System Maintenance

Environment

Software used when creating my program

- Python 3 (the programming Language)
- Idle (for writing the main program python file)
- Notepad ++ (for writing the rest of the python files)
- PyQt4 (the library used for creating the graphical user interface (GUI))
- SQLite3 (The library used to create the sql database in python)
- CX_Freeze (For compiling my program)

Explanation of Usage

Python 3:

- The language I'm most familiar with
- Open source meaning it's free for commercial use
- Portable meaning it's compatible with most operating systems, even Dos
- Object Oriented and Event Driven meaning it was easy to create this sort of program
- A vast amount of pre existing libraries meaning a large amount of modules are usable

IDLE:

- Automatically downloaded with Python
- Designed especially for python
- Easy to execute code as you don't have to assign a path like in notepad++ or sublime text
- User friendly design and layout
- Has a debug feature making testing easier

Notepad++:

- Tabular layout which makes it more manageable to work with multiple files at once
- More advanced features than IDLE like blank operations

PyQt4:

- Designed to easily allow the implementation of GUI in Python

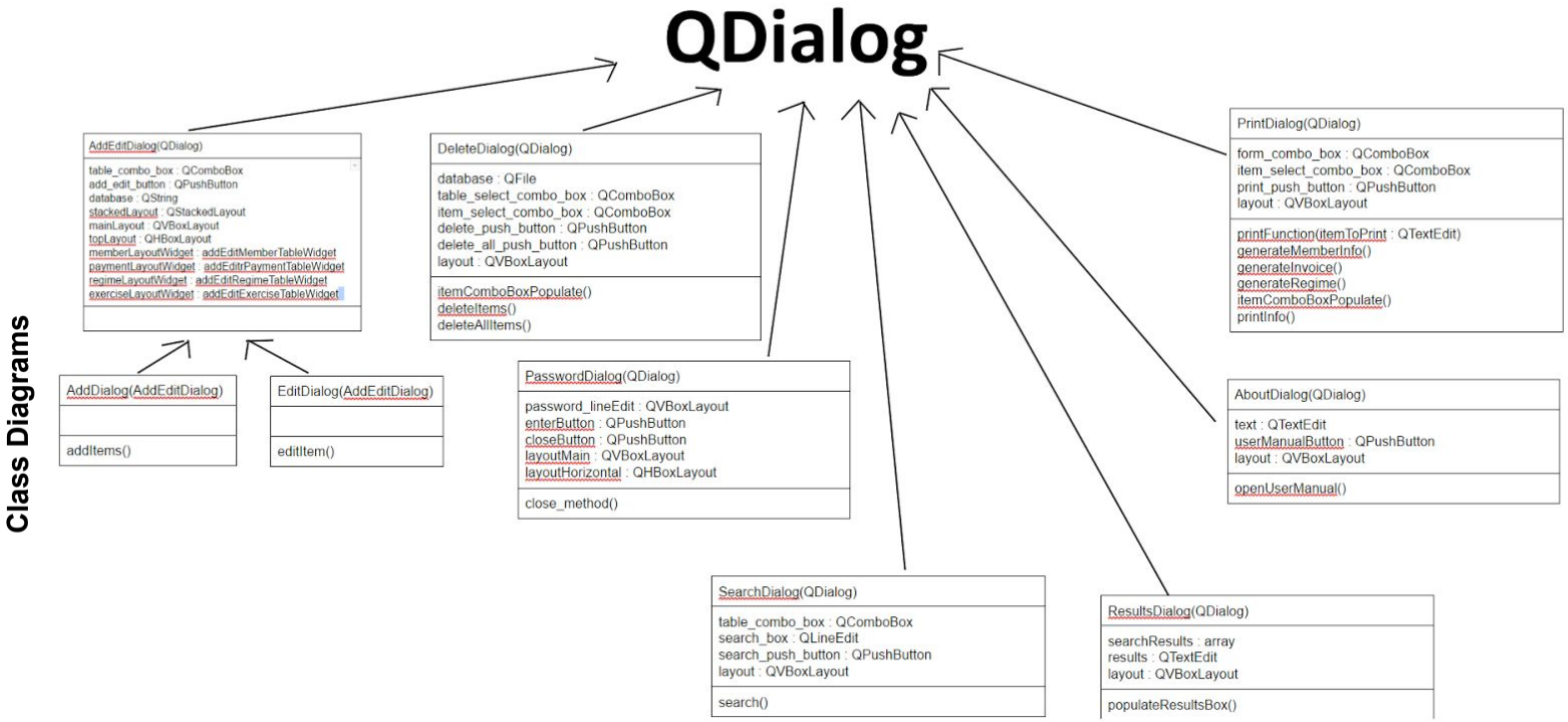
SQLite3:

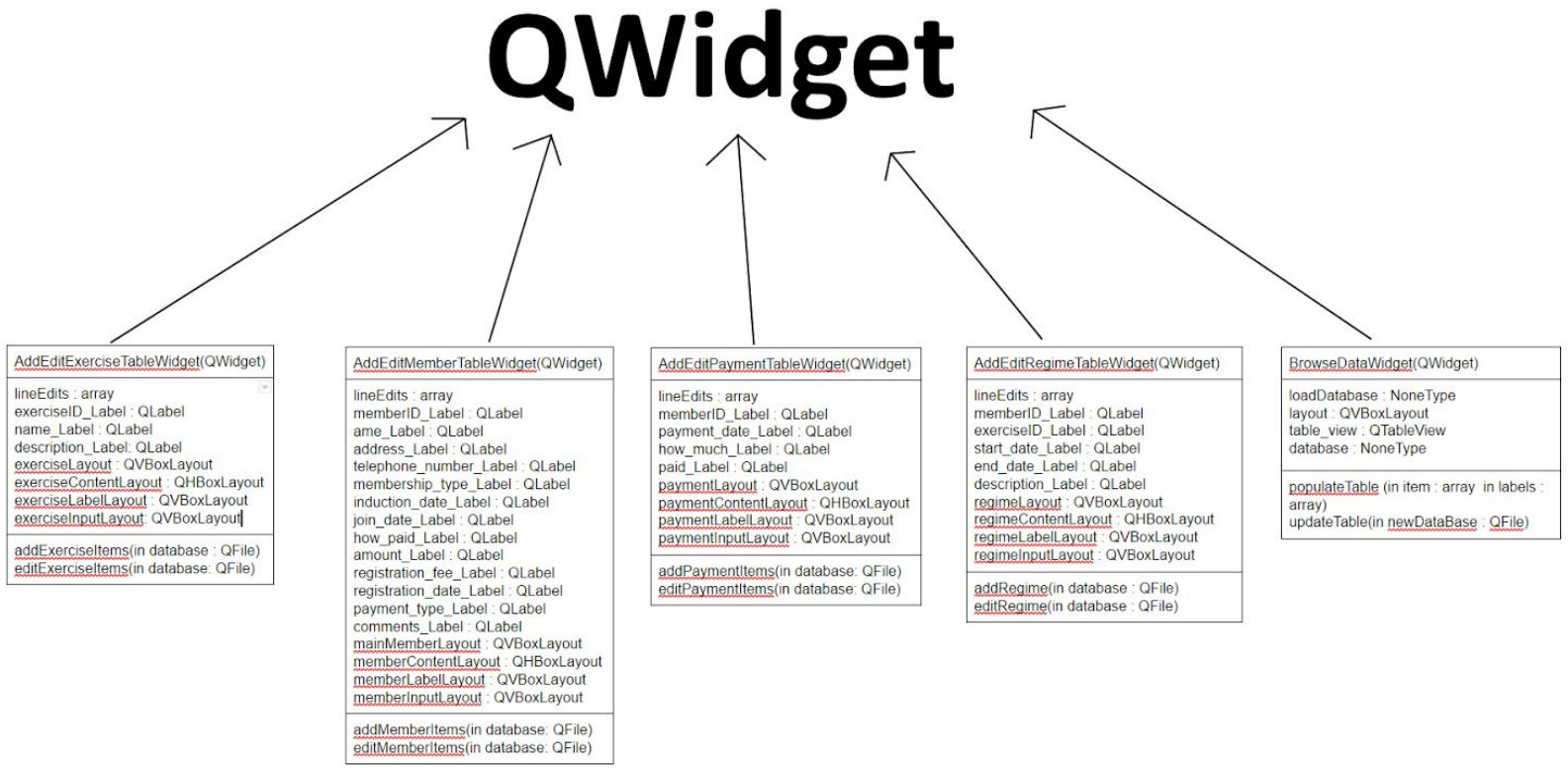
- Simple but usable version of mySQL allowing me to make databases easily but lacks some more advanced features
- Lacks security but this has been made up for by security in the main program those this made it easier to create the program and the database
- Already installed with Python3

CX_Freeze

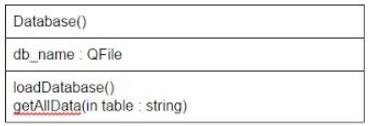
- Used to compile python programs
- Compatible with Python 3 unlike most python compilers that only work with Python 2
- Make it easier to distribute to my client and to install on my client's computer
- Allows the inclusion of images, databases and other file types

Class Diagrams

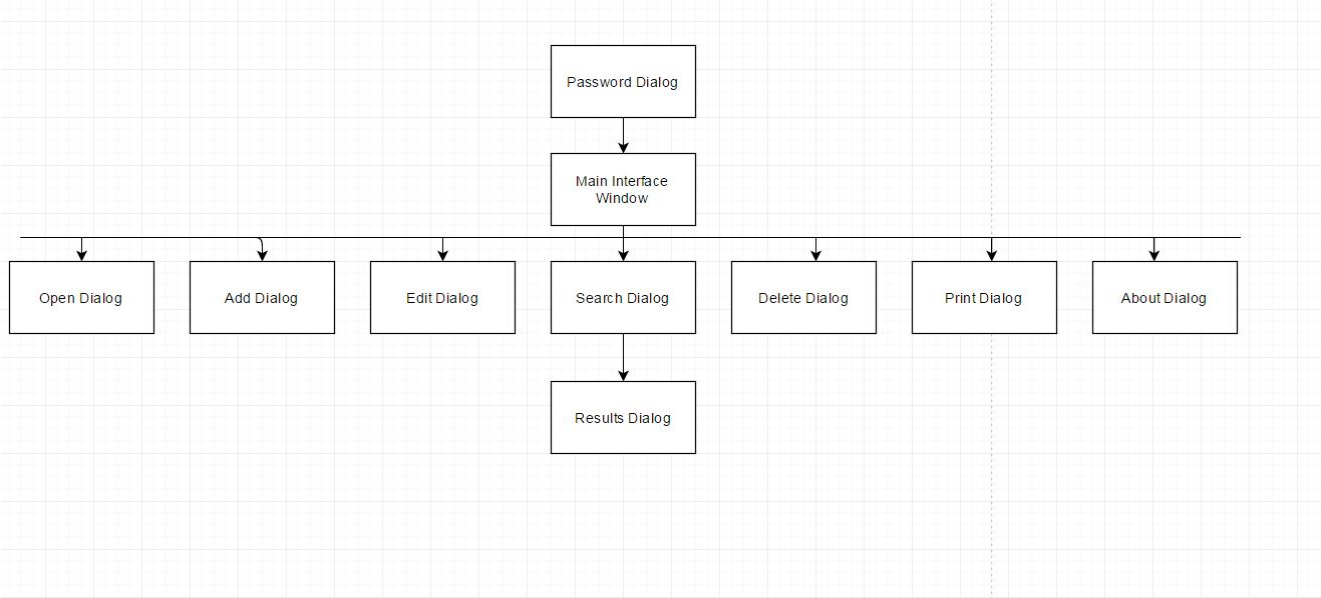



QWidget
AddEditExerciseTableWidget(QWidget)
lineEdits : array
exerciseID_Label : QLabel
name_Label : QLabel
description_Label: QLabel
exerciseLayout : QVBoxLayout
exerciseContentLayout : QHBoxLayout
exerciseLabelLayout : QVBoxLayout
exerciseInputLayout: QVBoxLayout
addExerciseItems(in database : QFile)
editExerciseItems(in database: QFile)
AddEditMemberTableWidget(QWidget)
lineEdits : array
memberID_Label : QLabel
ame_Label : QLabel
address_Label : QLabel
telephone_number_Label : QLabel
membership_type_Label : QLabel
induction_date_Label : QLabel
join_date_Label : QLabel
how_paid_Label : QLabel
amount_Label : QLabel
registration_fee_Label : QLabel
registration_date_Label : QLabel
payment_type_Label : QLabel
comments_Label : QLabel
mainMemberLayout : QVBoxLayout
memberContentLayout : QHBoxLayout
memberLabelLayout : QVBoxLayout
memberInputLayout : QVBoxLayout
addMemberItems(in database: QFile)
editMemberItems(in database: QFile)
AddEditPaymentTableWidget(QWidget)
lineEdits : array
memberID_Label : QLabel
payment_date_Label : QLabel
how_much_Label : QLabel
paid_Label : QLabel
paymentLayout : QVBoxLayout
paymentContentLayout : QHBoxLayout
paymentLabelLayout : QVBoxLayout
paymentInputLayout : QVBoxLayout
addPaymentItems(in database: QFile)
editPaymentItems(in database: QFile)
AddEditRegimeTableWidget(QWidget)
lineEdits : array
memberID_Label : QLabel
exerciseID_Label : QLabel
start_date_Label : QLabel
end_date_Label : QLabel
description_Label : QLabel
regimeLayout : QVBoxLayout
regimeContentLayout : QHBoxLayout
regimeLabelLayout : QVBoxLayout
regimeInputLayout : QVBoxLayout
addRegime(in database : QFile)
editRegime(in database : QFile)
BrowseDataWidget(QWidget)
loadDatabase : NoneType
layout : QVBoxLayout
table_view : QTableView
database : NoneType
populateTable (in item : array in labels : array)
updateTable(in newDataBase : QFile)


QMainWindow
AppWindow(QMainWindow)
file : NoneType
open_push_button : QPushButton
add_push_button : QPushButton
edit_push_button : QPushButton
search_push_button : QPushButton
delete_push_button : QPushButton
print_push_button : QPushButton
tab_bar : QTabWidget
tabs : array
tabNames : array
labels : nested array
toolBar : QMenuBar
layout1 : QHBoxLayout
layout2 : QHBoxLayout
layout3 : QVBoxLayout
mainWidget : QWidget
open_file_menu()
delete()
print_stuff()
search()
edit()
add()
about_the_program()
password(currentPass : string)
main()


Database()
db_name : QFile
loadDatabase()
getAllData(in table : string)

Navigation Diagram



CODE STRUCTURE

All of the code reviewed in this part of the system maintenance section is available in the Implementation section.

Add Edit Member Table Widget

This code is structured as a class so that it can parent both add and edit table widgets in the system. I did this because not only does it prevent duplication of code in the system as the Member Table Widget needs to be used for the Add and Edit input forms. This also reduced development time and makes debugging errors easier and quicker and still provides the full functionality required.

Add Edit Payment Table Widget

This code is structured as a class so that it can parent both add and edit table widgets in the system. I did this because not only does it prevent duplication of code in the system as the Payment Table Widget needs to be used for the Add and Edit input forms. This also reduced development time and makes debugging errors easier and quicker and still provides the full functionality required.

Add Edit Regime Table Widget

This code is structured as a class so that it can parent both add and edit table widgets in the system. I did this because not only does it prevent duplication of code in the system as the Regime Table Widget needs to be used for the Add and Edit input forms. This also reduced

development time and makes debugging errors easier and quicker and still provides the full functionality required.

Add Edit Exercise Table Widget
This code is structured as a class so that it can parent both add and edit table widgets in the system. I did this because not only does it prevent duplication of code in the system as the Exercise Table Widget needs to be used for the Add and Edit input forms. This also reduced development time and makes debugging errors easier and quicker and still provides the full functionality required.

Browse Data Widget
This code is used every time a database is loaded so it's structured to be efficient. It does this by inheriting functions from the gym database class and its only use is to load a database and set its layout.

Gym Password Dialog Class
This code is used frequently throughout the program when it's first opened and when the delete function is started. Its structured to display the password dialog, close, and if the password was entered incorrectly the repetition of the window is performed in a for loop where the class is called and not within the class itself.

Gym Search Dialog Class
This code is used to search through the database. It calls certain function from the Search Function in order for the code to be easily reusable.

Gym Search Results Dialog Class
This code is used to display the results of a user's search query.

Gym About Dialog Class
This code is used to display the about section of the system. It imports the webbrowser module in order to open a link to the user manual.

Gym Add Dialog Class
This class contains a small amount of code just to make the Gym Add Edit Dialog Class (which it inherits) specific to an Add dialog.

Gym Add Edit Dialog Class
This code is structured as a class so that it can parent both add and edit dialog classes in the system. I did this because not only does it prevent duplication of code in the system as it needs to be used for the Add and Edit input forms. This also reduced development time and makes debugging errors easier and quicker and still provides the full functionality required.

Gym Edit Dialog Class
This class contains a small amount of code just to make the Gym Add Edit Dialog Class (which it inherits) specific to an Edit dialog.

Gym Database Class
This code is used to load a database and to use sql to get data from a specific table in the database. Its methods are inherited by the Browse Data Widget to separate the PyQt operations and the SQL functions to make the code more maintainable and more reusable.

Gym Delete Dialog Class
This code is used to delete items through the database. It calls certain function from the Delete Function in order for the code to be easily reusable.

VARIABLE LIST

AddEditMemberTableWidget

Variable	**Purpose**
self.lineEdits	An array to store a series of other variables assigned to the values self.enter_text{0}.format(count)
self.memberID_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a memberID
self.name_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a name
self.address_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter an address
self.telephone_number_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a telephone number
self.membership_type_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a membership type
self.induction_date_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter an

	induction date
self.join_date_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a join date
self.how_paid_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a method of payment
self.amount_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a amount of money
self.registration_fee_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a registration fee
self.registration_date_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a registration date
self.payment_type_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a payment type
self.comments_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter any additional comments
self.mainMemberLayout	The main Vertical Box layout for the member table widget for all the other layouts to be added to
self.memberContentLayout	A Horizontal Box layout for the label and input layouts to be added to
self.memberLabelLayout	A Vertical Box layout for all the Label QLabels to be added to
self.memberInputLabelLayout	A Vertical Box Layout for all the Input QLineEdits to be added to
columns	A local variable to list the required names of all the columns in the member table to pass into an sql statement
items	A string for all the sql for the items being edited to be passed into the editMember

	function
count	A variable for counting through the number 1 through 12

AddEditPaymentTableWidget

Variable	**Purpose**
self.lineEdits	An array to store a series of other variables assigned to the values self.enter_text{0}.format(count)
self.memberID_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a memberID
self.payment_date_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a payment date
self.how_much_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter an amount of money
self.paid_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a boolean value describing whether a client has paid
self.paymentLayout	The main Vertical Box layout for the payment table widget for all the other layouts to be added to
self.paymentContentLayout	A Horizontal Box layout for the label and input layouts to be added to
self.paymentLabelLayout	A Vertical Box layout for all the Label QLabels to be added to
self.paymentInputLabelLayout	A Vertical Box Layout for all the Input QLineEdits to be added to
columns	A local variable to list the required names of all the columns in the payment table to pass into an sql statement

items	A string for all the sql for the items being edited to be passed into the editPayment function
count	A variable for counting through the number 2 through 3

AddEditRegimeTableWidget

Variable	**Purpose**
self.lineEdits	An array to store a series of other variables assigned to the values self.enter_text{0}.format(count)
self.memberID_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a memberID
self.exerciseID_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter an exerciseID
self.start_date_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a start date
self.end_date_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter an end date
self.description_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a description
self.regimeLayout	The main Vertical Box layout for the regimetable widget for all the other layouts to be added to
self.regimeContentLayout	A Horizontal Box layout for the label and input layouts to be added to
self.regimeLabelLayout	A Vertical Box layout for all the Label QLabels to be added to
self.regimeInputLabelLayout	A Vertical Box Layout for all the Input QLineEdits to be added to

columns	A local variable to list the required names of all the columns in the regime table to pass into an sql statement
items	A string for all the sql for the items being edited to be passed into the editRegimefunction
count	A variable for counting through the number 2 through 4

AddEditExerciseTableWidget

self.lineEdits	An array to store a series of other variables assigned to the values self.enter_text{0}.format(count)
self.exerciseID_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a exerciseID
self.name_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a name
self.description_Label	A string of text letting the user know that the QLineEdit adjacent is for the user to enter a description of an exercise
self.exerciseLayout	The main Vertical Box layout for the exercise table widget for all the other layouts to be added to
self.exerciseContentLayout	A Horizontal Box layout for the label and input layouts to be added to
self.exerciseLabelLayout	A Vertical Box layout for all the Label QLabels to be added to
self.exerciseInputLabelLayout	A Vertical Box Layout for all the Input QLineEdits to be added to
columns	A local variable to list the required names of all the columns in the exercise table to pass into an sql statement
items	A string for all the sql for the items being

	edited to be passed into the editExerciseFunction
count	A variable for counting through the number 1 through 2

BrowseDataWidget

Variable	**Purpose**
self.loadDataBase	A variable for the currently loaded database
self.layout	A Vertical Box layout for the databrowser
self.table_view	A table view to display the sql table appropriately
self.database	A variable for the database in use
data	A local variable to store all the data from the current database in an array
model	A local variable to set the model of the format of the data from the sql database
row	A local variable initially set to 0 to increment to set items in each row
item	A local variable that stores the current item from all the data array
column	A local variable initially set to 0 to increment to set items in each column
standardItem	A local variable to set the item to the table view

PrintDialog

Variable	**Purpose**
self.form_combo_box	A combo box to hold the different forms available for the user to print

self.item_select_combo_box	A combo box to hold the different items available for the user to print
self.print_push_button	A Push Button for the user to confirm what they want to print and open a print dialog
self.layout	A Vertical Box layout for the PrintDialog
dialog	A local variable that's assigned the QPrintDialog
itemToPrint	A QTextEdit of the items to print passed into the function
columns	A local variable that stores all the column names for the member table
info	A local variable that holds the return value from a function that gets all the required data
infoToPrint	A local variable for a text edit to hold all the information being printed
infoString	A local variable that’s a string for all the data going into the text edit
item	A local variable that's assigned an item from the info array
count	A local variable that’s incremented to assign a value from the columns array to add to the infoString variable
piece	A local variable that’s assigned to to each piece of data in the item array
name	A local variable that assigns the list of the member's name
items	A local variable that assigns the items returned by the getItems() function
list	A local variable that's assigned to the list inside the name variable
word	A local variable that’s assigned to the name inside the variable list

PasswordDialog

Variable	Purpose
self.password_lineEdit	A QLineEdit for the password to be entered by the user
self.enterButton	A QPushButton for the user to click to enter the password they've input
self.closeButton	A QPushButton that allows the user to close the dialog
self.layoutMain	The main layout for the dialog box
self.layoutHorizontal	A Horizontal layout for the enter and close push buttons to be added to

ResultsDialog

Variable	Purpose
self.searchResults	An array variable for assigning the search results from the search function
self.results	A QTextEdit that stores all the results in a text format
self.layout	the layout for the results dialog
string	A Local variable which stores each item in a string format
item	A local variable that's assigned to each result from the results array

SearchDialog

Variable	Purpose
self.table_combo_box	A combo box for allowing the user to select the table they want
self.search_box	A line edit for the user to enter their search

	term
self.search_push_button	A push button for the user to confirm their search
self.layout	A vertical box layout for all the widgets to be added to
results	A local variable that assigns all the data returned by the SearchQuery function
resultsDialog	A local variable assigned to an execution of the resultsDialog imported from the gym_search_results_dialog_class file

AboutDialog

Variable	**Purpose**
self.text	A text edit to store the appropriate text to describe the program to the user
self.userManualButton	A push button allowing the user to open a link to download a pdf of the user manual
self.layout	A Vertical layout for all the widgets to be added to

AddDialog

Variable	**Purpose**
No Variables	

EditDialog

Variable	**Purpose**
No Variables	

AddEditDialog

Variable	Purpose
self.table_combo_box	A combo box to allow the user to select a table
self.add_edit_button	A push button that confirms the user's input data
self.database	A variable that's assigned the current database
self.stackedLayout	A stacked layout to assign all of the different widgets used
self.mainLayout	The main layout for all the other layouts to be added to
self.topLayout	The layout for the add_edit_button and table_combo_box to be added to
self.memberLayoutWidget	A widget that contains all the widgets for an input form for a member input
self.paymentLayoutWidget	A widget that contains all the widgets for an input form for a payment input
self.regimeLayoutWidget	A widget that contains all the widgets for an input form for a regime input
self.exerciseLayoutWidget	A widget that contains all the widgets for an input form for an exercise input

Database

Variable	Purpose
self.db_name	A variable for assigning the name of the opened database
db	A local variable for assigning the open database
cursor	A local variable for assigning the cursor for the sql statements
allData	A local variable for assigning all the data from the database

query	A local variable for assigning the query
data	A local variable to assign an array of data in a for loop
item	A local variable to assign an item from the data array in a for loop

DeleteDialog

Variable	**Purpose**
self.table_select_combo_box	A Combo box allowing the user to select a table
self.item_select_combo_box	A combo box allowing the user to select an item
self.delete_push_button	A push button to confirm the users wants to delete a piece of data
self.delete_all_push_button	A push button for the user the delete all data in a table
self.layout	A variable for assigning the dialogs Vertical layout
items	A local variable for assigning an array of all the items from the selected table
count	A local variable to be incremented in a for loop
sep	A local variable assigned to a string that indicates where an item should be split to be passed as a parameter into a function

AppWindow

Variable	**Purpose**
self.file	A variable that stores the opened database file
self.open_push_button	A push button that lets the user start the open function

self.add_push_button	A push button that lets the user start the add function
self.edit_push_button	A push button that lets the user start the edit function
self.delete_push_button	A push button that lets the user start the delete function
self.search_push_button	A push button that lets the user start the search function
self.print_push_button	A push button that lets the user start the print function
self.tab_bar	A tab bar that allows the user to select a table to view
self.tabs	A variable that assigns the tabs
self.tabNames	A variable that assigns an array for the tab names
count	A local variable that increments from 1 in a for loop
self.labels	A variable to assign an array for the labels
self.toolBar	A toolbar that contains file and help options
self.file_menu	The file menu in the toolbar
self.help_menu	The help menu in the toolbar
self.about	The about option from the help menu
self.open_shortcut	The open option from the file menu
self.add_shortcut	The add option from the file menu
self.edit_shortcut	The edit option from the file menu
self.delete_shortcut	The delete option from the file menu
self.search_shortcut	The search option from the file menu
self.print_shortcut	The print option from the file menu
self.layout1	The main vertical layout for main window

self.layout2	The second layout for the main window
self.layout3	The third layout for the main window
self.mainWidget	The main widget for the app window
delete_dialog	A local variable that assigns a delete dialog
print_dialog	A local variable that assigns a print dialog
search_dialog	A local variable that assigns a search dialog
edit_dialog	A local variable that assigns an edit dialog
add_dialog	A local variable that assigns an add dialog
about_dialog	A local variable that assigns an about dialog
passWord	A local variable that assigns what the current password for the current password dialog
currentPass	A local variable that assigns the return from the print_dialog that returns what the user enters into the password dialog
password_dialog	A local variable that assigns a password dialog
gym_program	A local variable that assigns the main execution of the program
gym_window	A local variable that assigns the main window of the program

EXPLANATION OF DETAILED ALGORITHMS

Main Window Class

```
1.  import sys
2.
3.  from gym_delete_dialog_class import *
4.  from gym_print_dialog_class import *
5.  from gym_search_dialog_class import *
6.  from gym_edit_dialog_class import *
7.  from gym_add_edit_dialog_class import *
8.  from gym_add_dialog_class import *
9.  from gym_about_dialog_class import *
10. from gym_password_dialog_class import *
11.
12. from PyQt4.QtCore import *
13. from PyQt4.QtGui import *
14.
15. class AppWindow(QMainWindow):
16.     """creates the main window"""
17.
18.     #constructor
19.     def __init__(self):
20.         super().__init__()
21.         self.setWindowTitle("Gym Database Management System")#sets the title for the window
22.         self.setWindowIcon(QIcon("Logo.png"))#sets the window icon
23.
24.         #variable for database
25.         self.file = None
26.
27.         #toolbars
28.         self.open_push_button = QPushButton("Open")#sets the main button for opening the Open GUI and
    function
29.         self.add_push_button = QPushButton("Add")#sets the main button for opening the Add GUI and function
30.         self.edit_push_button = QPushButton("Edit")#sets the main button for opening the Edit GUI and function
31.         self.delete_push_button = QPushButton("Delete")#sets the main button for opening the Delete GUI and
    function
```

```
32.         self.search_push_button = QPushButton("Search")#sets the main button for opening the Search GUI and
    function
33.         self.print_push_button = QPushButton("Print")#sets the main button for opening the Print GUI and
    function
34.
35.         self.tab_bar = QTabWidget() #creates the widget to display the tables in a tabular layout
36.
37.         self.tabs = {}#creates a dictionary for the table names/tab names
38.         self.tabNames = ['Members','Payments','Regime','Exercise']#The 4 tab/table names
39.         for count in range(4):
40.             self.tabs["{0}".format(self.tabNames[count])] = BrowseDataWidget()
41.
    self.tab_bar.addTab(self.tabs["{0}".format(self.tabNames[count])],"{0}".format(self.tabNames[count]))
42.         #a for loop that creates a new tab and adds a "BrowseDataWidget" into each of them
43.
44.         self.labels = {"Members":["MemberID","Name","Address","Telephone Number","Membership
    Type","Induction Date","Join Date","How Paid", "Amount", "RegistrationFee","Registration
    Date","PaymentType","Comments"],
45.                        "Payments":["MemberID","Payment Date","How Much","Paid"],
46.                        "Regime":["MemberID","ExerciseID","Specific Description", "Start Date", "End Date"],
47.                        "Exercise":["ExerciseID","Name","Description"]}
48.         #labels for the table headers that are referenced later in the program
49.
50.         self.toolBar = QMenuBar()#creates a menu bar
51.         self.file_menu = self.toolBar.addMenu("File")#adds File option to the tool bar
52.         self.help_menu = self.toolBar.addMenu("Help")#adds Help option to the tool bar
53.         self.about = self.help_menu.addAction("About Gym Database Management System 9001")#adds options
    into the Help menu
54.         self.open_shortcut = self.file_menu.addAction("Open")#adds Open shortcut to the File menu
55.         self.add_shortcut = self.file_menu.addAction("Add")#adds Add shortcut to the File menu
56.         self.edit_shortcut = self.file_menu.addAction("Edit")#adds Edit shortcut to the File menu
57.         self.delete_shortcut = self.file_menu.addAction("Delete")#adds Delete shortcut to the File menu
58.         self.search_shortcut = self.file_menu.addAction("Search")#adds Search shortcut to the File menu
59.         self.print_shortcut = self.file_menu.addAction("Print")#adds Print shortcut to the File menu
60.
61.
62.         #layout
63.         self.layout1 = QHBoxLayout()
64.         self.layout2 = QHBoxLayout()
```

```
65.         self.layout3 = QVBoxLayout()
66.         self.layout1.addWidget(self.open_push_button)
67.         self.layout1.addWidget(self.add_push_button)
68.         self.layout1.addWidget(self.edit_push_button)
69.
70.         self.layout2.addWidget(self.delete_push_button)
71.         self.layout2.addWidget(self.search_push_button)
72.         self.layout2.addWidget(self.print_push_button)
73.
74.         self.layout3.addWidget(self.tab_bar)
75.         self.layout3.addLayout(self.layout1)
76.         self.layout3.addLayout(self.layout2)
77.
78.         self.mainWidget = QWidget()
79.         self.setMenuWidget(self.toolBar)
80.         self.mainWidget.setLayout(self.layout3)
81.         self.setCentralWidget(self.mainWidget)
82.
83.         #connections
84.         #connections linking the main pushbuttons to their respective functions
85.         self.open_push_button.clicked.connect(self.open_file_menu)
86.         self.delete_push_button.clicked.connect(self.delete)
87.         self.print_push_button.clicked.connect(self.print_stuff)
88.         self.search_push_button.clicked.connect(self.search)
89.         self.edit_push_button.clicked.connect(self.edit)
90.         self.add_push_button.clicked.connect(self.add)
91.         self.about.triggered.connect(self.about_the_program)
92.         self.open_shortcut.triggered.connect(self.open_file_menu)
93.         self.add_shortcut.triggered.connect(self.add)
94.         self.edit_shortcut.triggered.connect(self.edit)
95.         self.delete_shortcut.triggered.connect(self.delete)
96.         self.search_shortcut.triggered.connect(self.search)
97.         self.print_shortcut.triggered.connect(self.print_stuff)
98.         #Keyboard Shortcuts for accessing the 6 main Functions
99.         self.connect(QShortcut(QKeySequence("Ctrl+o"), self), SIGNAL('activated()'), self.open_file_menu)
100.        self.connect(QShortcut(QKeySequence("Ctrl+a"), self), SIGNAL('activated()'), self.add)
101.        self.connect(QShortcut(QKeySequence("Ctrl+e"), self), SIGNAL('activated()'), self.edit)
102.        self.connect(QShortcut(QKeySequence("Ctrl+d"), self), SIGNAL('activated()'), self.delete)
103.        self.connect(QShortcut(QKeySequence("Ctrl+s"), self), SIGNAL('activated()'), self.search)
```

```
104.        self.connect(QShortcut(QKeySequence("Ctrl+p"), self), SIGNAL('activated()'), self.print_stuff)
105.        self.connect(QShortcut(QKeySequence("Ctrl+h"), self), SIGNAL('activated()'), self.about_the_program)
106.
107.
108.
109.    def open_file_menu(self):
110.        self.file = QFileDialog.getOpenFileName(caption="Open Database",filter = "Database file (*.db
    *.dat)")#opens the windows folder tree allowing the user to select the database they want to open. File type
    restricted to .db or .dat files
111.        try: #Restricts the user to only opening correct database or a version of it
112.            for item in self.tabNames:
113.                self.tabs["{0}".format(item)].UpdateTable(self.file)#loads selected database into tabs
114.                self.tabs["{0}".format(item)].PopulateTable(item, self.labels[item])#populates the tabs with
    relevent information
115.        except NameError:
116.            return
117.        except sqlite3.OperationalError:
118.            return
119.
120.
121.    def delete(self):
122.        self.password("niel")#sets delete password to "niel" and opens the password function
123.        delete_dialog = DeleteDialog(self.file)#opens delete dialog
124.        delete_dialog.exec_()
125.
126.    def print_stuff(self):
127.        print_dialog = PrintDialog(self.file)#opens print dialog
128.        print_dialog.exec_()
129.
130.    def search(self):
131.        search_dialog = SearchDialog(self.file)#opens search dialog
132.        search_dialog.exec_()
133.
134.    def edit(self):
135.        edit_dialog = EditDialog(self.file)#opens edit dialog
136.        edit_dialog.exec_()
137.
138.    def add(self):
139.        add_dialog = AddDialog(self.file)#opens add dialog
```

```
140.        add_dialog.exec_()
141.
142.    def about_the_program(self):
143.        about_dialog = AboutDialog()#opens about dialog
144.        about_dialog.exec_()
145.
146.    def password(self,currentPass):
147.        passWord = ""#sets entered password to the wrong password
148.        while passWord != currentPass:#while the correct password hasn't been entered
149.            password_dialog = PasswordDialog()#opens password dialog
150.            password_dialog.exec_()
151.            passWord = password_dialog.close_method()
152.
153.
154.def main():
155.    gym_program = QApplication(sys.argv)#creates application
156.    gym_window = AppWindow()#creates Main Window
157.    gym_window.resize(700,600)#Locks inital window size
158.    password_dialog = gym_window.password("a")
159.    gym_window.show()
160.    gym_window.raise_()
161.    gym_program.exec_()
162.
163.
164.if __name__ == "__main__":
165.    main()
```

This is the main window class for the main interface and the centre of the entire program. Here I use several functions to allow the user to start each of the main functions and several layouts to appropriately assign the buttons, the toolbar and the Table View. In this code the class inherits QMainWindow instead of QDialog since it's the main gui.

Print Dialog Class

```
1.  from PyQt4.QtGui import *
2.  from PyQt4.QtCore import *
3.  from gym_print_sql_function import *
4.  from gym_delete_function import *
5.
6.  class PrintDialog(QDialog):
```

```
7.      """This class creatres the dialog window for creating forms and printing them"""
8.
9.      def __init__(self,database):
10.         super().__init__()
11.
12.         self.database = database
13.
14.         #create widgets
15.         self.form_combo_box = QComboBox()
16.         self.item_select_combo_box = QComboBox()
17.         self.print_push_button = QPushButton("Print")
18.
19.         self.form_combo_box.addItem("Select Form")
20.         self.form_combo_box.addItem("Invoice")
21.         self.form_combo_box.addItem("Member Details")
22.         self.form_combo_box.addItem("Regime")
23.
24.
25.         #create layout
26.         self.layout = QVBoxLayout()
27.
28.         #add widgets to layout
29.         self.layout.addWidget(self.form_combo_box)
30.         self.layout.addWidget(self.item_select_combo_box)
31.         self.layout.addWidget(self.print_push_button)
32.
33.         #set the window layout
34.         self.setLayout(self.layout)
35.         self.setWindowTitle("Print")
36.         self.setWindowIcon(QIcon("Hugobells.png"))
37.
38.         #connections
39.         self.form_combo_box.currentIndexChanged.connect(self.itemComboBoxPopulate)
40.         self.print_push_button.clicked.connect(self.printInfo)
41.
42.     def printFunction(self,itemToPrint):
43.             #function that sends the textEdit to be printed and allows the user to select a printer through the print dialog
44.         dialog = QPrintDialog()
45.         if dialog.exec_() == QDialog.Accepted:
```

```
            itemToPrint.print_(dialog.printer())

    def generateMemberInfo(self):
                #method that creates the textEdit containg the correct information for a member info print out
        columns
=["MemberID","Name","Address","TelephoneNumber","MembershipType","InductionDate","JoinDate","HowPaid","Amount","
RegistrationFee","RegistrationDate","PaymentType","Comments"]
        sep = "."
        info = getMemberInfo(self.database, str(self.item_select_combo_box.currentText()).split(sep,1)[0])
        infoToPrint = QTextEdit()
        infoString = ""
        for item in info:
            count = 0
            for piece in item:
                infoToPrint.setText(infoToPrint.toPlainText()+columns[count]+" : "+str(piece)+"\n")
                count += 1
        self.printFunction(infoToPrint)

    def generateInvoice(self):
                #method that creates the textEdit containg the correct information for a print out of an invoice
        columns = ["MemberID","Payment Date","How Much","Paid"]
        info,name = getInvoice(self.database, str(self.item_select_combo_box.currentText()).split(sep,1)[0])
        infoToPrint = QTextEdit()
        for list in name:
            for word in name:
                for item in word:
                    infoToPrint.setText("Member Name : "+str(item)+"\n\n")
        for item in info:
            count = 0
            for piece in item:
                infoToPrint.setText(infoToPrint.toPlainText()+columns[count]+" : "+str(piece)+"\n")
                count += 1
            infoToPrint.setText(infoToPrint.toPlainText()+"\n")
        self.printFunction(infoToPrint)

    def generateRegime(self):
                #method that creates the textEdit containg the correct information for a members regime print out
        columns = ["MemberID","ExerciseID","Description","Start Date","End Date"]
        info,name = getRegime(self.database, str(self.item_select_combo_box.currentText()).split(sep,1)[0])
```

```
        infoToPrint = QTextEdit()
        for list in name:
            for word in name:
                for item in word:
                    infoToPrint.setText("Member Name : "+str(item)+"\n\n")
        for item in info:
            count = 0
            for piece in item:
                infoToPrint.setText(infoToPrint.toPlainText()+columns[count]+" : "+str(piece)+"\n")
                count += 1
            infoToPrint.setText(infoToPrint.toPlainText()+"\n")
        self.printFunction(infoToPrint)

    def itemComboBoxPopulate(self):
        if self.form_combo_box.currentIndex() == 0:
            self.item_select_combo_box.clear()
            self.item_select_combo_box.addItem("Select Item")
        else:
            items = getItems(self.database, "MEMBERS")
            self.item_select_combo_box.clear()
            for count in range(len(items)):
                self.item_select_combo_box.addItem(items[count])

    def printInfo(self):
        if self.form_combo_box.currentIndex() == 1:
            self.generateInvoice()
        if self.form_combo_box.currentIndex() == 2:
            self.generateMemberInfo()
        if self.form_combo_box.currentIndex() == 3:
            self.generateRegime()
```

This code is for the print dialog. This uses 3 methods for getting and organising the information for the 3 different printable forms as well as another method for actually printing the form. These methods inherit multiple functions from the gym_print_sql_function.

Gym Edit Function

```
import sqlite3

def editMember(database,items,memberID):
    #function for editing items in the member table
    con = sqlite3.connect(database)

    with con:

        sql = "UPDATE Members SET "+items+" WHERE MemberID = "+memberID
        cur = con.cursor()
        cur.execute(sql)

def editPayment(database,items,memberID,paymentDate):
    #function for editing items in the payment table
    con = sqlite3.connect(database)

    with con:
        sql = "UPDATE Payments SET "+items+" WHERE MemberID="+memberID+" AND PaymentDate = "+paymentDate
        cur = con.cursor()
        cur.execute(sql)

def editRegime(database,items,memberID,exerciseID):
    #function for editing items in the regimetable
    con = sqlite3.connect(database)

    with con:
        sql = "UPDATE Regime SET "+items+" WHERE MemberID = "+memberID+" AND ExerciseID = "+exerciseID
        cur = con.cursor()
        cur.execute(sql)

def editExercise(database,items,exerciseID):
    #function for editing items in the exercise table
    con = sqlite3.connect(database)

    with con:
        sql = "UPDATE Exercise SET "+items+"WHERE EXERCISEID = "+exerciseID
        cur = con.cursor()
        cur.execute(sql)
```

This code is the sql for the Gym Edit Function. There is 4 different functions for each table.

All of them update their respective tables by passing in a string of items made out of the

inputs from the Gym Edit Dialog where their respective primary key(s) are the same as another parameter(s).