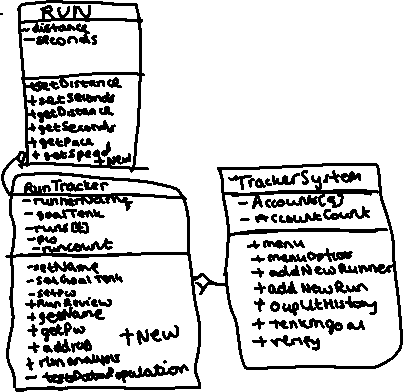
**Skeleton Program**

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  Methods: SetDistance(s), SetSeconds(s), GetDistance(f), GetSeconds(f), GetPace(f), GetSpeed(f), OutputRun(s)  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 |
| Distance | this stores the distance they ran (entered in metres) | No Validation in this class but in Run Tracker class, we added a validation so that a suitable distance was entered | L | Private – integer |
| seconds | this stores the time it took for them to run it in (entered in seconds) | No Validation in this class but in Run Tracker class, we added a validation so that a suitable time was entered | L | Private – integer … crash if decimal? |
|  |  |  |  |  |
|  |  |  |  |  |

**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 |
| Constructor | P |  |  |  |
| GetDistance | F |  | The distance | literally gets the distance |
| SetDistance | P | newValue |  | Makes sure in a range? Make sure not negative? |
| GetSecond | F |  | The seconds | literally gets the timing |
| SetSecond | P | newValue |  |  |
| GetPace | P |  |  | -(seconds/60) means in minutes (distance/1000) means in kilometres  -Time / Distance |
| GetSpeed | P |  |  | -Distance Over Time  -Converts time from m/s to km /h with help of Static Convert |
|  |  |  |  |  |

**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)  ! ! 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 |
| Accounts() | This stores the pre-set data of Simon and Annie along with any other data when added |  |  |  |
| AccountCount | This counts the number of accounts there are – which is necessary for all the loops and stuff |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**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 |
| Constructor | P |  |  | stores the pre set data for Simon and Annie |
| Menu | F |  | Returns true or false after running all the tasks if applicable | Case 0 – false (and so ends the function there and so can exit)  Provides the options for user |
| Menu Options |  | P |  | Prints the options of what the user can do {sub} |
| AddNewRun | P | Run Tracker 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 |
| Verify | F |  | Returns location of user in the array | If names match and password math then returns location in accounts array  Make sure username is unique |
| Add New Runner | P |  |  | Allows user to add a new runner. It allows input of name, 10km goal time, password |
| Output History | P | Run Tracker Class |  | 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) |
| Ten km goal | P | Run Tracker 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 |
|  |  |  |  |  |

**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)  ! ! 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 |
| runnerName | for name |  |  |  |
| goalTenK | Store the 10km goal |  |  |  |
| Runs(4) |  |  |  |  |
| Pw | For password |  |  |  |
| runCount | Stores the number of runs per runner |  |  |  |

**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 |
| AddRun | P | Run Class |  | !Try Catch – to ensure integer added!  Goes through adding a new run and storing it and then calculating pace and speed  Calculate pace and speed 🡪 useless?  Data Validation – data type to be added  Make sure not 0  Index out of bounds error by adding too many runs??!! |
| constructor | P | newName, newGoal, newPw, test |  |  |
| SetName | F | inputName | Returns a Boolean value | Ensure there is a name in. Returns true or false value. It is never called  Returns whether or not they input a name or nothing in the insert a name section |
| SetGoalTenk | F | inputGoal | Returns a Boolean value | Ability to reset goal  stores inputGoal as the goal |
| setPW | F | inputPw | Returns a Boolean value | !! Add validation  stores inputPw as the password |
| RunReview | P | Run Class |  | Outputs how much ran (km) and time (S), speed, pace |
| GetName | F |  | runnerName | returns name of person |
| GetPW | F |  | pW | returns password of person |
| RunAnalysis | P | Cpnvert Static Class |  | Returns analysis of the last run.  When we add a runner, have an empty data of run stuff, so run analysis will crash if ran before adding a new run.  Converts latest data into speed. Compares it with goal speed and then outputs a message on whether or not goal is met. |
| TestDataPopulation | P | Run Class |  | stores pre set data for Simon and Anne and one other person?!?! |

**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  ! ! Static Function ! ! – used without creating an instance of the class |

**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 |
| Mps2kmph | SHARED FUNCTION | InputSpeed | The converted value | Converts speed into km/h {shared function} |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Module Module1

Sub Main()



Dim NewSystem As New TrackerSystem

Console.WriteLine("Welcome to Addidas System Tracker")



Do

Loop While NewSystem.Menu()

End Sub

End Module

Public Class Run

Private distance As Integer

Private seconds As Integer

Public Sub SetDistance(ByVal newValue As Integer)

distance = newValue

End Sub

Public Sub SetSeconds(ByVal newValue As Integer)

seconds = newValue

End Sub

Public Function GetDistance() As Integer

Return distance

End Function

Public Function GetSeconds() As Integer

Return seconds

End Function

Public Function GetPace() As Double 'changed to double from integer

Return (seconds / 60) / (distance / 1000)

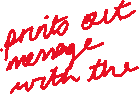
End Function

Public Function GetSpeed() As Double 'changed to double from decimal

Return Convert.mps2kmph(distance / seconds)

End Function

Public Sub OutputRun()



If seconds / 60 = 1 Then



Console.WriteLine("You ran " & distance / 1000 & " km in " & seconds / 60 & " Minute")



Else



Console.WriteLine("You ran " & distance / 1000 & " km in " & seconds / 60 & " Minutes")



End If



Console.WriteLine(" You average speed was " & GetSpeed())

Console.WriteLine(" You average pace was " & GetPace())

End Sub

End Class

Public Class RunTracker

Private runnerName As String

Private goalTenK As Integer '3000s = 50mins

Private runs(4) As Run

Private pw As String

Private runCount As Integer = 0

Public Sub New(ByVal newName As String, ByVal newGoal As Integer, ByVal newPw As String, ByVal test As Boolean)

runnerName = newName

goalTenK = newGoal

pw = newPw

If test = True Then

TestDataPopulation()

End If

End Sub

Public Function SetName(ByVal inputName As String) As Boolean

runnerName = inputName

If runnerName <> "" Then

Return True

End If

Return False

End Function

Public Function SetGoalTenK(ByVal inputGoal As Integer) As Boolean

goalTenK = inputGoal

Return True

End Function

Public Function SetPw(ByVal inputPw As String) As Boolean

pw = inputPw

Return True

End Function

Public Sub RunReview()

If runCount > 0 Then

For x = 0 To runCount - 1

runs(x).OutputRun()

Next

End If

End Sub

Public Function GetName() As String

Return runnerName

End Function

Public Function GetPw() As String

Return pw

End Function

Public Function GetKmGoal() As String

Return goalTenK

End Function

Public Sub AddRun()

Dim value As Integer

runs(runCount) = New Run()



Dim satisfy As Boolean = 0



Dim distance As Integer



Dim timing As Integer



Try



runs(runCount) = New Run()



Catch ex As Exception



Console.WriteLine("")



Dim temp(runCount - 1) As Run



For x = 0 To runCount - 1



temp(x) = runs(x)



Next



ReDim runs(runCount)



For x = 0 To runCount - 1



runs(x) = temp(x)



Next



runs(runCount) = New Run()



End Try



Do



Do



Try



Console.WriteLine("Enter the DISTANCE you ran, in meters")



value = Console.ReadLine()



Catch ex As Exception



MsgBox("Not entered a suitable answer. Please try again. ")



End Try



Loop Until value > 0



distance = value



runs(runCount).SetDistance(value)

Do



Try



Console.WriteLine("Enter the TIME you ran, in seconds")



value = Console.ReadLine()

Catch ex As Exception



MsgBox("Not entered a suitable answer. Please try again. ")



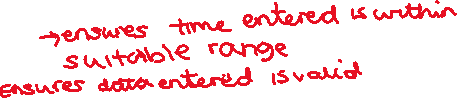
End Try



Loop Until value > 0.0 And value < 60 \* 60 \* 80



timing = value



runs(runCount).SetSeconds(value)

If distance / timing >= 36.0 Then



satisfy = 1



Console.WriteLine("The data you have entered suggests that you have magically been able to run more than 36 km / h which is wonderful but we don't believe you. Please input your data again! :)")



End If

Loop Until satisfy = 0



'runs(runCount).GetPace() 'not needed and so deleted (on version 2)

'runs(runCount).GetSpeed()



runCount += 1

End Sub

Public Sub RunAnalysis()



Dim goalSpeed As Double = Convert.mps2kmph(10000 / goalTenK)



If runCount = 0 Then



Console.WriteLine("WRONG ")



Else



Dim latestSpeed As Double = runs(runCount - 1).GetSpeed

If goalSpeed > latestSpeed Then

Console.WriteLine("You have not met your goal speed")

Else

Console.WriteLine("You have met your goal speed")

End If

End If

End Sub

Private Sub TestDataPopulation()

runs(0) = New Run

runs(0).SetDistance(1000)

runs(0).SetSeconds(240)

runs(1) = New Run

runs(1).SetDistance(2000)

runs(1).SetSeconds(720)

runs(2) = New Run

runs(2).SetDistance(3000)

runs(2).SetSeconds(1200)

runCount = 3

End Sub

Public Sub SaveFile()



Dim filename As String = "c:\data.txt"

Dim file As System.IO.StreamWriter



file = My.Computer.FileSystem.OpenTextFileWriter(filename, False)



file.WriteLine("Name: " & GetName())



file.WriteLine("Password: " & GetPw())

file.WriteLine("10KM Goal: " & GetKmGoal())



For x = 0 To runCount - 1

file.WriteLine("Distance: " & runs(x).GetDistance / 1000)



file.WriteLine("Time: " & runs(x).GetSeconds / 60)



file.WriteLine("Speed: " & runs(x).GetSpeed)



file.WriteLine("Pace: " & runs(x).GetPace)



file.WriteLine()



Next



file.WriteLine()



file.Close()



End Sub



End Class

Imports System



Imports System.IO



Public Class TrackerSystem

Private Accounts(1) As RunTracker

Private AccountCount As Integer = 0

Public Sub New()

Accounts(0) = New RunTracker("Simon", "5000", "fish", True)

Accounts(1) = New RunTracker("Annie", "2500", "fish", True)

AccountCount = 2

End Sub

Public Function Menu() As Boolean

Dim choice As Integer

MenuOptions()

choice = Console.ReadLine()

Select Case choice

Case 0

Return False

Case 1

AddNewRunner()

Case 2

AddNewRun()

Case 3

OutputHistory()

Case 4

TenKmGoal()

Case 5



Update()



Case 6



RemoveRunner()



Case 7



Save()



Case Else

Console.WriteLine("Not an Option")

End Select

Return True

End Function

Public Sub MenuOptions()

Console.WriteLine("")

Console.WriteLine("---------------------------------------")

Console.WriteLine("Choose 1 add a new runner")

Console.WriteLine("Choose 2 add a new run")

Console.WriteLine("Choose 3 for your run history")

Console.WriteLine("Choose 4 for your 10 km goal analysis")

Console.WriteLine("Choose 5 to update data")



Console.WriteLine("Choose 6 to remove a runner")



Console.WriteLine("Choose 7 to save")



Console.WriteLine("Choose 0 to exit")

End Sub

Public Sub RemoveRunner()



Dim name As String



Dim password As String



Console.WriteLine("What is your name?")



name = Console.ReadLine()



Console.WriteLine("What is your password?")



password = Console.ReadLine()



For x = 0 To AccountCount - 1



Next



End Sub



Public Sub AddNewRunner()

Dim newName As String

Dim newGoal As Integer

Dim newPw As String

Dim check As Boolean = False



Dim currentChar As Integer



Dim addedNumber As Boolean = True



Dim count As Integer = 0



Dim total As Integer = 2



Dim passwordCorrect As String



Dim satisfy As Boolean = 0



Dim counter As Integer = 1



Dim oldaccounts(AccountCount - 1) As RunTracker



For x = 0 To AccountCount - 1



oldaccounts(x) = Accounts(x)



Next



ReDim Accounts(AccountCount)



For x = 0 To AccountCount - 1



Accounts(x) = oldaccounts(x)



Next



Console.WriteLine("Enter Your Name")

Do

Do

newName = Console.ReadLine()



While counter <> 0



For x = 0 To AccountCount - 1



counter = 0



If newName = Accounts(x).GetName() Then



Console.WriteLine("That username has already been taken. Please enter another username.")



newName = Console.ReadLine()



counter = 1



End If



Next



End While



addedNumber = True

For x = 1 To Len(newName)



currentChar = (Asc(Mid(newName, x, 1)))



If currentChar > 47 And currentChar < 58 And count = 0 Then



Console.WriteLine("You have entered an integer as part of your name. Please try again. ")



count = count + 1



addedNumber = False



End If



Next



Loop Until addedNumber = True

Do



Try



Console.WriteLine("Enter Your 10 Km goal time (in seconds)")

newGoal = Console.ReadLine

Catch ex As Exception



MsgBox("Not entered an integer")



End Try



Loop Until newGoal > 0



Do



Console.WriteLine("Enter Your Password")

newPw = Console.ReadLine

check = PasswordChecker(newPw)



Do While check = False

Console.WriteLine("Enter a new password. It is not a strong enough password.")

newPw = Console.ReadLine



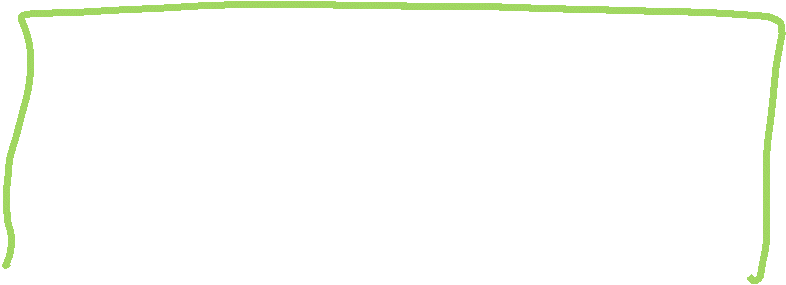
check = PasswordChecker(newPw)



Loop

Console.WriteLine("Please re enter your password (validating your password for you)")

passwordCorrect = Console.ReadLine()



If passwordCorrect <> newPw Then

satisfy = 1

Console.WriteLine("Your passwords do not match. Please reinput again. ")

End If

Loop Until satisfy = 0

Accounts(AccountCount) = New RunTracker(newName, newGoal, newPw, False)

Loop Until Accounts(AccountCount).GetName <> ""

AccountCount += 1

End Sub



Public Function PasswordChecker(ByVal newPw As String) As Boolean



Dim upC As Integer



Dim lowC As Integer

Dim num As Integer

Dim sym As Integer

Dim points As Integer

' Length

If Len(newPw) >= 8 Then



points = points + 1

End If

For x = 1 To Len(newPw)

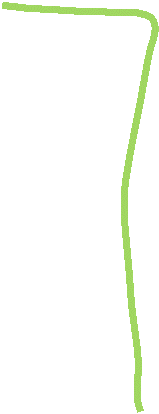
' Uppercase



If Asc(Mid(newPw, x, 1)) > 64 And Asc(Mid(newPw, x, 1)) < 91 Then

upC = upC + 1

If upC = 1 Then



points = points + 1

End If

End If

' Lowercase

If Asc(Mid(newPw, x, 1)) > 96 And Asc(Mid(newPw, x, 1)) < 123 Then

lowC = lowC + 1

If lowC = 1 Then

points = points + 1

End If

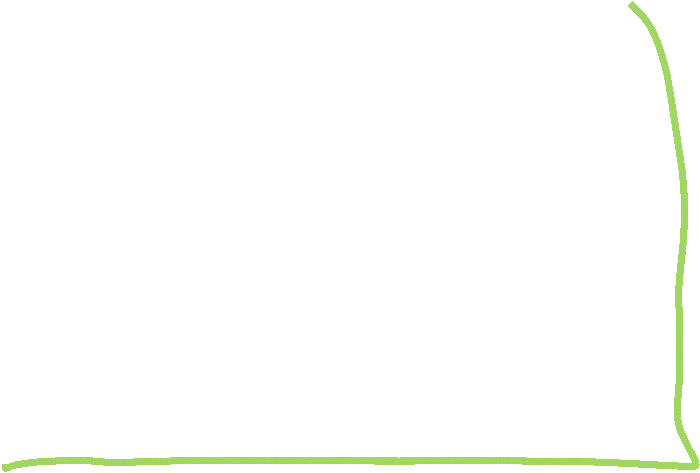
End If

' Numbers

If Asc(Mid(newPw, x, 1)) >= 0 And Asc(Mid(newPw, x, 1)) < 48 Then

num = num + 1

If num = 1 Then



points = points + 1

End If



End If

' Symbols

If Asc(Mid(newPw, x, 1)) > 47 And Asc(Mid(newPw, x, 1)) < 58 Then

sym = sym + 1

If sym = 1 Then

points = points + 1

End If

End If

Next

If points > 3 Then

Return True

End If

Return False

End Function

Public Sub AddNewRun()

Dim foundLocation As Integer

foundLocation = Verify()

If foundLocation > -1 Then

Accounts(foundLocation).AddRun()

End If

End Sub

Public Sub OutputHistory()

Dim foundLocation As Integer

foundLocation = Verify()

If foundLocation > -1 Then

Accounts(foundLocation).RunReview()

End If

End Sub

Public Sub TenKmGoal()

Dim foundLocation As Integer

foundLocation = Verify()

If foundLocation > -1 Then

Accounts(foundLocation).RunAnalysis()

End If

End Sub

Public Function Verify() As Integer

Dim inputName As String

Dim inputPw As String

Console.WriteLine("Enter Your Name")

inputName = Console.ReadLine()

Console.WriteLine("Enter Your Password")

inputPw = Console.ReadLine()

For x = 0 To AccountCount - 1

If inputName = Accounts(x).GetName Then

If inputPw = Accounts(x).GetPw Then

Return x

End If

End If

Next

Console.WriteLine("Incorrect Username or password")

Return -1

End Function

Public Sub Update()

Dim newName As String



Dim newGoal As Integer

Dim choice As String

Dim nmCheck As Integer



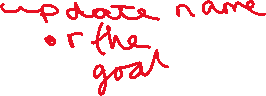
Dim check As Boolean

Console.WriteLine("UPDATE DATA")

nmCheck = Verify()

If nmCheck > -1 Then

Console.WriteLine("Choose 1 to update name



Choose 2 to update goal")

choice = Console.ReadLine()

Select Case choice

Case 1

Console.WriteLine("Enter new name")

check = Accounts(nmCheck).SetName(Console.ReadLine())



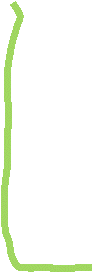
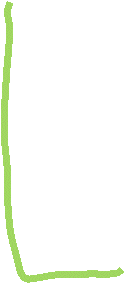
If check = True Then

Console.WriteLine("Name changed")

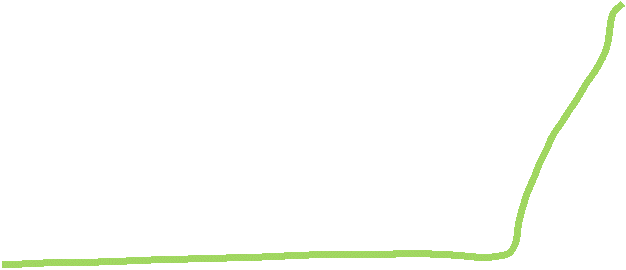


End If

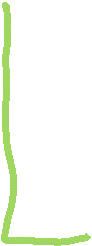
Case 2



Console.WriteLine("Enter new goal")



check = Accounts(nmCheck).SetGoalTenK(Console.ReadLine())



If check = True Then

Console.WriteLine("Goal changed")

End If

Case Else

End Select

End If

End Sub



Public Sub Save()



For x = 0 To AccountCount - 1

Accounts(x).SaveFile()



Next

End Sub

End Class

Public Class Convert

Public Shared Function mps2kmph(ByVal inputSpeed) As Double

Return inputSpeed \* 3.6 'converts meters/sec value to km/h

End Function

End Class