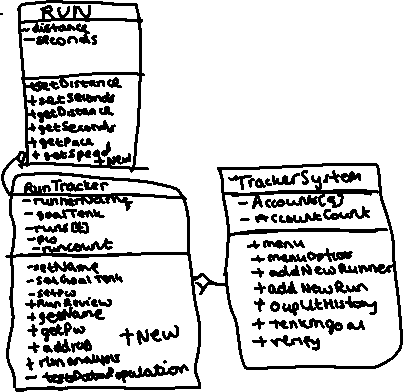
**Skeleton Program**

You will be receiving a skeleton program to research over the holidays. You should use the following pages to make notes on the program. Identify what it does, how it works, any errors and what improvements you would make.

**Explain the system, its purpose, what it does, data input, data returned**

|  |
| --- |
| The purpose of this system is to help Adidas sports system. It allows the user to input their name, add a run they did in metres and seconds, outputs its history and provides a 10km goal analysis. Data input: name, password, distance, speed. Data returned: 10km analysis, and returns the run history when asked, average speed, average pace. |

**Class Diagram**



|  |
| --- |
|  |

**Explain any inheritance**

|  |
| --- |
| There is no inheritance, just association aggregation and static. Association aggregation means that if RUN is deleted then RUN TRACKER and TRACKER SYSTEM are not deleted. Static method (Convert) can be run without being instantiated. |

**Class**

List the variables involved in the program, their role, any validation performed or required, scope, and what they do

|  |  |
| --- | --- |
| Identifier | What it does (inheritance, aggregation etc) |
| Run | Properties/ Variables: distance, seconds, newValue  Methods: SetDistance(s), SetSeconds(s), GetDistance(f), GetSeconds(f), GetPace(f), GetSpeed(f), OutputRun(s)  Distance – this stores the distance they ran (entered in metres)  Seconds – this stores the time it took for them to run it in (entered in seconds)  Get Distance – [ByVal newValue] literally gets the distance  Get Seconds – [ByVal newValue] literally gets the timing  !! Error – wrong data  Set Distance – saves the distance {sub}  Set Seconds – saves the timing {sub}  Get Pace – (seconds/60) means in minutes (distance/1000) means in kilometres   * Time / Distance   Get Speed – Distance Over Time   * Converts time from m/s to km /h with help of Static Convert   OutputRun – [uses GetSpeed and GetPace] to output distance in km, time in minutes, average speed and average pace {sub}  ! ! Association Aggregation with Run Tracker ! ! |
|  |  |

**Properties**

List the variables involved in the program, their role, any validation performed or required, scope, and what they do

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Role | Validation | Scope (G/L) | What it does |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Methods**

List the subroutines involved in the program, parameters, return type, type, and what they do

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Type  (F/P) | Parameters | Return | What it does |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Class**

List the variables involved in the program, their role, any validation performed or required, scope, and what they do

|  |  |
| --- | --- |
| Identifier | What it does (inheritance, aggregation etc) |
| TrackerSystem | Properties/ Variables: accounts(9), AccountCount  Methods: Constructor(s), Menu(f) ,Menu Options(s), AddNewRunner(s), AddNewRun(s), OutputHistory(s), TenKMGoal(s), Verify(f)  Constructor – stores the pre set data for Simon and Annie {sub}  Menu – allows user to choose what they want to do from the menu  MenuOptions – Prints the options of what the user can do {sub}  Add New Runner – Allows user to add a new runner. It allows input of name, 10km goal time, password {sub}  AddNewRun – [ByVal RunTracker Class] Checks username and password are in the database (if not then there is an output message). Then it allows user to input run and stores the data in the array Accounts {sub}  OutputHistory – [ByVal RunTrackerClass ]Checks username and password are in the database (if not then there is an output message). Then it outputs all the data about runs previously for that particular runner (since prints according to array location)  TenKmGoal – [ByVal RunTracker Class] Checks username and password are in the database (if not then there is an output message). Then it outputs whether or not the 10km goal has been met  ! ! Association Aggregation with RunTracker ! ! |

**Properties**

List the variables involved in the program, their role, any validation performed or required, scope, and what they do

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Role | Validation | Scope (G/L) | What it does |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Methods**

List the subroutines involved in the program, parameters, return type, type, and what they do

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Type  (F/P) | Parameters | Return | What it does |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Class**

List the variables involved in the program, their role, any validation performed or required, scope, and what they do

|  |  |
| --- | --- |
| Identifier | What it does (inheritance, aggregation etc) |
| RunTracker | Properties/ Variables: runnerName, goalTenK, runs(4), pw, runCount  Methods: Constructor(s), SetName(f) ,SetGoalTenK(f), SetPw(f), RunReview(s), GetName(s), GetPw(f), AddRun(s), RunAnalysis(s), TestDataPopulation(s)  Constructor– [ByVal newname, newGoal, newPw, test] !!! {sub}  SetName[ByVal inputName] – Returns whether or not they input a name or nothing in the insert a name section  setGoalTenK – [ByVal inputGoal ] – stores inputGoal as the goal  setPw [inputPw] – stores inputPw as the password  runReview - [ByVal Run Class] Outputs how much ran (km) and time (S), speed, pace {sub}  GetName – returns name of person  GetPw – returns password of person  AddRun [ByVal Run Class] – goes through adding a new run and storing it and then calculating pace and speed {sub}  RunAnalysis [ByVal Convert Static Class] – converts latest data into speed. Compares it with goal speed and then outputs a message on whether or not goal is met {sub}  TestDataPopulation [ByVal Run Class] – stores pre set data for Simon and Anne and one other person?!?!? {sub}  ! ! Association Aggregation with Run ! !  ! ! Uses static class Convert ! ! |

**Properties**

List the variables involved in the program, their role, any validation performed or required, scope, and what they do

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Role | Validation | Scope (G/L) | What it does |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Methods**

List the subroutines involved in the program, parameters, return type, type, and what they do

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Type  (F/P) | Parameters | Return | What it does |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Class**

List the variables involved in the program, their role, any validation performed or required, scope, and what they do

|  |  |
| --- | --- |
| Identifier | What it does (inheritance, aggregation etc) |
| Converter | This converts the speed from meters per second to kilometres per hour  Mps2kmph – [ByVal inputSpeed] Converts speed into km/h {shared function}  ! ! Static Function ! ! |

**Properties**

List the variables involved in the program, their role, any validation performed or required, scope, and what they do

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Role | Validation | Scope (G/L) | What it does |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Methods**

List the subroutines involved in the program, parameters, return type, type, and what they do

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Type  (F/P) | Parameters | Return | What it does |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Errors**

|  |  |  |
| --- | --- | --- |
| **Where** | **Why** | **Solution** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Improvements**

|  |  |  |
| --- | --- | --- |
| **Where** | **Why** | **Solution** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |