```
ValueRange possibleFirstName = service.spreadsheets().values().get(spreadsheetId, "Goals!B3:B").execute();
List<List<Object>> firstNames = possibleFirstName.getValues();
Object firstName = firstNames.get(firstNames.size() - 1);
String userFirstName = firstName.toString().substring(1, firstName.toString().length() - 1);
if(currentUser.getFirstName().equals("")){
    currentUser.setFirstName(userFirstName);
}
```

This bit of code gets the possibleFirstNames from the spreadsheet API that I used. Then it goes through all them and checks to see if the most recent user has inputted a name, if not then it will use the last name used on the google sheets.

```
boolean compareDate(Date dateA, Date dateB){
    boolean isLess = false;
    if(dateA.getYear() < dateB.getYear()){
        isLess = true;
    } else if (dateA.getYear() == dateB.getYear()){
        if(dateA.getMonth() < dateB.getMonth()) {
            isLess = true;
        } else if (dateA.getMonth() == dateB.getMonth()) {
            if(dateA.getDate() < dateB.getDate()){
                isLess = true;
            }
        }
    }
    return isLess;
}
```

This piece of code is used to determine whether a date is smaller than another. It had to be done this way because if we were comparing Mar. 1 and Feb. 2 it would say that Mar 1 is smaller if you purely look at dates since it is a one. So you have to check everything. I later realized that there was a function that does this for you but too little too late I guess

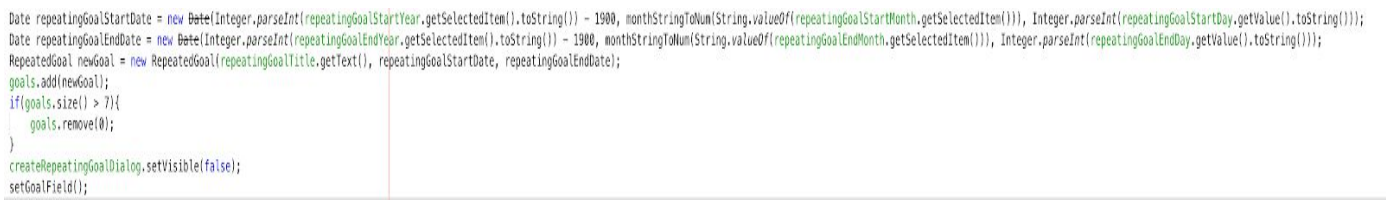
```
for(List row : currentGoals){
    currentGoal++;
    if(row.size() <  5){
        RepeatedGoal newGoal = new RepeatedGoal(row.get(0).toString(), df.parse(row.get(1).toString()), df.parse(row.get(2).toString()));
        goals.add(newGoal);
        if(goals.size() > 7){
            goals.remove(0);
        }
    } else {
        SingleGoal newGoal = new SingleGoal((row.get(0).toString()), df.parse(row.get(1).toString()));
        goals.add(newGoal);
        if(goals.size() > 7){
            goals.remove(0);
        }
    }
}
```

This code will get the current goals that you are using. So if it finds any goals under the

“Current Goals” section of my sheet then it will load them into the program. This was tricky to write as it was a bunch of casting and stuff as well as an enhanced for loop with nested if statements.

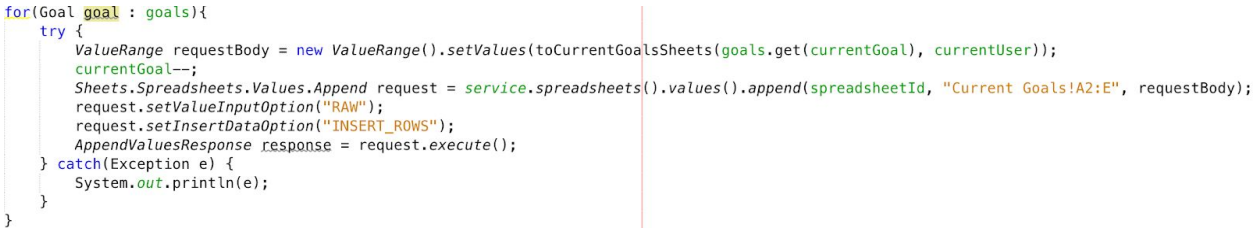
```
List<List<Object>> goalPartsList = new ArrayList<List<Object>>();
List<Object> goalPartsListInner = new ArrayList<Object>(Arrays.asList(goalParts));
goalPartsList.add(goalPartsListInner);
```

This bit of code was my solution as to how to get an List<> into a List<List<>> because a lot of API required you to use 2D lists and I had to handle most the data in a 1D array so conversion was key.



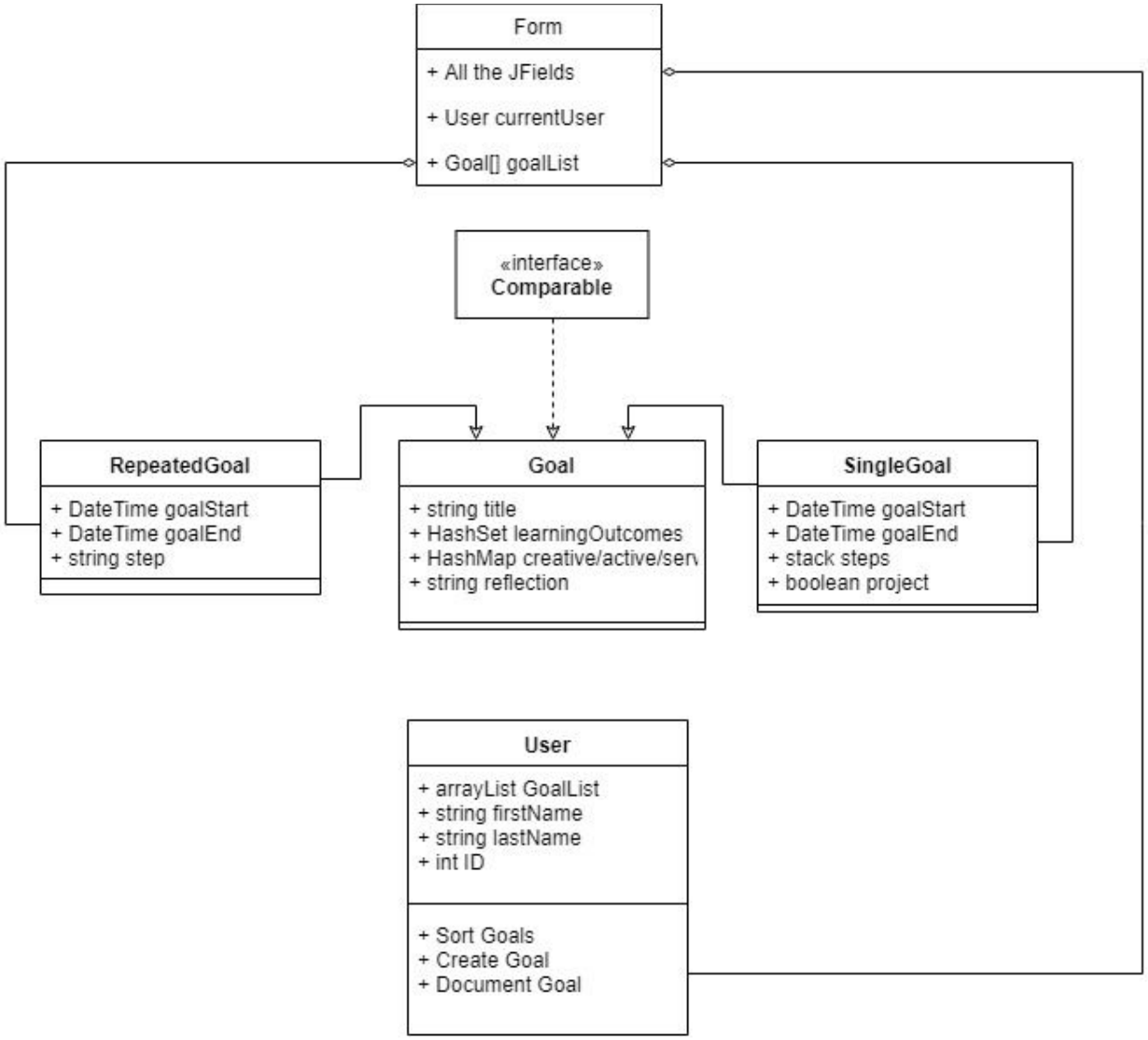
```
Date repeatingGoalStartDate = new Date(Integer.parseInt(repeatingGoalStartYear.getSelectedItem().toString()) - 1900, monthStringToNum(String.valueOf(repeatingGoalStartMonth.getSelectedItem())), Integer.parseInt(repeatingGoalStartDay.getValue().toString()));
Date repeatingGoalEndDate = new Date(Integer.parseInt(repeatingGoalEndYear.getSelectedItem().toString()) - 1900, monthStringToNum(String.valueOf(repeatingGoalEndMonth.getSelectedItem())), Integer.parseInt(repeatingGoalEndDay.getValue().toString()));
RepeatedGoal newGoal = new RepeatedGoal(repeatingGoalTitle.getText(), repeatingGoalStartDate, repeatingGoalEndDate);
goals.add(newGoal);
if(goals.size() > 7){
    goals.remove(0);
}
createRepeatingGoalDialog.setVisible(false);
setGoalField();
```

I finagled this piece of code for a while but couldn’t get it in a good position that would work. If you want to see it more than look for the CriteriaC1 in the IA folder. Anyway this little gem showed the struggle that was getting stuff out of text fields and into a usable format. As well it shows how I handled the goal bar (The thing that displays the current goals).



```
for(Goal goal : goals){
    try {
        ValueRange requestBody = new ValueRange().setValues(toCurrentGoalsSheets(goals.get(currentGoal), currentUser));
        currentGoal--;
        Sheets.Spreadsheets.Values.Append request = service.spreadsheets().values().append(spreadsheetId, "Current Goals!A2:E", requestBody);
        request.setValueInputOption("RAW");
        request.setInsertDataOption("INSERT_ROWS");
        AppendValuesResponse response = request.execute();
    } catch(Exception e) {
        System.out.println(e);
    }
}
```

Finally I have this piece that sets that current goals whenever you exit the program. This is an example of what I had to do in order to work with the API. It was a lot of added work that took a while to wrap my head around, but I ended up getting it done.



This is my class Diagram