List of Simple Techniques used

Encapsulation
Data Hiding
Polymorphism
Exception handling
Association

Technique:

Bufferedreader, File reader and String class Methods to Import data from CSV files

Success Criteria :

1. The client must be able to give a spreadsheet/csv files with duties
2. The client must be able to give another spreadsheet/csv files with teachers
3. The client must be able to give another spreadsheet/csv files with a list of Admins and Subject area Leaders

Source:

Lee, A., 2020. *Java: Read A CSV File Into An Array*. [video] Available at: <https://www.youtube.com/watch?v=-Aud0cDh-J8>.

Justification

My client has a lot of data that she needs to be imported. These are stored in excel file formats which can quickly and easily be converted to csv file formats. A Buffered reader object allows my code to read the data in the file which the file object is able to create. From this, the methods in the string class

Explanation:

For every different file that my client needs to upload, there will be a different function. In each of those functions, a new BufferedReader object needs to be created. Before that, the user is asked to input the file path which is stored in a string variable called path. After this, a new file object is created with the path which is then used to create the buffered reader object. The buffered reader object then reads every line which is stored in a String variable called Line. Using methods available in the String class, the line was split by “,” and then stored into an array from which data was extracted

Example:

```
public static boolean importPeriodSix() { //import teachers who have period six
System.out.println("Enter file path for list of teachers with period 6"); //output message
String path = in.next(); //read the path

    try {
        BufferedReader br = new BufferedReader(new FileReader(path)); //create a new bufferedReader
        br.readLine(); //read the first Line as it is not important
        while(br.ready()) { //while the BufferedReader is ready
            line = br.readLine(); //read the line
            String values[] = line.split(","); //split it and store in array
```

User was asked to input file path which was stored

New fileReader and BufferedReader was created

Data was read and stored in Line (Line is a global variable)

Data was then stored in a String [] using the method "split" in the String class

Technique : Use of PrintWriter and Stringbuilder to Create a CSV File

Source:

Jawad M, J., 2018. *Create CSV File Using Java*. [video] Available at: <https://www.youtube.com/watch?v=xlqKNTln2YA>.

Success Criteria:

4. The system should be able to produce a csv files with a list of duties
5. The system should be able to accurately allocate each teacher to each duty

14. The client should be able to tell the program where to store the final file and the name of the final file.

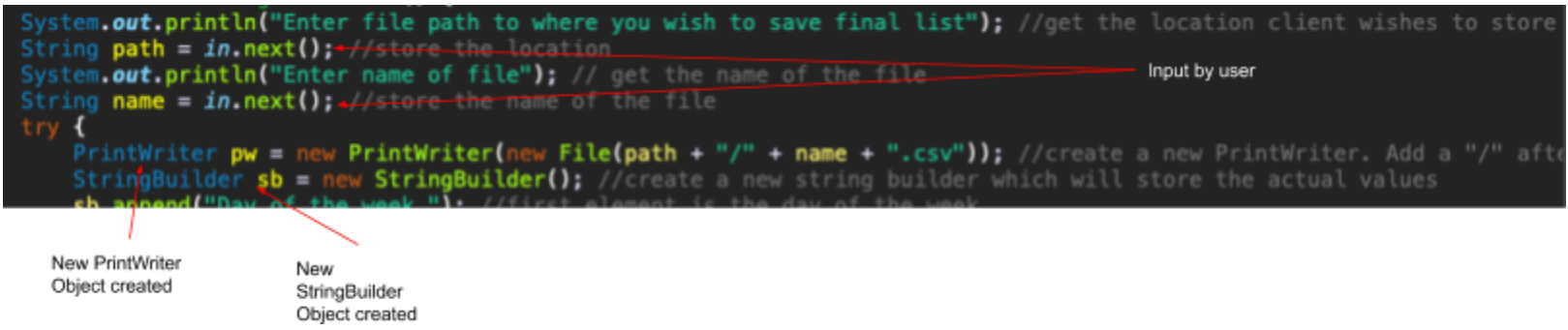
Justification:

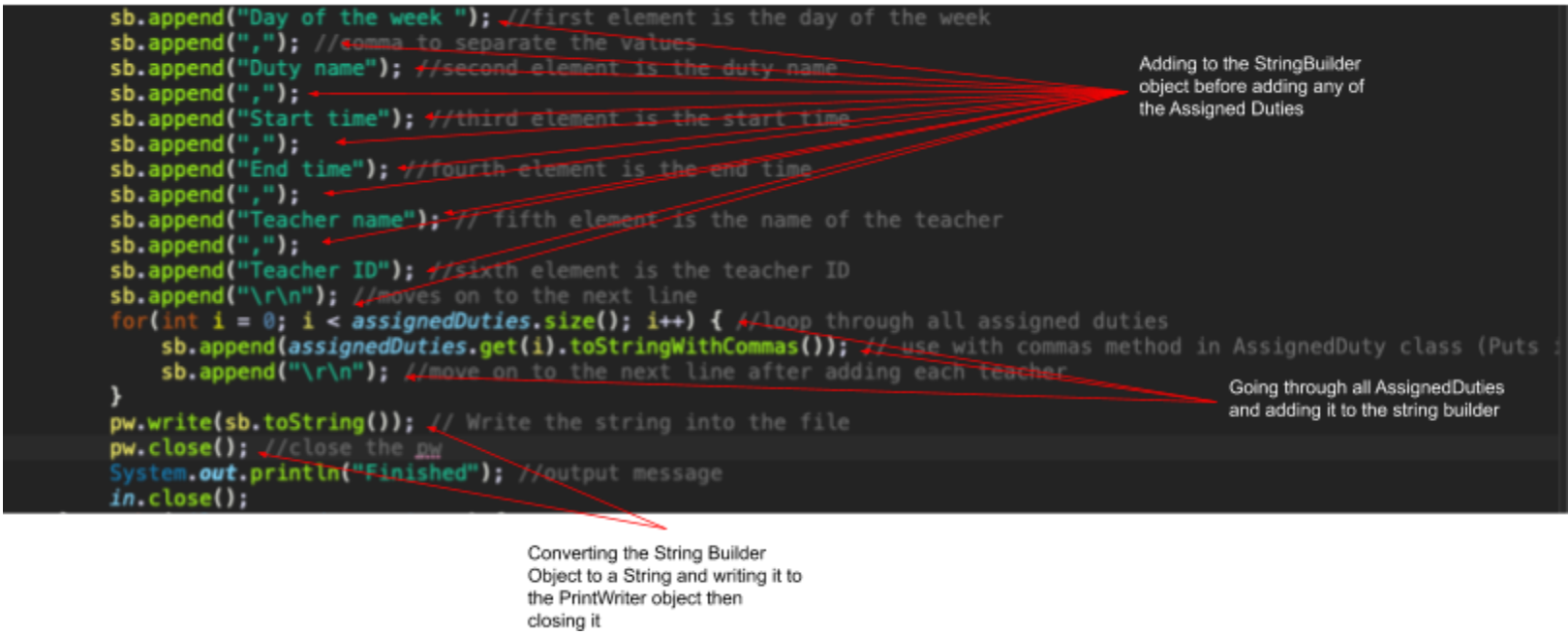
After my program has done all the processing that it is required to do, which is to allocate all the duties to all of the teachers, it needs to output all of this information for which my client can see. In addition, my client needs to send the finished list of duties with teachers to my school's IT team which will put the data on to the school system from where all teachers can access it. Therefore, my program is outputting a CSV file which includes all of the duties that have been assigned to which teacher and which have not.

Explanation

First, the user was prompted for a file path where she wished the final list to be stored which was stored into a variable. Then the user was asked for the name of the file. This information was then used to create a Printwriter object. After that, a String Builder object was created and using the append method, all the Assigned Duties were added to the string. After each teacher, a "/r/n" was added indicating a new line. All of the information for the assigned duties were also separated by a ",". Once all the assigned duties were added, the string builder was converted to a string using the toString method (in the StringBuilder class) which was then put into the printwriter using the write method in the PrintWriter class. Once that was done, the print writer was closed causing a CSV file to be created and stored in the client's desired location.

Example:





Technique : Use of ArrayLists to store imported data in the system and Collections class to sort the ArrayLists

Success Criteria

1. The client must be able to give a spreadsheet/csv files with duties
2. The client must be able to give another spreadsheet/csv files with teachers
3. The client must be able to give another spreadsheet/csv files with a list of Admins and Subject area Leaders
4. The system should be able to produce a csv files with a list of duties
5. The system should be able to accurately allocate most of the duties to a teacher.

Source : Java Illuminated,

Anderson., 2018. *Java Illuminated*. Burlington: Jones & Bartlett Learning.
Anderson, G., 2016. *Sort Arraylist Of Objects*. [video] Available at: <https://www.youtube.com/watch?v=wzWFQTLn8hI>.

Justification

Duties in my school are split between the Elementary, Middle and High school. This means that there is not a set number of duties done by the high school every year. In addition, the number of teachers in the school also changes year on year . Therefore through the use of ArrayLists which are dynamic data structures, this is avoided as they can grow and shrink. The sort method in the collections class was used to sort the duties by the number of teachers that can do the duty such that those duties with the fewest possible teachers are allocated first and those with the most area allocated last.

Explanation

In the top_Level class which acts as my main class, I have three main global ArrayLists called duties teachers, assignedDuties. This is such that when my client is importing the teachers. It extracts the information and adds it to the global arrayLists. After all of it is imported, my code looks at each duty and tries to assign it teachers based on lessons. Each teacher that can be assigned to the duty is stored in another ArrayList native to the duty. After this is done, the duties are sorted by the size of that ArrayList. The ArrayList is then also used for the assigning of the teachers to the duties.

Example

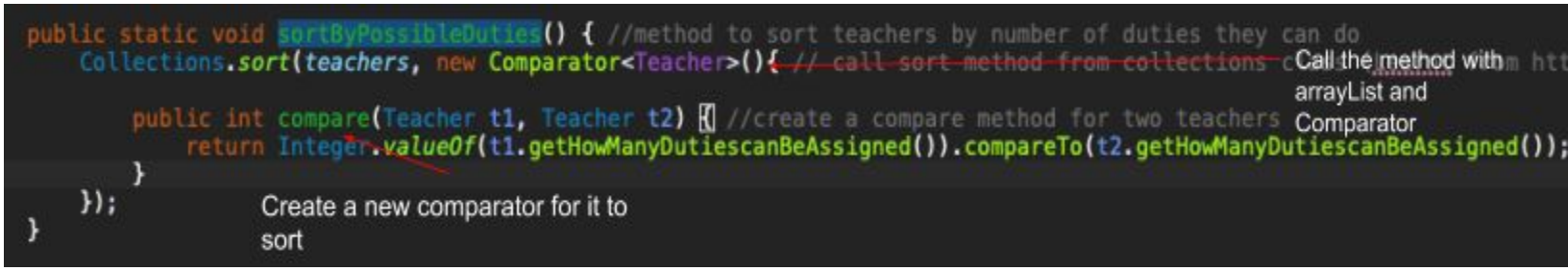
```
public static ArrayList<Teacher> teachers = new ArrayList<Teacher>(); // List of teachers
public static ArrayList<Duty> duties = new ArrayList<Duty>(); // list of duties
public static ArrayList<AssignedDuty> assignedDuties = new ArrayList<AssignedDuty>(); // list of
public static ArrayList<Integer> adminIds = new ArrayList<Integer>(); //list of adminIds
public static ArrayList<Subject> SubjectsWithDays = new ArrayList<Subject>();
```

Global ArrayLists I used in the Top_Level class

```
try {
    BufferedReader br = new BufferedReader(new FileReader(path)); //create a new buffered reader which reads the path
    int i = 0; //counter variable
    br.readLine(); //going through first line because it doesn't matter (days of the week)
    while (br.ready()) {//check if the the bufferedreader is ready
        line = br.readLine(); // read the next line
        String [] values = new String[34]; //initialize a values array with 33 elements (Max number of elements that a row can have (disc
        String [] read = line.split(","); //store the values of line into a new array called read;
        for (int b = 0; b < values.length; b++) { //loop through the whole of the values array
            if (b < read.length) { //check if the value of b is less than the length of the read array
                values[b] = read[b]; // if so add the value of read[b] into the values[b]
            } else {
                values[b] = ""; //otherwise set the values of value[b] to be empty
            }
        }
        if(isValidID(values[0]) == false) { //check if it is a valid ID.
            System.out.println("Error. ID " + values[0] + " of teacher " + values[1] + " is invalid. Please fix and re-enter"); //output
            return false; //return false as import is not successful.
        }
        int Id = Integer.parseInt(values[0]); //first element is ID
        if(isValidName(values[1]) == false) { //check if the name is valid
            System.out.println("Error. Name " + values[1] + " of teacher with ID " + values[0] + " is invalid. Please fix and re-enter");
            return false;
        }
        String name = values[1];// second element is Name

        int lessonsPerWeek = Integer.parseInt(values[2]); //third element is the number of lessons the teacher has
        boolean homebase = setLesson(values[3]); //fourth element is the teacher's homebase
        //Setting all the temp lessons to false to avoid NullPointer exception
        Lesson tempLessons [] = {allFalse, allFalse, allFalse, allFalse, allFalse};

        for(int j = 0; j < 5; j++) { //going through all the days in the week
            boolean one = setLesson(values[(j*6) + 4]); //get lesson one
            boolean two = setLesson(values[(j*6) + 5]); //get lesson two
            boolean three = setLesson(values[(j*6) + 6]); //get lesson three
            boolean four = setLesson(values[(j*6) + 7]); //get lesson four
            boolean lunch = setLesson(values[(j*6) + 8]); //get lunch lesson
            boolean five = setLesson(values[(j*6) + 9]); // get lesson 5
            Lesson temp = new Lesson(one, two, three, four, lunch, five, false); //create a new lesson (Lesson 6 is false for now)
            tempLessons[j] = temp; //set the day to be this
            /* the above is a calculation that will help me get each lesson on each day of the week. There are 6 lessons (Omitting lesson
             * that occur in each day. All of them will be stored in values. The calculation will help me get a specific lesson on one da
             * 3 on thursday will be the teacher's 21st lesson. In the values array that will be stored in index no 24. Using this calcul
             * index no 24 (thursday means j = 3 so 3*6 + 6 = 24)
             */
        }

        // set the teacher's lessons
        br.readLine(); //read the next line because the line does not have important information either (Teacher Name), and important fo
        Teacher temp = new Teacher(name, Id, lessonsPerWeek, homebase, tempLessons,theTimes); //create a new teacher with the attributes
        teachers.add(temp);//add the teacher into the arraylist

        i++;
    }
```

Above is extracting the information from the file

Creating and Adding the teacher into the ArrayList


public static void sortByPossibleDuties() { //method to sort teachers by number of duties they can do
Collections.sort(teachers, new Comparator<Teacher>(){ // call sort method from collections
public int compare(Teacher t1, Teacher t2) { //create a compare method for two teachers
return Integer.valueOf(t1.getHowManyDutiescanBeAssigned()).compareTo(t2.getHowManyDutiescanBeAssigned());
}
});
}
Call the method with arrayList and Comparator
Create a new comparator for it to sort

Technique : For/While loops and If Statements for validation and creation of main algorithm.

Success Criteria

5. The system should be able to accurately allocate most of the duties to a teacher.
6.The system should not assign duties to teachers who are have a subject area meeting during that time
7. The system should not assign more than one duty to teachers who are also admins
8. The system should not assign more than one duty to teachers who are also head of departments
9. The system should not assign more than 4 duties to any teacher
10. The system should not assign duties to teachers who work more than 21 hours
11. The system should not assign duties to teachers who have 4 or 5 lessons on the day
12. The system should be able to prevent the entering of invalid data
13. The system should be able to tell the client a mistake and occurred and where it has occurred (If it occurred in the csv file)

Source :

Anderson., 2018. *Java Illuminated*. Burlington: Jones & Bartlett Learning.

Justification.

Through the use of for loops if statements and while loops, the system was able to correctly allocate the duties. It was able to check whether or not a teacher had a particular reason as to why he/she could not do the duty and hence made sure was not allocated the duty. Another purpose of the if statements was to validate the data and output error messages if there were any problems entering the data. The loops allowed the system to go through each duty and go through each teacher allowing the system to check whether or not this particular teacher could do the duty. They also allowed for searching for teachers/duties by ID to uniquely locate each specific duty/ teacher and also allowed for methods to be called again in case an error occurred. Each of the search methods were Linear as teacher IDs were not in ascending orders and the duties needed to be sorted (before assigning) causing a binary search method to not work (As they would not be in order of ascending ID after sorting).

Explanation

The techniques were used in various different methods throughout the top_level class. For if statements, they were primarily used in the assignPossibleTeacher and assignTheTeacher methods which were the methods that were used to first assign the possible teachers to duties then sorted by fewest possible teachers and then actually assigning each duty with the teacher. The for loops were also primarily used in both of these as they allowed the algorithms to go through each duty and each teacher. The while loops were used to search for a teacher by ID as they would move on to the next teacher (In the global arrayList) if the ID trying to be found did not match the teacher's ID. The import method also returned a boolean. They would return

true if the import was successful and false if it was not with an error message as to why not. As such, the while loops were used here to call the method again if it returned false.

Example

```
for (int i = 0; i < duties.size(); i++) { // going through all duties
    Duty assigned = duties.get(i); // accessing duty
    Time dutyStartTime = assigned.getStartTime();// get the start time of the duty
    ArrayList<Teacher> possibleTeachers = new ArrayList<Teacher>(); //create a new array used to find probable teachers for the duty
    for (int j = 0; j < teachers.size(); j++) { // going through all teachers
        boolean canAssign = true; //Assume teacher can be assigned duty
        Teacher teacher = teachers.get(j); // accessing teacher
        if(teacher.getDutiesToBeAssigned() == 0) { //check if the teacher cannot do any duties
            canAssign = false; //cannot assign duty because teacher cannot have any duties
        }
        if (dutyStartTime.getHours() == 8) { // Check if the duty begins  before school. So begins before 9:00
            Lesson[] teacherLessons = teacher.getLessons();
            int index = searchForDay(assigned.getDayOfTheWeek());
            if(teacher.isHomebase() == true || teacherLessons[index].one == true) { //Check if teacher has a homebase or if they have
                canAssign = false; //Set can assign to false as the teacher cannot be assigned the duty
            }
```

Loops to go through all teachers and duties

Checking for conditions such that teacher cannot be assigned this duty.

Setting flag variable to false and not assigning duty to teacher

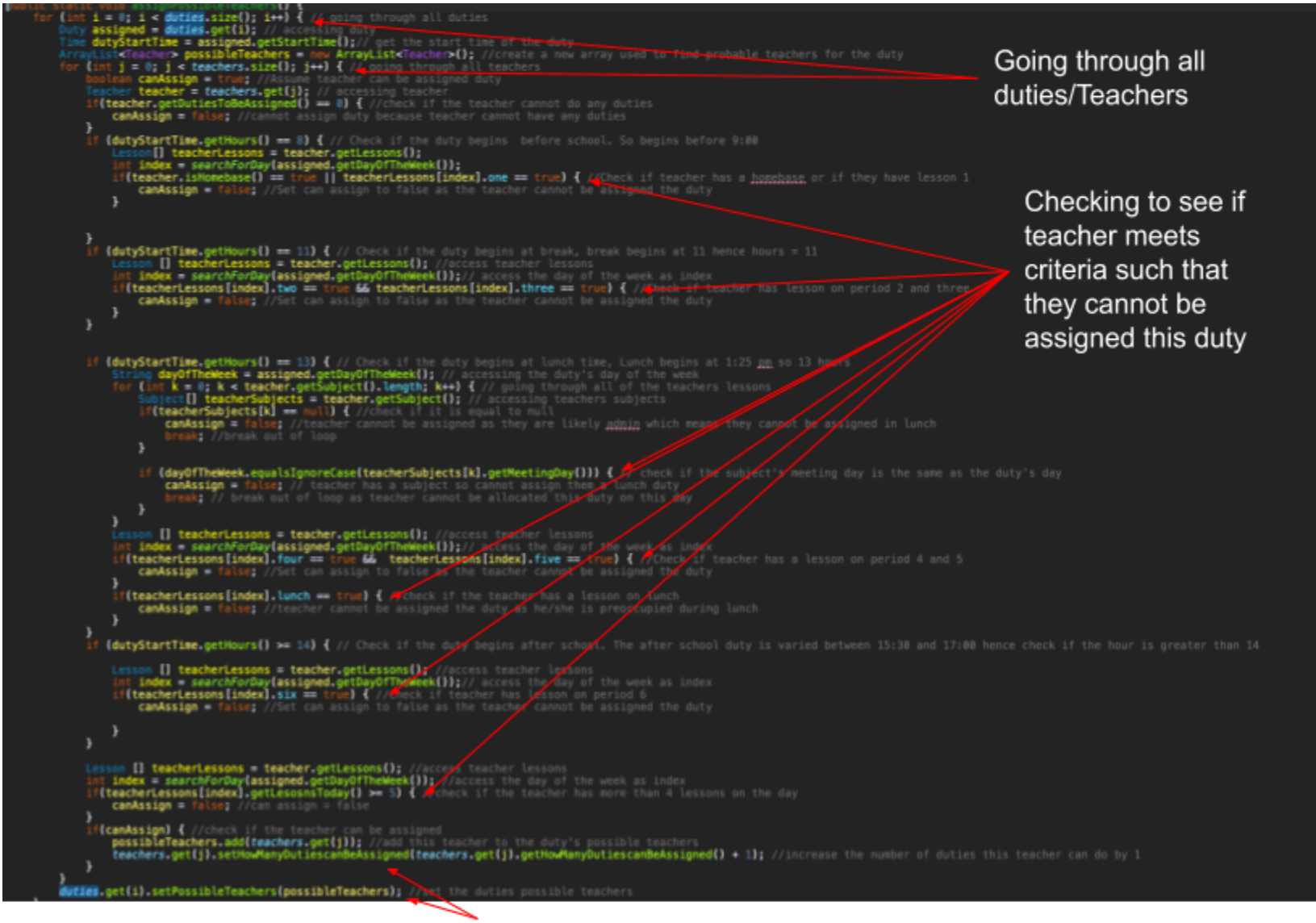
```
public static boolean isValidID(String id) { //validation for ID
    //Presence Check
    if (id == null) { //check if id is null
        return false; //Invalid ID
        }
    id = id.trim(); //remove unnecessary spaces
    //Length check
    if(id.length() != 5) { //check if length of ID is not 5
        return false; //Invalid ID
    }
```

Checking for invalid Types and returning false

```
public static int findTeacherByID(int id) { //method to find teacher by ID. I do not need to worry about validation as
    int i = 0; //counter variable
    while(teachers.get(i).getId() != id) { //while the teacher's ID does not equal the id we
        i++; //increase I
    }
    return i; //return i as teacher has been found
}
```

While the condition is not met

Increase i and move onto the next teacher



Going through all duties/Teachers

Checking to see if teacher meets criteria such that they cannot be assigned this duty

Adding teacher to the possible teachers arrayList. Go through all teachers and add all possible teachers to duty list