RFID Documentation

Criterion A - Planning

Scenario

The client is Mr Pham, a teacher at Canberra Grammar School. Every week, he is involved in a Robotics Co-Curricular group, on Wednesday afternoons and Friday mornings. A roll is marked manually during each session, but this can be inaccurate due to human error, and time-consuming. Physical rolls can also be lost or misplaced. After consulting with him, I suggested that the roll could be done online, by making use of Student ID cards (with RFID) and an RFID reader.

As the Robotics Co-Curricular group has a Raspberry Pi available to them, we decided that the most appropriate application would be a command line program, made using Java. The role would be stored in a database built in-house and hosted on a website programmed through Javascript and JSON.

Rationale for solution

This would solve the client's problems by:

- Storing the API data online, so that data would not ever be lost, and is easily accessible
- Automatically displaying every student in Robotics, so that it is easier to track absent students and follow up on their reason for absence
- Easy setup and management (removal and additional) of student rolls
- Capacity for students to automatically sign in, by entering in a function and swiping their ID cards
- Allow to check who is present and who is absent at an instant

This work could be split amongst three teams, by setting up:

- A Command Line App, that could run on a Raspberry Pi and use the RFID reader to update rolls automatically when a Student Card is scanned
- A database that manages the students who should be in attendance (the 'ideal' roll)
- A website that stores and displays the attendance roll

Command Line App

- Programmed in Java (JDK 1.8)
- Can interact with the RFID Reader
- Can send HTTP requests to the API and retrieve JSON data
- Parses JSON data
- Open source
- Not reliant on other external software
- Displays the current students present
- Can delete or add students from roll

Use of Java

Java is a good choice for the API and Command Line Interface because:

- It has a strict declaration of variable types
- The application can be neatly packaged in a single JAR file
- It has a vast array of data structures supporting OOP methods
- Allows use of objects which directly support functionality and the OOP methodology
- It allows for expansion of program
- There are many resources available to troubleshoot and debug code

Success Criteria

- Rolls are managed and able to be saved to memory.
- The functions are easy and intuitive to use in order to mark and delete students.
- Can add and delete students using RFID values and by scanning student cards.
- Displays name and other details from RFID values.
    - Takes in the raw RFID number as input and uses the API's data to convert it into a name
- Program is intuitive to use
    - Reader, delete, and exit functions are easy to access
- Program can be expanded and built upon
    - Program doesn't use external frameworks and is entirely independent in its development
    - Program has the capability to be expanded in the future to be able to post to an API

Criterion B - Solution overview

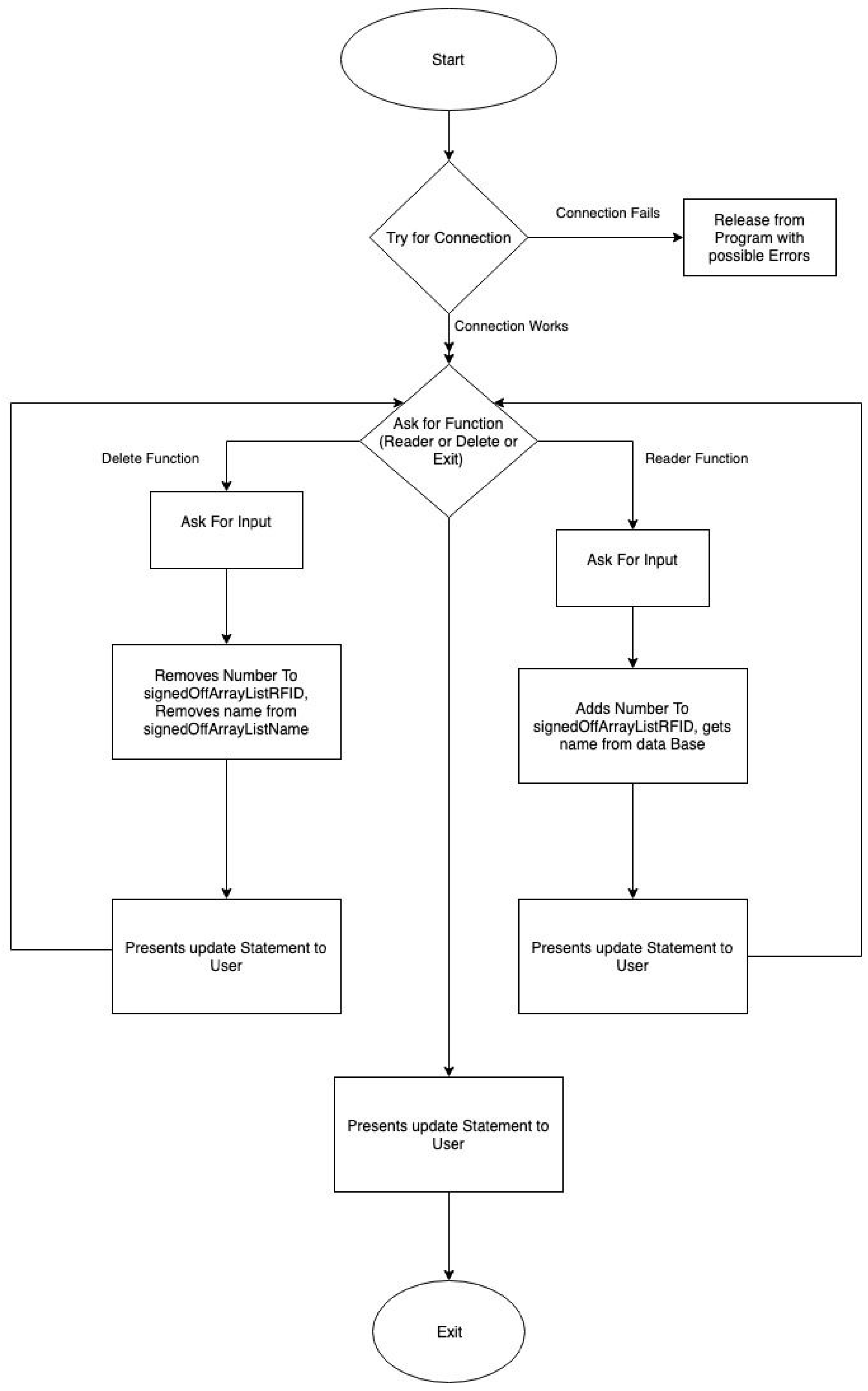
Record of tasks

[Record of Tasks Document]

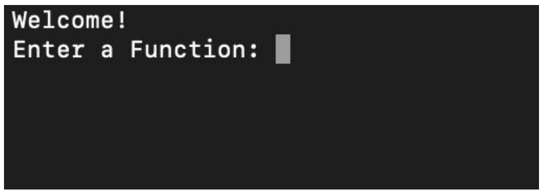
Test plan

Test Type	Nature Of Test	Input	Output
Rolls are managed and able to be saved to memory	There is an array that stores all students that have signed on.	“[RFID NUMBER]”	“[‘ACCORDING NAME’] has been signed in”
It’s intuitive to use and works without flaws	Teachers can use the functions “Reader”, “Delete”, or “Exit”, as well as “r”, “d”, “e” (while either uppercase or lowercase) to suit their preferences	“r” “d” “e”	“Enter a number: ” “Who do you want to sign out? ” “Exiting”
Able to interact with a directory and webpage	A try catch statement is used to connect to the web API, which contains all of the appropriate data	*No input, this occurs upon opening the program	“Welcome!” - If the program is successful in loading the API’s information, it will run the program as normal
Can add and delete students using RFID values and by scanning student cards	There is a function to add and remove students through scanning their students' cards and choosing the delete/add function.	“r” & “[RFID NUMBER]” “d” & “[RFID NUMBER]”	“[RFID NUMBER] has been signed in” “[RFID NUMBER] has signed out”
Displays name and other details from RFID values	There is a print function to display the signed on students stored in the signed on the array.	“r”	“[RFID NUMBER 1], [RFID NUMBER 2] … [RFID NUMBER n] are here”

Flowchart & Data Flow Diagram

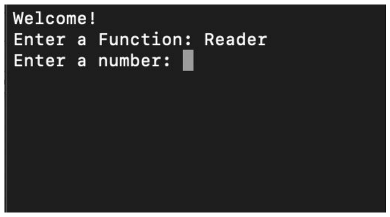

Start
Try for Connection
Connection Fails
Release from Program with possible Errors
Connection Works
Ask for Function (Reader or Delete or Exit)
Delete Function
Reader Function
Ask For Input
Removes Number To signedOffArrayListRFID, Removes name from signedOffArrayListName
Presents update Statement to User
Ask For Input
Adds Number To signedOffArrayListRFID, gets name from data Base
Presents update Statement to User
Presents update Statement to User
Exit

Graphical Visualisation



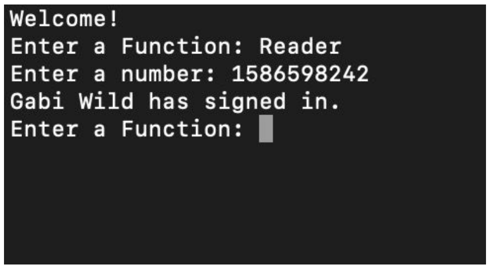
```
Welcome!
Enter a Function: 
```

- Upon opening the program, you are given a prompt to either type “Reader” or “r” to add new people from the API into the roll, “Delete” or “d” to remove people from the roll, and “Exit” or “e” to exit the program.



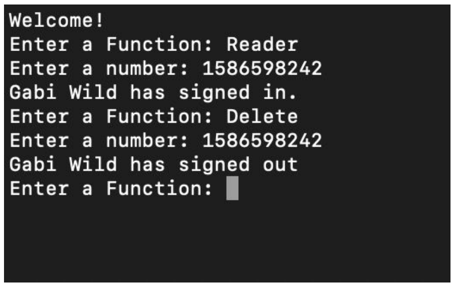
```
Welcome!
Enter a Function: Reader
Enter a number: 
```

- After typing “Reader” or “r”, you are prompted to enter a number. Using the RFID card reader, a CGS card will be scanned and its RFID value will be taken to be converted into a name on the roll.



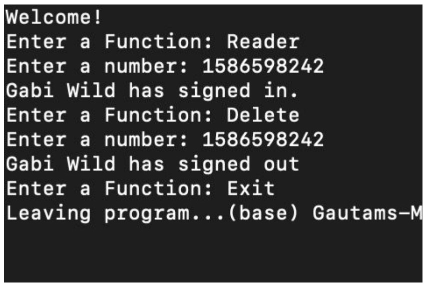
```
Welcome!
Enter a Function: Reader
Enter a number: 1586598242
Gabi Wild has signed in.
Enter a Function: 
```

- Once an RFID card is tapped to the RFID number, the program gets the according name from the API, held externally, and puts it into the present array. After this, the user is then prompted to choose another function, as mentioned above.



```
Welcome!
Enter a Function: Reader
Enter a number: 1586598242
Gabi Wild has signed in.
Enter a Function: Delete
Enter a number: 1586598242
Gabi Wild has signed out
Enter a Function: 
```

- Similarly, after typing “Delete” or “d”, you are prompted to enter a number. Using the RFID card reader, a CGS card will be scanned and its RFID value will be used to remove the respective person from the roll



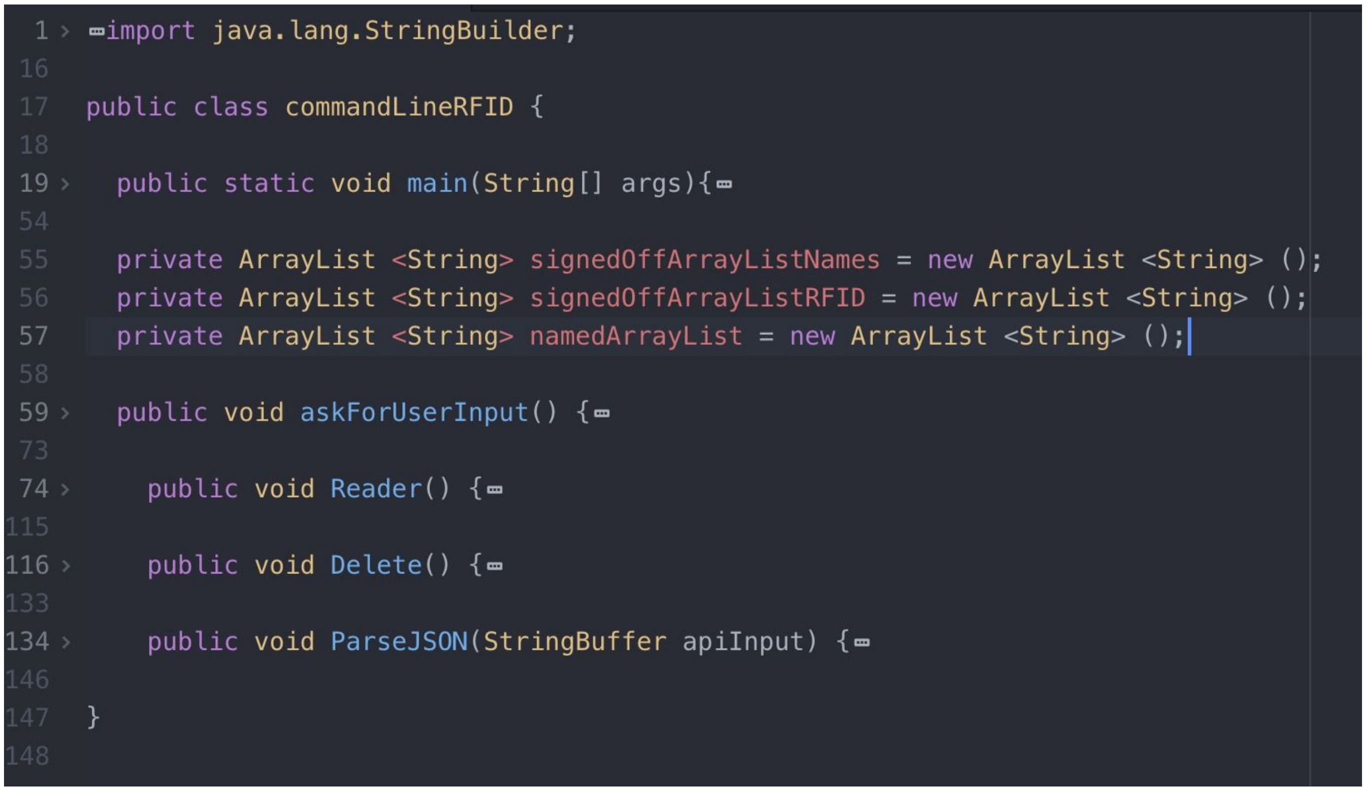
```
Welcome!
Enter a Function: Reader
Enter a number: 1586598242
Gabi Wild has signed in.
Enter a Function: Delete
Enter a number: 1586598242
Gabi Wild has signed out
Enter a Function: Exit
Leaving program...(base) Gautams-M
```

- After typing “Exit” or “e”, the system states that it is leaving the program, signalling the completion of its runtime.

Criterion C - Development

Screenshots of complex code

General Layout of Program



```
import java.lang.StringBuilder;

public class commandLineRFID {

  public static void main(String[] args){

  private ArrayList <String> signedOffArrayListNames = new ArrayList <String> ();
  private ArrayList <String> signedOffArrayListRFID = new ArrayList <String> ();
  private ArrayList <String> namedArrayList = new ArrayList <String> ();

  public void askForUserInput() {

    public void Reader() {

    public void Delete() {

    public void ParseJSON(StringBuffer apiInput) {

}
```

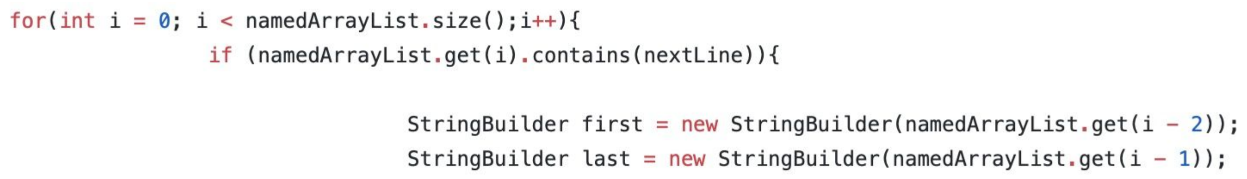
Techniques used

```
try {
        URL url = new URL("https://ibcsamazonec2.tk/students");
        HttpURLConnection con = (HttpURLConnection) url.openConnection();

        con.setRequestMethod("GET");
        Map<String, String> parameters = new HashMap<>();
        parameters.put("param1", "val");

        con.setDoInput(true);

        con.setRequestProperty("Content-Type", "application/json");
        String contentType = con.getHeaderField("Content-Type");

        BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
        String inputLine;
        StringBuffer content = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
content.append(inputLine);
        }
        program.ParseJSON(content);
        in.close();

        con.disconnect();
} catch (Exception e){
        System.out.print("Connection is not working");
        System.out.print(e);
}
```

- Try-catch statements
    - In this instance, a try-catch statement is used to attempt to connect to the appropriate web server and locate data from the API. If this data is successfully located, then the program will continue with it's running, and present the user with the option to run functions to sign in/out students. If the appropriate data is not found on the website, or the program cannot connect to the server, then the system will catch that, and proceed to inform the user that there has been an error and print the reason the error occurred.

```
private ArrayList <String> signedOffArrayListNames = new ArrayList <String> ();
private ArrayList <String> signedOffArrayListRFID = new ArrayList <String> ();
```

- ArrayLists
    - In this instance, three ArrayLists are declared which are used to programatically add new values into who is signed off (both their names and RFID values). As the program receives data from the API that could change constantly, it is important that the arrays can change dynamically according to the situation (the main application of this would be an increase in students in the API; if the array were to be of a fixed length, this wouldn't accomodate for a dynamically changing set of students).



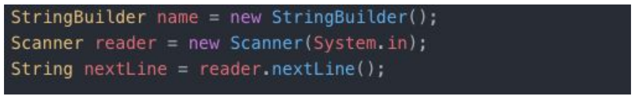
```
for(int i = 0; i < namedArrayList.size();i++){
                if (namedArrayList.get(i).contains(nextLine)){

                                StringBuilder first = new StringBuilder(namedArrayList.get(i - 2));
                                StringBuilder last = new StringBuilder(namedArrayList.get(i - 1));
```

- StringBuilders
    - In this instance, in order to parse the JSON data without using an external framework/software package, a StringBuilder is used to build the first and last name strings to put into the signed off array. As the information from the API is being parsed (through a series of loops), StringBuilders are employed to accumulate the appropriate characters to build into a single string and inputted into the signedOffArrayList.

Explanation of Key Algorithms

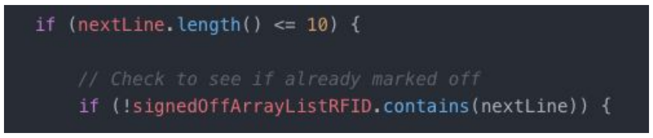
Reader Function

The reader function initially creates several variables which include: A StringBuilder, for the name, a scanner, and a string output.



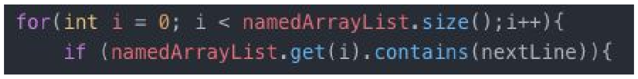
```
StringBuilder name = new StringBuilder();
Scanner reader = new Scanner(System.in);
String nextLine = reader.nextLine();
```

The user is then asked to put in an RFID number. Once the number is check to be of suitable length (RFID tags will be 10 characters long), the number is checked against the current list of marked off students.



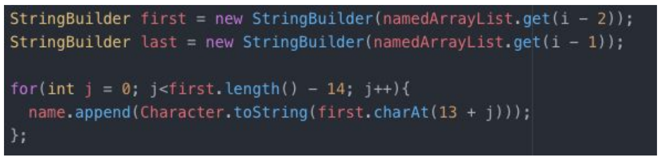
```
  if (nextLine.length() <= 10) {

      // Check to see if already marked off
      if (!signedOffArrayListRFID.contains(nextLine)) {
```

If the number is not in the system, the program finds the correlating object in the database. A loop is then initiated. The loop first checks the namedArray list.



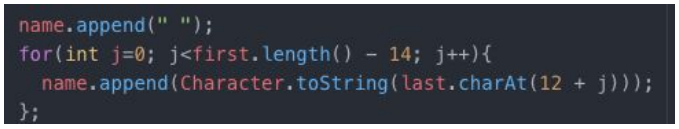
```
for(int i = 0; i < namedArrayList.size();i++){
    if (namedArrayList.get(i).contains(nextLine)){
```

The loop the initiates two string builders one for the last name and one for the first name. A loop is started, adding the characters to the new first name variable. As the original name from the data based = “firstname”:”NAME”, the loop runs from the first letter of the name. Hence the addition of 13 to the original j.



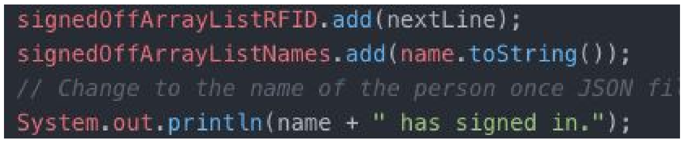
```
StringBuilder first = new StringBuilder(namedArrayList.get(i - 2));
StringBuilder last = new StringBuilder(namedArrayList.get(i - 1));

for(int j = 0; j<first.length() - 14; j++){
  name.append(Character.toString(first.charAt(13 + j)));
};
```

This is repeated with the last name.



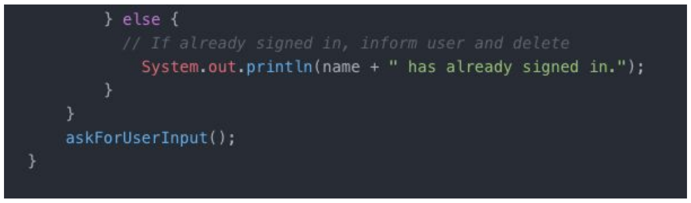
```
name.append(" ");
for(int j=0; j<first.length() - 14; j++){
  name.append(Character.toString(last.charAt(12 + j)));
};
```

The name and RFID value are added to global arrays to be used in other parts of the program. The user is updated on the progress.



```
signedOffArrayListRFID.add(nextLine);
signedOffArrayListNames.add(name.toString());
// Change to the name of the person once JSON fi
System.out.println(name + " has signed in.");
```

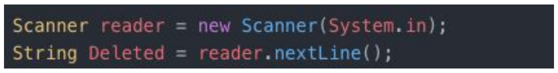
If the name is already in the local database the code is exited and the user is once again asked for input.



```
          } else {
            // If already signed in, inform user and delete
              System.out.println(name + " has already signed in.");
          }
      }
      askForUserInput();
  }
```

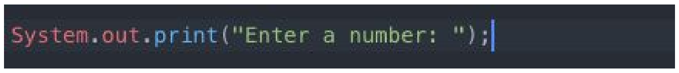
Delete Function

The function starts with the declaration of a scanner and a string of the scanners output.



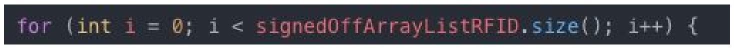
```
Scanner reader = new Scanner(System.in);
String Deleted = reader.nextLine();
```

The user is asked for an RFID number



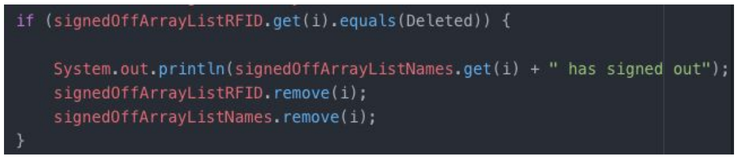
```
System.out.print("Enter a number: ");
```

A loop is initiated to search through the list of current students who have marked off.



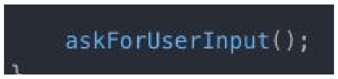
```
for (int i = 0; i < signedOffArrayListRFID.size(); i++) {
```

Once the program fines the number in the list which the user has entered, the program removes the number from the list and removes the name from the names list. It also informs the user of the progress.



```
if (signedOffArrayListRFID.get(i).equals(Deleted)) {

    System.out.println(signedOffArrayListNames.get(i) + " has signed out");
    signedOffArrayListRFID.remove(i);
    signedOffArrayListNames.remove(i);
}
```

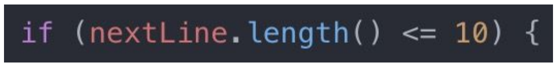
The loop ends and the askForUserInput function is called.



```
    askForUserInput();
```

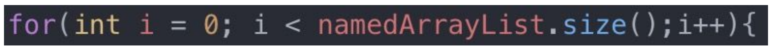
Parsing Function

Initially, an if statement is used to determine whether the amount of characters in the inputted value is an RFID value (which have 10 digit values). If it is, then it proceeds with the parsing. If it isn't then the if statement leaves the parsing algorithm.



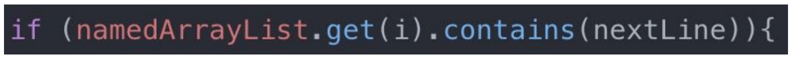
```
if (nextLine.length() <= 10) {
```

To start the parsing of the API, a for loop is created which parses through the characters of the API's received raw JSON data.



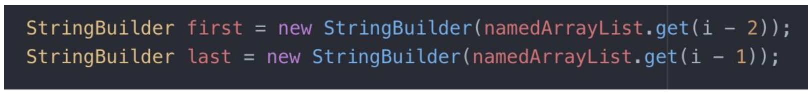
```
for(int i = 0; i < namedArrayList.size();i++){
```

An if statement is used to find out whether the raw input data contains a character that is contained in the raw data that is in the API. If it is, the parsing algorithm continues. If not, then the parsing algorithm is stopped as there is no data associated with the RFID number.



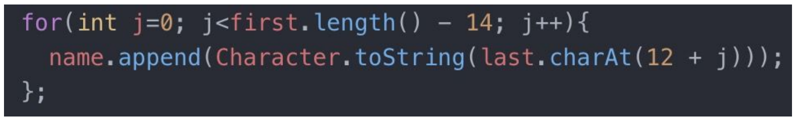
```
if (namedArrayList.get(i).contains(nextLine)){
```

To parse the API, two StringBuilders are defined, with the names "first" and "last". These are used when adding characters from the data, and ultimately used when displaying the people that are marked off.

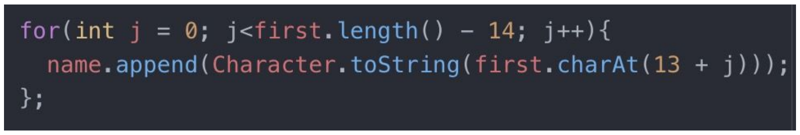


```
StringBuilder first = new StringBuilder(namedArrayList.get(i - 2));
StringBuilder last = new StringBuilder(namedArrayList.get(i - 1));
```

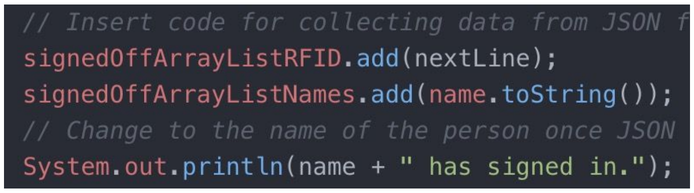
A for loop is employed to loop through the characters in the raw input, to see which characters are in the first name of the person; located with their according RFID number. Each character is added to the aforementioned StringBuilder, to be added to the signedOffArray


```java
for(int j=0; j<first.length() - 14; j++){
  name.append(Character.toString(last.charAt(12 + j)));
};
```

Another for loop is used for the same intention above to find the last names of the person located using their RFID number. This is going to be added back to the signedOffArray with the first name StringBuilder.

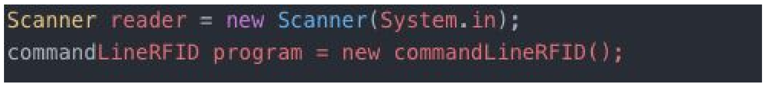

```java
for(int j = 0; j<first.length() - 14; j++){
  name.append(Character.toString(first.charAt(13 + j)));
};
```

Once the first and last names have been collated, both of them are added to the signOffArray and to the signedOffArratNames arrays. A print statement is also used in the terminal to show that the person has been marked off.

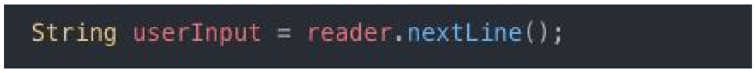

```java
// Insert code for collecting data from JSON f
signedOffArrayListRFID.add(nextLine);
signedOffArrayListNames.add(name.toString());
// Change to the name of the person once JSON
System.out.println(name + " has signed in.");
```

Ask for User Input Function

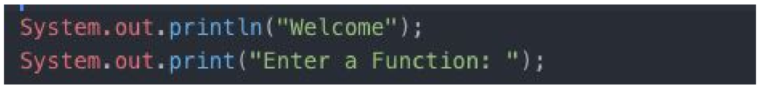
A scanner and a program are declared for use in the later.


```java
Scanner reader = new Scanner(System.in);
commandLineRFID program = new commandLineRFID();
```

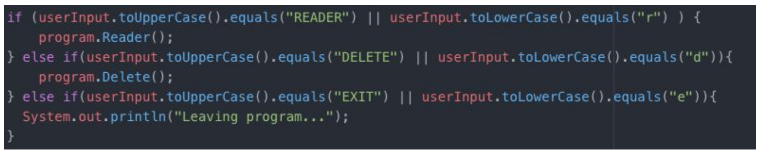
A string is declared, taking the input from the scanner


```java
 String userInput = reader.nextLine();
```

The user is informed of the progress


```java
System.out.println("Welcome");
System.out.print("Enter a Function: ");
```

An if statement is run, in the statement the program finds if the user wants to run a certain function via their input. If so, the function is run.


```
if (userInput.toUpperCase().equals("READER") || userInput.toLowerCase().equals("r") ) {
    program.Reader();
} else if(userInput.toUpperCase().equals("DELETE") || userInput.toLowerCase().equals("d")){
    program.Delete();
} else if(userInput.toUpperCase().equals("EXIT") || userInput.toLowerCase().equals("e")){
  System.out.println("Leaving program...");
}
```

Criterion D

Video

Video Link

Film edited by William Peterswald, filmed by Gautam Mishra and William Peterswald and acted by William Peterswald, Gautam Mishra and Ziling Ouyang.

Criterion E

Evaluation

Criteria	**Status**
Rolls are managed and able to be saved to memory.	Rolls are able to be saved to an array and managed.
The functions are easy and intuitive to use in order to mark and delete students.	Users are able to use simple command line functions to mark off and delete students.
Can add and delete students using RFID values and by scanning student cards.	There is added functionality of being able to use an RFID reader to mark off. Students can then be signed off by entering their names after running the delete function.
Displays name and other details from RFID values. - Takes in the raw RFID number as input and uses the API's data to convert it into a name	By inserting the function, users can display student details from RFID values. The program gets information from the API and converts the RFID values to an appropriate name.
Program is intuitive to use - Reader, delete, and exit functions are easy to access	There are functions that can easily be used to manipulate the application to perform the adjacent utilities, and these functions can be accessed using abbreviated versions of their names for ease and quickness of use.

Program can be expanded and built upon - Program doesn't use external frameworks and is entirely independent in its development ○ Program has the capability to be expanded in the future to be able to post to an API	There is no dependence on Maven / other external software.

Mr Pham said that the program is useful and fulfills the success criteria above as well as the needs for his Robotics group, but would be nice if the program would use JSON. This could be implemented in the future, and the current program allows for this expandability to occur.

Improvements and future opportunities for development:

1. Implement the option of using org.JSON to parse JSON and to send JSON data back to the API to make the system be able to store all data externally
2. To remove some of the hardcoded elements and make the code a bit more dynamic in its use and future development.