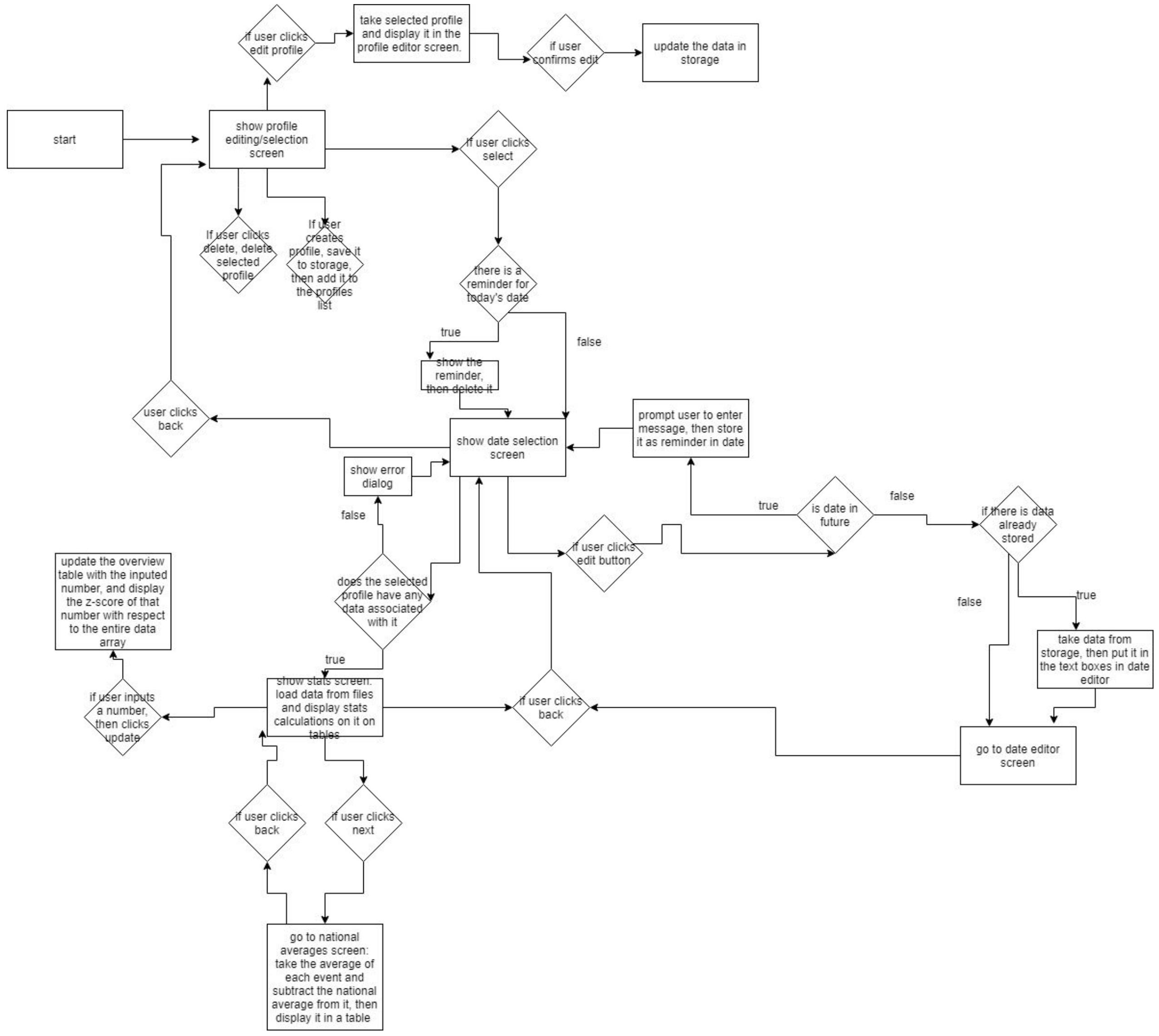
A flowchart that details how the program works.


start
show profile editing/selection screen
if user clicks edit profile
take selected profile and display it in the profile editor screen.
if user confirms edit
update the data in storage
if user clicks select
If user clicks delete, delete selected profile
If user creates profile, save it to storage, then add it to the profiles list
there is a reminder for today's date
true
false
show the reminder, then delete it
user clicks back
show date selection screen
prompt user to enter message, then store it as reminder in date
show error dialog
false
true
is date in future
false
if there is data already stored
if user clicks edit button
does the selected profile have any data associated with it
false
true
take data from storage, then put it in the text boxes in date editor
true
update the overview table with the inputed number, and display the z-score of that number with respect to the entire data array
if user inputs a number, then clicks update
show stats screen. load data from files and display stats calculations on it on tables
if user clicks back
go to date editor screen
if user clicks back
if user clicks next
go to national averages screen: take the average of each event and subtract the national average from it, then display it in a table

Tests

Italics = program output

Test Type	Nature of Test	Example
Check the the profile creation screen works	Creating a profile should add it to the profile list and save it to a .csv file. There should be only one name per profile, so if the user attempts to enter	User enters profile "Bob", age 14, gender male. *Profile stored* User enters profile "Alice", age 13, gender female. *Profile stored*

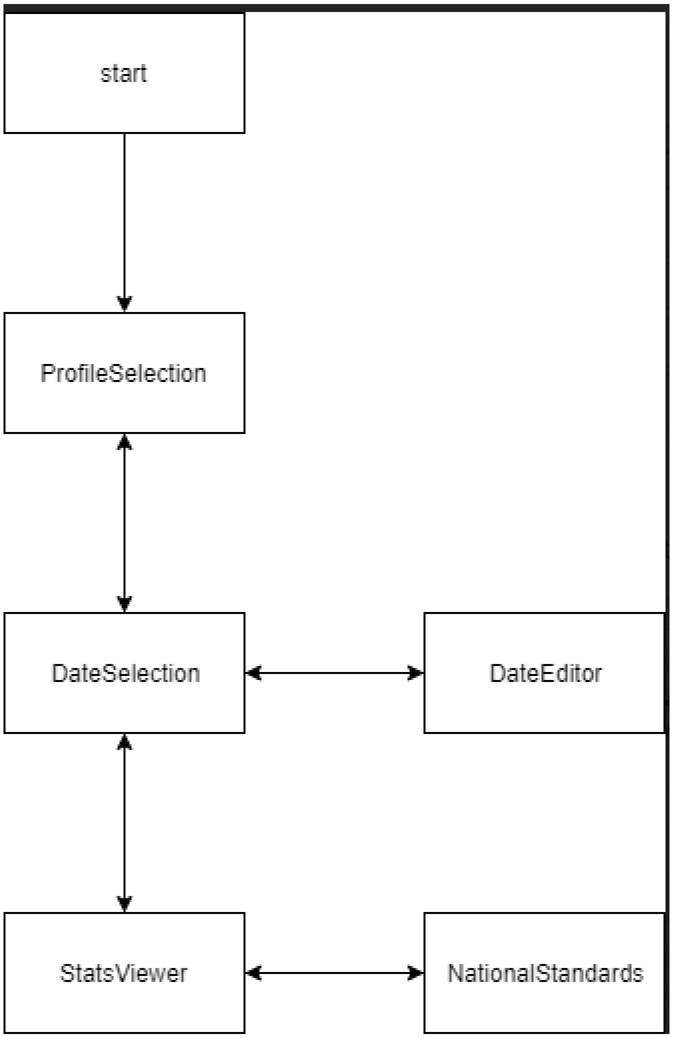
	the same name for different profiles, an error message will show.	User enters profile "Bob", age 15, gender male. *Error: name already taken*
Check that the profile editing function works	If the user selects a profile, then clicks edit, the create profile interface should switch to the edit profile interface, with the selected profile's current settings displayed in the proper fields. If the user edits the fields and presses "edit", the profile should be updated and the .csv file should be updated as well. No two profiles can share the same name.	User edits profile "Bob", age 14, gender male to "Alice", age 20, gender female. *Profile is updated* User edits profile "Alice", age 20, gender female to "Alice", age 10, gender female. *Profile is updated*
Check that the date selection screen works.	If the user selects a date in the future, they will be prompted to enter a reminder message. If there are any reminders that have passed when this screen is opened, display them, then delete them. If the user selects a date not in the future, go to the date editor screen for this date.	If the user selects a date in the future *Enter reminder text* If the user selects a date not in the future *go to date editor screen*
Check that the date editor screen works	If the user has selected a date with data saved into it, the fields in the UI should be filled with those numbers, or be left empty if the number is -1. If the user presses update, the numbers entered by the user should be saved into a .csv file. If a field is empty, save -1 instead.	User enters 3.14, 5, 1, [null], and 7.1 into the five fields, respectively. *Program stores 3.14, 5, 1, -1, and 7.1 into the corresponding .csv file*

Classes and Files Required

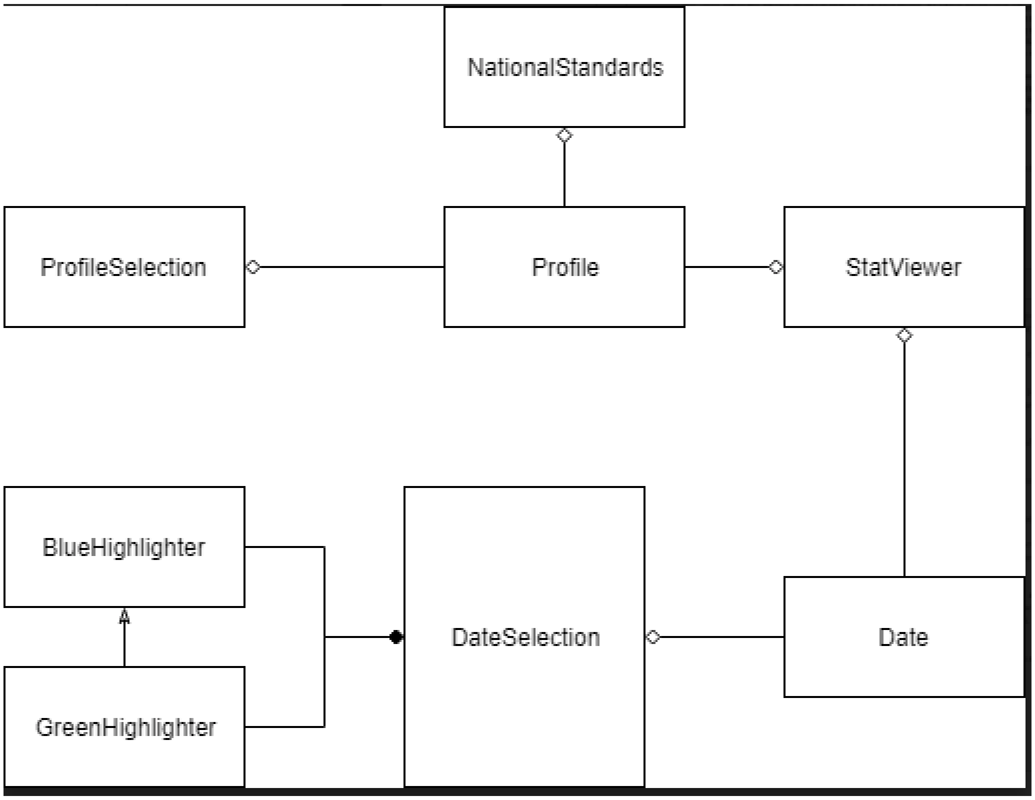
Class	**Description**
ProfileSelection extends JFrame	The class with the main method. Allows the user to edit, delete, and add profiles.
DateSelection extends JFrame	Allows the user to select dates. The user can either add a reminder for that date if it is in the future, or input swim times for that date otherwise.

Date	Class that defines the custom date object. A new date object is created when the user selects a date in DateSelection. Not to be confused with java.util.Date.
Profile	Class that defines the profile object. A new profile object is created when the user adds a profile.
DataManager	Stores and reads data from files.
Gender (enum)	either MALE or FEMALE
DateEditor extends JFrame	Allows the user to enter and edit swim times.
Stat	Class that contains static methods for statistic calculations.
StatsViewer extends JFrame	Displays statistical tables for information based on user's swim times. One table for overview, the rest for each stroke.
NationalStats extends JFrame	Display the national standards relative to the user's age and gender, and how the user's average swim times compare to said standards.
HighlightEvaluator implements IDateEvaluator	Use to higlight dates in JCalendar in blue.
HighlightEvaluator2 extends HighlightEvaluator	Use to highlight dates in JCalendar in green.

A flowchart that shows how the program switches between screens.



A UML program detailing relationships between objects and who uses them.

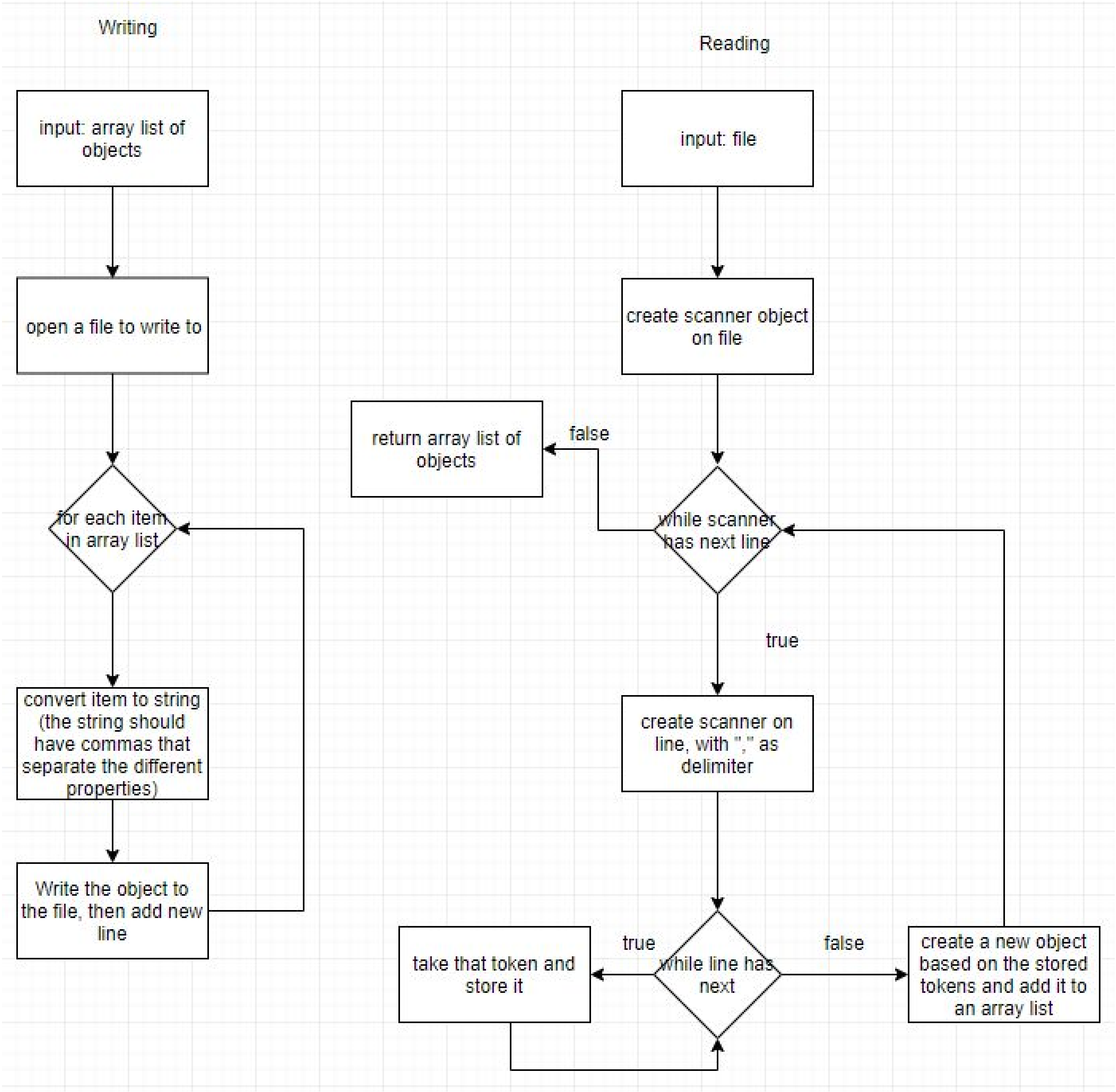


Data Storage

Data will be stored in .csv files because they can be easily read by a human. For storing swim times, the date of the time will be the file name (eg: swim times for 14 March, 2015 will be stored in "14 3 2015.csv"). Inside the file, the first cell in each row would be the profile UUID that determine who the numbers in that row would belong to. The next five cells in that row would be the numbers, one for each stroke.
For the national standards, numbers for boys will be stored in "boys.csv" and girls will be stored in "girls.csv". That way, I won't have to make a 3D data file (one dimension would be the stroke, the other would be the age). Data will be stored and retrieved as array list as opposed to arrays, because array lists are not constrained by size.

Algorithms

As storing data in .csv files are meant to be human-readable, the program will have to convert the data into strings before writing, and convert the strings back into data after reading. Each row in the file will represent one object. The program will take a list of the objects, convert it into a string, write it to the file along with a new line. When reading the data, a `Scanner` will extract the strings line by line and pass them into the constructor of that object class.


Writing
Reading
input: array list of objects
input: file
open a file to write to
create scanner object on file
return array list of objects
false
for each item in array list
while scanner has next line
true
convert item to string (the string should have commas that separate the different properties)
create scanner on line, with "," as delimiter
Write the object to the file, then add new line
true
false
take that token and store it
while line has next
create a new object based on the stored tokens and add it to an array list

Word Count: 236