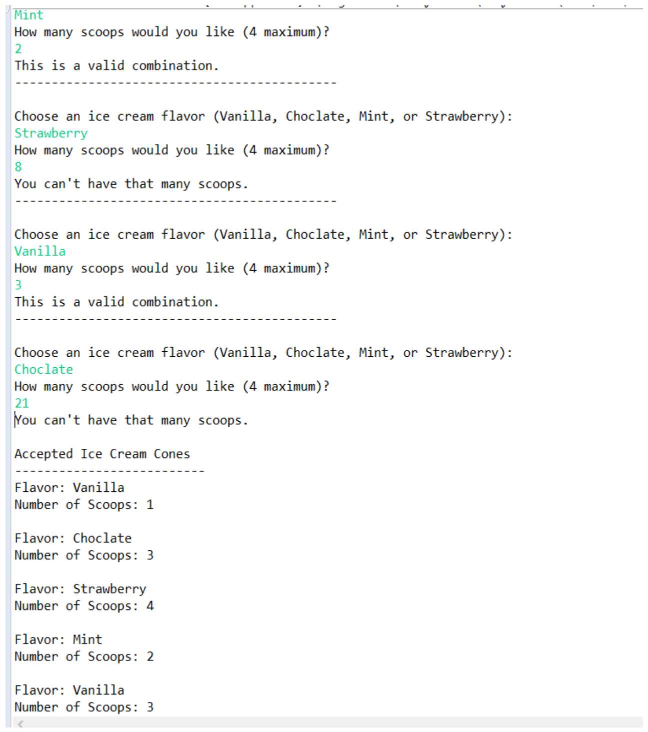
```
Enter your address:
428 Skyview Lane
Which service would you like to request (S, F, or T)
S
Would you like to add another subscription? (Y or N)
Y

Enter your address:
123 LemonDrop
Which service would you like to request (S, F, or T)
F
Would you like to add another subscription? (Y or N)
Y

Enter your address:
428 Skyview Lane
Invalid: One address cannot have more than one service.
Would you like to add another subscription? (Y or N)
Y

Enter your address:
456 BellHeaven
Which service would you like to request (S, F, or T)
T
Would you like to add another subscription? (Y or N)
N

Subscription List
-------------------------
Address: 428 Skyview Lane
Rate: $10.50
Service Type: Seven Day Subscriber

Address: 123 LemonDrop
Rate: $7.50
Service Type: Weekday Subscriber

Address: 456 BellHeaven
Rate: $4.50
Service Type: Weekend Subscriber
```

Problem 2

This problem was interesting to work through because it wasn’t something I’ve done before using this type of method. But I enjoyed the challenge and learning something new, to start I created a class called *IceCreamConeException* and extended it as an exception class. I then created the constructor for the class that would receive a string that would be an error message. Next, I created the *IceCreamCone* class and created private instance variables for the flavor, scoops, an array with my four selected flavors in it, and two exception messages using my *IceCreamConeException* class. Then I created a constructor for the class which would receive a flavor and number of scoops and included methods for *setFlavor* and *setScoops*. Which are two methods I created which both throw their own *IceCreamConeException* if the designated parameters aren’t met. Lastly, I created a *toString()* that would display the flavor and number of scoops in a nice format. I then created a client class called *IceCreamConeTester* and imported the scanner class. I created several variables that would be used to collect information including an array of maximum size 10 that would hold the list of *IceCreamCone* objects created. I then used a series of output and input statements to ask the user for a flavor and number of scoops. The program then enters a while loop that ends after 10 iterations the maximum number of *IceCreamCone* objects that can be put in the array. Within the while loop I created a Boolean variable that would be used later in the program. I then used the try and catch statements to create an *IceCreamCone* object and insert it into the array and then increment a variable that would control the index of the array. If an exception is caught the instance won’t be inserted into the array and it will be thrown so an error message will be displayed depending on what the user did wrong. Also, within the catch statement the boolean variable I created is set to false, but if there’s no exception since the variable remains true a message will be output telling the user they entered a valid combination. The count variable for the while loop is incremented and an if statement is used to see if the count variable is at 10. If so it will break out of the loop, if not then the loop will continue and repeat. Then lastly, I used a for loop to display all the valid *IceCreamCone* objects that the user created.



```
Choose an ice cream flavor (Vanilla, Choclate, Mint, or Strawberry):
Vanilla
How many scoops would you like (4 maximum)?
1
This is a valid combination.
--------------------------------------------

Choose an ice cream flavor (Vanilla, Choclate, Mint, or Strawberry):
Mint
How many scoops would you like (4 maximum)?
6
You can't have that many scoops.
--------------------------------------------

Choose an ice cream flavor (Vanilla, Choclate, Mint, or Strawberry):
Choclate
How many scoops would you like (4 maximum)?
3
This is a valid combination.
--------------------------------------------

Choose an ice cream flavor (Vanilla, Choclate, Mint, or Strawberry):
Lemon
How many scoops would you like (4 maximum)?
1
This is not an acceptable flavor.
--------------------------------------------

Choose an ice cream flavor (Vanilla, Choclate, Mint, or Strawberry):
Apple
How many scoops would you like (4 maximum)?
4
This is not an acceptable flavor.
--------------------------------------------

Choose an ice cream flavor (Vanilla, Choclate, Mint, or Strawberry):
Strawberry
How many scoops would you like (4 maximum)?
4
This is a valid combination.
--------------------------------------------
```



```
Mint
How many scoops would you like (4 maximum)?
2
This is a valid combination.
--------------------------------------------

Choose an ice cream flavor (Vanilla, Choclate, Mint, or Strawberry):
Strawberry
How many scoops would you like (4 maximum)?
8
You can't have that many scoops.
--------------------------------------------

Choose an ice cream flavor (Vanilla, Choclate, Mint, or Strawberry):
Vanilla
How many scoops would you like (4 maximum)?
3
This is a valid combination.
--------------------------------------------

Choose an ice cream flavor (Vanilla, Choclate, Mint, or Strawberry):
Choclate
How many scoops would you like (4 maximum)?
21
You can't have that many scoops.

Accepted Ice Cream Cones
--------------------------
Flavor: Vanilla
Number of Scoops: 1

Flavor: Choclate
Number of Scoops: 3

Flavor: Strawberry
Number of Scoops: 4

Flavor: Mint
Number of Scoops: 2

Flavor: Vanilla
Number of Scoops: 3
```