Reflection Report:

Name: Carter Fitzgerald
ID: u3240494
Assignment: ST2 Assignment 1
Date: 31/03/23

Problems:

Throughout my development of the assignment program I came across a large number of bugs and problems, listed below are some of the bigger problems I faced or bugs I faced once all the code had been written.

- The first problem I encountered was when writing code for task 1, since each flat can have any number of tenants from 1 to 99, you can just make a class and have a tenant 1 field and a tenant 2 field etc.. So I thought about it and I could just store all the details other than the tenants of the flats as fields in a flat class then the last field for the class would store an arraylist of the tenants which use the flat. This allows me to no matter the amount of tenants in 1 building I just keep adding to the arraylist. Then I can clear the arraylist before making the next flat instance which gives me a new list of tenants for the new instance.

```
String[] splitArray = line.split(",");
int size = splitArray.length;

int int_buildingNo = Integer.parseInt(splitArray[1]);
int int_CReading = Integer.parseInt(splitArray[3]);
int int_PReading = Integer.parseInt(splitArray[5]);

ArrayList<String> tenantAList = new ArrayList<String>();

for(int i = 7; i < size; i++)
{
    tenantAList.add(splitArray[i]);
}

Flat Flat = new Flat(splitArray[0], int_buildingNo,
splitArray[2], int_CReading, splitArray[4], int_PReading,
splitArray[6], tenantAList);

flats.add(Flat);
```

- The next major problem I had was when I needed to count the meters read in by the file reader. Since I chose to store the tenants for each flat in an arraylist and I had all my flats stored in another arraylist. I had to loop through all the flats in the flats arraylist and count all the elements of the arraylist of tenants in that element of the arraylist. To do this I used a for each loop which let me loop through all the elements in the flats arraylist then I created another arraylist which was used to store all the tenants/meters and as I went through each elements in flats I just added all the elements of tenant arraylist to the list to be counted then added that to a count variable which stores the amount of meters read in.

```
int count = 0;
for (Flat Flats:flats) {
    ArrayList<Integer> arrayListToCount = Flats.getTenant();
    count += arrayListToCount.size();
}
System.out.printf("Number of Meters Read in is: %d%n", count);
```

- After getting task 1 to work or one part of task 1, on the Dev 0 files I attempted to run the program using the Dev 1 files, this threw an error, as I was trying to access the second element in an arraylist when only 1 existed. This was caused because of the 2 empty lines at the end of 1 of the Dev 1 files. My first solution to this was to just remove the empty lines from the file, which solved my problem. But later in the project when the files got rereleased, I decided to change my program to only run if the line in the file wasn't empty. This meant I had to change to a buffered reader which checks if the line is empty if it is it skips it otherwise it runs the code to create classes from the files.

```
try {
    BufferedReader reader = new BufferedReader(new FileReader(file));
    String line = null;

    while ((line = reader.readLine()) != null) {
        if (line.isEmpty()) {
            continue;
        } else {
            String[] splitArray = line.split(",");
            int size = splitArray.length;
```

- A problem I had much later in the project was when I was programming for task 4 and I needed information from both the flats file and the meter file. This meant I had to access the instances of objects that were connected to one another. (Eg. The tenants that were listed for a certain apartment) At the time, a while after coding task 1 I thought that my arraylist in flats of tenants was an arraylist of tenant objects, which caused a lot of problems. Because actually I was just storing the meter numbers of each of the tenants in that apartment, meaning I had to use those meter numbers to associate the flat class with the tenant class. Below you can see the main code which solves this problem, above this code is a for each loop which is looping through every flat's bill in an arraylist and it checks to see if it is the building input by the user for task 4. Then comes the code below which is a for each loop of the tenant meter numbers stored in the matching flat bill. Then it takes the meter number, sets it as a variable and then it loops through the arraylist of all the tenants and checks to see if the meter number in the instance of the

tenant class matches the one from the flat's bill and if it does it proceeds with the program. Seen below.

```
for (Object tenantID : Bills.gettenants()) {
    String id = (String) tenantID;
    for (Tenant tenant : tenants) {
        if (tenant.getmeterNo().equals(id)) {
            honorificT = tenant.gethonorific();
            fNameT = tenant.getfName();
```

- Another problem I had was when I was doing task 5 which is basically task 4 but for every flat read in. The problem I was having was I was accessing a lot of variables and arraylists and then when I would access them a second time the contents of these variables and arraylists would be not what they should be. A specific example was when I was calculating the tenant bill for task 5 it was giving me a cumulative tenant bill as it processed each flat in the file. Instead of just the tenant bill for that flat. This was because I wasn't resetting the values which were creating the value to be stored in the bill class which was being displayed. An easy fix at the start of every loop reset all variables being used to 0. Seen below is me resetting the values for a bunch of variables which are needed in each iteration of the loop.

```
for (BC_Bill Bills: bills) {
    billusage = Bills.getusage();
    billTUsage = rate * billusage;
    totalCulmative = 0.00;
    baseCulmative = 0.00;
    usageCulmative = 0;
    adjCulmative = 0.00;
    double diff = 0.00;
```

- After coding task 5 to work with the Dev 1 files as I was comparing with the assignment examples, I tried my code on the test files and prod files and immediately an error was apparent. I never bothered to check if meter files had reached the cap (999999) and reset to (000000) so basically my usage numbers were way off in a few select cases. Regarding tenant meters. To fix this I coded an if statement which checks to see if the current meter reading is lower than the previous which will tell you if it reset and if it did it applies calculations to get the correct usage value. Seen Below. Furthermore, after fixing this it worked with the test file, but I believe it still had an issue because the prod file had a flat meter which reset, and I did apply the if statement when checking flat meters, so I added that in. Which solved my problem.

```
if (cMeterReadingT < pMeterReadingT) {
        usage = (cMeterReadingT + 1000000) - pMeterReadingT;
    } else {
        usage = cMeterReadingT - pMeterReadingT;
    }
```

- The last major problem I had was with testing task 5, I was testing task 5 and I noticed that when I would set the files to prod and run task 5 I would get the correct result and then if I switched to the test files and ran task 5 it would still give me the prod flats. After a bunch of print statements to check that it was only applying the correct flats and the right file. I discovered that my arraylists were still holding all the instances of the flat object. The solution I used was to just clear all my arraylists whenever the files being used are changed in the menu. This solved my problem.

```
public void opt0()
{
    // 0 - Set Dev0 environment
    System.out.printf("Dev0 Environment Selected\n");
    flatFileName = "Dev0_Flat.txt";
    meterFileName = "Dev0_Meter.txt";
    flats.clear();
    tenants.clear();
    bills.clear();
    tBills.clear();
}
```

Data Structure Choice:

My program is made up of five classes:

- Main
- Flat
- Tenant
- BC_Bill
- Tenant_Bill

The reason I chose to make each of these things was because they are nouns and not verbs and in the case of all but main I wanted to have the data stored in a place that was structured and could be easily looped through, compared, and accessed when needed. As noted above in the problems section, I had difficulty linking the flat and tenant classes for task 4.

When looking through my program you would also notice that all the lists are arraylists and not regular arrays, the reason I chose to go with arraylists was because of the variety of data being

put into them as well as the quantity of data wasn't enough to have an impact on the speed of the program. For example the flats could have any number of tenants from 1 to 99 so it was easy to just store them in an arraylist. Same goes for storing objects. 1 file might have 10 flats and the other might have 100. It just made it easy to have everything stored in arraylists. Also it seemed to be heavily hinted to use arraylists in the assignment outline/description.

Sort Routine Choice:

My program sorts the tenants using a bubble sort. The reason I chose to go with a bubble sort was because it was simple and easy for me to understand as well as a small amount of code. Furthermore, due to the limited dataset size the sort was going to be extremely fast whichever sort routine you chose to use therefore I chose an easy one which I understood well. If we reuse code from this assignment in the future, I am up for the challenge of coding a quicksort routine though. The only issue I had when coding this sort routine was that I was trying to compare strings and I needed to compare integers, so I had to view the meter numbers without the m at the beginning. Solution was to use the substring method. Sort routine code can be seen below.

```
public static void bubbleSort (ArrayList<Tenant> list) {
    int length = list.size();
    boolean swapped;
    do {
        swapped = false;
        for (int i = 1; i < length; i++) {
            int a = Integer.parseInt(((Tenant)list.get(i-1)).getmeterNo().substring(1));
            int b = Integer.parseInt(((Tenant)list.get(i)).getmeterNo().substring(1));
            if (a > b) {
                Tenant temp = list.get(i-1);
                list.set(i-1, list.get(i));
                list.set(i, temp);
                swapped = true;
            }
        }
        length--;
    } while (swapped);
}
```

Search Technique Choice:

My program uses a binary search and a sequential search as part of task 3. The reason I used these searches was because that was what was used in the example for the assignment. Interestingly I believe both the Dev files it is faster to use the sequential search but on the test and prod files the binary search becomes faster than sequential.

```java
public void sequentialSearch(String find) {
    for (int i = 0; i < tenants.size(); i++) {
        Tenant tenant = tenants.get(i);
        if (tenant.getmeterNo() == find) {
            break;
        }
    }
}

public void binarySearch(String find, int start, int end) {
    int intFind = Integer.parseInt(find.substring(1));
    while (start <= end) {
        int middle = start + (end - start) / 2;
        int middleCheck = Integer.parseInt(tenants.get(middle).getmeterNo().substring(1));
        if (middleCheck == intFind) {
            return;
        } else if (middleCheck > intFind) {
            end = middle - 1;
        } else {
            start = middle + 1;
        }
    }
}
```

Source Code Borrowed:

```
public static void readFile(String file) {
      try {
         File myObj = new File(file);
         Scanner myReader = new Scanner(myObj);
         while (myReader.hasNextLine()) {
         String data = myReader.nextLine();
         System.out.println("Line >> "+data);
         }
         myReader.close();
      } catch (FileNotFoundException e) {
         System.out.println("An error occurred.");
         e.printStackTrace();
      }
}
```

Above was some source code I borrowed which was for reading in files, however as I had the problem with empty lines in files, I ended up replacing this code with a buffered reader anyway so this was mostly not used in the final program.

```
@Override
public String toString() {
return "Student No: " + this.getStudentNo() + ", Student Name: " + this.getStudentName();
```

}

Secondly I was having the problem, when I wanted to check if objects were being stored in my arraylists so I tried printing them out and I just got the pointers to the location where they were stored in the computer's memory. So I looked up how I could print information which would be useful to me from each class so I could identify them. This was the code I found. It is from the link below.
https://stackoverflow.com/questions/13001427/printing-out-all-the-objects-in-array-list

What I learned:

Well I think this assignment, including watching tutorials and lectures, took me around 30+ hrs, so the important lesson which I followed after learning from previous units, get the work done well before the due date, which I achieved.

Furthermore, having never coded in java before this was my first major project using java, so that was a big learning experience and I learned lots of new things, I further developed my understanding of algorithms and data structures and I had fun.

Also, I have learned that programming is really enjoyable to me, the project may have taken 30+ hrs but to me it was really enjoyable and I looked forward to coding everytime I went to work on it.

Finally, this program thoroughly instilled the importance of testing to me, I probably spent a bit under 10 hrs coding and thinking about code and 10 hrs or more testing that code. Almost every time you make a change to your program a bug will occur and when you fix that you will find 2 more bugs. So it just reinforced my knowledge of the importance of testing to me.

Things to Change or Improve Next Time:

I think I would change the name of the tenant class to meter because this made it confusing occasionally when thinking about the program because I was trying to identify meters but they were stored in a class called tenant.

Also, the bubble sort algorithm isn't the fastest sort algorithm so if I had more time or it was more necessary I could have used an algorithm like quicksort to be even faster. Also I am sure that somehow the binary search could be made quicker if I had more time.

Also, a lot of my program relies on the user inputting the correct thing and so if I had more time I would use try and catch statements to make sure the user enters the right thing and knows what is wrong about what they are entering.

Testing Screenshots:

Task 1 Testing:

Test	Option	File	Notes
Task1a	Read Flats	Prod_flat.txt	
Task1b	Read Meters	Prod_Meter.txt	

Task 1a:

```
Select Option:
f
Reading flat file prod_Flat.txt
Number of Flats Read in is: 150
Number of Meters Read in is: 1101
Total sum (checksum) of all current flat readings is: 8.1717218E+07
Total sum (checksum) of all current flat readings is: 81,717,218
```

Task 1b:

```
Select Option:
m
Reading meter file prod_Meter.txt

Number of Meters Read in is: 1110
Total sum (checksum) of all current meter readings is: 5.97019713E+08
Total sum (checksum) of all current meter readings is: 597,019,713
```

Task 2 Testing:

Test	Option	File	Notes
Task2a	Compute BC Bill For one Flat (Task2)	Prod_flat.txt	
Task2b	Compute BC Bill For All Flats (Task2)	Prod_flat.txt	

Task 2a:

```
Select Option:
C
Compute bill for one Block of flats:
Enter Street Number: 10
Enter Street Name: Balmoral Drive
Showing Bill for: 10 Balmoral Drive
-----------------------------
Current  meter reading 956067  08/04/2022
Previous meter reading 937230  11/01/2022
Usage                   18837
Rate                0.205/kwh
Bill Usage                  $ 3861.58
```

Task 2b:

```
Select Option:
A
Compute bill for all Blocks of flats:
       Total for All Flats
-------------------------
Total             :     129,969.39
Records Processed:            150
```

Task 3 Testing:

Test	Option	File	Notes
Task3a	S	Prod_Meter.txt	
Task3b	P	Prod_Meter.txt	

Task 3a:

```
Select Option:
S
 Check[ 0] Lewis Pressman m163962 223579 Fri Apr 08 00:00:00 AEST 2022
 Check[ 9] Rose Wilton m163971 307056 Fri Apr 08 00:00:00 AEST 2022
  Last[ 1109] Stanley Rathbone m290279 993326 Thu Apr 07 00:00:00 AEST 2022
```

Task 3b:

```
Select Option:
P
 Check[ 0] Lewis Pressman m163962 223579 Fri Apr 08 00:00:00 AEST 2022
 Check[ 9] Rose Wilton m163971 307056 Fri Apr 08 00:00:00 AEST 2022
  Last[ 1109] Stanley Rathbone m290279 993326 Thu Apr 07 00:00:00 AEST 2022

Find Sequential: 32 milliseconds.
Find Binary    : 14 milliseconds.
```

Task 4 Testing:

Test	Option	File	Notes
Task4a	Compute Full Bill For One Flat (Task4)	test_flat.txt, test_Meter.txt	Beaconsfield Road,9
Task4b	Compute Full Bill For One Flat (Task4)	Prod_flat.txt, Prod_Meter.txt	The Causeway,12

Task 4a:

```
Select Option:
O
Compute bill for one Block of flats:
Enter Street Number: 9
Enter Street Name: Beaconsfield Road
Showing Bill for:  9 Beaconsfield Road
-----------------------------
Current  meter reading 334187  07/04/2022
Previous meter reading 325084  09/01/2022
Usage                    9103
Rate                0.205/kwh
Bill Usage                 $ 1866.11

Tenant                  Meter   Curr   Prev     Usage Pcnt%    Base     Adj    Total
-------------------------------------------------------------------------------
      Mr Ronald Doherty m163994 621996 620692   1304  14.57    $267.32  $4.60   $271.92
       Miss Tracy Denon m163991 769262 767730   1532  17.12    $314.06  $5.40   $319.46
  Mr Timothy Cartwright m164032 390961 389456   1505  16.82    $308.53  $5.31   $313.83
        Mr David Purvis m164012 940746 939469   1277  14.27    $261.78  $4.50   $266.29
      Miss Bella Mellor m163997 235907 234330   1577  17.62    $323.28  $5.56   $328.85
      Mr Gary Netherton m163968 803769 802015   1754  19.60    $359.57  $6.19   $365.76

Total Tenant bills (metered)     1834.54
Total Tenant bills Diff            31.57
Total Tenant bills Adjusted      1866.11
```

Task 4b:

```
Select Option:
0
Compute bill for one Block of flats:
Enter Street Number: 12
Enter Street Name: The Causeway
Showing Bill for: 12  The Causeway
----------------------------
Current  meter reading 917074  07/04/2022
Previous meter reading 902811  11/01/2022
Usage                   14263
Rate                0.205/kwh
Bill Usage                  $ 2923.92

Tenant                  Meter   Curr   Prev     Usage Pcnt%    Base     Adj    Total
-------------------------------------------------------------------------------
        Mrs Lisa Newbry m164904 786070 784712   1358  9.50     $278.39  $0.00   $278.39
         Mr Quincy East m164237 993574 991796   1778  12.44    $364.49  $0.00   $364.49
    Mr Dennis Pattishal m164221 715790 714651   1139  7.97     $233.49  $0.00   $233.49
    Mr Ricky Saddlemore m164951 483774 481979   1795  12.56    $367.97  $0.00   $367.97
       Miss Adele Dobbs m290258 769019 767905   1114  7.79     $228.37  $0.00   $228.37
      Mr Larry Mulgroon m290137 763259 761804   1455  10.18    $298.28  $0.00   $298.28
Miss Shirley Desborough m164902 553255 551864   1391  9.73     $285.16  $0.00   $285.16
      Mr Phillip Hagman m164034 685800 684600   1200  8.40     $246.00  $0.00   $246.00
        Mr Zak Talisman m164984 501013 499194   1819  12.73    $372.90  $0.00   $372.90
    Miss Lisa Dickinson m164041 880587 879342   1245  8.71     $255.23  $0.00   $255.23

Total Tenant bills (metered)     2930.27
Total Tenant bills Diff            -6.35
Total Tenant bills Adjusted      2930.27
```

Task 5 Testing:

Test	Option	File	Notes
Task5a	Compute Full Bill For All Flats (Task5)	test_flat.txt, test_Meter.txt	Full Screen shot all flats
Task5b	Compute Full Bill For All Flats (Task5)	Prod_flat.txt, Prod_Meter.txt	Just Screen shot the totals bit not all the mess

Task 5a:

```
0   Set prod environment
Select Option:
5
List Adjusted bill for all Blocks of flats

Flat Address                    BC Bill   Difference   DiffAdj   Tenant Bill
-------------------------------------------------------------------------------

11   Adelaide Street            347.47$       9.43$     9.43$       347.47$
7    Alma Street                519.06$      13.12$    13.12$       519.06$
8    Alma Street                984.00$      20.70$    20.70$       984.00$
11   Alma Street               1160.50$      31.16$    31.16$      1160.50$
10   Arundel Road              1991.17$      20.09$    20.09$      1991.17$
18   Arundel Road              1282.07$       0.00$    -3.08$      1285.15$
5    Ash Grove                  299.51$       7.38$     7.38$       299.51$
17   Ash Grove                 1354.44$      36.49$    36.49$      1354.44$
10   Balmoral Drive             312.01$       1.85$     1.85$       312.01$
11   Balmoral Drive            1153.13$       0.00$    -8.61$      1161.74$
12   Balmoral Drive             354.45$       8.61$     8.61$       354.45$
9    Beaconsfield Road         1866.11$      31.57$    31.57$      1866.11$
34   Beechwood Drive           1826.55$      33.42$    33.42$      1826.55$
6    Birch Avenue              1865.09$      29.93$    29.93$      1865.09$
12   Birch Grove                325.13$       4.31$     4.31$       325.13$
16   Blackthorn Close          1374.12$      26.86$    26.86$      1374.11$
20   Blackthorn Close           980.52$      22.35$    22.35$       980.52$
10   Bright Street              563.75$       0.00$    -1.64$       565.39$
5    Burnside                   580.56$       1.64$     1.64$       580.56$
7    Burnside                   920.24$      14.56$    14.56$       920.25$
12   Burnside                  1053.09$       1.23$     1.23$      1053.09$
6    Cambridge Street          1085.48$      12.10$    12.10$      1085.48$
10   Cambridge Street          1024.39$      13.33$    13.33$      1024.39$
47   Cambridge Street          1678.13$      14.76$    14.76$      1678.13$
4    Campbell Close             868.38$      20.50$    20.50$       868.38$
9    Campbell Close            1146.98$       4.92$     4.92$      1146.97$
10   Cavendish Road             648.42$       0.00$    -6.36$       654.77$
13   Cavendish Road             670.96$       0.00$    -4.31$       675.27$
14   Cavendish Road             237.60$       2.46$     2.46$       237.60$
-------------------------------------------------------------------------------
                             28,473.27$     382.74$   358.75$    28,497.26$
```

Task 5b:

10	Richmond Avenue	3901.35$	0.00$	-26.65$	3928.00$
4	Riverside	2820.80$	79.13$	79.13$	2820.80$
7	Russell Road	4902.99$	0.00$	-15.58$	4918.57$
13	Russell Street	279.62$	3.49$	3.49$	279.62$
8	Scott Close	676.91$	17.63$	17.63$	676.91$
10	Scott Close	679.37$	12.50$	12.50$	679.37$
12	Severn Road	1128.52$	21.52$	21.52$	1128.52$
20	Severn Road	2711.54$	45.51$	45.51$	2711.53$
78	Shakespeare Road	1265.05$	0.20$	0.20$	1265.05$
17	Spencer Road	1232.25$	30.34$	30.34$	1232.25$
9	Spencer Street	2967.79$	3.90$	3.90$	2967.79$
11	Spring Street	810.16$	0.00$	-1.02$	811.19$
15	Spring Street	561.29$	10.25$	10.25$	561.29$
201	St Andrew's Close	300.73$	5.94$	5.94$	300.73$
10	The Causeway	719.96$	19.68$	19.68$	719.96$
61	The Causeway	759.32$	3.49$	3.49$	759.32$
12	The Causeway	2923.92$	0.00$	-6.35$	2930.27$
6	The Oaks	1756.65$	0.00$	0.00$	1756.65$
18	The Oaks	1109.05$	1.43$	1.43$	1109.05$
6	The Street	874.73$	14.76$	14.76$	874.73$
14	Union Street	1386.21$	0.00$	-6.97$	1393.18$
10	Waterside	1525.61$	0.00$	-8.20$	1533.81$
12	Waterside	962.06$	1.84$	1.84$	962.06$
9	West Avenue	1233.28$	13.74$	13.74$	1233.28$
15	West Avenue	1344.60$	0.00$	-3.48$	1348.08$
11	West Drive	1283.92$	24.81$	24.81$	1283.92$
14	West Drive	373.72$	0.00$	-3.07$	376.79$
14	Willow Avenue	1235.13$	19.68$	19.68$	1235.13$
5	Willow Close	1654.76$	27.27$	27.27$	1654.76$
12	Wilson Road	556.17$	11.89$	11.89$	556.17$
92	Wilson Road	1424.34$	0.00$	-10.45$	1434.79$
14	Windmill Lane	275.93$	7.38$	7.38$	275.93$
10	Woodside	566.82$	0.00$	-4.72$	571.54$
16	Woodside	4047.93$	105.78$	105.78$	4047.93$
9	Woodside Road	551.25$	7.79$	7.79$	551.25$
16	Worcester Road	926.81$	14.97$	14.97$	926.81$
		334,969.38$	3,764.21$	3,398.49$	335,335.11$