

System Maintenance

1. Environment

1.1 Software

This is a complete list of the software I used when creating my program:

- Python 3 (the programming language)
- IDLE (to write Python scripts)
- PyQt 4 (to produce user interface elements)
- SQLite 3 (for databases)
- CX_Freeze (for compiling my program)
- Microsoft Paint (for creating graphics)
- Adobe Fireworks CS5 (for adding transparency to graphics)

1.2 Usage Explanation

Python 3:

- The language I'm most familiar and practiced in
- Python is open source, meaning it's free to use however I like so there are no restrictions, including for commercial use.
- Compatible with all systems, so should my client change system at any point, the program can still be used.
- There is a vast number of pre-existing modules in library, meaning I was able to use a lot of modules from that
- It's an object oriented programming language

IDLE:

- Designed especially for python scripting
- Easy to use as there is no complicated interface or obscure features
- Colour coded text and automatic indentation to make programming easier
- Has a debug feature making testing easier

PyQ 4t:

- Designed specifically for implementing GUI in python

SQLite 3

- More simplified version of MySQL, making it easier to use as there are fewer features

- Does not require a log in or password. This is fine as my client asked for all areas of the program to be accessible by anyone (see section 1.4 in Project Analysis). It also makes the building and testing of the system easier and faster as you do not have to log in each time
- Compatible with Python 3

CX_Freeze:

- Used for compiling python files
- Compatible with Python 3, unlike most python compilers
- Compiles all modules used with the program, so nothing needed to install and use the resulting program
- Includes the option to include other files when creating the compiled program.

Microsoft Paint

- Easy to create the simply graphics I wanted
- Do not have to export images
- Good for creating low detailed, small graphics

Adobe Fireworks CS5

- Wand tool allows for simple selection and deletion of areas, making image transparency easy to create
- Image can be saved straight over original, unlike with Photoshop which requires exporting the image first.

1.3 Features Used

Python 3:

- This is the language I used to write my program so naturally it is the main basis for it. Specifically I used the built in random module for generating some random numbers in my code in a specified range.

IDLE:

- I used this to write my python script in and run it
- I used the colour coded text and automatic indentation to make programming easier

PyQt 4:

- I used this to create the GUI for my program. I utilized the QtCore and QtGui modules. Specifically, I used the module for creating planes (with QRectF) and for specifying the size of objects, ect. (with QSize).
- I used the QtGui modules for creating all my widgets and windows, as well as the graphical displays. The menu bar was also created using this module (QMenuBar).

SQLite 3:

- I used the SQLite syntax for creating, reading and writing to and from my database.

CX_Freeze:

- I used this to compile all my program files into a Windows installer file.
- I made use of the ability to compile other files, specifically image files, along with the program files.

Microsoft Paint:

- I used this to create the graphics for my program that I read from image files.

Adobe Fireworks CS5:

- I used this simply for the wand tool to make my images I made in Paint transparent.

2. System Overview

2.1 The Graphic Display

The graphic display is made up of a visual representation of the fish and plants present in the pond, a text box containing the total number and type of the plants and fish currently in the pond, a text box containing the status of the pump and filter, and finally a visual representation of the pond's water level and temperature.

All the data required for the graphic displayed is read from the database after the user has either created a new pond or selected the pond they wish to open from the 'open pond' window. The data required for the display is as followed: pond water level, pond temperature, pump status of pump matching pump ID in pond, filter status of filter matching filter ID in pond, a list of fish with pond ID matching the ID of the pond, a list of plants with pond ID matching the ID of the pond.

2.2 Manipulating the Pond Object

In order for the program to be useful, the user must be able to manipulate the pond object. In order to create a new pond, the user must select the "New Pond" option in the file menu. From there, the information is filled out and the pond is saved to the database as well as opening in the program (visible in the graphic display).

An existing pond may be opened by clicking the "open pond" option. From here, the user must select a pond from the list by double clicking on it, and the pond's information will be read into the program from the database.

An existing pond's settings may be changed by selecting the "Edit pond settings" option from the manage pond menu. After the information has been inputted by the user in the same way as making a new pond, it's saved to the pond attributes with a series of setter functions. The database is then updated with an update function, writing over the record.

2.3 Adding and removing fish and plants

Fish and plants can be added and removed from the pond at any time. There are 4 different types of plants – rooted floating plant, true floating plant, emergent plant and submersed plant – and 6 different types of fish – goldfish, tench, orfe, koi, rudd and sharks. All fish types have different growth rates, food requirements and starting sizes. All plant types have different growth rates and light needs.

To add a fish, the user must select "Add fish" from the manage pond life menu in the manage pond menu, then select the type of fish they wish to add from the drop down menu and enter a name. The fish will be stored in the database and a graphical representation is added to the pond graphic.

To remove a fish, the user must select "remove fish" from the manage pond life menu in the manage pond menu, then select the fish they wish to remove by double clicking on an item from the displayed list.

To add a plant to the pond, the user must select "Add plant" from the manage pond life menu in the manage pond menu, then select the type of plant they wish to add from the drop down menu. The plant will be stored in the database and a graphical representation is added to the pond graphic.

To remove a plant, the user has to select "remove plant" from the manage pond life menu in the manage pond menu, then double click on the plant they wish to from the list provided.

2.4 Feeding fish

Fish need to be fed in order for them to live and grow. There are two ways to feed fish in the system. The first way is manually. Manual feeding asks the user for an amount of food they'd like to feed the fish with. It then feeds all the fish in the pond using that amount of food. If there's not enough to feed all the fish, some fish will remain hungry. Feeding the fish manually increments the day by 1.

The second way to feed fish in the pond is by automatically feed them. Automatically feeding calculates the exact amount of food the fish need and feeds them for 7 days.

3. Code Structure

3.1 Function – openWindow()

```
def openWindow():
    #when pond is opened
    def onOpen():
        self.window.close() #hides window
        self.selected=self.window.tree.currentItem()
        pondID=self.selected.data(0,0)
        depth=self.selected.data(1,0)
        length=self.selected.data(2,0)
        width=self.selected.data(3,0)
        temp=self.selected.data(4,0)
        level=self.selected.data(5,0)
        pump=self.selected.data(6,0)
        Filter=self.selected.data(7,0)
        days=self.selected.data(8,0)

    self.mainPond=Pond(pondID=pondID,depth=depth,width=width,temp=temp,waterLevel=level,pump=pump,filter=Filter,days=days, length=length)
    self.createUI()

def tree():
    self.window.tree.setHeaderItem(self.window.header) #setting header
    results=pondSimDatabase.getPonds(db,cursor)
    for count in range(0,len(results)):
        self.window.treeList=QTreeWidgetItem() #list items
        self.window.treeList.setText(0,str(results[count][0]))
        self.window.treeList.setText(1,str(results[count][1]))
        self.window.treeList.setText(2,str(results[count][2]))
        self.window.treeList.setText(3,str(results[count][3]))
        self.window.treeList.setText(4,str(results[count][4]))
        self.window.treeList.setText(5,str(results[count][5]))
        self.window.treeList.setText(6,str(results[count][6]))
        self.window.treeList.setText(7,str(results[count][7]))
        self.window.treeList.setText(8,str(results[count][8]))
        self.window.tree.addTopLevelItem(self.window.treeList)

self.window=QWidget()
self.window.setWindowTitle("Pond Simulator 3000 - Open Pond")
self.window.setFixedWidth(600)
self.window.setFixedHeight(400)
self.window.show()
self.window.layout = QVBoxLayout()
self.window.setLayout(self.window.layout)

self.window.tree=QTreeWidgetItem()
self.window.tree.setColumnCount(9)
self.window.tree.setColumnWidth(0,80)

self.window.header=QTreeWidgetItem() #headers
self.window.header.setText(0,'PondID')
self.window.header.setText(1,'Pond Depth')
```

```

self.window.header.setText(2, 'Pond Length')
self.window.header.setText(3, 'Pond Width')
self.window.header.setText(4, 'Water Temp.')
self.window.header.setText(5, 'Water Level')
self.window.header.setText(6, 'PumpID')
self.window.header.setText(7, 'FilterID')
self.window.header.setText(8, 'Days')
tree()

self.window.button=QPushButton("Cancel") #making button
self.window.button.setFixedWidth(100) #changing button size
self.window.button.layout=QHBoxLayout() #creating layout for button
self.window.buttonWidget=QWidget() #creating blank widget
self.window.blank=QWidget()
self.window.buttonWidget.setLayout(self.window.button.layout) #setting
layout to blank widget
self.window.button.layout.addWidget(self.window.blank) #adding button
layout
self.window.button.layout.addWidget(self.window.button) #adding button
layout

self.window.button.clicked.connect(self.window.close) #when button is
clicked, window is hidden
self.window.layout.addWidget(self.window.tree) #adding tree list to layout
self.window.layout.addWidget(self.window.buttonWidget) #adding button to
layout
self.window.tree.itemDoubleClicked.connect(onOpen) #when item double clicked

```

In reference to lines 47 to 116 of the GUI in the section appendix, `openWindow()` is a function that is called when the user selects the “Open Pond” option from the menu bar. Two functions are included within the function (`onOpen()` and `tree()`).

It was necessary make this section of the code a function as it can be called repeatedly throughout the running of the program by the user. Similarly, `onOpen()` is only called when the user has selected a pond to open.

Looking at the `tree()` function, I think there’s no real reason for the section of code to be a function but I decided to separate it to make it easier to distinguish and understand.

3.2 Function – savePond()

```

def savePond():
    try:
        self.mainPond.updatePond(db,cursor)
    except:
        self.errorWindow=QWidget()
        self.errorWindow.setWindowTitle("Error")
        self.errorWindow.setFixedWidth(250)
        self.errorWindow.setFixedHeight(150)
        self.errorWindow.show()
        self.errorWindow.layout = QVBoxLayout()
        self.errorWindow.setLayout(self.errorWindow.layout)
        self.errorWindow.label=QLabel("You must open a pond before it can
be saved")
        self.errorWindow.label.setWordWrap(True)
        self.errorWindow.widget=QWidget()
        self.errorWindow.widgetLayout=QGridLayout()
        self.errorWindow.widget.setLayout(self.errorWindow.widgetLayout)

```

```

self.errorWindow.okay=QPushButton("OK")
self.errorWindow.okay.setFixedWidth(50)
self.errorWindow.okay.clicked.connect(self.errorWindow.close)
blank=QWidget()
blank.setFixedWidth(100)
blankTwo=QWidget()
blankTwo.setFixedWidth(100)
self.errorWindow.widgetLayout.addWidget(blank,0,2)
self.errorWindow.widgetLayout.addWidget(blankTwo,0,0)
self.errorWindow.widgetLayout.addWidget(self.errorWindow.okay,0,1)
self.errorWindow.layout.addWidget(self.errorWindow.label)
self.errorWindow.layout.addWidget(self.errorWindow.widget)

```

In reference to lines 18 to 45 in the GUI section of the appendix, the `savePond()` function is called when a user selects to save the pond from the file menu on the menu bar. I made this a function for 2 reasons – the first being that parameters cannot be given to a function acting in response to an event (see line 118 of the GUI section – `savePond()` is given to the trigger as an event but cannot contain parameters), the second reason being I was able to add an exception error to display an error message to the user should they try to save a pond when one is not open.

3.3 Function – saveFishTypes()

```

def saveFishTypes(db,cursor):
    newGold=Goldfish(db,cursor)
    newTench=Tench(db,cursor)
    newOrfe=Orfe(db,cursor)
    newKoi=Koi(db,cursor)
    newRudd=Rudd(db,cursor)
    newShark=Shark(db,cursor)
    try:
        newGold.save(db, cursor)
        newTench.save(db, cursor)
        newOrfe.save(db, cursor)
        newKoi.save(db, cursor)
        newRudd.save(db, cursor)
        newShark.save(db, cursor)
    except:
        pass

```

In reference to lines 171 to 186 in the fish module in the section appendix, this function is required to save the allowed fish types to the database when the program is opened.

I made this section of a code a function so that it can be accessed by the main module, GUI, when the program is first run and the database is created.

3.4 Class – Fish

```

class Fish:
    """A simulation of a generic fish"""
    def __init__(self, db, cursor):
        self._fishName=None
        self._fishType="fish"
        self._foodNeed=0
        self._growthRate=0

```

```
        self._fishSize=0
        self._hunger=0
        self._status="good"
        self._daysAlive=0
        self._growth=0
        self._maxDaysAlive=100
        self._fishID=None

    def getFishID(self):
        return self._fishID

    def setFishName(self,x):
        self._fishName=x

    def getFishName(self):
        return self._fishName

    def setFishType(self,x):
        self._fishType=x

    def getFishType(self):
        return self._fishType

    def setFoodNeed(self,x):
        self._foodNeed=x

    def getFoodNeed(self):
        return self._foodNeed

    def setGrowthRate(self,x):
        self._growthRate=x

    def getGrowthRate(self):
        return self._growthRate

    def setFishSize(self,x):
        self._setFishSize=x

    def getFishSize(self):
        return self._fishSize

    def setHunger(self,x):
        self._hunger=x

    def getHunger(self):
        return self._hunger

    def setStatus(self,growth,daysAlive):
        pass

    def getStatus(self):
        return self._status

    def setDaysAlive(self,x):
        self._daysAlive=x

    def getDaysAlive(self):
        return self._daysAlive

    def setGrowth(self,x):
        self._growth=x
```



```
def getGrowth(self):
    return self._growth

def grow(self, food):
    pass

def saveFish(self, pondID, db, cursor):
    sql = """insert into fish(fishName, hunger, status, daysAlive,
growth, fishType, pondID) values
    (
'{0}','{1}','{2}','{3}','{4}','{5}','{6}')""".format(self._fishName,
self._hunger, self._status, self._daysAlive, self._growth, self._fishType,
pondID)
    cursor.execute(sql)
    cursor.execute("""select last_insert_rowid()""")
    self._fishID = cursor.fetchall()[0][0]
    db.commit()
```

In reference to lines 3 to 84 in the fish module in the section appendix, this is the base class for all the different fish objects. There are 6 different types of fish that can be added to my pond, so there are 6 corresponding fish objects. I made a base fish class to be inherited by the other fish classes in order to stop unnecessary duplication of code, as all the fish class methods would need to be added to the 6 different fish classes.

Fish is a class as all fish objects need specific attributes about them to be stored, so they can be added to the database correctly.

3.5 Class – Pond

```
class Pond:
    """A simulation of a pond"""
    def __init__(self, width, length, depth, temp, waterLevel, pump,
filter, pondID, days):
        self._pondWidth=width
        self._pondLength=length
        self._pondDepth=depth
        self._waterTemp=temp
        self._waterLevel=waterLevel
        self._pump=pump
        self._filter=filter
        self._pondID=pondID
        self._days=days

    def setPondID(self,x):
        self._pondID=x

    def getPondID(self):
        return self._pondID

    def setPondWidth(self, x): #where x is the desired pond width
        self._pondWidth=x

    def getPondWidth(self):
        return self._pondWidth

    def setPondLength(self, x):
        self._pondLength=x

    def getPondLength(self):
        return self._pondLength

    def setPondDepth(self,x):
        self._pondDepth=x

    def getPondDepth(self):
        return self._pondDepth

    def setWaterLevel(self,x):
        self._waterLevel=x

    def getWaterLevel(self):
        return self._waterLevel

    def setWaterTemp(self, x):
        self._waterTemp=x

    def getWaterTemp(self):
        return self._waterTemp

    def setPump(self, x): #where x is the pump object
        self._pump=x

    def getPump(self):
        return self._pump

    def setFilter(self, x):
```

```

        self._filter=x

    def getFilter(self):
        return self._filter

    def setDays(self, x):
        self._days=x

    def getDays(self):
        return self._days

    def addFish(self, db, cursor,choice,name):
        def chooseFish(db, cursor,choice):
            if choice=="Goldfish":
                newFish=fish.Goldfish(db, cursor)
            elif choice=="Tench":
                newFish=fish.Tench(db,cursor)
            elif choice=="Orfe":
                newFish=fish.Orfe(db,cursor)
            elif choice=="Koi":
                newFish=fish.Koi(db,cursor)
            elif choice=="Rudd":
                newFish=fish.Rudd(db,cursor)
            elif choice=="Shark":
                newFish=fish.Shark(db,cursor)
            return newFish

        newFish=chooseFish(db, cursor,choice)
        newFish.setFishName(name)
        newFish.saveFish(self._pondID, db, cursor)

    def removeFish(self,db, cursor):
        pondID=pond.getPondID()
        fishTup=pondSimDatabase.fetchFish(db, cursor, pondID)
        correct=False

        #displaying list of fish in pond

    print('{0:^10}|{1:^10}|{2:^15}|{3:^15}|{4:^15}|{5:^15}|{6:^10}|'.format('FishID', 'fishName', 'Hunger', 'Status', 'daysAlive', 'growth', 'FishType'))
        for each in range(0,len(fishTup)):
            print('{0:^12} {1:^10} {2:^10} {3:^20} {4:^10} {5:^20} {6:^5}'.format(fishTup[each][0],fishTup[each][1],fishTup[each][2],fishTup[each][3],fishTup[each][4],fishTup[each][5],fishTup[each][6]))
        print('')

        try:
            while correct==False:
                IDchoice=int(input('Enter the ID of the fish you wish to
remove: '))
                if IDchoice>len(fishTup)or IDchoice<0:
                    print("That ID was not found on the list")
                else:
                    correct=True
                    pondSimDatabase.deleteFish(db, cursor, IDchoice)
            except:
                print("That was not a valid ID - please enter a number")

    def removePlant(self,db,cursor):
        pondID=pond.getPondID()

```

```

plantsTup=pondSimDatabase.fetchPlants(db, cursor, pondID)
correct=False

print('{0:^10}|{1:^10}|{2:^15}|{3:^15}|{4:^15}|{5:^15}|'.format('PlantID',
'PlantType','Status','LightNeed','daysAlive','growth'))
    for each in range(0,len(plantsTup)):
        print('{0:^12} {1:^10} {2:^10} {3:^20} {4:^10}
{5:^20}'.format(plantsTup[each][0],plantsTup[each][1],plantsTup[each][2],pl
antsTup[each][3],plantsTup[each][4],plantsTup[each][5]))
    print('')

    try:
        while correct==False:
            IDchoice=int(input('Enter the ID of the plant you wish to
remove: '))
            if IDchoice>len(plantTup)or IDchoice<0:
                print("That ID was not found on the list")
            else:
                correct=True
                pondSimDatabase.deletePlant(db, cursor, IDchoice)
    except:
        print("That was not a valid ID - please enter a number")

def addPlant(self, db, cursor,choice):
def choosePlant(db,cursor,choice):
    if choice=="Rooted Floater":
        newPlant=plant.RootedFloater(db,cursor)
    elif choice=="True Floater":
        newPlant=plant.TrueFloater(db,cursor)
    elif choice=="Emergent":
        newPlant=plant.Emergent(db,cursor)
    elif choice=="Submersed":
        newPlant=plant.Submersed(db,cursor)
    return newPlant
newPlant=choosePlant(db,cursor,choice)
newPlant.savePlant(self._pondID, db, cursor)

def editPond(self):
    waterTemp=input("Water Temperature:")
    self.setWaterTemp(waterTemp)

    pondWidth=input("Pond Width:")
    self.setPondWidth(pondWidth)

    pondLength=input("Pond Length:")
    self.setPondLength(pondLength)

    pondDepth=input("Pond Depth:")
    self.setPondDepth(pondDepth)

    self.setWaterLevel(int(pondDepth)-0.5)

def savePond(self,db,cursor):
    #saving pond
    try:
        self._pumpID=self._pump.getPumpID()
    except:
        self._pumpID=None

    try:

```

```

        self._filterID=self._filter.getFilterID()
    except:
        self._filterID=None

    sql = """insert into pond(pondDepth, pondLength, pondWidth,
waterTemp, waterLevel, pumpID, filterID, days) values
        (
'{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}')""".format(self._pondDepth,
self._pondLength, self._pondWidth, self._waterTemp, self._waterLevel,
self._pump, self._filter, self._days)
    cursor.execute(sql)
    cursor.execute("""select last_insert_rowid()""")
    self._pondID = cursor.fetchall()[0][0]
    db.commit()

def updatePond(self,db,cursor):
    try:
        self._pumpID=self.getPump()
    except:
        self._pumpID=None

    try:
        self._filterID=self.getFilter()
    except:
        self._filterID=None

    try:
        tup=(self._pondDepth, self._pondLength, self._pondWidth,
self._waterTemp, self._waterLevel, self._days, self._pumpID,
self._filterID, self._pondID)
        sql = """update pond
                set pondDepth=?, pondLength=?, pondWidth=?,
waterTemp=?, waterLevel=?, days=?, pumpID=?, filterID=?
                where pondID=?"""
        cursor.execute(sql,tup)
        db.commit()
    except:
        pass

```

In reference to lines 9 to 206 in the pond module in the section, appendix, this is a class for defining and accessing ponds. Clearly this section needed to be a class as I previously identified ponds as objects which require specific attributes.

As well as the getter and setter methods, the pond class contains several methods; `updatePond()`, `savePond()`, `editPond()`, `addPlant()`, `removePlant()`, `addFish()`, `removeFish()`. These methods are all part of the pond class as they all require access to the pond object to work. The methods `addPlant()`, `removePlant()`, `addFish()`, `removeFish()` could have been including in the fish and plant modules instead, but I decided to include them with the pond module as it seemed easier to think of the fish and plants belonging to a pond.

The `incrementDay()` function (lines 433 to 520) could also have been included in the pond module.

4. Variable Listing

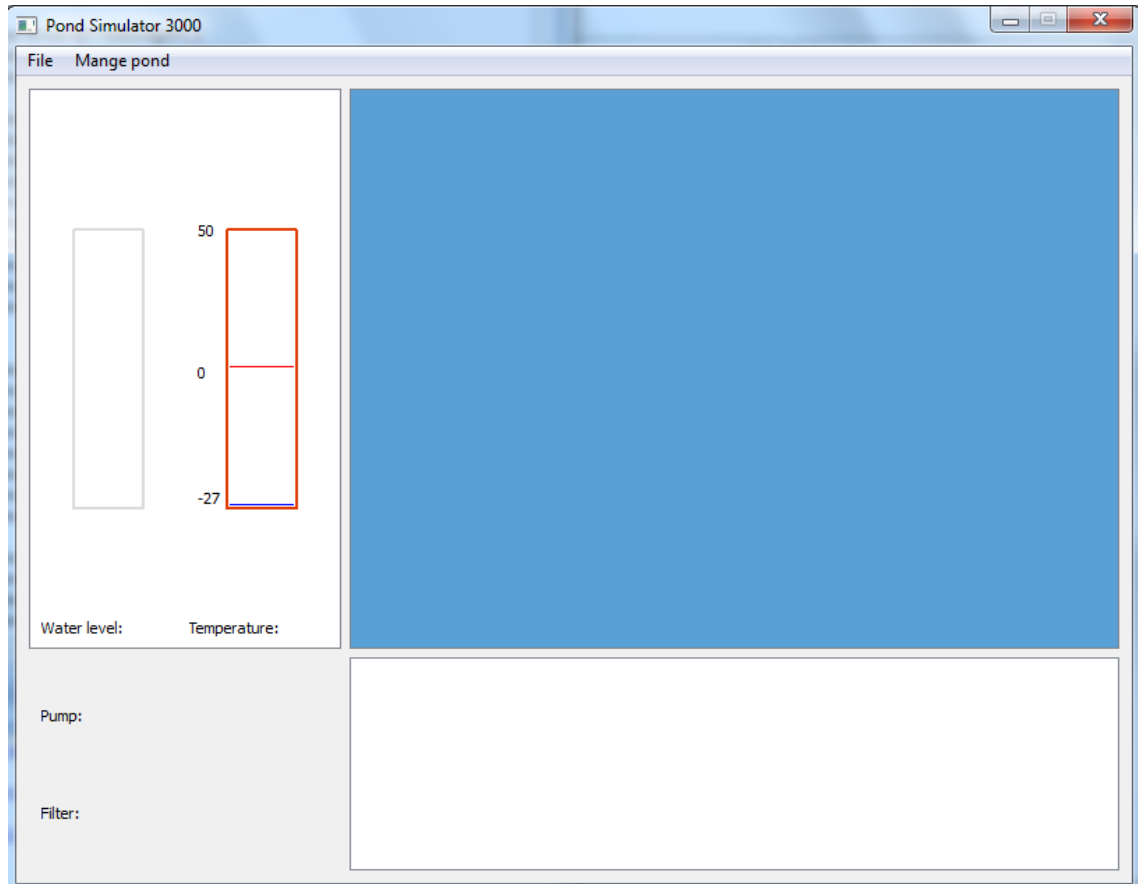
The following is a table containing a complete list of variables in my program. The data dictionary can be found in Design Section 10.4.

Variable Name	Purpose	Location in code
<code>self.errorWindow</code>	Blank widget for error message	gui.py, line 22
<code>self.errorWindow.layout</code>	Layout for blank widget	Gui.py, line 27
<code>self.errorWindow.label</code>	Label displaying error message to be added to layout	Gui.py, 29
<code>self.errorWindow.okay</code>	Push button to be added to layout under label	Gui.py, line 34
<code>self.selected</code>	Getting the selected pond to open	Gui.py, line 51
<code>pondID</code>	Getting the pond id from the selected pond	Gui.py, line 52
<code>depth</code>	Getting depth from selected pond	Gui.py, line 53
<code>length</code>	Getting length from selected pond	Gui.py, line 54
<code>width</code>	Getting width from selected pond	Gui.py, line 55
<code>temp</code>	Getting temp from selected pond	Gui.py, line 56
<code>level</code>	Getting water level from selected pond	Gui.py, line 57
<code>pump</code>	Getting pump ID from selected pond	Gui.py, line 58
<code>filter</code>	Getting filter ID from selected pond	Gui.py, line 59
<code>days</code>	Getting days from selected pond	Gui.py, line 60
<code>self.mainPond</code>	The pond currently open in the program	Gui.py, lines 61, 282
<code>results</code>	A list of all the ponds in the database	Gui.py, line 66
<code>self.window.treeList</code>	A QTreeWidgetItem for displaying the list of ponds in the open pond window	Gui.py, line 68
<code>self.window</code>	Blanket widget window	Gui.py, line 80, 119, 292, 447, 481, 611, 650, 817, 841
<code>db</code>	Allows me to connect to the database	Gui.py, line 1292
<code>cursor</code>	Allows me to execute sql methods in my code using the <code>cursor.execute()</code> method	Gui.py, line 1293
<code>application</code>	My actual application object that everything is built on	Gui.py, line 1298
<code>mainWindow</code>	The main window of my program	Gui.py, line 1299
<code>goldfishTotal</code>	The total number of	Gui.py, line 1215, 1224

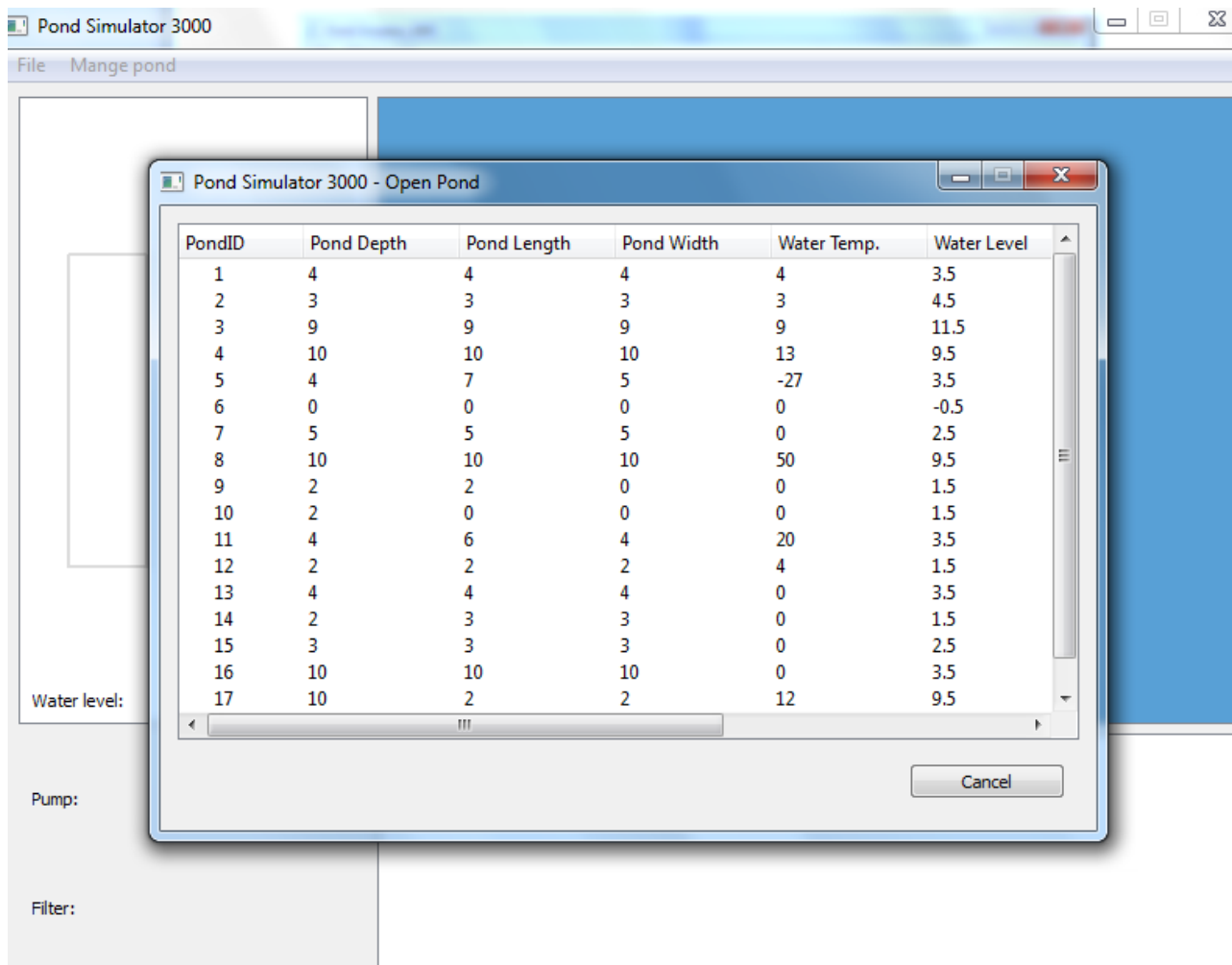
	goldfish in the pond, to be displayed to the user in the main window	
tenchTotal	The total number of tench in the pond, to be displayed to the user in the main window	Gui.py, line 1216, 1226
orfeTotal	The total number of orfe in the pond, to be displayed to the user in the main window	Gui.py, line 1217, 1228
koiTotal	The total number of koi in the pond, to be displayed to the user in the main window	Gui.py, line 1218, 1230
ruddTotal	The total number of rudd in the pond, to be displayed to the user in the main window	Gui.py, line 1219, 1232
sharkTotal	The total number of shark in the pond, to be displayed to the user in the main window	Gui.py, line 1220, 1234
emergentTotal	The total number of emergent plants in the pond, to be displayed to the user in the main window	Gui.py, line 1257, 1264
rootedTotal	The total number of rooted floating plants in the pond, to be displayed to the user in the main window	Gui.py, line 1258, 1266
trueTotal	The total number of true floating plants in the pond, to be displayed to the user in the main window	Gui.py, line 1259, 1268
submersedTotal	The total number of submersed plants in the pond, to be displayed to the user in the main window	Gui.py, line 1260, 1270

5. System Evidence

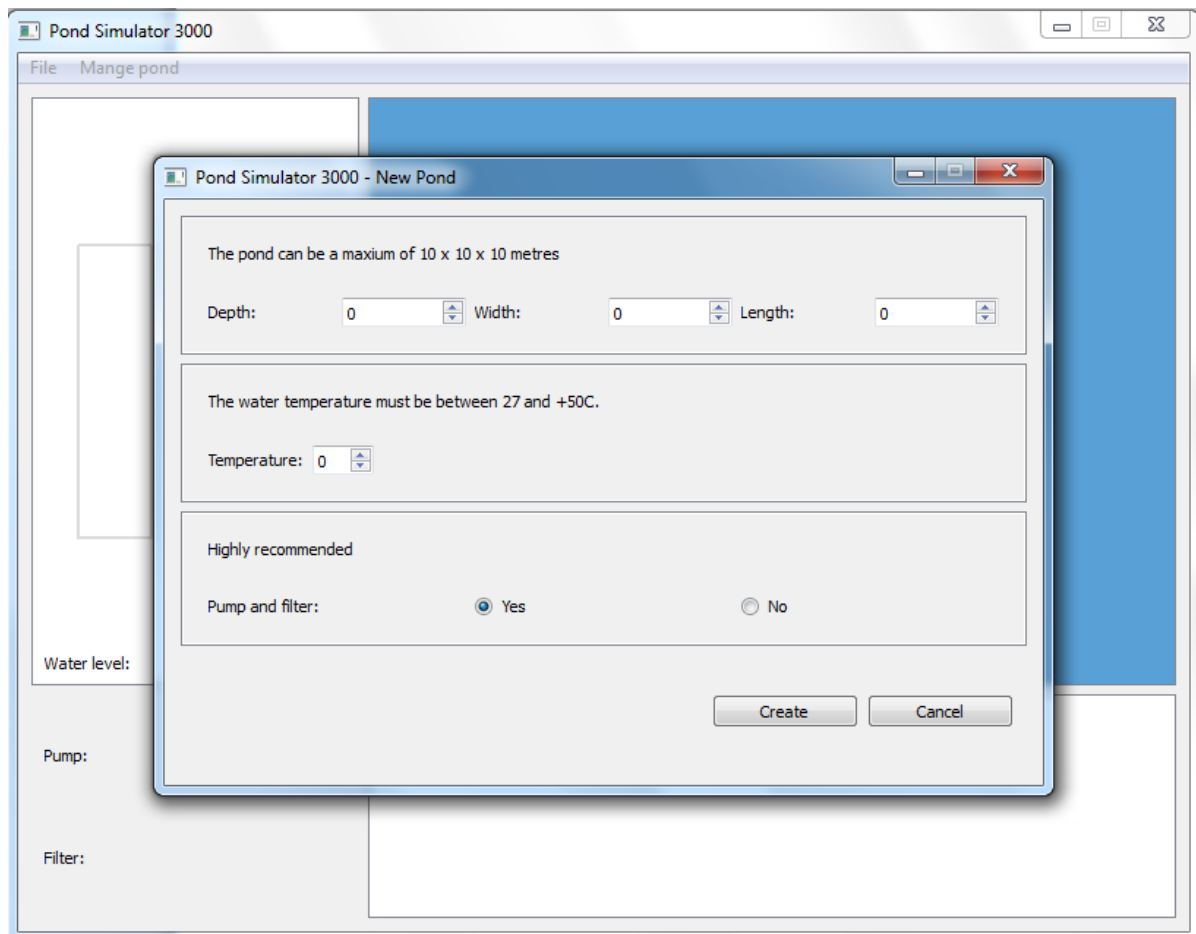
5.1 User Interface



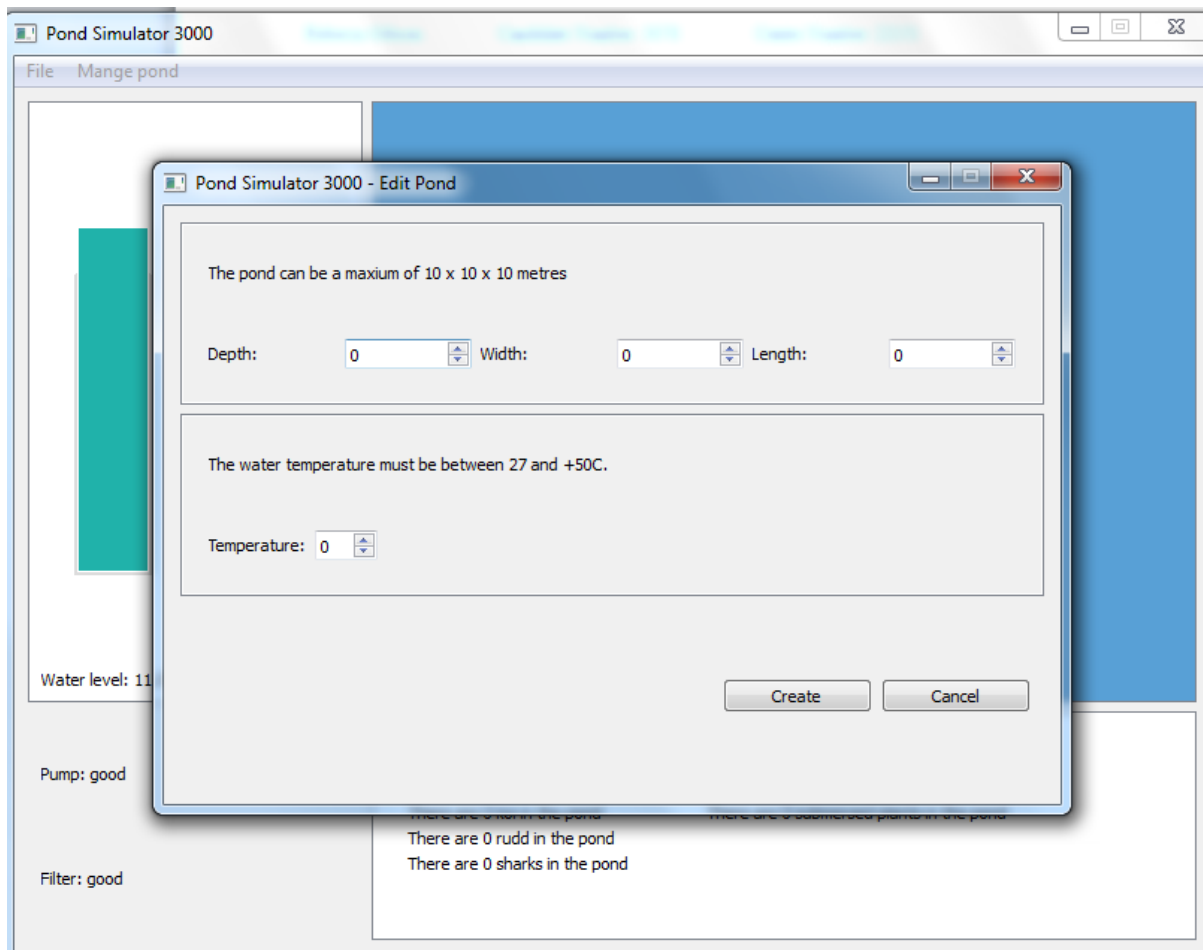
The main window of the program as it appears on opening.



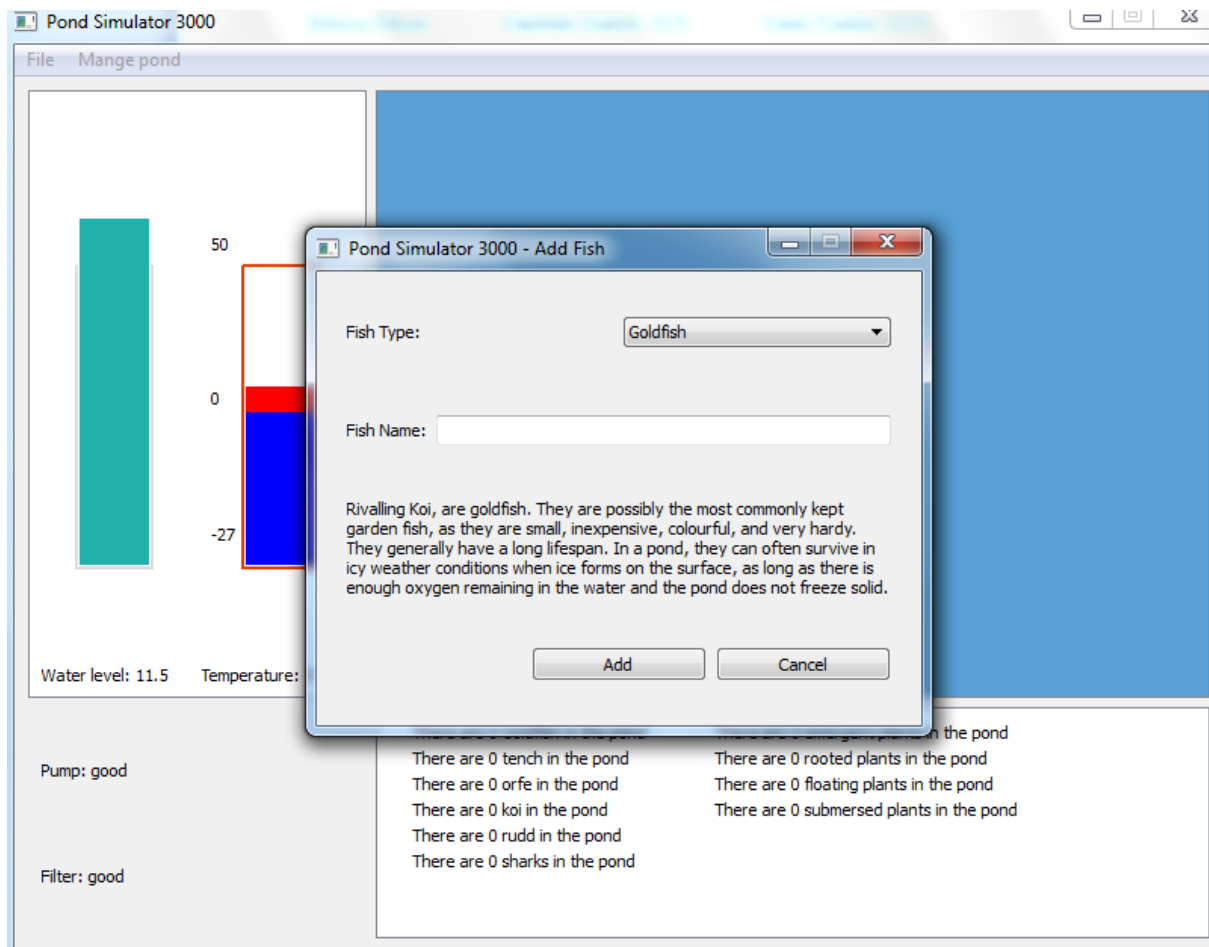
The open pond window



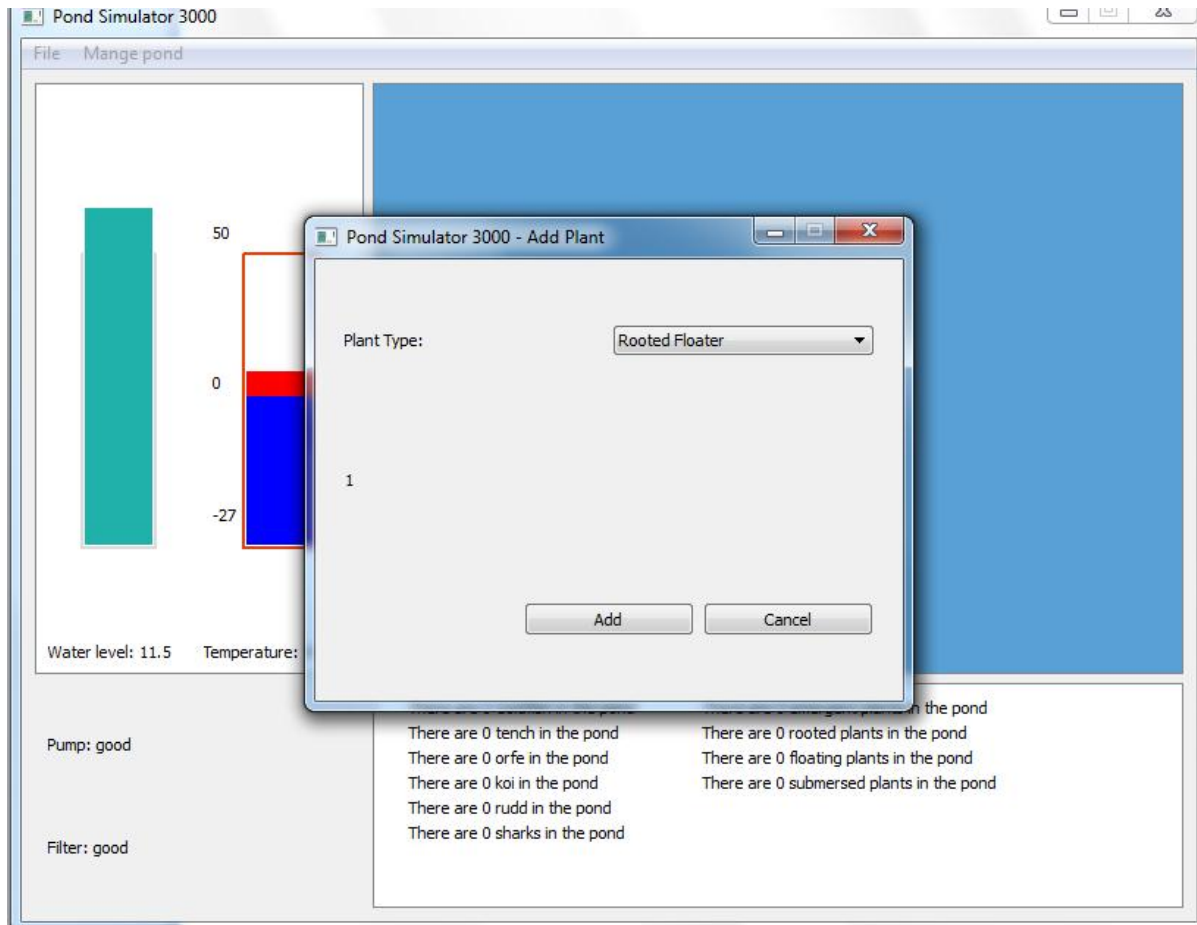
New pond window



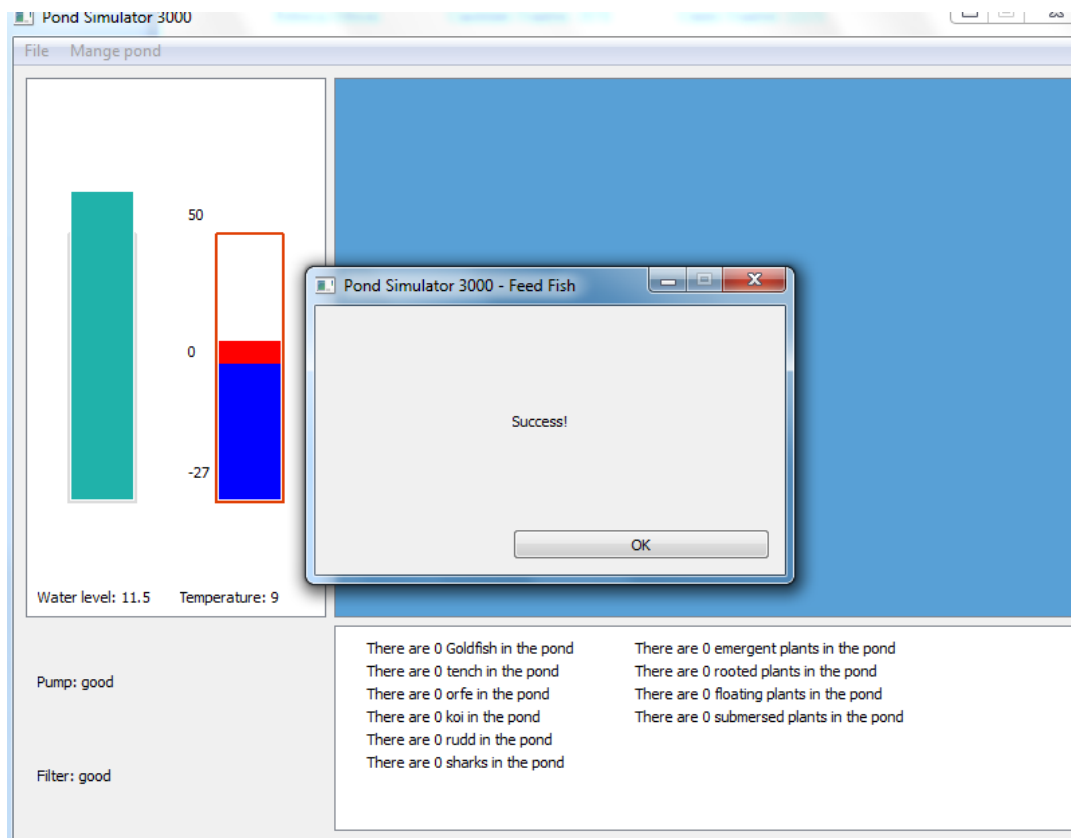
Edit pond window



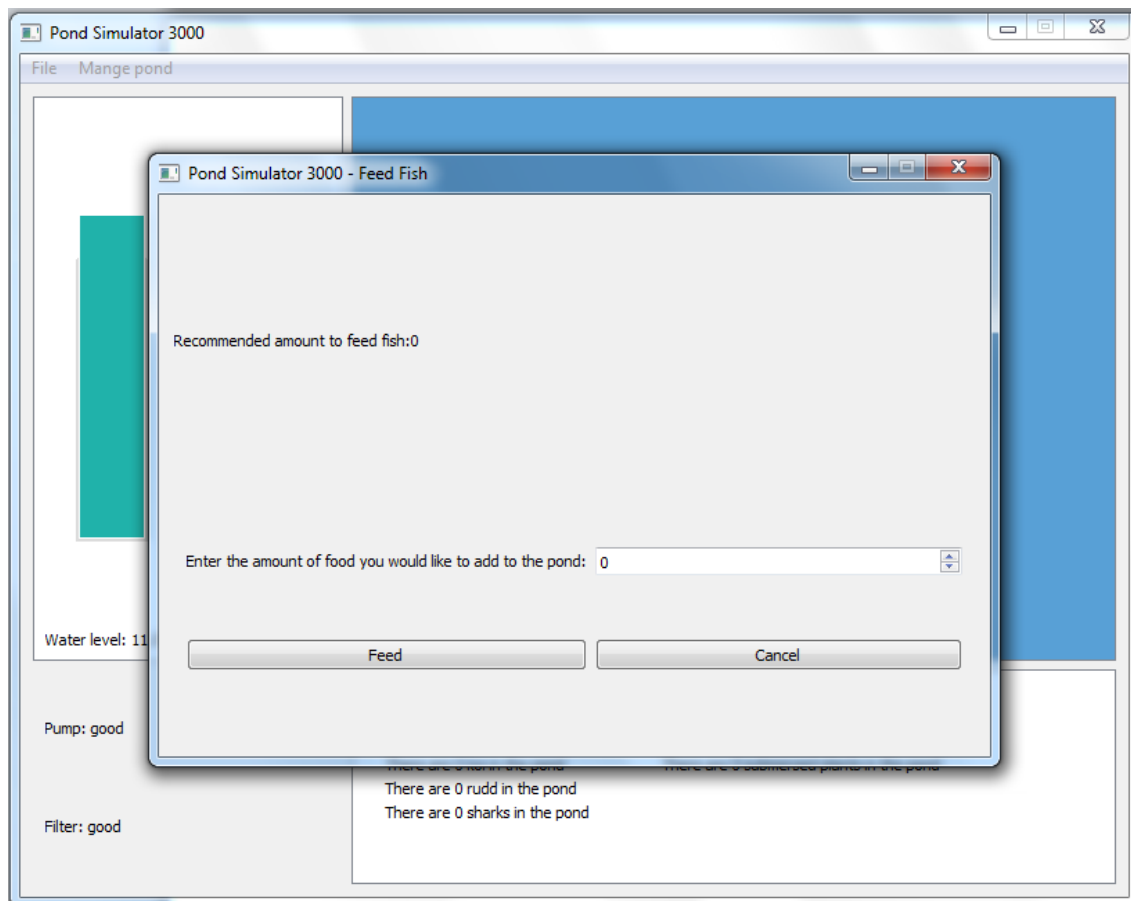
Add fish window



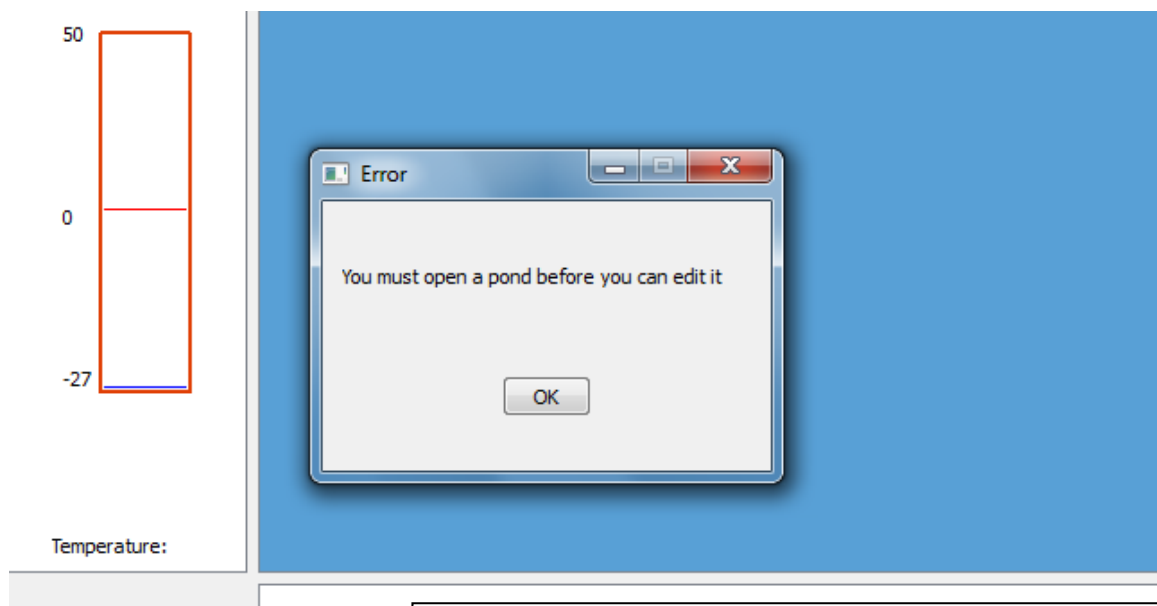
Add plant window



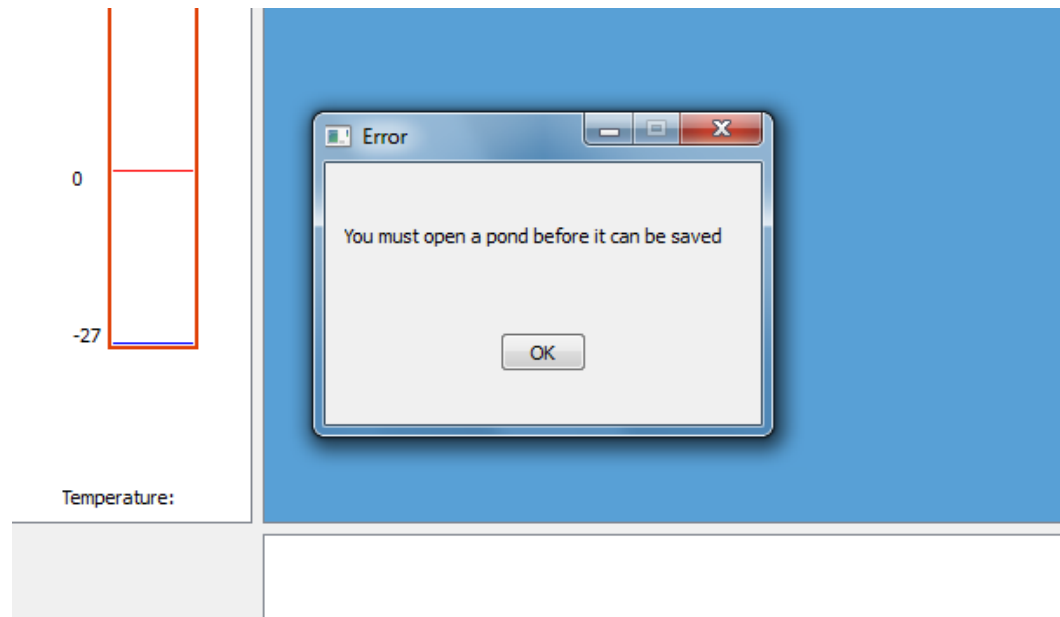
Message telling user fish have been fed successfully



Feed fish manually window

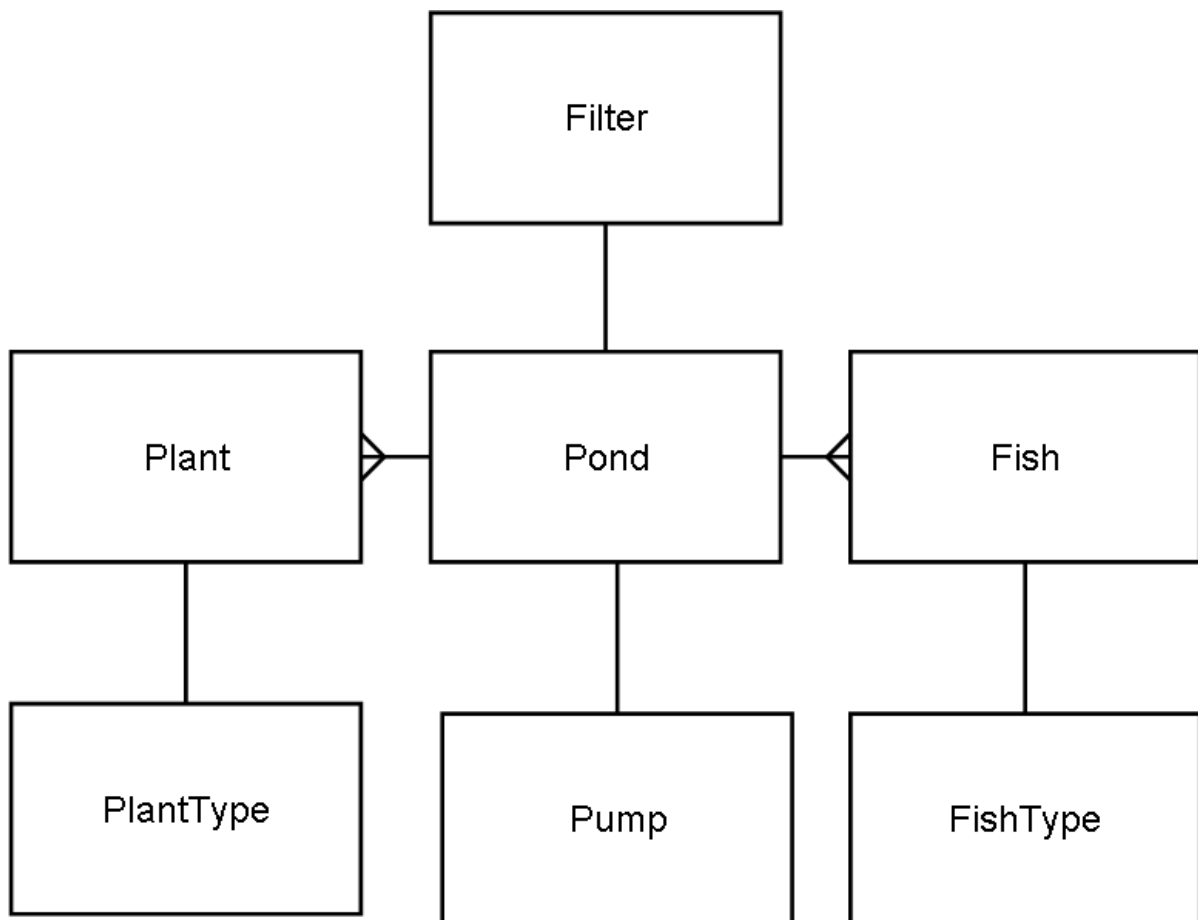


Error message displays when user attempts to edit the pond when one is not selected.



Error message displays when use attempts to save the pond when one is not selected.

5.2 ER Diagram



5.3 Database Table Views

SQLite Database Browser - E:\program\pond.db

File Edit View Help

Database Structure Browse Data Execute SQL

Table: filter

New Record Delete Record

filterID	filterStatus
1	Broken
2	good
3	good
4	good
5	good
6	good
7	good
8	good
9	good
10	good
11	good
12	good
13	good
14	good
15	good
16	good
17	good

< 1 - 17 of 17 >

Go to: 0

SQLite Database Browser - E:\program\pond.db

File Edit View Help

Database Structure Browse Data Execute SQL

Table: pump

New Record Delete Record

pumpID	pumpStatus
1	Broken
2	good
3	Broken
4	good
5	good
6	good
7	Broken
8	good
9	Broken
10	Broken
11	Broken
12	Broken
13	good
14	good
15	Broken
16	good
17	good

< 1 - 17 of 17 >

Go to: 0

SQLite Database Browser - E:\program\pond.db

File Edit View Help

Database Structure Browse Data Execute SQL

Table: fishType

New Record Delete Record

	fishType	foodNeed	growthRate	fishSize
1	Goldfish	5	1	0.5
2	Tench	4	0.5	1
3	Orfe	3	3	1
4	Koi	8	2	0.5
5	Rudd	2	0.5	0.25
6	Shark	1000	1	100

< 1 - 6 of 6 >

Go to: 0

SQLite Database Browser - E:\program\pond.db

File Edit View Help

Database Structure Browse Data Execute SQL

Table: plantType

New Record Delete Record

	plantType	lightRequiremen	growthRate
1	Rooted Floater	7	2
2	True Floater	2	2
3	Emergent	2	1
4	Submersed	2	1

< 1 - 4 of 4 >

Go to: 0

SQLite Database Browser - E:\program\pond.db

File Edit View Help

Database Structure Browse Data Execute SQL

Table: pond

New Record Delete Record

	pondID	pondDepth	pondLength	pondWidth	waterTemp	waterLevel	pumpID	filterID	Days
1	1	4	4	4	4	3.5	None	None	228
2	2	3	3	3	3	4.5			0
3	3	9	9	9	9	11.5	3	3	7
4	4	10	10	10	13	9.5			3
5	5	4	7	5	-27	3.5	5	5	0
6	6	0	0	0	0	-0.5			0
7	7	5	5	5	5	2.5	1	1	81
8	8	10	10	10	50	9.5	7	7	2
9	9	2	2	0	0	1.5	8		11
10	10	2	0	0	0	1.5	9	9	0
11	11	4	6	4	20	3.5	10		62
12	12	2	2	2	4	1.5	11		12
13	13	4	4	4	0	3.5	12		16
14	14	2	3	3	0	1.5			1
15	15	3	3	3	0	2.5			1
16	16	10	10	10	0	3.5	15	15	14
17	17	10	2	2	12	9.5	16	16	3
18	18	5	5	5	5	4.5	17	17	0

1 - 18 of 18

Go to: 0

SQLite Database Browser - E:\program\pond.db

File Edit View Help

Database Structure Browse Data Execute SQL

Table: fish

New Record Delete Record

	fishID	fishName	hunger	status	daysAlive	growth	fishType	pondID
1	20			3 good		3	9 Orfe	17
2	21			8 good		3	6 Koi	17
3	22			2 good		3	1.5 Rudd	17

1 - 3 of 3

Go to: 0

SQLite Database Browser - E:\program\pond.db

File Edit View Help

Database Structure Browse Data Execute SQL

Table: plant

New Record Delete Record

plantID	plantType	status	lightNeed	daysAlive	growth	PondID
1	1 Emergent	good		0	205	205 1
2	2 Rooted Floater	good		0	147	147 1
3	3 Emergent	good		0	147	147 1
4	6 Rooted Floater	good		0	0	0 5
5	7 Emergent	good		0	69	69 11

< 1 - 5 of 5 >

Go to: 0

5.4 Database SQL

5.4.1 Pond Database Table

```
create table pond(
pondID integer,
pondDepth integer,
pondLength integer,
pondWidth integer,
waterTemp integer,
waterLevel integer,
pumpID integer,
filterID integer,
Days integer,
primary key(pondID),
foreign key(pumpID) references pump(pumpID) on update cascade on delete restrict,
foreign key(filterID) references filter(filterID) on update cascade on delete restrict)
```

5.4.2 Fish Type Database Table

```
create table fishType(
fishType text,
foodNeed integer,
growthRate integer,
fishSize integer,
```

```
primary key(fishType))
```

5.4.3 Fish Database Table

```
create table fish(  
fishID integer,  
fishName text,  
hunger integer,  
status text,  
daysAlive integer,  
growth integer,  
fishType text,  
pondID integer,  
primary key(fishID),  
foreign key(fishType) references fishType(fishType),  
foreign key(pondID) references pond(pondID))
```

5.4.4 Plant Type Database Table

```
create table plantType(  
plantType text,  
lightRequirement integer,  
growthRate integer,  
primary key(plantType))
```

5.4.5 Plant Database Table

```
create table plant(  
plantID integer,  
plantType text,  
status text,  
lightNeed integer,  
daysAlive integer,  
growth integer,  
PondID integer,  
primary key(plantID),  
foreign key(plantType) references plantType(plantType),  
foreign key(pondID) references pondID(pondID))
```

5.4.6 Pump Database Table

```
create table pump(  
pumpID integer,  
pumpStatus text,  
primary key (pumpID))
```

5.4.7 Filter Database Table

```
create table filter(  
filterID integer,  
filterStatus text,  
primary key (filterID))
```

5.5 SQL Queries

5.5.1 Fetch all Ponds

```
select *  
from pond
```

Reference: getPond function in pondSimDatabase, lines 4 to 9

5.5.1 Fetch Pond where ID is user selection

```
select *  
from pond  
where pondID=?
```

Reference: fetchPond function in pondSimDatabase, lines 11 to 18

5.5.2 Fetch Pump where pond ID is user selection

```
select *  
from pump  
where pumpID=?
```

Reference: fetchPump function in pondSimDatabase, lines 20 to 27

5.5.3 Fetch Filter where pond ID is user selection

```
select *  
from filter  
where filterID=?
```

Reference: fetchFilter function in pondSimDatabase, lines 29 to 36

5.5.4 Fetch all fish from specified pond

```
select *  
from fish  
where pondID=?
```

Reference: fetchFish function in pondSimDatabase, lines 38 to 45

5.5.5 Fetch all Plants from a specified Pond

```
select *  
from plant  
where pondID=?
```

Reference: fetchPlants function in pondSimDatabase, lines 47 to 54

5.5.6 Fetch everything from the PlantType Table

```
select *  
from plantType
```

Reference: fetchPlantType function in pondSimDatabase, lines 65 to 70

5.5.7 Fetch everything from the FishType table

```
select *  
from fishType
```

Reference: fetchFishType function in pondSimDatabase, lines 72 to 77

5.5.8 Fetch all Growth Rates for the Different Fish Types

```
select growthRate  
from fishType
```

Reference: fetchGrowthRate function in pondSimDatabase, lines 79 to 84

5.5.9 Deleting a Specified Pond

```
delete  
from pond  
where pondID=?
```

Reference: deletePond function in pondSimDatabase, lines 102 to 108

5.5.10 Fetching all pond IDs

```
select pondID  
from pond
```

Reference: fetchPondID function in pondSimDatabase, lines 110 to 115

5.5.11 Deleting a Specified Fish from Fish Database

```
delete  
from fish  
where fishID=?
```

Reference: deleteFish function in pondSimDatabase, lines 117 to 123

5.5.12 Deleting all Fish from Specified Pond

```
delete
from fish
where pondID=?
```

Reference: deleteFishFromPond function in pondSimDatabase, lines 125 to 131

5.5.13 Deleting all plants from Specified Pond

```
delete
from plant
where pondID=?
```

Reference: deletePlantFromPond function in pondSimDatabase, lines 133 to 139

5.5.14 Deleting Specified Plant

```
delete
from plant
where plantID=?
```

Reference: deletePlant function in pondSimDatabase, lines 141 to 147

5.5.15 Delete Pump

```
delete
from pump
where pumpID=?
```

Reference: deletePump function in pondSimDatabase, lines 149 to 155

5.5.16 Delete Filter

```
delete
from filter
where filterID=?
```

Reference: deleteFilter function in pondSimDatabase, lines 157 to 163

5.5.17 Save Fish to Database

```
sql = """insert into fish(fishName, hunger, status, daysAlive, growth,
fishType, pondID) values
(
'{0}','{1}','{2}','{3}','{4}','{5}','{6}')""".format(self._fishName,
self._hunger, self._status, self._daysAlive, self._growth, self._fishType,
pondID)
```

I included some of the surrounding python code for this query to make it easier to understand.

Reference: saveFish method in the fish class in the fish module, lines 78 to 84

5.5.18 Save Plants to Database

```
sql = """insert into plant(status, lightNeed, daysAlive, growth, plantType,
PondID) values
(
'{0}','{1}','{2}','{3}','{4}','{5}')""".format(self._status,
self._lightNeed, self._daysAlive, self._growth, self._plantType, pondID)
```

Reference: savePlant method in the plant class in the plant module, lines 46 to 52

5.5.19 Save Pond to Database

```
sql = """insert into pond(pondDepth, pondLength, pondWidth, waterTemp,
waterLevel, pumpID, filterID, days) values
(
'{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}')""".format(self._pondDepth,
self._pondLength, self._pondWidth, self._waterTemp, self._waterLevel,
self._pump, self._filter, self._days)
```

Reference: savePond method in the pond class in the pond module, lines 169 to 186

5.5.19 Update Pond

```
update pond
set pondDepth=?, pondLength=?, pondWidth=?, waterTemp=?, waterLevel=?,
days=?, pumpID=?, filterID=?
where pondID="""
```

Reference: updatePond method in the pond class in the pond module, lines 188 to 206

5.5.20 Update Filter

```
update filter
set filterStatus=?
where filterID=?
```

Reference: updateFilter method in the filter class in the pond module, lines 231 to 237

5.5.21 Update Pump

```
update pump
set pumpStatus=?
where pumpID=?
```

Reference: updatePump method in the pump class in the pond module, lines 262 to 268

5.5.22 Update Fish

```
update fish
set hunger=?, daysAlive=?
where fishID=""
```

Reference: updateFish function in the incrementDay function in the pond module, lines 434 to 439

5.5.23 Update Plant

```
update plant
set daysAlive=?, growth=?
where plantID=""
```

Reference: updatePlant function in the incrementDay function in the pond module, lines 442 to 446

6. Testing

6.1 Summary of Results

My testing proved my program to be reliable and robust as results were as expected and the program did not crash during testing (see 3.1 Actual Results, page 75).

The weaknesses of my testing program are that I neglected to test several features of my GUI, such as the radio buttons. I also did not test specific queries, although a lot of my tests relied on them working so I believe most or all to be correct. I failed to test if trying to perform certain functions while a pond is not open would cause the program to crash – this would obviously affect the robustness of my program

The main strength of my testing was when I tested the GUI features of my program – it proved that my program could be navigated correctly and easily through the menu bar and buttons, and that the graphical display is correct. The testing revealed that all of my buttons and menu options were linked to their correct places, and that the graphical display displayed all of the fish and plants correctly. My testing also allowed me to identify the problem discussed below.

6.2 Known Issues

Test 4.1 - The fish status did not change after feeding

I intended the status of a fish to change to dead from alive once the fish reached over 20 hunger, however this only happens when the day is incremented and not when the fish are fed (or not fed) even though the day is incremented as part of that function. I'm not sure how to fix this error other than checking the fish's hunger level in the manual feeding function (10.5 Pond Module, page 167 – lines 648 to 692) as well as the increment day function. The fish's status might not be saving properly.

7. Code Explanations

7.1 Difficult Sections

The following section is lines 1099 to 1208 in the GUI code section, which can be found in the section appendix. The purpose of this section of code is to draw the water and temperature meters and display them in the graphic display.

```

-----GUI for meters-----
#water meter outline
self.meter.waterMeter=QRectF(10,-50,50,200) #creating rect
self.meter.waterMeter.pen=QPen() #creating pen
self.meter.waterMeter.color=QColor() #creating colour for pen
self.meter.waterMeter.color.setNamedColor('#DCDCDC')
self.meter.waterMeter.pen.setColor(self.meter.waterMeter.color) #setting color of pen
self.meter.waterMeter.pen.setWidth(2) #setting line width
self.meter.waterMeter.brush=QBrush() #creating blank fill
self.meter.addRect(self.meter.waterMeter,self.meter.waterMeter.pen,self.meter.waterMeter.brush) #drawing rect

#temperature meter outline
self.meter.tempMeter=QRectF(120,-50,50,200)
self.meter.tempMeter.pen=QPen()
self.meter.tempMeter.color=QColor() #creating colour for pen
self.meter.tempMeter.color.setNamedColor('#E04006')
self.meter.tempMeter.pen.setColor(self.meter.tempMeter.color) #setting color of pen
self.meter.tempMeter.pen.setWidth(2) #setting line width
self.meter.tempMeter.brush=QBrush()
self.meter.addRect(self.meter.tempMeter,self.meter.tempMeter.pen,self.meter.tempMeter.brush)

#water meter fill
try:
    number=float(self.mainPond.getWaterLevel())
    number=number*20
except:
    number=0
self.meter.waterMeterFill=QRectF(11,-50,47,number)
self.meter.waterMeterFill.moveBottom(148)
self.meter.waterMeterFill.pen=QPen() #creating pen
self.meter.waterMeterFill.color=QColor() #creating colour for pen
self.meter.waterMeterFill.color.setNamedColor('FFFFFF')
self.meter.color=QColor()
self.meter.color.setNamedColor('lightseagreen')
self.meter.waterMeterFill.pen.setColor(self.meter.waterMeterFill.color) #setting color of pen
self.meter.waterMeterFill.brush=QBrush() #creating blank fill
self.meter.waterMeterFill.brush.setColor(self.meter.color)
self.meter.waterMeterFill.brush.setStyle(Qt.SolidPattern)
self.meter.addRect(self.meter.waterMeterFill,self.meter.waterMeterFill.pen,self.meter.waterMeterFill.brush) #drawing rect

```

Defining the size of the rect for the water meter outline

Drawing the rectangle requires a pen, which requires a colour and a width (the default is black and 1) and a brush (acts as the fill – default is white)

The temperature outline is made in the same way

The height of the rectangle of the water fill is the water level multiplied by 20 – this is because 10 x 20 is 200 (the max height of the rectangle)

The brush of the water fill rectangle is given a colour and the fill style is set to solid

```
#water temperature fill
```

```
try:
    number=float(self.mainPond.getWaterTemp())
    if number>=0:
        number=number*2
    else:
        number=0
except:
    number=0
```

This determines the height of the rectangle. If the water temperature is greater than 0, it means the temperature is positive. The number is times by two to fit in the box properly

```
self.meter.tempMeterFill=QRectF(122,-50,45,number)
self.meter.tempMeterFill.moveBottom(48)
self.meter.tempMeterFill.pen=QPen() #creating pen
color=QColor() #creating colour for pen
color.setNamedColor('FFFFFF')
self.meter.tempMeterFill.color=QColor()
self.meter.tempMeterFill.color.setNamedColor('red')
self.meter.tempMeterFill.pen.setColor(self.meter.tempMeterFill.color) #setting color of pen
self.meter.tempMeterFill.brush=QBrush() #creating blank fill
self.meter.tempMeterFill.brush.setColor(self.meter.tempMeterFill.color)
self.meter.tempMeterFill.brush.setStyle(Qt.SolidPattern)
self.meter.addRect(self.meter.tempMeterFill,self.meter.tempMeterFill.pen,self.meter.tempMeterFill.brush) #drawing rect
```

The rectangle is drawn in the middle of the outline rectangle

```
#water temperature minus fill
```

```
try:
    number=float(self.mainPond.getWaterTemp())
    if number<0:
        number=-number*2
    else:
        number=100
except:
    number=0
```

This determines the height of the rectangle. If the water temperature is less than 0, the height is minus the water temperature multiplied by 2, otherwise the height is 100 (the full height of the box)

```
self.meter.tempMeterFill=QRectF(122,-50,45,number)
self.meter.tempMeterFill.moveBottom(147)
self.meter.tempMeterFill.pen=QPen() #creating pen
color=QColor() #creating colour for pen
color.setNamedColor('FFFFFF')
self.meter.tempMeterFill.color=QColor()
self.meter.tempMeterFill.color.setNamedColor('blue')
self.meter.tempMeterFill.pen.setColor(self.meter.tempMeterFill.color) #setting color of pen
self.meter.tempMeterFill.brush=QBrush() #creating blank fill
self.meter.tempMeterFill.brush.setColor(self.meter.tempMeterFill.color)
self.meter.tempMeterFill.brush.setStyle(Qt.SolidPattern)
self.meter.addRect(self.meter.tempMeterFill,self.meter.tempMeterFill.pen,self.meter.tempMeterFill.brush) #drawing rect
```

```
self.widgetM=QWidget() #creating widget
try:
    level=str(self.mainPond.getWaterLevel())
except:
    level=""
try:
    temp=str(self.mainPond.getWaterTemp())
except:
    temp=""
self.waterLabel=QLabel("Water level: "+level)#making labels
self.tempLabel=QLabel("Temperature: "+temp)
self.zeroLabel=QLabel(" 0")
self.zeroLabel.setFixedHeight(8)
self.fiftyLabel=QLabel(" 50")
self.fiftyLabel.setFixedHeight(185)
```

7.2. Self-created Algorithms

7.2.1 Automatically Grow Pond

Reference: 10.5 Pond Module 167, line 695 to 732.

The following function looks up what plants and fish are currently in the pond and grows them with the exact required food for 7 days.

```
def automaticGrow(db,cursor,pond):
    #feeds fish the required food amount for 7 days.
```

```
    def updateFish(tup):
        sql = """update fish
                set hunger=?, growth=?, status=?
                where fishID=?"""
        cursor.execute(sql,tup)
        db.commit()
```

This function will save the fish information to the database

Fetches all the fish in the pond from the database

```
    fishList=pondSimDatabase.fetchFishInfo(db,cursor,pond.getPondID())
    growthList=pondSimDatabase.fetchGrowthRate(db,cursor)
```

Fetches all the growth rates of the fish

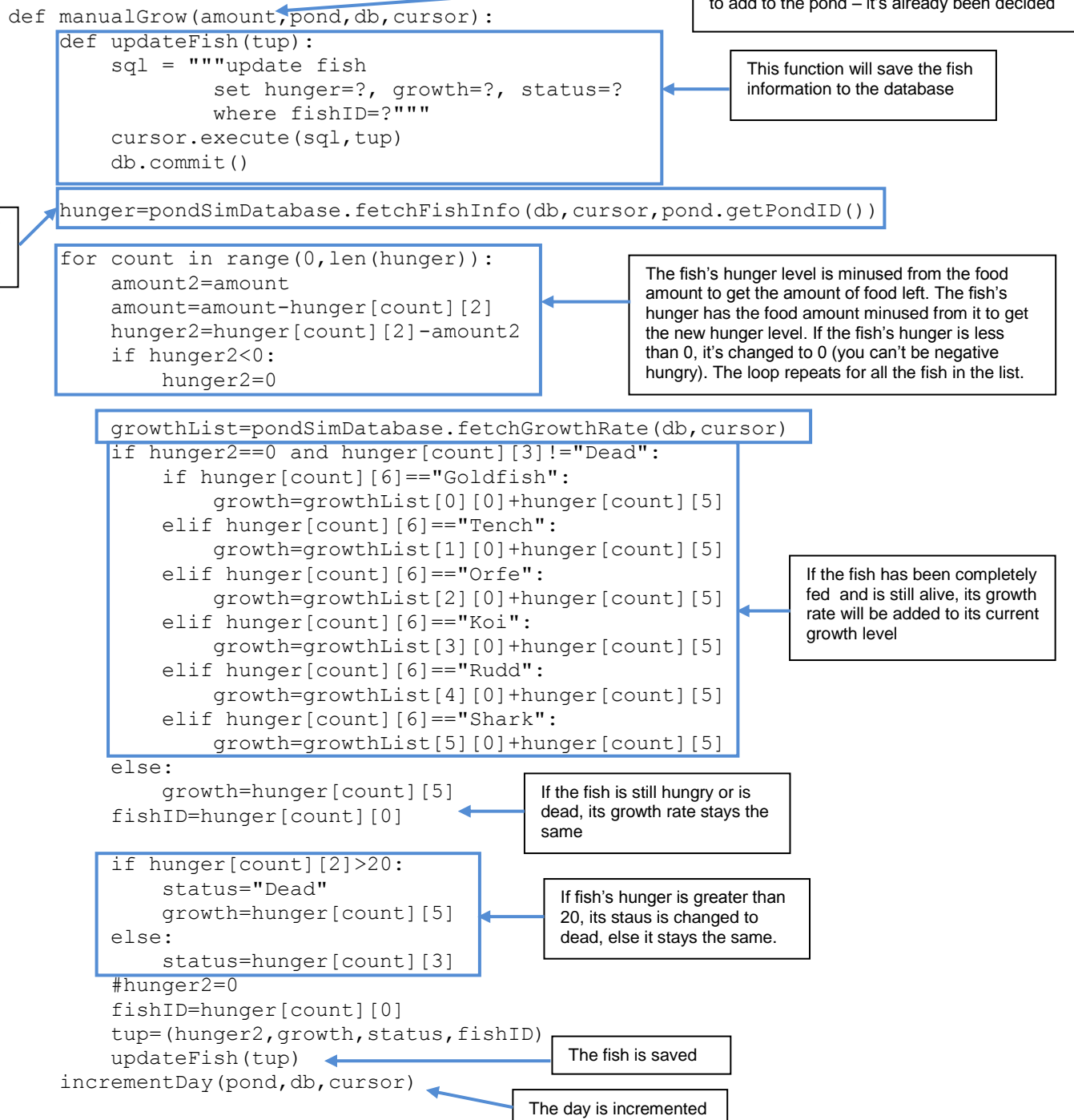
```
    for count in range(0,7):
        for count in range(0,len(fishList)):
            if fishList[count][3]!="Dead":
                if fishList[count][6]=="Goldfish":
                    growth=growthList[0][0]+fishList[count][5]
                elif fishList[count][6]=="Tench":
                    growth=growthList[1][0]+fishList[count][5]
                elif fishList[count][6]=="Orfe":
                    growth=growthList[2][0]+fishList[count][5]
                elif fishList[count][6]=="Koi":
                    growth=growthList[3][0]+fishList[count][5]
                elif fishList[count][6]=="Rudd":
                    growth=growthList[4][0]+fishList[count][5]
                elif fishList[count][6]=="Shark":
                    growth=growthList[5][0]+fishList[count][5]
            else:
                growth=fishList[count][5]
            if fishList[count][2]>20:
                status="Dead"
                growth=fishList[count][5]
            else:
                status=fishList[count][3]
            hunger=0
            fishID=fishList[count][0]
            tup=(hunger,growth,status,fishID)
            updateFish(tup)
        incrementDay(pond,db,cursor)
```

Goes through each fish at a time and adds their growth rate to their growth level, if they are not dead. If they are dead, their growth level stays the same. If the fish is over 20 hunger, their status is changed to dead. Hunger is set to 0, and "updateFish" is called to save the changed data to the database. The day is incremented. This whole loop repeats 7 time to simulate the 7 days.

7.2.2 Manually Grow Pond

Reference: 10.5 Pond Module 167, line 648 to 692

The following function receives an input from the user for how food they want to add to the pond, if the fish are completely fed, they will grow and the pond will increment by one day.



7.2.3 Increment Day

Reference: 10.5 Pond Module 167, line 433 to 520

The following function increments the number of days the pond has been around and all the plants and fish in it.

```
def incrementDay(mainPond,db,cursor):
```

```
    def updateFish(tup):
        sql = """update fish
                set hunger=?, daysAlive=?
                where fishID=?"""
        cursor.execute(sql,tup)
        db.commit()
```

Updates the fish in the database

```
    def updatePlants(tup):
        sql="""update plant
            set daysAlive=?, growth=?
            where plantID=?"""
        cursor.execute(sql,tup)
        db.commit()
```

Updates the plants in the database

Fetching list of all fish, list of all plants, list of the plant type information, list of the fish type informaion

```
    fishList=pondSimDatabase.fetchFishInfo(db,cursor,mainPond.getPondID())
    plantList=pondSimDatabase.fetchPlantInfo(db,cursor,mainPond.getPondID())
    plantType=pondSimDatabase.fetchPlantType(db,cursor)
    foodNeed=pondSimDatabase.fetchFoodNeed(db,cursor)
```

```
    for count in range(0,len(fishList)):
        daysAlive=fishList[count][4]+1
        if fishList[count][6]=="Goldfish":
            hunger=fishList[count][2]+foodNeed[0][1]
        elif fishList[count][6]=="Tench":
            hunger=fishList[count][2]+foodNeed[1][1]
        elif fishList[count][6]=="Orfe":
            hunger=fishList[count][2]+foodNeed[2][1]
        elif fishList[count][6]=="Koi":
            hunger=fishList[count][2]+foodNeed[3][1]
        elif fishList[count][6]=="Rudd":
            hunger=fishList[count][2]+foodNeed[4][1]
        elif fishList[count][6]=="Shark":
            hunger=fishList[count][2]+foodNeed[5][1]
        if fishList[count][3]=="Dead":
            hunger=0
            daysAlive=0
        elif fishList[count][3]=="good":
            daysAlive=fishList[count][4]+1
        tup=(hunger,daysAlive,fishList[count][0])
        updateFish(tup)
```

Goes through each fish in the fish list and adds the food need of that fish type to the fish's hunger level to calculate the new hunger level.

If the fish is dead, hunger is set to 0 and days alive is set to 0, else the days alive is incremeneted by 1.

The fish is updated in the database

Continues next page

```

for count in range(0,len(plantList)):
    daysAlive=plantList[count][4]+1
    if plantList[count][1]=="Rooted Floater":
        growth=plantList[count][5]+plantType[0][1]
    elif plantList[count][1]=="True Floater":
        growth=plantList[count][5]+plantType[1][1]
    elif plantList[count][1]=="Emergent":
        growth=plantList[count][5]+plantType[2][1]
    elif plantList[count][1]=="submersed":
        growth=plantList[count][5]+plantType[3][1]
    if plantList[count][2]=="Dead":
        daysAlive=0
    else:
        daysAlive=plantList[count][4]+1
    tup=(daysAlive,plantList[count][5]+1,plantList[count][0])
    updatePlants(tup)

```

Goes through each plant in the plant list and adds the plant's growthrate to the growth of the plant.

If the plant is dead, the number of days alive is set to 0, else it's incremented by 1.

The plant is updated in the database

```

randomNo=random.randint(0,20)
if randomNo==1:
    try:
        pump=pondSimDatabase.fetchPump(db,cursor,mainPond.getPump())
    #fetching pump data
        pump=Pump(pump[0][1], pump[0][0]) #creating pump object from
    pump data
        pumpStatus=pump.getPumpStatus()
        pump.setPumpStatus("Broken")
        pump.updatePump(db,cursor)
        mainPond.setPump(pump.getPumpID())
    except:
        pass

```

Generates a random number between 1 and 20. If number equals 1, the pump's status is changed to "Broken" and the change is saved to the database.

```

randomNo=random.randint(0,20)
if randomNo==1:
    try:
        filter=pondSimDatabase.fetchFilter(db,cursor,self.mainPond.getFilter())
    #fetching pump data
        filter=Filter(filter[0][1], filter[0][0]) #creating pump object
    from pump data
        filterStatus=filter.getFilterStatus()
        filter.setFilterStatus("Broken")
        filter.updateFilter(db,cursor)
        mainPond.setFilter(filter.getFilterID())
    except:
        pass

    mainPond.setDays(int(mainPond.getDays())+1)
    mainPond.updatePond(db,cursor)

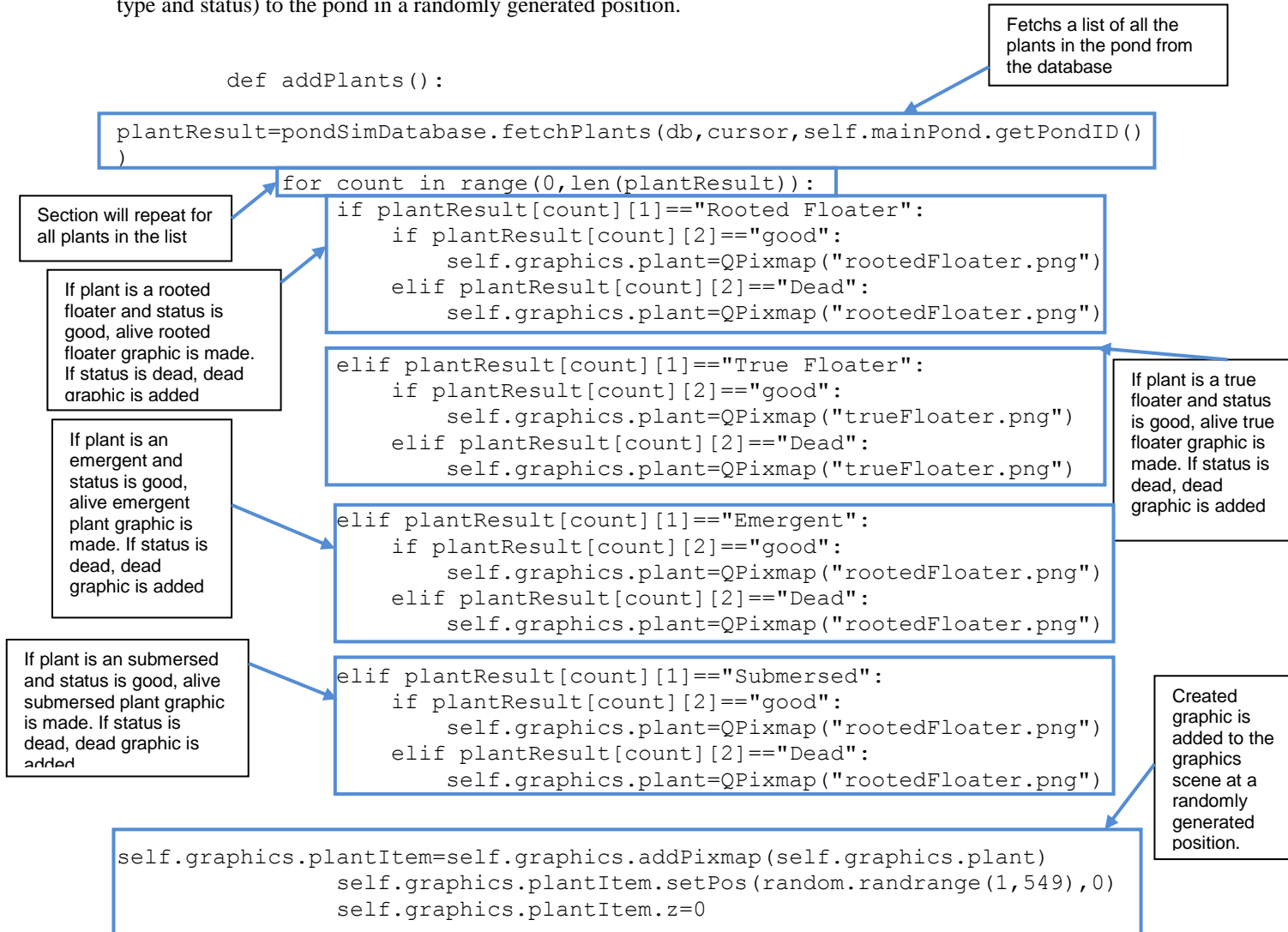
```

Generates a random number between 1 and 20. If number equals 1, the filter's status is changed to "Broken" and the change is saved to the database.

7.2.4 Add Plants

Reference: 10.2 GUI, page 139 – lines 1023 to 1063

The following function fetches the plants stored in the database and adds an image of them (depending on their type and status) to the pond in a randomly generated position.



7.2.5 Add Fish

Reference: 10.2 GUI, page 139 – lines 976 to 1031

The following function fetches the fish stored in the database and adds an image of them (depending on the type and status) to the pond in a randomly generated position.

```
def addFish():
```

```
    fishResults=pondSimDatabase.fetchFish(db,cursor,self.mainPond.getPondID())
```

Fetches a list of all the fish in the pond from the database

```
    for count in range(0,len(fishResults)):
```

If fish is a goldfish and status is good, alive goldfish graphic is made. If status is dead, dead graphic is added

```
        if fishResults[count][6]=="Goldfish":
            if fishResults[count][3]=="good":
                self.graphics.fish=QPixmap("fish.png")
            elif fishResults[count][3]=="Dead":
                self.graphics.fish=QPixmap("dead.png")
```

Adds fish graphic to graphics scene at a random position

```
    self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
```

```
    self.graphics.fishItem.setPos(random.randrange(1,549),random.randrange(1,399)) #adds the fish graphic at a random location
    self.graphics.fishItem.z=2
```

If fish is a tench and status is good, alive tench graphic is made. If status is dead, dead graphic is added

```
        elif fishResults[count][6]=="Tench":
            if fishResults[count][3]=="good":
                self.graphics.fish=QPixmap("fish.png")
            elif fishResults[count][3]=="Dead":
                self.graphics.fish=QPixmap("dead.png")
```

```
    self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
```

```
    self.graphics.fishItem.setPos(random.randrange(1,549),random.randrange(1,399)) #adds the fish graphic at a random location
    self.graphics.fishItem.z=2
```

If fish is an orfe and status is good, alive orfe graphic is made. If status is dead, dead graphic is added

```
        elif fishResults[count][6]=="Orfe":
            if fishResults[count][3]=="good":
                self.graphics.fish=QPixmap("fish.png")
            elif fishResults[count][3]=="Dead":
                self.graphics.fish=QPixmap("dead.png")
```

```
    self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
```

```
    self.graphics.fishItem.setPos(random.randrange(1,549),random.randrange(1,399)) #adds the fish graphic at a random location
    self.graphics.fishItem.z=2
```

If fish is a koi and status is good, alive koi graphic is made. If status is dead, dead graphic is added

```
        elif fishResults[count][6]=="Koi":
            if fishResults[count][3]=="good":
                self.graphics.fish=QPixmap("fish.png")
            elif fishResults[count][3]=="Dead":
                self.graphics.fish=QPixmap("dead.png")
```

```
    self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
```

```
    self.graphics.fishItem.setPos(random.randrange(1,549),random.randrange(1,399)) #adds the fish graphic at a random location
    self.graphics.fishItem.z=2
```

If fish is a rudd and status is good, alive rudd graphic is made. If status is dead, dead graphic is added

```
elif fishResults[count][6]=="Rudd":
    if fishResults[count][3]=="good":
        self.graphics.fish=QPixmap("fish.png")
    elif fishResults[count][3]=="Dead":
        self.graphics.fish=QPixmap("dead.png")
```

```
self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
```

```
self.graphics.fishItem.setPos(random.randrange(1,549),random.randrange(1,399)) #adds the fish graphic at a random location
self.graphics.fishItem.z=2
```

If fish is a shark and status is good, alive shark graphic is made. If status is dead, dead graphic is added

```
elif fishResults[count][6]=="Shark":
    if fishResults[count][3]=="good":
        self.graphics.fish=QPixmap("shark.png")
    elif fishResults[count][3]=="Dead":
        self.graphics.fish=QPixmap("sharkDead.png")
```

```
self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
self.graphics.fishItem.setPos(random.randrange(-10,10),random.randrange(-20,20)) #adds the fish graphic at a random location
self.graphics.fishItem.z=2
```

8. Settings

In order for my program to run correctly, Python 3.2 must be installed on the machine as well the modules SQLite3 and PyQt4

9. Acknowledgements

I referenced this page on ZetCode when drawing my graphics <http://zetcode.com/tutorials/pyqt4/drawing/>

The code I looked at is as follows:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

"""
ZetCode PyQt4 tutorial

This example draws three rectangles in three
different colors.

author: Jan Bodnar
website: zetcode.com
last edited: September 2011
"""

import sys
from PyQt4 import QtGui, QtCore

class Example(QtGui.QWidget):
```

```
def __init__(self):
    super(Example, self).__init__()

    self.initUI()

def initUI(self):

    self.setGeometry(300, 300, 350, 100)
    self.setWindowTitle('Colors')
    self.show()

def paintEvent(self, e):

    qp = QtGui.QPainter()
    qp.begin(self)
    self.drawRectangles(qp)
    qp.end()

def drawRectangles(self, qp):

    color = QtGui.QColor(0, 0, 0)
    color.setNamedColor('#d4d4d4')
    qp.setPen(color)

    qp.setBrush(QtGui.QColor(200, 0, 0))
    qp.drawRect(10, 15, 90, 60)

    qp.setBrush(QtGui.QColor(255, 80, 0, 160))
    qp.drawRect(130, 15, 90, 60)

    qp.setBrush(QtGui.QColor(25, 0, 90, 200))
    qp.drawRect(250, 15, 90, 60)

def main():

    app = QtGui.QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```

10. Code Listing Appendix

10.1 databaseCreation Module

```
1  import sqlite3
2  import fish
3  import plant
4
5  #creating databases
6  def createPondTable(db,cursor):
7      sql = """create table pond(
8          pondID integer,
9          pondDepth integer,
10         pondLength integer,
11         pondWidth integer,
12         waterTemp integer,
13         waterLevel integer,
14         pumpID integer,
15         filterID integer,
16         Days integer,
17         primary key(pondID),
18         foreign key(pumpID) references pump(pumpID) on update cascade on
delete restrict,
19         foreign key(filterID) references filter(filterID) on update cascade
on delete restrict)"""
20     cursor.execute(sql)
21
22 def createFishTypeTable(db,cursor):
23     sql = """create table fishType(
24         fishType text,
25         foodNeed integer,
26         growthRate integer,
27         fishSize integer,
28         primary key(fishType))"""
29     cursor.execute(sql)
30
31 def createFishTable(db,cursor):
32     sql = """create table fish(
33         fishID integer,
34         fishName text,
35         hunger integer,
36         status text,
37         daysAlive integer,
38         growth integer,
39         fishType text,
40         pondID integer,
41         primary key(fishID),
42         foreign key(fishType) references fishType(fishType),
43         foreign key(pondID) references pond(pondID))"""
44     cursor.execute(sql)
45
46 def createPlantTypeTable(db,cursor):
47     sql = """create table plantType(
48         plantType text,
49         lightRequirement integer,
50         growthRate integer,
51         primary key(plantType))"""
52     cursor.execute(sql)
53
54 def createPlantTable(db,cursor):
55     sql = """create table plant(
56         plantID integer,
57         plantType text,
58         status text,
```

```

59         lightNeed integer,
60         daysAlive integer,
61         growth integer,
62         PondID integer,
63         primary key(plantID),
64         foreign key(plantType) references plantType(plantType),
65         foreign key(pondID) references pondID(pondID) """
66     cursor.execute(sql)
67
68 def createPumpTable(db,cursor):
69     sql = """create table pump(
70         pumpID integer,
71         pumpStatus text,
72         primary key (pumpID) """
73     cursor.execute(sql)
74
75 def createFilterTable(db,cursor):
76     sql = """create table filter(
77         filterID integer,
78         filterStatus text,
79         primary key (filterID) """
80     cursor.execute(sql)
81
82 def createDatabases():
83     db=sqlite3.connect('pond.db')
84     cursor=db.cursor()
85     try:
86         createPumpTable(db,cursor)
87     except:
88         pass
89
90     try:
91         createFilterTable(db,cursor)
92     except:
93         pass
94
95     try:
96         createFishTypeTable(db,cursor)
97     except:
98         pass
99
100    try:
101        createPlantTypeTable(db,cursor)
102    except:
103        pass
104
105    try:
106        createPondTable(db,cursor)
107    except:
108        pass
109
110    try:
111        createFishTable(db,cursor)
112    except:
113        pass
114
115    try:
116        createPlantTable(db,cursor)
117    except:
118        pass
119    fish.saveFishTypes(db,cursor)
120    plant.savePlantTypes(db,cursor)
121    cursor.close()

```

10.2 GUI

```
1 from PyQt4.QtGui import *
```



```
2  from PyQt4.QtCore import *
3  import sys
4  import pond
5  import sqlite3
6  import pondSimDatabase
7  import databaseCreation
8  import fish
9  import plant
10 import random
11
12 class MainWindow(QMainWindow): #main window class inherits from QMainWindow
13     def __init__(self): #constructor
14         super(MainWindow,self).__init__() #call parent constructor
15         self.createUI() #call creatUI method to create user interface for the
main window
16
17     def createUI(self):
18         def savePond():
19             try:
20                 self.mainPond.updatePond(db,cursor)
21             except:
22                 self.errorWindow=QWidget()
23                 self.errorWindow.setWindowTitle("Error")
24                 self.errorWindow.setFixedWidth(250)
25                 self.errorWindow.setFixedHeight(150)
26                 self.errorWindow.show()
27                 self.errorWindow.layout = QVBoxLayout()
28                 self.errorWindow.setLayout(self.errorWindow.layout)
29                 self.errorWindow.label=QLabel("You must open a pond before it
can be saved")
30                 self.errorWindow.label.setWordWrap(True)
31                 self.errorWindow.widget=QWidget()
32                 self.errorWindow.widgetLayout=QGridLayout()
33
34                 self.errorWindow.widget.setLayout(self.errorWindow.widgetLayout)
35                 self.errorWindow.okay=QPushButton("OK")
36                 self.errorWindow.okay.setFixedWidth(50)
37                 self.errorWindow.okay.clicked.connect(self.errorWindow.close)
38                 blank=QWidget()
39                 blank.setFixedWidth(100)
40                 blankTwo=QWidget()
41                 blankTwo.setFixedWidth(100)
42                 self.errorWindow.widgetLayout.addWidget(blank,0,2)
43                 self.errorWindow.widgetLayout.addWidget(blankTwo,0,0)
44
45                 self.errorWindow.widgetLayout.addWidget(self.errorWindow.okay,0,1)
46                 self.errorWindow.layout.addWidget(self.errorWindow.label)
47                 self.errorWindow.layout.addWidget(self.errorWindow.widget)
48
49         def openWindow():
50             #when pond is opened
51             def onOpen():
52                 self.window.close() #hides window
53                 self.selected=self.window.tree.currentItem()
54                 pondID=self.selected.data(0,0)
55                 depth=self.selected.data(1,0)
56                 length=self.selected.data(2,0)
57                 width=self.selected.data(3,0)
58                 temp=self.selected.data(4,0)
59                 level=self.selected.data(5,0)
60                 pump=self.selected.data(6,0)
61                 Filter=self.selected.data(7,0)
62                 days=self.selected.data(8,0)
63
64                 self.mainPond=pond.Pond(pondID=pondID,depth=depth,width=width,temp=temp,waterLevel=
level,pump=pump,filter=Filter,days=days, length=length)
65                 self.createUI()
```

```

63
64     def tree():
65         self.window.tree.setHeaderItem(self.window.header) #setting
header
66         results=pondSimDatabase.getPonds(db,cursor)
67         for count in range(0,len(results)):
68             self.window.treeList=QTreeWidgetItem()#list items
69             self.window.treeList.setText(0,str(results[count][0]))
70             self.window.treeList.setText(1,str(results[count][1]))
71             self.window.treeList.setText(2,str(results[count][2]))
72             self.window.treeList.setText(3,str(results[count][3]))
73             self.window.treeList.setText(4,str(results[count][4]))
74             self.window.treeList.setText(5,str(results[count][5]))
75             self.window.treeList.setText(6,str(results[count][6]))
76             self.window.treeList.setText(7,str(results[count][7]))
77             self.window.treeList.setText(8,str(results[count][8]))
78             self.window.tree.addTopLevelItem(self.window.treeList)
79
80     self.window=QWidget()
81     self.window.setWindowTitle("Pond Simulator 3000 - Open Pond")
82     self.window.setFixedWidth(600)
83     self.window.setFixedHeight(400)
84     self.window.show()
85     self.window.layout = QVBoxLayout()
86     self.window.setLayout(self.window.layout)
87
88     self.window.tree=QTreeWidgetItem()
89     self.window.tree.setColumnCount(9)
90     self.window.tree.setColumnWidth(0,80)
91
92     self.window.header=QTreeWidgetItem() #headers
93     self.window.header.setText(0,'PondID')
94     self.window.header.setText(1,'Pond Depth')
95     self.window.header.setText(2,'Pond Length')
96     self.window.header.setText(3,'Pond Width')
97     self.window.header.setText(4,'Water Temp.')
98     self.window.header.setText(5,'Water Level')
99     self.window.header.setText(6,'PumpID')
100    self.window.header.setText(7,'FilterID')
101    self.window.header.setText(8,'Days')
102    tree()
103
104    self.window.button=QPushButton("Cancel") #making button
105    self.window.button.setFixedWidth(100) #changing button size
106    self.window.button.layout=QHBoxLayout() #creating layout for button
107    self.window.buttonWidget=QWidget() #creating blank widget
108    self.window.blank=QWidget()
109    self.window.buttonWidget.setLayout(self.window.button.layout)
#setting layout to blank widget
110    self.window.button.layout.addWidget(self.window.blank) #adding
button layout
111    self.window.button.layout.addWidget(self.window.button) #adding
button layout
112
113    self.window.button.clicked.connect(self.window.close) #when button
is clicked, window is hidden
114    self.window.layout.addWidget(self.window.tree) #adding tree list to
layout
115    self.window.layout.addWidget(self.window.buttonWidget) #adding
button to layout
116    self.window.tree.itemDoubleClicked.connect(onOpen)#when item double
clicked
117
118    def newPondWindow():
119        self.window=QWidget()
120        self.window.setWindowTitle("Pond Simulator 3000 - New Pond")
121        self.window.setFixedWidth(600)

```

```
122         self.window.setFixedHeight(400)
123         self.window.show()
124         self.window.layout = QVBoxLayout()
125         self.window.setLayout(self.window.layout)
126
127         #box one
128         self.window.mainWidget=QFrame()
129         self.window.mainWidget.setLineWidth(1)
130         self.window.mainWidget.setFrameShape(QFrame.StyledPanel)
131         self.window.mainWidgetLayout=QVBoxLayout()
132         self.window.mainWidget.setLayout(self.window.mainWidgetLayout)
133
134         self.window.labelWidget=QWidget()
135         self.window.labelWidgetLayout=QHBoxLayout()
136         self.window.labelWidget.setLayout(self.window.labelWidgetLayout)
137         self.window.labelWidgetLayout.addWidget(QLabel("The pond can be a
maximum of 10 x 10 x 10 metres"))
138
139         self.window.editWidget=QWidget()
140         self.window.editWidgetLayout=QGridLayout()
141         self.window.editWidget.setLayout(self.window.editWidgetLayout)
142         self.window.editWidgetLayout.addWidget(QLabel("Depth:"),0,0)
143         self.window.depthLine=QSpinBox()
144         self.window.depthLine.setMaximum(10)
145         self.window.depthLine.setMinimum(0.5)
146         self.window.editWidgetLayout.addWidget(self.window.depthLine,0,1)
147         self.window.editWidgetLayout.addWidget(QLabel("Width:"),0,2)
148         self.window.widthLine=QSpinBox()
149         self.window.widthLine.setMaximum(10)
150         self.window.widthLine.setMinimum(0.5)
151         self.window.editWidgetLayout.addWidget(self.window.widthLine,0,3)
152         self.window.editWidgetLayout.addWidget(QLabel("Length:"),0,4)
153         self.window.lengthLine=QSpinBox()
154         self.window.lengthLine.setMaximum(10)
155         self.window.lengthLine.setMinimum(0.5)
156         self.window.editWidgetLayout.addWidget(self.window.lengthLine,0,5)
157
158         self.window.mainWidgetLayout.addWidget(self.window.labelWidget)
159         self.window.mainWidgetLayout.addWidget(self.window.editWidget)
160         self.window.layout.addWidget(self.window.mainWidget)
161
162         #box two
163         self.window.mainWidget=QFrame()
164         self.window.mainWidget.setLineWidth(1)
165         self.window.mainWidget.setFrameShape(QFrame.StyledPanel)
166         self.window.mainWidgetLayout=QVBoxLayout()
167         self.window.mainWidget.setLayout(self.window.mainWidgetLayout)
168
169         self.window.labelWidget=QWidget()
170         self.window.labelWidgetLayout=QHBoxLayout()
171         self.window.labelWidget.setLayout(self.window.labelWidgetLayout)
172         self.window.labelWidgetLayout.addWidget(QLabel("The water
temperature must be between 27 and +50C."))
173
174         self.window.editWidget=QWidget()
175         self.window.editWidgetLayout=QGridLayout()
176         self.window.editWidget.setLayout(self.window.editWidgetLayout)
177         self.window.editWidgetLayout.addWidget(QLabel("Temperature:"),0,0)
178         self.window.tempLine=QSpinBox()
179         self.window.tempLine.setMaximum(50)
180         self.window.tempLine.setMinimum(-27)
181         self.window.editWidgetLayout.addWidget(self.window.tempLine,0,1)
182         self.window.blank=QWidget()
183         self.window.blank.setFixedWidth(2000)
184         self.window.editWidgetLayout.addWidget(self.window.blank,0,2)
185
186         self.window.mainWidgetLayout.addWidget(self.window.labelWidget)
```

```

187         self.window.mainWidgetLayout.addWidget(self.window.editWidget)
188         self.window.layout.addWidget(self.window.mainWidget)
189
190         #box three
191         self.window.mainWidget=QFrame()
192         self.window.mainWidget.setLineWidth(1)
193         self.window.mainWidget.setFrameShape(QFrame.StyledPanel)
194         self.window.mainWidgetLayout=QVBoxLayout()
195         self.window.mainWidget.setLayout(self.window.mainWidgetLayout)
196
197         self.window.labelWidget=QWidget()
198         self.window.labelWidgetLayout=QHBoxLayout()
199         self.window.labelWidget.setLayout(self.window.labelWidgetLayout)
200         self.window.labelWidgetLayout.addWidget(QLabel("Highly
recommended"))
201
202         self.window.editWidget=QWidget()
203         self.window.editWidgetLayout=QGridLayout()
204         self.window.editWidget.setLayout(self.window.editWidgetLayout)
205         self.window.editWidgetLayout.addWidget(QLabel("Pump and
filter:"),0,0)
206
207         self.window.Radio1 = QRadioButton("Yes")
208         self.window.Radio1.setChecked(True)
209         self.window.Radio2 = QRadioButton("No")
210         self.window.ButtonGroup = QButtonGroup()
211
212         self.window.ButtonGroup.addButton(self.window.Radio1)
213         self.window.ButtonGroup.setId(self.window.Radio1,0)
214         self.window.ButtonGroup.addButton(self.window.Radio2)
215         self.window.ButtonGroup.setId(self.window.Radio2,1)
216         self.window.editWidgetLayout.addWidget(self.window.Radio1,0,1)
217         self.window.editWidgetLayout.addWidget(self.window.Radio2,0,2)
218
219         self.window.mainWidgetLayout.addWidget(self.window.labelWidget)
220         self.window.mainWidgetLayout.addWidget(self.window.editWidget)
221         self.window.layout.addWidget(self.window.mainWidget)
222
223         #buttons
224         self.createButton=QPushButton("Create")
225         self.createButton.setFixedWidth(100)
226         self.cancelButton=QPushButton("Cancel")
227         self.cancelButton.setFixedWidth(100)
228         self.blank=QWidget()
229         self.blank.setFixedWidth(2000)
230         self.buttonWidget=QWidget()
231         self.buttonLayout=QGridLayout()
232         self.buttonWidget.setLayout(self.buttonLayout)
233         self.buttonLayout.addWidget(self.blank,0,0)
234         self.buttonLayout.addWidget(self.createButton,0,1)
235         self.buttonLayout.addWidget(self.cancelButton,0,2)
236         self.window.layout.addWidget(self.buttonWidget)
237
238         def error():
239             self.errorWindow=QWidget()
240             self.errorWindow.setWindowTitle("Error")
241             self.errorWindow.setFixedWidth(250)
242             self.errorWindow.setFixedHeight(150)
243             self.errorWindow.show()
244             self.errorWindow.layout = QVBoxLayout()
245             self.errorWindow.setLayout(self.errorWindow.layout)
246             self.errorWindow.label=QLabel("Length, width and depth values
must all be greater than 0.")
247             self.errorWindow.label.setWordWrap(True)
248             self.errorWindow.widget=QWidget()
249             self.errorWindow.widgetLayout=QGridLayout()

```

```

250 self.errorWindow.widget.setLayout(self.errorWindow.widgetLayout)
251     self.errorWindow.okay=QPushButton("OK")
252     self.errorWindow.okay.setFixedWidth(50)
253     self.errorWindow.okay.clicked.connect(self.errorWindow.close)
254     blank=QWidget()
255     blank.setFixedWidth(100)
256     blankTwo=QWidget()
257     blankTwo.setFixedWidth(100)
258     self.errorWindow.widgetLayout.addWidget(blank,0,2)
259     self.errorWindow.widgetLayout.addWidget(blankTwo,0,0)
260
261 self.errorWindow.widgetLayout.addWidget(self.errorWindow.okay,0,1)
262     self.errorWindow.layout.addWidget(self.errorWindow.label)
263     self.errorWindow.layout.addWidget(self.errorWindow.widget)
264
265     def getInput():
266         depth=self.window.depthLine.text()
267         length=self.window.lengthLine.text()
268         width=self.window.widthLine.text()
269         temp=self.window.tempLine.text()
270         if int(depth) and int(length) and int(width)!=0:
271             result=self.window.ButtonGroup.checkedId()
272             if int(result)==0:
273                 pump=pond.Pump(status="good",pumpID=None)
274                 filter=pond.Filter(status="good",filterID=None)
275                 filter.saveFilter(db,cursor)
276                 filterID=filter.getFilterID()
277                 pump.savePump(db,cursor)
278                 pumpID=pump.getPumpID()
279             else:
280                 pump=None
281                 filter=None
282
283 self.mainPond=pond.Pond(pondID=None,depth=int(depth),width=int(width),temp=int(temp)
284 ),waterLevel=int(depth)-0.5,pump=pumpID,filter=filterID,days=0, length=length)
285     self.mainPond.savePond(db,cursor)
286     self.window.close()
287     self.createUI()
288     else:
289         error()
290 self.createButton.clicked.connect(getInput)
291 self.cancelButton.clicked.connect(self.window.close)
292
293 def addFishWindow():
294     self.window=QWidget()
295     self.window.setWindowTitle("Pond Simulator 3000 - Add Fish")
296     self.window.setFixedWidth(400)
297     self.window.setFixedHeight(300)
298     self.window.show()
299     self.window.layout = QVBoxLayout()
300     self.window.setLayout(self.window.layout)
301
302     self.window.dropLayout=QGridLayout()
303     self.window.widget=QWidget()
304     self.window.widget.setLayout(self.window.dropLayout)
305     self.window.dropLayout.addWidget(QLabel("Fish Type:"),0,0)
306     self.window.dropDown=QComboBox()
307     plantList=pondSimDatabase.fetchFishType(db,cursor)
308     for count in range(0,len(plantList)):
309         self.window.dropDown.addItem(plantList[count][0])
310     self.window.dropLayout.addWidget(self.window.dropDown,0,1)
311     self.window.layout.addWidget(self.window.widget)
312
313     self.window.editWidget=QWidget()
314     self.window.editLayout=QGridLayout()

```

```

313         self.window.editWidget.setLayout(self.window.editLayout)
314         self.nameEdit=QLineEdit()
315         self.nameEdit.setMaxLength(10)
316         self.window.editLayout.addWidget(QLabel("Fish Name:"),0,0)
317         self.window.editLayout.addWidget(self.nameEdit,0,1)
318
319         self.window.layout.addWidget(self.window.editWidget)
320
321         self.window.widget=QWidget()
322
323         def updateLabel():
324             text=str(self.window.dropDown.currentText())
325             if text!="Goldfish":
326                 goldfishLabel.hide()
327             else:
328                 goldfishLabel.show()
329             if text!="Rudd":
330                 ruddLabel.hide()
331             else:
332                 ruddLabel.show()
333             if text!="Tench":
334                 tenchLabel.hide()
335             else:
336                 tenchLabel.show()
337             if text!="Orfe":
338                 orfeLabel.hide()
339             else:
340                 orfeLabel.show()
341             if text!="Koi":
342                 koiLabel.hide()
343             else:
344                 koiLabel.show()
345             if text!="Shark":
346                 sharkLabel.hide()
347             else:
348                 sharkLabel.show()
349
350         def error():
351             self.errorWindow=QWidget()
352             self.errorWindow.setWindowTitle("Error")
353             self.errorWindow.setFixedWidth(250)
354             self.errorWindow.setFixedHeight(150)
355             self.errorWindow.show()
356             self.errorWindow.layout = QVBoxLayout()
357             self.errorWindow.setLayout(self.errorWindow.layout)
358             self.errorWindow.label=QLabel("You must open or create a pond
before adding anything")
359             self.errorWindow.label.setWordWrap(True)
360             self.errorWindow.widget=QWidget()
361             self.errorWindow.widgetLayout=QGridLayout()
362
363             self.errorWindow.widget.setLayout(self.errorWindow.widgetLayout)
364             self.errorWindow.okay=QPushButton("OK")
365             self.errorWindow.okay.setFixedWidth(50)
366             self.errorWindow.okay.clicked.connect(self.errorWindow.close)
367             blank=QWidget()
368             blank.setFixedWidth(100)
369             blankTwo=QWidget()
370             blankTwo.setFixedWidth(100)
371             self.errorWindow.widgetLayout.addWidget(blank,0,2)
372             self.errorWindow.widgetLayout.addWidget(blankTwo,0,0)
373
374             self.errorWindow.widgetLayout.addWidget(self.errorWindow.okay,0,1)
375             self.errorWindow.layout.addWidget(self.errorWindow.label)
376             self.errorWindow.layout.addWidget(self.errorWindow.widget)
377
378         def addPondFish():

```

```

377         try:
378             text=str(self.window.dropDown.currentText())
379             name=self.nameEdit.text()
380             self.mainPond.addFish(db,cursor,text,name)
381             self.window.close()
382             self.createUI()
383         except:
384             error()
385
386         self.window.labelLayout=QVBoxLayout()
387         self.window.widget.setLayout(self.window.labelLayout)
388
389         goldfishLabel=QLabel("Rivalling Koi, are goldfish. They are
possibly the most commonly kept garden fish, as they are small, inexpensive,
colourful, and very hardy. They generally have a long lifespan. In a pond, they can
often survive in icy weather conditions when ice forms on the surface, as long as
there is enough oxygen remaining in the water and the pond does not freeze solid.")
390         ruddLabel=QLabel("Rudd are a fantastic alternative to Orfe if you
have a smaller pond. Rudds are acclimatised to the British weather and therefore
are a hardy fish. They are a smaller growing surface fish which needs less oxygen.
Rudd generally only achieve a size of 5-6 inches (12-15cm), however, they can get
larger than this in large well filtered ponds.. Similar to Orfe, Rudd live much
happier in small shoals. Rudd also like a varied diet, consisting of pellets,
stick, and flaked dried preparations. They also like to feed from small larvae in
the pond.")
391         tenchLabel=QLabel("Tench are peaceful bottom feeding fish. . Tench
are acclimatised to British weather. Although when living in lakes and large pools
they grow to excess of 12 inches (30cm), rarely will they get as big as this in
most garden ponds.Whilst it is a bit of a myth that tench will clean the bottom of
your pond, they will disturb the silt and debris at the bottom, allowing the filter
system to remove the waste more efficiently. Tench will thrive in well filtered
large ponds, and are not ideally suited to small goldfish ponds. Tench are at their
most happiest with sinking foods, and love scavenging around the bottom for worms
and algae. The most common Tench are green, although golden and red can be
purchased.")
392         orfeLabel=QLabel("Orfe are fast growing, very active and a welcome
addition to a garden pond. Although they are suitable for ponds, they do have
slightly different requirements to goldfish. Orfe thrive well in faster moving
water that is highly oxygenated. As they are a surface dwelling fish, therefore
very oxygen dependant, ponds suitable for Orfe, should be well filtered, and
oxygenated. Although they can survive in still water, they are much happier in
moving water. Orfe can grow to lengths in excess of 24 inches (60cm), therefore
they do require more space than goldfish. Orfe need to be kept in small shoals.
They will feel much more secure, therefore the general health of the fish will be
much improved. Orfe like a varied diet, consisting of pellets, stick, and flaked
dried preparations. This fish will also be more than happy munching on small larvae
in the pond.")
393         koiLabel=QLabel("Koi fish are essentially an ornamental
domesticated version of the common carp, and are sometimes known as
Japanese Carp. They originate from Eastern Asia,
Aral, Black and Caspian Seas. They are closely related to goldfish. A koi is
considered a symbol of love and friendship. They are generally the most popular
fish in the world to keep in ponds.Koi are freshwater, bottom dwelling fish,
capable of living in a wide range of conditions. It is generally thought there are
THIRTEEN different classifications of Koi. The Japanese classify koi according to
various features, including colour, pattern, scale type and arrangement. However,
within each classification there are different types of Koi. Koi come in many
different colours and patterns, the more common colours are white, black, red,
yellow, blue, and cream.")
394         sharkLabel=QLabel("It's a shark. It will break your pond.")
395
396         goldfishLabel.setWordWrap(True)
397         ruddLabel.setWordWrap(True)
398         tenchLabel.setWordWrap(True)
399         orfeLabel.setWordWrap(True)
400         koiLabel.setWordWrap(True)
401         sharkLabel.setWordWrap(True)

```

```

402
403     self.window.labelLayout.addWidget(goldfishLabel)
404     self.window.labelLayout.addWidget(ruddLabel)
405     self.window.labelLayout.addWidget(tenchLabel)
406     self.window.labelLayout.addWidget(orfeLabel)
407     self.window.labelLayout.addWidget(koiLabel)
408     self.window.labelLayout.addWidget(sharkLabel)
409
410     self.window.layout.addWidget(self.window.widget)
411     updateLabel()
412     self.window.dropDown.currentIndexChanged.connect(updateLabel)
413     self.window.layout.addWidget(self.window.widget)
414
415     self.window.buttonLayout=QGridLayout()
416     self.window.buttonWidget=QWidget()
417     self.window.buttonWidget.setLayout(self.window.buttonLayout)
418     self.window.buttonLayout.addWidget(QWidget(),0,0)
419     self.window.addButton=QPushButton("Add")
420     self.window.cancelButton=QPushButton("Cancel")
421     self.window.addButton.clicked.connect(addPondFish)
422     self.window.cancelButton.clicked.connect(self.window.close)
423     self.window.buttonLayout.addWidget(self.window.addButton,0,1)
424     self.window.buttonLayout.addWidget(self.window.cancelButton,0,2)
425     self.window.layout.addWidget(self.window.buttonWidget)
426
427 def removeFishWindow():
428     def tree():
429         self.window.tree.setHeaderItem(self.window.header) #setting
header
430         pondID=self.mainPond.getPondID()
431         fishTup=pondSimDatabase.fetchFish(db, cursor, pondID)
432         for count in range(0,len(fishTup)):
433             self.window.treeList=QTreeWidgetItem()#list items
434             self.window.treeList.setText(0,str(fishTup[count][0]))
435             self.window.treeList.setText(1,str(fishTup[count][1]))
436             self.window.treeList.setText(2,str(fishTup[count][3]))
437             self.window.treeList.setText(3,str(fishTup[count][6]))
438             self.window.tree.addTopLevelItem(self.window.treeList)
439
440     def onRemove():
441         self.window.close() #closes window
442         self.selected=self.window.tree.currentItem()
443         ID=self.selected.data(0,0)
444         pondSimDatabase.deleteFish(db, cursor, ID)
445         self.createUI()
446
447     self.window=QWidget()
448     self.window.setWindowTitle("Pond Simulator 3000 - Open Pond")
449     self.window.setFixedWidth(600)
450     self.window.setFixedHeight(400)
451     self.window.show()
452     self.window.layout = QVBoxLayout()
453     self.window.setLayout(self.window.layout)
454
455     self.window.tree=QTreeWidgetItem()
456     self.window.tree.setColumnCount(4)
457     self.window.tree.setColumnWidth(0,80)
458
459     self.window.header=QTreeWidgetItem() #headers
460     self.window.header.setText(0,'ID')
461     self.window.header.setText(1,'Name')
462     self.window.header.setText(2,'Status')
463     self.window.header.setText(3,'Type')
464     tree()
465
466     self.window.button=QPushButton("Cancel") #making button
467     self.window.button.setFixedWidth(100) #changing button size

```



```

468         self.window.button.layout=QHBoxLayout() #creating layout for button
469         self.window.buttonWidget=QWidget() #creating blank widget
470         self.window.blank=QWidget()
471         self.window.buttonWidget.setLayout(self.window.button.layout)
#setting layout to blank widget
472         self.window.button.layout.addWidget(self.window.blank) #adding
button layout
473         self.window.button.layout.addWidget(self.window.button) #adding
button layout
474
475         self.window.button.clicked.connect(self.window.close) #when button
is clicked, window is hidden
476         self.window.layout.addWidget(self.window.tree) #adding tree list to
layout
477         self.window.layout.addWidget(self.window.buttonWidget) #adding
button to layout
478         self.window.tree.itemDoubleClicked.connect(onRemove)#when item
double clicked
479
480     def addPlantWindow():
481         self.window=QWidget()
482         self.window.setWindowTitle("Pond Simulator 3000 - Add Plant")
483         self.window.setFixedWidth(400)
484         self.window.setFixedHeight(300)
485         self.window.show()
486         self.window.layout = QVBoxLayout()
487         self.window.setLayout(self.window.layout)
488
489         self.window.dropLayout=QGridLayout()
490         self.window.widget=QWidget()
491         self.window.widget.setLayout(self.window.dropLayout)
492         self.window.dropLayout.addWidget(QLabel("Plant Type:"),0,0)
493         self.window.dropDown=QComboBox()
494         plantList=pondSimDatabase.fetchPlantType(db,cursor)
495         for count in range(0,len(plantList)):
496             self.window.dropDown.addItem(plantList[count][0])
497         self.window.dropLayout.addWidget(self.window.dropDown,0,1)
498         self.window.layout.addWidget(self.window.widget)
499
500     def updateLabel():
501         text=str(self.window.dropDown.currentText())
502         if text!="Rooted Floater":
503             rootedLabel.hide()
504         else:
505             rootedLabel.show()
506         if text!="True Floater":
507             trueLabel.hide()
508         else:
509             trueLabel.show()
510         if text!="Emergent":
511             emergentLabel.hide()
512         else:
513             emergentLabel.show()
514         if text!="Submersed":
515             submersedLabel.hide()
516         else:
517             submersedLabel.show()
518
519     def error():
520         self.errorWindow=QWidget()
521         self.errorWindow.setWindowTitle("Error")
522         self.errorWindow.setFixedWidth(250)
523         self.errorWindow.setFixedHeight(150)
524         self.errorWindow.show()
525         self.errorWindow.layout = QVBoxLayout()
526         self.errorWindow.setLayout(self.errorWindow.layout)

```

```
527         self.errorWindow.label=QLabel("You must open or create a pond
before adding anything")
528         self.errorWindow.label.setWordWrap(True)
529         self.errorWindow.widget=QWidget()
530         self.errorWindow.widgetLayout=QGridLayout()
531
self.errorWindow.widget.setLayout(self.errorWindow.widgetLayout)
532         self.errorWindow.okay=QPushButton("OK")
533         self.errorWindow.okay.setFixedWidth(50)
534         self.errorWindow.okay.clicked.connect(self.errorWindow.close)
535         blank=QWidget()
536         blank.setFixedWidth(100)
537         blankTwo=QWidget()
538         blankTwo.setFixedWidth(100)
539         self.errorWindow.widgetLayout.addWidget(blank,0,2)
540         self.errorWindow.widgetLayout.addWidget(blankTwo,0,0)
541
self.errorWindow.widgetLayout.addWidget(self.errorWindow.okay,0,1)
542         self.errorWindow.layout.addWidget(self.errorWindow.label)
543         self.errorWindow.layout.addWidget(self.errorWindow.widget)
544
545     def addPondPlant():
546         try:
547             text=str(self.window.dropDown.currentText())
548             self.mainPond.addPlant(db,cursor,text)
549             self.window.close()
550             self.createUI()
551         except:
552             error()
553
554         self.window.widget=QWidget()
555         self.window.labelLayout=QVBoxLayout()
556         self.window.widget.setLayout(self.window.labelLayout)
557
558         rootedLabel=QLabel("1")
559         trueLabel=QLabel("2")
560         emergentLabel=QLabel("3")
561         submersedLabel=QLabel("4")
562
563         rootedLabel.setWordWrap(True)
564         trueLabel.setWordWrap(True)
565         emergentLabel.setWordWrap(True)
566         submersedLabel.setWordWrap(True)
567
568         self.window.labelLayout.addWidget(rootedLabel)
569         self.window.labelLayout.addWidget(trueLabel)
570         self.window.labelLayout.addWidget(emergentLabel)
571         self.window.labelLayout.addWidget(submersedLabel)
572
573         self.window.layout.addWidget(self.window.widget)
574         updateLabel()
575         self.window.dropDown.currentIndexChanged.connect(updateLabel)
576         self.window.layout.addWidget(self.window.widget)
577
578         self.window.buttonLayout=QGridLayout()
579         self.window.buttonWidget=QWidget()
580         self.window.buttonWidget.setLayout(self.window.buttonLayout)
581         self.window.buttonLayout.addWidget(QWidget(),0,0)
582         self.window.addButton=QPushButton("Add")
583         self.window.cancelButton=QPushButton("Cancel")
584         self.window.addButton.clicked.connect(addPondPlant)
585         self.window.cancelButton.clicked.connect(self.window.close)
586         self.window.buttonLayout.addWidget(self.window.addButton,0,1)
587         self.window.buttonLayout.addWidget(self.window.cancelButton,0,2)
588         self.window.layout.addWidget(self.window.buttonWidget)
589
590 #####
```

```
591
592     def removePlantWindow():
593         def tree():
594             self.window.tree.setHeaderItem(self.window.header) #setting
header
595             pondID=self.mainPond.getPondID()
596             plantTup=pondSimDatabase.fetchPlants(db, cursor, pondID)
597             for count in range(0,len(plantTup)):
598                 self.window.treeList=QTreeWidgetItem()#list items
599                 self.window.treeList.setText(0,str(plantTup[count][0]))
600                 self.window.treeList.setText(1,str(plantTup[count][2]))
601                 self.window.treeList.setText(2,str(plantTup[count][1]))
602                 self.window.tree.addTopLevelItem(self.window.treeList)
603
604         def onRemove():
605             self.window.close() #closes window
606             self.selected=self.window.tree.currentItem()
607             ID=self.selected.data(0,0)
608             pondSimDatabase.deletePlant(db, cursor, ID)
609             self.createUI()
610
611         self.window=QWidget()
612         self.window.setWindowTitle("Pond Simulator 3000 - Remove Plant")
613         self.window.setFixedWidth(600)
614         self.window.setFixedHeight(400)
615         self.window.show()
616         self.window.layout = QVBoxLayout()
617         self.window.setLayout(self.window.layout)
618
619         self.window.tree=QTreeWidgetItem()
620         self.window.tree.setColumnCount(3)
621         self.window.tree.setColumnWidth(0,80)
622
623         self.window.header=QTreeWidgetItem() #headers
624         self.window.header.setText(0,'ID')
625         self.window.header.setText(1,'Status')
626         self.window.header.setText(2,'Type')
627         tree()
628
629         self.window.button=QPushButton("Cancel") #making button
630         self.window.button.setFixedWidth(100) #changing button size
631         self.window.button.layout=QHBoxLayout() #creating layout for button
632         self.window.buttonWidget=QWidget() #creating blank widget
633         self.window.blank=QWidget()
634         self.window.buttonWidget.setLayout(self.window.button.layout)
#setting layout to blank widget
635         self.window.button.layout.addWidget(self.window.blank) #adding
button layout
636         self.window.button.layout.addWidget(self.window.button) #adding
button layout
637
638         self.window.button.clicked.connect(self.window.close) #when button
is clicked, window is hidden
639         self.window.layout.addWidget(self.window.tree) #adding tree list to
layout
640         self.window.layout.addWidget(self.window.buttonWidget) #adding
button to layout
641         self.window.tree.itemDoubleClicked.connect(onRemove)#when item
double clicked
642
643
644         def incrementDay():
645             pond.incrementDay(self.mainPond,db,cursor)
646
647         def editPondWindow():
648             try:
649                 self.mainPond.updatePond(db,cursor)
```

```
650         self.window=QWidget()
651         self.window.setWindowTitle("Pond Simulator 3000 - Edit Pond")
652         self.window.setFixedWidth(600)
653         self.window.setFixedHeight(400)
654         self.window.show()
655         self.window.layout = QVBoxLayout()
656         self.window.setLayout(self.window.layout)
657
658         #box one
659         self.window.mainWidget=QFrame()
660         self.window.mainWidget.setLineWidth(1)
661         self.window.mainWidget.setFrameShape(QFrame.StyledPanel)
662         self.window.mainWidgetLayout=QVBoxLayout()
663         self.window.mainWidget.setLayout(self.window.mainWidgetLayout)
664
665         self.window.labelWidget=QWidget()
666         self.window.labelWidgetLayout=QHBoxLayout()
667
668         self.window.labelWidget.setLayout(self.window.labelWidgetLayout)
669         self.window.labelWidgetLayout.addWidget(QLabel("The pond can be
670 a maxium of 10 x 10 x 10 metres"))
671
672         self.window.editWidget=QWidget()
673         self.window.editWidgetLayout=QGridLayout()
674         self.window.editWidget.setLayout(self.window.editWidgetLayout)
675         self.window.editWidgetLayout.addWidget(QLabel("Depth:"),0,0)
676         self.window.depthLine=QSpinBox()
677         self.window.depthLine.setMaximum(10)
678         self.window.depthLine.setMinimum(0.5)
679
680         self.window.editWidgetLayout.addWidget(self.window.depthLine,0,1)
681         self.window.editWidgetLayout.addWidget(QLabel("Width:"),0,2)
682         self.window.widthLine=QSpinBox()
683         self.window.widthLine.setMaximum(10)
684         self.window.widthLine.setMinimum(0.5)
685
686         self.window.editWidgetLayout.addWidget(self.window.widthLine,0,3)
687         self.window.editWidgetLayout.addWidget(QLabel("Length:"),0,4)
688         self.window.lengthLine=QSpinBox()
689         self.window.lengthLine.setMaximum(10)
690         self.window.lengthLine.setMinimum(0.5)
691
692         self.window.editWidgetLayout.addWidget(self.window.lengthLine,0,5)
693
694         self.window.mainWidgetLayout.addWidget(self.window.labelWidget)
695         self.window.mainWidgetLayout.addWidget(self.window.editWidget)
696         self.window.layout.addWidget(self.window.mainWidget)
697
698         #box two
699         self.window.mainWidget=QFrame()
700         self.window.mainWidget.setLineWidth(1)
701         self.window.mainWidget.setFrameShape(QFrame.StyledPanel)
702         self.window.mainWidgetLayout=QVBoxLayout()
703         self.window.mainWidget.setLayout(self.window.mainWidgetLayout)
704
705         self.window.labelWidget=QWidget()
706         self.window.labelWidgetLayout=QHBoxLayout()
707
708         self.window.labelWidget.setLayout(self.window.labelWidgetLayout)
709         self.window.labelWidgetLayout.addWidget(QLabel("The water
710 temperature must be between 27 and +50C."))
711
712         self.window.editWidget=QWidget()
713         self.window.editWidgetLayout=QGridLayout()
714         self.window.editWidget.setLayout(self.window.editWidgetLayout)
715
716         self.window.editWidgetLayout.addWidget(QLabel("Temperature:"),0,0)
```

```

709         self.window.tempLine=QSpinBox()
710         self.window.tempLine.setMaximum (50)
711         self.window.tempLine.setMinimum(-27)
712
713     self.window.editWidgetLayout.addWidget(self.window.tempLine,0,1)
714     self.window.blank=QWidget()
715     self.window.editWidgetLayout.addWidget(self.window.blank,0,2)
716
717     self.window.mainWidgetLayout.addWidget(self.window.labelWidget)
718     self.window.mainWidgetLayout.addWidget(self.window.editWidget)
719     self.window.layout.addWidget(self.window.mainWidget)
720
721     #buttons
722     self.createButton=QPushButton("Create")
723     self.createButton.setFixedWidth(100)
724     self.cancelButton=QPushButton("Cancel")
725     self.cancelButton.setFixedWidth(100)
726     self.blank=QWidget()
727     self.blank.setFixedWidth(2000)
728     self.buttonWidget=QWidget()
729     self.buttonLayout=QGridLayout()
730     self.buttonWidget.setLayout(self.buttonLayout)
731     self.buttonLayout.addWidget(self.blank,0,0)
732     self.buttonLayout.addWidget(self.createButton,0,1)
733     self.buttonLayout.addWidget(self.cancelButton,0,2)
734     self.window.layout.addWidget(self.buttonWidget)
735
736     def error():
737         self.errorWindow=QWidget()
738         self.errorWindow.setWindowTitle("Error")
739         self.errorWindow.setFixedWidth(250)
740         self.errorWindow.setFixedHeight(150)
741         self.errorWindow.show()
742         self.errorWindow.layout = QVBoxLayout()
743         self.errorWindow.setLayout(self.errorWindow.layout)
744         self.errorWindow.label=QLabel("Length, width and depth
values must all be greater than 0.")
745         self.errorWindow.label.setWordWrap(True)
746         self.errorWindow.widget=QWidget()
747         self.errorWindow.widgetLayout=QGridLayout()
748
749     self.errorWindow.widget.setLayout(self.errorWindow.widgetLayout)
750     self.errorWindow.okay=QPushButton("OK")
751     self.errorWindow.okay.setFixedWidth(50)
752
753     self.errorWindow.okay.clicked.connect(self.errorWindow.close)
754     blank=QWidget()
755     blank.setFixedWidth(100)
756     blankTwo=QWidget()
757     blankTwo.setFixedWidth(100)
758     self.errorWindow.widgetLayout.addWidget(blank,0,2)
759     self.errorWindow.widgetLayout.addWidget(blankTwo,0,0)
760
761     self.errorWindow.widgetLayout.addWidget(self.errorWindow.okay,0,1)
762     self.errorWindow.layout.addWidget(self.errorWindow.label)
763     self.errorWindow.layout.addWidget(self.errorWindow.widget)
764
765     def getInput():
766         #getting user inputs from screen
767         depth=self.window.depthLine.text()
768         length=self.window.lengthLine.text()
769         width=self.window.widthLine.text()
770         temp=self.window.tempLine.text()
771         #setting inputs to pond
772         if int(depth)!=0 and int(length)!=0 and int(width)!=0:
773             self.mainPond.setPondDepth(int(depth))

```

```

771         self.mainPond.setPondLength(int(length))
772         self.mainPond.setPondWidth(int(width))
773         self.mainPond.setWaterTemp(int(temp))
774         self.mainPond.setWaterLevel(int(depth)-0.5)
775
776         #saving pond
777         self.mainPond.updatePond(db,cursor)
778         self.window.close()
779         #updated GUI
780         self.createUI()
781     else:
782         error() #error message displays if any of the
dimentions are 0
783
784     #button events
785     self.createButton.clicked.connect(getInput)
786     self.cancelButton.clicked.connect(self.window.close)
787
788     except:
789         self.errorWindow=QWidget()
790         self.errorWindow.setWindowTitle("Error")
791         self.errorWindow.setFixedWidth(250)
792         self.errorWindow.setFixedHeight(150)
793         self.errorWindow.show()
794         self.errorWindow.layout = QVBoxLayout()
795         self.errorWindow.setLayout(self.errorWindow.layout)
796         self.errorWindow.label=QLabel("You must open a pond before you
can edit it")
797         self.errorWindow.label.setWordWrap(True)
798         self.errorWindow.widget=QWidget()
799         self.errorWindow.widgetLayout=QGridLayout()
800
801         self.errorWindow.widget.setLayout(self.errorWindow.widgetLayout)
802         self.errorWindow.okay=QPushButton("OK")
803         self.errorWindow.okay.setFixedWidth(50)
804         self.errorWindow.okay.clicked.connect(self.errorWindow.close)
805         blank=QWidget()
806         blank.setFixedWidth(100)
807         blankTwo=QWidget()
808         blankTwo.setFixedWidth(100)
809         self.errorWindow.widgetLayout.addWidget(blank,0,2)
810         self.errorWindow.widgetLayout.addWidget(blankTwo,0,0)
811         self.errorWindow.widgetLayout.addWidget(self.errorWindow.okay,0,1)
812         self.errorWindow.layout.addWidget(self.errorWindow.label)
813         self.errorWindow.layout.addWidget(self.errorWindow.widget)
814
815     def autoFeedWindow():
816         pond.automaticGrow(db,cursor,self.mainPond)
817         self.window=QWidget()
818         self.window.setWindowTitle("Pond Simulator 3000 - Feed Fish")
819         self.window.setFixedWidth(350)
820         self.window.setFixedHeight(200)
821         self.window.show()
822         self.window.layout = QGridLayout()
823         self.window.setLayout(self.window.layout)
824         self.blank=QWidget()
825         self.blank.setFixedWidth(130)
826         self.window.layout.addWidget(self.blank,0,0)
827         #self.window.layout.addWidget(self.blank,0,2)
828         self.window.layout.addWidget(QLabel("Success!"),0,1)
829         button=QPushButton("OK")
830         button.clicked.connect(self.window.close)
831         self.window.layout.addWidget(button,1,1)
832
833

```

```
834     def manualFeedWindow():
835         def manualGrow():
836             amount=self.line.text()
837             amount=int(amount)
838             pond.manualGrow(amount,self.mainPond,db,cursor)
839             self.window.close()
840
841         self.window=QWidget()
842         self.window.setWindowTitle("Pond Simulator 3000 - Feed Fish")
843         self.window.setFixedWidth(600)
844         self.window.setFixedHeight(400)
845         self.window.show()
846         self.window.layout = QVBoxLayout()
847         self.window.setLayout(self.window.layout)
848
849         totalHunger=0
850
851     hunger=pondSimDatabase.fetchFishInfo(db,cursor,self.mainPond.getPondID())
852     for count in range(0,len(hunger)):
853         try:
854             totalHunger=hunger[count][2]+totalHunger
855         except:
856             pass
857
858     self.window.layout.addWidget(QLabel("Recommended amount to feed
fish:"+str(totalHunger)))
859
860     self.editWidget=QWidget()
861     self.editWidget.layout=QGridLayout()
862     self.editWidget.setLayout(self.editWidget.layout)
863     self.editWidget.layout.addWidget(QLabel("Enter the amount of food
you would like to add to the pond:"),0,0)
864     self.line=QSpinBox()
865     self.line.setMaximum(400)
866     self.line.setMinimum(0)
867     self.editWidget.layout.addWidget(self.line,0,1)
868     self.window.layout.addWidget(self.editWidget)
869     feedButton=QPushButton("Feed")
870     cancelButton=QPushButton("Cancel")
871     self.editWidget.layout.addWidget(feedButton,1,0)
872     self.editWidget.layout.addWidget(cancelButton,1,1)
873     feedButton.clicked.connect(manualGrow)
874     cancelButton.clicked.connect(self.window.close)
875
876
877 #-----
-----
878     #setting title
879     self.setWindowTitle('Pond Simulator 3000')
880
881     #setting fixed screen size
882     self.setFixedWidth(800)
883     self.setFixedHeight(600)
884
885     #creating menubar
886     menu=QMenuBar()
887     self.setMenuBar(menu)
888
889     #adding menus
890     fileMenu=menu.addMenu("File")
891     manageMenu=menu.addMenu("Mange pond")
892
893     #adding actions to file menu
894     newAction=fileMenu.addAction("New Pond")
895     openAction=fileMenu.addAction("Open Pond")
896     saveAction=fileMenu.addAction("Save Pond")
```

```
897         exitAction=fileMenu.addAction("Exit program")
898
899         #adding actions to manage menu
900         mpaf=manageMenu.addMenu("Manage pump and filter")
901         eps=manageMenu.addAction("Edit pond settings")
902         increment=manageMenu.addAction("Increment day")
903         mpl=manageMenu.addMenu("Manage pond life")
904         mpaf.addAction("Add pump and filter")
905         mpaf.addAction("Remove pump and filter")
906         mpaf.addAction("Edit pump and filter")
907         addFish=mpl.addAction("Add fish")
908         addPlant=mpl.addAction("Add plant")
909         removeFish=mpl.addAction("Remove fish")
910         removePlant=mpl.addAction("Remove plant")
911         ffm=mpl.addMenu("Feed fish")
912         autoFeed=ffm.addAction("Automatically feed")
913         manualFeed=ffm.addAction("Manually feed")
914
915         #file menu action events
916         exitAction.triggered.connect(application.closeAllWindows)
917         openAction.triggered.connect(openWindow)
918         saveAction.triggered.connect(savePond)
919         newAction.triggered.connect(newPondWindow)
920
921         #manage menu action events
922         addFish.triggered.connect(addFishWindow)
923         addPlant.triggered.connect(addPlantWindow)
924         increment.triggered.connect(incrementDay)
925         eps.triggered.connect(editPondWindow)
926         removeFish.triggered.connect(removeFishWindow)
927         removePlant.triggered.connect(removePlantWindow)
928         autoFeed.triggered.connect(autoFeedWindow)
929         manualFeed.triggered.connect(manualFeedWindow)
930
931         #creating main layout
932         self.layout = QGridLayout()
933         self.mainWidget = QWidget()
934         self.mainWidget.setLayout(self.layout)
935
936         #creating graphics scenes for each widget
937         self.meter=QGraphicsScene()
938         self.graphics=QGraphicsScene()
939         self.graphics.setSceneRect(0,0,550,400)
940         self.scene4=QGraphicsScene()
941         self.scene4.setSceneRect(0,0,550,150)
942         #making each scene viewable
943         self.meterView=QGraphicsView(self.meter)
944         self.graphicsView=QGraphicsView(self.graphics)
945         self.view4=QGraphicsView(self.scene4)
946
947         #adding each viewable scene to its respective widget
948         self.layout.addWidget(self.meterView,0,0)
949         self.layout.addWidget(self.graphicsView,0,1)
950         self.layout.addWidget(self.view4,1,1)
951
952         #setting the main widget
953         self.setCentralWidget(self.mainWidget)
954
955         #-----GUI for graphical display-----
956         try:
957             pump=pondSimDatabase.fetchPump(db,cursor,self.mainPond.getPump())
#fetching pump data
958             pump=pond.Pump(pump[0][1], pump[0][0]) #creating pump object from
pump data
959             pumpStatus=pump.getPumpStatus()
960         except:
961             pumpStatus=""
```



```
962
963         try:
964 filter=pondSimDatabase.fetchFilter(db,cursor,self.mainPond.getFilter()) #fetching
pump data
965             filter=pond.Filter(filter[0][1], filter[0][0]) #creating pump
object from pump data
966             filterStatus=filter.getFilterStatus()
967         except:
968             filterStatus=""
969
970         if filterStatus=="Broken":
971             self.graphics.background=QPixmap("background2.png")
972         else:
973             self.graphics.background=QPixmap("background.png") #creates
background
974
self.graphics.backgroundItem=self.graphics.addPixmap(self.graphics.background)
975
976         def addFish():
977 fishResults=pondSimDatabase.fetchFish(db,cursor,self.mainPond.getPondID())
978             for count in range(0,len(fishResults)):
979                 if fishResults[count][6]=="Goldfish":
980                     if fishResults[count][3]=="good":
981                         self.graphics.fish=QPixmap("fish.png")
982                     elif fishResults[count][3]=="Dead":
983                         self.graphics.fish=QPixmap("dead.png")
984
self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
985
self.graphics.fishItem.setPos(random.randrange(1,549),random.randrange(1,399))
#adds the fish graphic at a random location
986                 self.graphics.fishItem.z=2
987
988                 elif fishResults[count][6]=="Tench":
989                     if fishResults[count][3]=="good":
990                         self.graphics.fish=QPixmap("fish.png")
991                     elif fishResults[count][3]=="Dead":
992                         self.graphics.fish=QPixmap("dead.png")
993
self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
994
self.graphics.fishItem.setPos(random.randrange(1,549),random.randrange(1,399))
#adds the fish graphic at a random location
995                 self.graphics.fishItem.z=2
996
997                 elif fishResults[count][6]=="Orfe":
998                     if fishResults[count][3]=="good":
999                         self.graphics.fish=QPixmap("fish.png")
1000                     elif fishResults[count][3]=="Dead":
1001                         self.graphics.fish=QPixmap("dead.png")
1002
self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
1003
self.graphics.fishItem.setPos(random.randrange(1,549),random.randrange(1,399))
#adds the fish graphic at a random location
1004                 self.graphics.fishItem.z=2
1005
1006                 elif fishResults[count][6]=="Koi":
1007                     if fishResults[count][3]=="good":
1008                         self.graphics.fish=QPixmap("fish.png")
1009                     elif fishResults[count][3]=="Dead":
1010                         self.graphics.fish=QPixmap("dead.png")
1011
self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
```

```

1012
self.graphics.fishItem.setPos(random.randrange(1,549),random.randrange(1,399))
#adds the fish graphic at a random location
1013         self.graphics.fishItem.z=2
1014
1015         elif fishResults[count][6]=="Rudd":
1016             if fishResults[count][3]=="good":
1017                 self.graphics.fish=QPixmap("fish.png")
1018             elif fishResults[count][3]=="Dead":
1019                 self.graphics.fish=QPixmap("dead.png")
1020
self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
1021
self.graphics.fishItem.setPos(random.randrange(1,549),random.randrange(1,399))
#adds the fish graphic at a random location
1022         self.graphics.fishItem.z=2
1023
1024         elif fishResults[count][6]=="Shark":
1025             if fishResults[count][3]=="good":
1026                 self.graphics.fish=QPixmap("shark.png")
1027             elif fishResults[count][3]=="Dead":
1028                 self.graphics.fish=QPixmap("sharkDead.png")
1029
self.graphics.fishItem=self.graphics.addPixmap(self.graphics.fish)
1030         self.graphics.fishItem.setPos(random.randrange(-
10,10),random.randrange(-20,20)) #adds the fish graphic at a random location
1031         self.graphics.fishItem.z=2
1032
1033
1034     def addPlants():
1035
plantResult=pondSimDatabase.fetchPlants(db,cursor,self.mainPond.getPondID())
1036         for count in range(0,len(plantResult)):
1037             if plantResult[count][1]=="Rooted Floater":
1038                 if plantResult[count][2]=="good":
1039                     self.graphics.plant=QPixmap("rootedFloater.png")
1040                 elif plantResult[count][2]=="Dead":
1041                     self.graphics.plant=QPixmap("rootedFloater.png")
1042
1043             elif plantResult[count][1]=="True Floater":
1044                 if plantResult[count][2]=="good":
1045                     self.graphics.plant=QPixmap("trueFloater.png")
1046                 elif plantResult[count][2]=="Dead":
1047                     self.graphics.plant=QPixmap("trueFloater.png")
1048
1049             elif plantResult[count][1]=="Emergent":
1050                 if plantResult[count][2]=="good":
1051                     self.graphics.plant=QPixmap("rootedFloater.png")
1052                 elif plantResult[count][2]=="Dead":
1053                     self.graphics.plant=QPixmap("rootedFloater.png")
1054
1055             elif plantResult[count][1]=="Submersed":
1056                 if plantResult[count][2]=="good":
1057                     self.graphics.plant=QPixmap("rootedFloater.png")
1058                 elif plantResult[count][2]=="Dead":
1059                     self.graphics.plant=QPixmap("rootedFloater.png")
1060
self.graphics.plantItem=self.graphics.addPixmap(self.graphics.plant)
1061         self.graphics.plantItem.setPos(random.randrange(1,549),0)
1062         self.graphics.plantItem.z=0
1063
1064
1065     try:
1066         addFish()
1067     except:
1068         pass
1069

```

```
1070         try:
1071             addPlants()
1072         except:
1073             pass
1074
1075 #-----GUI for pump and filter status-----
1076         self.widgetF=QWidget() #creating widget
1077         try:
1078             pump=pondSimDatabase.fetchPump(db,cursor,self.mainPond.getPump())
#fetching pump data
1079             pump=pond.Pump(pump[0][1], pump[0][0]) #creating pump object from
pump data
1080             pumpStatus=pump.getPumpStatus()
1081         except:
1082             pumpStatus=""
1083
1084         try:
1085             filter=pondSimDatabase.fetchFilter(db,cursor,self.mainPond.getFilter()) #fetching
pump data
1086             filter=pond.Filter(filter[0][1], filter[0][0]) #creating pump
object from pump data
1087             filterStatus=filter.getFilterStatus()
1088         except:
1089             filterStatus=""
1090
1091         self.pumpLabel=QLabel("Pump: "+pumpStatus)#making labels
1092         self.filterLabel=QLabel("Filter: "+filterStatus)
1093         self.layoutF=QGridLayout() #making layout
1094         self.widgetF.setLayout(self.layoutF) #giving widget layout
1095         self.layoutF.addWidget(self.pumpLabel,0,0) #adding labelsto layout
1096         self.layoutF.addWidget(self.filterLabel,1,0)
1097         self.layout.addWidget(self.widgetF,1,0) #adding widget to main layout
1098
1099 #-----GUI for meters-----
1100         #water meter outline
1101         self.meter.waterMeter=QRectF(10,-50,50,200) #creating rect
1102         self.meter.waterMeter.pen=QPen() #creating pen
1103         self.meter.waterMeter.color=QColor() #creating colour for pen
1104         self.meter.waterMeter.color.setNamedColor('#DCDCDC')
1105         self.meter.waterMeter.pen.setColor(self.meter.waterMeter.color)
#setting color of pen
1106         self.meter.waterMeter.pen.setWidth(2) #setting line width
1107         self.meter.waterMeter.brush=QBrush() #creating blank fill
1108
1109         self.meter.addRect(self.meter.waterMeter,self.meter.waterMeter.pen,self.meter.water
Meter.brush) #drawing rect
1110
1111         #temperature meter outline
1112         self.meter.tempMeter=QRectF(120,-50,50,200)
1113         self.meter.tempMeter.pen=QPen()
1114         self.meter.tempMeter.color=QColor() #creating colour for pen
1115         self.meter.tempMeter.color.setNamedColor('#E04006')
1116         self.meter.tempMeter.pen.setColor(self.meter.tempMeter.color) #setting
color of pen
1117         self.meter.tempMeter.pen.setWidth(2) #setting line width
1118         self.meter.tempMeter.brush=QBrush()
1119
1120         self.meter.addRect(self.meter.tempMeter,self.meter.tempMeter.pen,self.meter.tempMet
er.brush)
1121
1122         #water meter fill
1123         try:
1124             number=float(self.mainPond.getWaterLevel())
1125             number=number*20
1126         except:
```

```
1126         number=0
1127     self.meter.waterMeterFill=QRectF(11,-50,47,number)
1128     self.meter.waterMeterFill.moveBottom(148)
1129     self.meter.waterMeterFill.pen=QPen() #creating pen
1130     self.meter.waterMeterFill.color=QColor() #creating colour for pen
1131     self.meter.waterMeterFill.color.setNamedColor('#FFFFFF')
1132     self.meter.color=QColor()
1133     self.meter.color.setNamedColor('lightseagreen')
1134     self.meter.waterMeterFill.pen.setColor(self.meter.waterMeterFill.color)
#setting color of pen
1135     self.meter.waterMeterFill.brush=QBrush() #creating blank fill
1136     self.meter.waterMeterFill.brush.setColor(self.meter.color)
1137     self.meter.waterMeterFill.brush.setStyle(Qt.SolidPattern)
1138
self.meter.addRect(self.meter.waterMeterFill,self.meter.waterMeterFill.pen,self.meter.waterMeterFill.brush) #drawing rect
1139
1140     #water temperature fill
1141     try:
1142         number=float(self.mainPond.getWaterTemp())
1143         if number>=0:
1144             number=number*2
1145         else:
1146             number=0
1147     except:
1148         number=0
1149     self.meter.tempMeterFill=QRectF(122,-50,45,number)
1150     self.meter.tempMeterFill.moveBottom(48)
1151     self.meter.tempMeterFill.pen=QPen() #creating pen
1152     color=QColor() #creating colour for pen
1153     color.setNamedColor('#FFFFFF')
1154     self.meter.tempMeterFill.color=QColor()
1155     self.meter.tempMeterFill.color.setNamedColor('red')
1156     self.meter.tempMeterFill.pen.setColor(self.meter.tempMeterFill.color)
#setting color of pen
1157     self.meter.tempMeterFill.brush=QBrush() #creating blank fill
1158     self.meter.tempMeterFill.brush.setColor(self.meter.tempMeterFill.color)
1159     self.meter.tempMeterFill.brush.setStyle(Qt.SolidPattern)
1160
self.meter.addRect(self.meter.tempMeterFill,self.meter.tempMeterFill.pen,self.meter.tempMeterFill.brush) #drawing rect
1161
1162     #water temperature minus fill
1163     try:
1164         number=float(self.mainPond.getWaterTemp())
1165         if number<0:
1166             number=-number*2
1167         else:
1168             number=100
1169     except:
1170         number=0
1171     self.meter.tempMeterFill=QRectF(122,-50,45,number)
1172     self.meter.tempMeterFill.moveBottom(147)
1173     self.meter.tempMeterFill.pen=QPen() #creating pen
1174     color=QColor() #creating colour for pen
1175     color.setNamedColor('#FFFFFF')
1176     self.meter.tempMeterFill.color=QColor()
1177     self.meter.tempMeterFill.color.setNamedColor('blue')
1178     self.meter.tempMeterFill.pen.setColor(self.meter.tempMeterFill.color)
#setting color of pen
1179     self.meter.tempMeterFill.brush=QBrush() #creating blank fill
1180     self.meter.tempMeterFill.brush.setColor(self.meter.tempMeterFill.color)
1181     self.meter.tempMeterFill.brush.setStyle(Qt.SolidPattern)
1182
self.meter.addRect(self.meter.tempMeterFill,self.meter.tempMeterFill.pen,self.meter.tempMeterFill.brush) #drawing rect
1183
```

```

1184         self.widgetM=QWidget() #creating widget
1185         try:
1186             level=str(self.mainPond.getWaterLevel())
1187         except:
1188             level=""
1189         try:
1190             temp=str(self.mainPond.getWaterTemp())
1191         except:
1192             temp=""
1193         self.waterLabel=QLabel("Water level: "+level)#making labels
1194         self.tempLabel=QLabel("Temperature: "+temp)
1195         self.zeroLabel=QLabel("  0")
1196         self.zeroLabel.setFixedHeight(8)
1197         self.fiftyLabel=QLabel("  50")
1198         self.fiftyLabel.setFixedHeight(185)
1199         self.sevenLabel=QLabel(" -27")
1200         self.sevenLabel.setFixedHeight(160)
1201         self.layoutM=QGridLayout() #making layout
1202         self.widgetM.setLayout(self.layoutM) #giving widget layout
1203         self.layoutM.addWidget(self.waterLabel,3,0) #adding labels to layout
1204         self.layoutM.addWidget(self.tempLabel,3,1)
1205         self.layoutM.addWidget(self.fiftyLabel,0,1)
1206         self.layoutM.addWidget(self.zeroLabel,1,1)
1207         self.layoutM.addWidget(self.sevenLabel,2,1)
1208         self.layout.addWidget(self.widgetM,0,0) #adding widget to main layout
1209
1210#-----GUI for about the pond-----
1211         try:
1212
1213         fishList=pondSimDatabase.fetchFish(db,cursor,self.mainPond.getPondID())
1214
1215         plantList=pondSimDatabase.fetchPlants(db,cursor,self.mainPond.getPondID())
1216
1217         goldfishTotal=0
1218         tenchTotal=0
1219         orfeTotal=0
1220         koiTotal=0
1221         ruddTotal=0
1222         sharkTotal=0
1223
1224         for count in range(0,len(fishList)):
1225             if fishList[count][6]=="Goldfish":
1226                 goldfishTotal+=1
1227             elif fishList[count][6]=="Tench":
1228                 tenchTotal+=1
1229             elif fishList[count][6]=="Orfe":
1230                 orfeTotal+=1
1231             elif fishList[count][6]=="Koi":
1232                 koiTotal+=1
1233             elif fishList[count][6]=="Rudd":
1234                 ruddTotal+=1
1235             elif fishList[count][6]=="Shark":
1236                 sharkTotal+=1
1237
1238         self.scene4.goldfish=QGraphicsTextItem("There are "+str(goldfishTotal)+" Goldfish in the pond")
1239         self.scene4.tench=QGraphicsTextItem("There are "+str(tenchTotal)+" tench in the pond")
1240         self.scene4.orfe=QGraphicsTextItem("There are "+str(orfeTotal)+" orfe in the pond")
1241         self.scene4.koi=QGraphicsTextItem("There are "+str(koiTotal)+" koi in the pond")
1242         self.scene4.rudd=QGraphicsTextItem("There are "+str(ruddTotal)+" rudd in the pond")
1243         self.scene4.shark=QGraphicsTextItem("There are "+str(sharkTotal)+" sharks in the pond")
1244

```

```
1243         self.scene4.goldfish.setPos(20,5)
1244         self.scene4.tench.setPos(20,22)
1245         self.scene4.orfe.setPos(20,39)
1246         self.scene4.koi.setPos(20,56)
1247         self.scene4.rudd.setPos(20,73)
1248         self.scene4.shark.setPos(20,90)
1249
1250         self.scene4.addItem(self.scene4.goldfish)
1251         self.scene4.addItem(self.scene4.tench)
1252         self.scene4.addItem(self.scene4.orfe)
1253         self.scene4.addItem(self.scene4.koi)
1254         self.scene4.addItem(self.scene4.rudd)
1255         self.scene4.addItem(self.scene4.shark)
1256
1257         emergentTotal=0
1258         rootedTotal=0
1259         trueTotal=0
1260         submersedTotal=0
1261
1262         for count in range(0,len(plantList)):
1263             if plantList[count][1]=="Emergent":
1264                 emergentTotal+=1
1265             elif plantList[count][1]=="Rooted Floater":
1266                 rootedTotal+=1
1267             elif plantList[count][1]=="True Floater":
1268                 trueTotal+=1
1269             elif plantList[count][1]=="Submersed":
1270                 submersedTotal+=1
1271
1272         self.scene4.emergent=QGraphicsTextItem("There are
1273         "+str(emergentTotal)+" emergent plants in the pond")
1274         self.scene4.rooted=QGraphicsTextItem("There are
1275         "+str(rootedTotal)+" rooted plants in the pond")
1276         self.scene4.true=QGraphicsTextItem("There are "+str(trueTotal)+"
1277         floating plants in the pond")
1278         self.scene4.submersed=QGraphicsTextItem("There are
1279         "+str(submersedTotal)+" submersed plants in the pond")
1280
1281         self.scene4.emergent.setPos(220,5)
1282         self.scene4.rooted.setPos(220,22)
1283         self.scene4.true.setPos(220,39)
1284         self.scene4.submersed.setPos(220,56)
1285
1286         self.scene4.addItem(self.scene4.emergent)
1287         self.scene4.addItem(self.scene4.rooted)
1288         self.scene4.addItem(self.scene4.true)
1289         self.scene4.addItem(self.scene4.submersed)
1290
1291     except:
1292         pass
1293
1294 if __name__=="__main__":
1295     db=sqlite3.connect('pond.db') #connecting to database
1296     cursor=db.cursor()
1297     databaseCreation.createDatabases()
1298     fish.saveFishTypes(db,cursor)
1299     plant.savePlantTypes(db,cursor)
1300
1301     application=QApplication(sys.argv) #create application
1302     mainWindow=MainWindow() #creat new window instance
1303     mainWindow.show() #make instance visable
1304     mainWindow.raise_() #raise instance to top of window stack
1305     application.exec_() #monitor for events
```

10.3 Fish Module

```
1  import sqlite3
2
3  class Fish:
4      """A simulation of a generic fish"""
5      def __init__(self, db, cursor):
6          self._fishName=None
7          self._fishType="fish"
8          self._foodNeed=0
9          self._growthRate=0
10         self._fishSize=0
11         self._hunger=0
12         self._status="good"
13         self._daysAlive=0
14         self._growth=0
15         self._maxDaysAlive=100
16         self._fishID=None
17
18     def getFishID(self):
19         return self._fishID
20
21     def setFishName(self,x):
22         self._fishName=x
23
24     def getFishName(self):
25         return self._fishName
26
27     def setFishType(self,x):
28         self._fishType=x
29
30     def getFishType(self):
31         return self._fishType
32
33     def setFoodNeed(self,x):
34         self._foodNeed=x
35
36     def getFoodNeed(self):
37         return self._foodNeed
38
39     def setGrowthRate(self,x):
40         self._growthRate=x
41
42     def getGrowthRate(self):
43         return self._growthRate
44
45     def setFishSize(self,x):
46         self._setFishSize=x
47
48     def getFishSize(self):
49         return self._fishSize
50
51     def setHunger(self,x):
52         self._hunger=x
53
54     def getHunger(self):
55         return self._hunger
56
57     def setStatus(self,growth,daysAlive):
58         pass
59
60     def getStatus(self):
61         return self._status
62
63     def setDaysAlive(self,x):
64         self._daysAlive=x
65
```

```
66     def getDaysAlive(self):
67         return self._daysAlive
68
69     def setGrowth(self,x):
70         self._growth=x
71
72     def getGrowth(self):
73         return self._growth
74
75     def grow(self,food):
76         pass
77
78     def saveFish(self, pondID, db, cursor):
79         sql = """insert into fish(fishName, hunger, status, daysAlive, growth,
fishType, pondID) values
80             (
81             '{0}', '{1}', '{2}', '{3}', '{4}', '{5}', '{6}')""".format(self._fishName, self._hunger,
self._status, self._daysAlive, self._growth, self._fishType, pondID)
81         cursor.execute(sql)
82         cursor.execute("""select last_insert_rowid()""")
83         self._fishID = cursor.fetchall()[0][0]
84         db.commit()
85
86 class Goldfish(Fish):
87     """A goldfish fish type"""
88     def __init__(self, db, cursor):
89         Fish.__init__(self, db, cursor)
90         self._fishType="Goldfish"
91         self._foodNeed=5
92         self._growthRate=1
93         self._fishSize=0.5
94     def save(self, db, cursor):
95         sql = """insert into fishType(fishType,foodNeed, growthRate, fishSize)
values
96             ('{0}', '{1}', '{2}', '{3}')""".format(self._fishType, self._foodNeed,
self._growthRate, self._fishSize)
97         cursor.execute(sql)
98         db.commit()
99
100 class Tench(Fish):
101     """A tench fish type"""
102     def __init__(self, db, cursor):
103         Fish.__init__(self, db, cursor)
104         self._fishType="Tench"
105         self._foodNeed=4
106         self._growthRate=0.5
107         self._fishSize=1
108     def save(self, db, cursor):
109         sql = """insert into fishType(fishType,foodNeed, growthRate, fishSize)
values
110             ('{0}', '{1}', '{2}', '{3}')""".format(self._fishType, self._foodNeed,
self._growthRate, self._fishSize)
111         cursor.execute(sql)
112         db.commit()
113
114 class Orfe(Fish):
115     """A Orfe fish type"""
116     def __init__(self, db, cursor):
117         Fish.__init__(self, db, cursor)
118         self._fishType="Orfe"
119         self._foodNeed=3
120         self._growthRate=3
121         self._fishSize=1
122     def save(self, db, cursor):
123         sql = """insert into fishType(fishType,foodNeed, growthRate, fishSize)
values
```



```
124         ('{0}','{1}','{2}','{3}').format(self._fishType, self._foodNeed,
self._growthRate, self._fishSize)
125         cursor.execute(sql)
126         db.commit()
127
128class Koi(Fish):
129     """A koi fish type"""
130     def __init__(self, db, cursor):
131         Fish.__init__(self, db, cursor)
132         self._fishType="Koi"
133         self._foodNeed=8
134         self._growthRate=2
135         self._fishSize=0.5
136     def save(self, db, cursor):
137         sql = """insert into fishType(fishType,foodNeed, growthRate, fishSize)
values
138         ('{0}','{1}','{2}','{3}').format(self._fishType, self._foodNeed,
self._growthRate, self._fishSize)
139         cursor.execute(sql)
140         db.commit()
141
142class Rudd(Fish):
143     """A rudd fish type"""
144     def __init__(self, db, cursor):
145         Fish.__init__(self, db, cursor)
146         self._fishType="Rudd"
147         self._foodNeed=2
148         self._growthRate=0.5
149         self._fishSize=0.25
150     def save(self, db, cursor):
151         sql = """insert into fishType(fishType,foodNeed, growthRate, fishSize)
values
152         ('{0}','{1}','{2}','{3}').format(self._fishType, self._foodNeed,
self._growthRate, self._fishSize)
153         cursor.execute(sql)
154         db.commit()
155
156class Shark(Fish):
157     """A shark fish type"""
158     def __init__(self, db, cursor):
159         Fish.__init__(self, db, cursor)
160         self._fishType="Shark"
161         self._foodNeed=1000
162         self._growthRate=1
163         self._fishSize=100
164     def save(self, db, cursor):
165         sql = """insert into fishType(fishType,foodNeed, growthRate, fishSize)
values
166         ('{0}','{1}','{2}','{3}').format(self._fishType, self._foodNeed,
self._growthRate, self._fishSize)
167         cursor.execute(sql)
168         db.commit()
169
170
171def saveFishTypes(db,cursor):
172     newGold=Goldfish(db,cursor)
173     newTench=Tench(db,cursor)
174     newOrfe=Orfe(db,cursor)
175     newKoi=Koi(db,cursor)
176     newRudd=Rudd(db,cursor)
177     newShark=Shark(db,cursor)
178     try:
179         newGold.save(db, cursor)
180         newTench.save(db, cursor)
181         newOrfe.save(db, cursor)
182         newKoi.save(db, cursor)
183         newRudd.save(db, cursor)
```

```
184         newShark.save(db, cursor)
185     except:
186         pass
187
188
189 if __name__ == "__main__":
190     db=sqlite3.connect('pond.db')
191     cursor=db.cursor()
192     saveFishTypes(db,cursor)
193     cursor.close()
```

10.4 Plant Module

```
1  import sqlite3
2
3  class Plant:
4      """A simulation of a generic plant, requires a plantType object"""
5      def __init__(self, db, cursor):
6          self._status="good"
7          self._lightNeed=0
8          self._daysAlive=0
9          self._growth=0
10         self._plantID=None
11         self._plantType="Plant"
12
13     def getPlantID(self):
14         return self._plantID
15
16     def setPlantType(self,x):
17         self._plantType=x
18
19     def getPlantType(self):
20         return self._plantType
21
22     def setStatus(self,x):
23         self._status=x
24
25     def getStatus(self):
26         return self._status
27
28     def setLightNeed(self,x):
29         self._lightNeed=x
30
31     def getLightNeed(self):
32         return self._lighNeed
33
34     def setDaysAlive(self,x):
35         self._daysAlive=x
36
37     def getDaysAlive(self):
38         return self._daysAlive
39
40     def setGrowth(self,x):
41         self._growth=x
42
43     def getGrowth(self):
44         return self._growth
45
46     def savePlant(self, pondID, db,cursor):
47         sql = """insert into plant(status, lightNeed, daysAlive, growth,
plantType, PondID) values
48             ('{0}','{1}','{2}','{3}','{4}','{5}')""".format(self._status,
self._lightNeed, self._daysAlive, self._growth, self._plantType, pondID)
49         cursor.execute(sql)
50         cursor.execute("""select last_insert_rowid()""")
```

```

51         self._plantID = cursor.fetchall()[0][0]
52         db.commit()
53
54     class RootedFloater(Plant):
55         """A rooted floater plant type"""
56         def __init__(self,db,cursor):
57             Plant.__init__(self,db,cursor)
58             self._plantType="Rooted Floater"
59             self._lightRequirement=7
60             self._growthRate=2
61         def save(self, db, cursor):
62             sql = """insert into plantType(plantType, lightRequirement, growthRate)
values
63                 ('{0}','{1}','{2}')""".format(self._plantType,
self._lightRequirement, self._growthRate)
64             cursor.execute(sql)
65             db.commit()
66
67     class TrueFloater(Plant):
68         """A true floater plant type"""
69         def __init__(self,db,cursor):
70             Plant.__init__(self,db,cursor)
71             self._plantType="True Floater"
72             self._lightRequirement=2
73             self._growthRate=2
74         def save(self, db, cursor):
75             sql = """insert into plantType(plantType, lightRequirement, growthRate)
values
76                 ('{0}','{1}','{2}')""".format(self._plantType,
self._lightRequirement, self._growthRate)
77             cursor.execute(sql)
78             db.commit()
79
80     class Emergent(Plant):
81         """An emergent plant type"""
82         def __init__(self,db,cursor):
83             Plant.__init__(self,db,cursor)
84             self._plantType="Emergent"
85             self._lightRequirement=2
86             self._growthRate=1
87         def save(self, db, cursor):
88             sql = """insert into plantType(plantType, lightRequirement, growthRate)
values
89                 ('{0}','{1}','{2}')""".format(self._plantType,
self._lightRequirement, self._growthRate)
90             cursor.execute(sql)
91             db.commit()
92
93     class Submersed(Plant):
94         """A Submersed plant type"""
95         def __init__(self,db,cursor):
96             Plant.__init__(self,db,cursor)
97             self._plantType="Submersed"
98             self._lightRequirement=2
99             self._growthRate=1
100         def save(self, db, cursor):
101             sql = """insert into plantType(plantType, lightRequirement, growthRate)
values
102                 ('{0}','{1}','{2}')""".format(self._plantType,
self._lightRequirement, self._growthRate)
103             cursor.execute(sql)
104             db.commit()
105
106 def savePlantTypes(db, cursor):
107     newRooted=RootedFloater(db,cursor)
108     newFloat=TrueFloater(db, cursor)
109     newEmergent=Emergent(db,cursor)

```

```
110     newSubmersed=Submersed(db,cursor)
111     try:
112         newRooted.save(db, cursor)
113         newFloat.save(db, cursor)
114         newEmergent.save(db, cursor)
115         newSubmersed.save(db, cursor)
116     except:
117         pass
118
119
120 if __name__ == "__main__":
121     db=sqlite3.connect('pond.db')
122     cursor=db.cursor()
123     savePlantTypes(db, cursor)
124     cursor.close()
```

10.5 Pond Module

```
1  import sqlite3
2  import pondSimDatabase
3  import plant
4  import fish
5  import databaseCreation
6  import random
7
8  #classes
9  class Pond:
10     """A simulation of a pond"""
11     def __init__(self, width, length, depth, temp, waterLevel, pump, filter,
12 pondID, days):
13         self._pondWidth=width
14         self._pondLength=length
15         self._pondDepth=depth
16         self._waterTemp=temp
17         self._waterLevel=waterLevel
18         self._pump=pump
19         self._filter=filter
20         self._pondID=pondID
21         self._days=days
22
23     def setPondID(self,x):
24         self._pondID=x
25
26     def getPondID(self):
27         return self._pondID
28
29     def setPondWidth(self, x): #where x is the desired pond width
30         self._pondWidth=x
31
32     def getPondWidth(self):
33         return self._pondWidth
34
35     def setPondLength(self, x):
36         self._pondLength=x
37
38     def getPondLength(self):
39         return self._pondLength
40
41     def setPondDepth(self,x):
42         self._pondDepth=x
43
44     def getPondDepth(self):
45         return self._pondDepth
46
47     def setWaterLevel(self,x):
48         self._waterLevel=x
```

```

48
49     def getWaterLevel(self):
50         return self._waterLevel
51
52     def setWaterTemp(self, x):
53         self._waterTemp=x
54
55     def getWaterTemp(self):
56         return self._waterTemp
57
58     def setPump(self, x): #where x is the pump object
59         self._pump=x
60
61     def getPump(self):
62         return self._pump
63
64     def setFilter(self, x):
65         self._filter=x
66
67     def getFilter(self):
68         return self._filter
69
70     def setDays(self, x):
71         self._days=x
72
73     def getDays(self):
74         return self._days
75
76     def addFish(self, db, cursor,choice,name):
77         def chooseFish(db, cursor,choice):
78             if choice=="Goldfish":
79                 newFish=fish.Goldfish(db, cursor)
80             elif choice=="Tench":
81                 newFish=fish.Tench(db,cursor)
82             elif choice=="Orfe":
83                 newFish=fish.Orfe(db,cursor)
84             elif choice=="Koi":
85                 newFish=fish.Koi(db,cursor)
86             elif choice=="Rudd":
87                 newFish=fish.Rudd(db,cursor)
88             elif choice=="Shark":
89                 newFish=fish.Shark(db,cursor)
90             return newFish
91
92         newFish=chooseFish(db, cursor,choice)
93         newFish.setFishName(name)
94         newFish.saveFish(self._pondID, db, cursor)
95
96     def removeFish(self,db, cursor):
97         pondID=pond.getPondID()
98         fishTup=pondSimDatabase.fetchFish(db, cursor, pondID)
99         correct=False
100
101         #displaying list of fish in pond
102
103     print('{0:^10}|{1:^10}|{2:^15}|{3:^15}|{4:^15}|{5:^15}|{6:^10}|'.format('FishID','
fishName','Hunger','Status','daysAlive','growth','FishType'))
104     for each in range(0,len(fishTup)):
105         print('{0:^12} {1:^10} {2:^10} {3:^20} {4:^10} {5:^20}
{6:^5}'.format(fishTup[each][0],fishTup[each][1],fishTup[each][2],fishTup[each][3],
fishTup[each][4],fishTup[each][5],fishTup[each][6]))
106     print('')
107     try:
108         while correct==False:
109             IDchoice=int(input('Enter the ID of the fish you wish to remove:
'))

```

```

110         if IDchoice>len(fishTup)or IDchoice<0:
111             print("That ID was not found on the list")
112         else:
113             correct=True
114             pondSimDatabase.deleteFish(db, cursor, IDchoice)
115     except:
116         print("That was not a valid ID - please enter a number")
117
118
119     def removePlant(self,db,cursor):
120         pondID=pond.getPondID()
121         plantsTup=pondSimDatabase.fetchPlants(db, cursor, pondID)
122         correct=False
123
124     print('{0:^10}|{1:^10}|{2:^15}|{3:^15}|{4:^15}|{5:^15}|'.format('PlantID','PlantTy
125     pe','Status','LightNeed','daysAlive','growth'))
126     for each in range(0,len(plantsTup)):
127         print('{0:^12} {1:^10} {2:^10} {3:^20} {4:^10}
128         {5:^20}'.format(plantsTup[each][0],plantsTup[each][1],plantsTup[each][2],plantsTup[
129         each][3],plantsTup[each][4],plantsTup[each][5]))
130         print('')
131
132     try:
133         while correct==False:
134             IDchoice=int(input('Enter the ID of the plant you wish to
135             remove: '))
136             if IDchoice>len(plantTup)or IDchoice<0:
137                 print("That ID was not found on the list")
138             else:
139                 correct=True
140                 pondSimDatabase.deletePlant(db, cursor, IDchoice)
141     except:
142         print("That was not a valid ID - please enter a number")
143
144     def addPlant(self, db, cursor,choice):
145         def choosePlant(db,cursor,choice):
146             if choice=="Rooted Floater":
147                 newPlant=plant.RootedFloater(db,cursor)
148             elif choice=="True Floater":
149                 newPlant=plant.TrueFloater(db,cursor)
150             elif choice=="Emergent":
151                 newPlant=plant.Emergent(db,cursor)
152             elif choice=="Submersed":
153                 newPlant=plant.Submersed(db,cursor)
154             return newPlant
155         newPlant=choosePlant(db,cursor,choice)
156         newPlant.savePlant(self._pondID, db, cursor)
157
158     def editPond(self):
159         waterTemp=input("Water Temperature:")
160         self.setWaterTemp(waterTemp)
161
162         pondWidth=input("Pond Width:")
163         self.setPondWidth(pondWidth)
164
165         pondLength=input("Pond Length:")
166         self.setPondLength(pondLength)
167
168         pondDepth=input("Pond Depth:")
169         self.setPondDepth(pondDepth)
170
171         self.setWaterLevel(int(pondDepth)-0.5)
172
173     def savePond(self,db,cursor):
174         #saving pond
175         try:

```

```

172         self._pumpID=self._pump.getPumpID()
173     except:
174         self._pumpID=None
175
176     try:
177         self._filterID=self._filter.getFilterID()
178     except:
179         self._filterID=None
180
181     sql = """insert into pond(pondDepth, pondLength, pondWidth, waterTemp,
waterLevel, pumpID, filterID, days) values
182         (
183         '{0}', '{1}', '{2}', '{3}', '{4}', '{5}', '{6}', '{7}').format(self._pondDepth,
self._pondLength, self._pondWidth, self._waterTemp, self._waterLevel, self._pump,
self._filter, self._days)
183         cursor.execute(sql)
184         cursor.execute("""select last_insert_rowid()""")
185         self._pondID = cursor.fetchall()[0][0]
186         db.commit()
187
188     def updatePond(self,db,cursor):
189         try:
190             self._pumpID=self.getPump()
191         except:
192             self._pumpID=None
193
194         try:
195             self._filterID=self.getFilter()
196         except:
197             self._filterID=None
198         try:
199             tup=(self._pondDepth, self._pondLength, self._pondWidth,
self._waterTemp, self._waterLevel, self._days, self._pumpID, self._filterID,
self._pondID)
200             sql = """update pond
201                 set pondDepth=?, pondLength=?, pondWidth=?, waterTemp=?,
waterLevel=?, days=?, pumpID=?, filterID=?
202                 where pondID=?"""
203             cursor.execute(sql,tup)
204             db.commit()
205         except:
206             pass
207
208 class Filter:
209     """A simulation of a pond filter"""
210     def __init__(self, status, filterID):
211         self._filterStatus=status
212         self._filterID=filterID
213
214     def getFilterID(self):
215         return self._filterID
216
217     def setFilterStatus(self,x):
218         self._filterStatus=x
219
220     def getFilterStatus(self):
221         return self._filterStatus
222
223     def saveFilter(self,db,cursor):
224         sql = """insert into filter(filterStatus) values
225             ( '{0}').format(self._filterStatus)
226         cursor.execute(sql)
227         cursor.execute("""select last_insert_rowid()""")
228         self._filterID = cursor.fetchall()[0][0]
229         db.commit()
230
231     def updateFilter(self,db,cursor):

```

```
232         tup=(self._filterStatus,self._filterID)
233         sql="""update filter
234             set filterStatus=?
235             where filterID=?"""
236         cursor.execute(sql,tup)
237         db.commit()
238
239 class Pump:
240     """A simulation of a pump"""
241     def __init__(self, status, pumpID):
242         self._pumpStatus=status
243         self._pumpID=pumpID
244
245     def getPumpID(self):
246         return self._pumpID
247
248     def setPumpStatus(self,x):
249         self._pumpStatus=x
250
251     def getPumpStatus(self):
252         return self._pumpStatus
253
254     def savePump(self,db,cursor):
255         sql = """insert into pump(pumpStatus) values
256             ( '{0} ' )""".format(self._pumpStatus)
257         cursor.execute(sql)
258         cursor.execute("""select last_insert_rowid()""")
259         self._pumpID = cursor.fetchall()[0][0]
260         db.commit()
261
262     def updatePump(self,db,cursor):
263         tup=(self._pumpStatus,self._pumpID)
264         sql="""update pump
265             set pumpStatus=?
266             where pumpID=?"""
267         cursor.execute(sql,tup)
268         db.commit()
269
270 #creating a pond
271 def createPond():
272     width=0
273     while width>10 or width<1:
274         try:
275             width=int(input('Enter the width of the pond: '))
276             if width>10 or width<1:
277                 print("The width of the pond must be between 1 and 10 (1 = 1m)")
278         except:
279             print("The width of the pond must be a number between 1 and 10 (1 =
1m)")
280     print('')
281
282     length=0
283     while length>10 or length<1:
284         try:
285             length=int(input('Enter the length of the pond: '))
286             if length>10 or length<1:
287                 print("The length of the pond must be between 1 and 10 (1 =
1m)")
288         except:
289             print("The length of the pond must be a number between 1 and 10 (1 =
1m)")
290     print('')
291
292     depth=0
293     while depth>10 or depth<1:
294         try:
295             depth=int(input('Enter the depth of the pond: '))
```



```
296         if depth>10 or depth<1:
297             print("The depth of the pond must be between 1 and 10 (1 = 1m)")
298     except:
299         print("The depth of the pond must be a number between 1 and 10 (1 =
1m)")
300     print('')
301
302     temp=100
303     while temp>50 or temp<-27:
304         try:
305             temp=int(input('Enter the starting temperature of the pond: '))
306             if temp>50 or temp<-27:
307                 print("The water temperature must be between -27C")
308         except:
309             print("The water temperature must be a number between -27C")
310     print('')
311
312     pandf=4
313     while pandf>1 or pandf<0:
314         try:
315             pandf=int(input('Press 1 to add a pump and filter to the pond or 0
to not: '))
316             if pandf>1 or pandf<0:
317                 print("The value entered was not 1 or 0. Try again")
318         except:
319             print("The value entered was not 1 or 0. Try again")
320     print('')
321
322     waterLevel=float(depth)-0.5
323     if int(pandf)==1:
324         pump=Pump("good",None)
325         pump.savePump()
326         filter=Filter("good",None)
327         filter.saveFilter()
328     else:
329         pump=None
330         filter=None
331     newPond=Pond(width, length, depth, temp, waterLevel, pump, filter,None,0)
332     newPond.savePond()
333     return newPond
334
335#main menu - first thing displayed
336def mainMenu():
337     def displayMenu():
338         print('')
339         print("1. Create new pond")
340         print("2. Open pond")
341         print("3. Delete Pond")
342         print("4. Exit program")
343         print('')
344
345     def getMenuChoice():
346         choice=0
347         while choice==0:
348             try:
349                 choice=int(input("What would you like to do? "))
350                 if choice>4 or choice<1:
351                     print("You must enter an option between 1 and 4")
352             except:
353                 print("You must enter a number between 1 and 4")
354         return choice
355
356     displayMenu()
357     choice=getMenuChoice()
358     return choice
359
360#opening an existing pond
```

```

361def openPond():
362    #fetching and displaying all ponds in database
363    correct=False
364    results=pondSimDatabase.getPonds(db,cursor)
365    #getting ID from user
366    pondIDs=pondSimDatabase.fetchPondID(db,cursor)
367
368    try:
369        while correct==False:
370            IDchoice=int(input('Enter the ID of the pond you wish to open: '))
371            for count in range(0,len(pondIDs)):
372                if IDchoice==pondIDs[count][0]:
373                    correct=True
374            if correct==False:
375                print("That ID was not found on the list")
376    except:
377        print("That was not a valid ID - please enter a number")
378
379    results=pondSimDatabase.fetchPond(db,cursor,str(IDchoice))#fetching pond
data
380    try:
381        pump=pondSimDatabase.fetchPump(db,cursor,str(results[0][6])) #fetching
pump data
382        pump=Pump(pump[0][1], pump[0][0]) #creating pump object from pump data
383        filter=pondSimDatabase.fetchFilter(db,cursor,str(results[0][7]))
384        filter=Filter(filter[0][1], filter[0][0])
385    except:
386        pump=None
387        filter=None
388
389    pondDepth=results[0][1]
390    pondWidth=results[0][3]
391    pondLength=results[0][2]
392    waterTemp=results[0][4]
393    waterLevel=results[0][5]
394    pondID=results[0][0]
395    days=results[0][8]
396
397    pond=Pond(width=pondWidth, length=pondLength, depth=pondDepth,
temp=waterTemp, waterLevel=waterLevel, pump=pump, filter=filter, pondID=pondID,
days=days) #creating pond from data
398    return pond
399
400#deleted a pond for the pond table
401def deletePond(ID):
402    if(ID == None):
403        correct=False
404        results=pondSimDatabase.getPonds(db,cursor)
405        print('')
406
407    print('{0:^10}|{1:^10}|{2:^15}|{3:^15}|{4:^15}|{5:^15}|{6:^10}|{7:^10}|'.format('P
ondID','pondDepth','PondLength','PondWidth','WaterTemp','waterLevel','pump','filter
'))
407        for each in range(0,len(results)):
408            print('{0:^12} {1:^10} {2:^10} {3:^20} {4:^10} {5:^20} {6:^5}
{7:^15}'.format(results[each][0],results[each][1],results[each][2],results[each][3]
,results[each][4],results[each][5],results[each][6],results[each][7]))
409        print('')
410        #getting ID from user
411        try:
412            while correct==False:
413                IDchoice=int(input('Enter the ID of the pond you wish to delete:
'))
414                if IDchoice>len(results)or IDchoice<0:
415                    print("That ID was not found on the list")
416                else:
417                    correct=True

```

```
418     except:
419         print("That was not a valid ID - please enter a number")
420     filter=pond.getFilter()
421     filterID=filter.getFilterID()
422     pump=pond.getPump()
423     pumpID=pump.getPumpID()
424     pondSimDatabase.deletePond(db, cursor, IDchoice)
425     pondSimDatabase.deleteFishFromPond(db, cursor, IDchoice)
426     pondSimDatabase.deletePlantFromPond(db, cursor, IDchoice)
427     pondSimDatabase.deletePump(db, cursor, pumpID)
428     pondSimDatabase.deleteFilter(db, cursor, filterID)
429 else:
430     pondSimDatabase.deletePond(db, cursor, ID)
431
432 #increment day
433 def incrementDay(mainPond, db, cursor):
434     def updateFish(tup):
435         sql = """update fish
436             set hunger=?, daysAlive=?
437             where fishID=?"""
438         cursor.execute(sql, tup)
439         db.commit()
440
441     def updatePlants(tup):
442         sql="""update plant
443             set daysAlive=?, growth=?
444             where plantID=?"""
445         cursor.execute(sql, tup)
446         db.commit()
447
448     fishList=pondSimDatabase.fetchFishInfo(db, cursor, mainPond.getPondID())
449     plantList=pondSimDatabase.fetchPlantInfo(db, cursor, mainPond.getPondID())
450     plantType=pondSimDatabase.fetchPlantType(db, cursor)
451     foodNeed=pondSimDatabase.fetchFoodNeed(db, cursor)
452
453     for count in range(0, len(fishList)):
454         daysAlive=fishList[count][4]+1
455         if fishList[count][6]=="Goldfish":
456             hunger=fishList[count][2]+foodNeed[0][1]
457         elif fishList[count][6]=="Tench":
458             hunger=fishList[count][2]+foodNeed[1][1]
459         elif fishList[count][6]=="Orfe":
460             hunger=fishList[count][2]+foodNeed[2][1]
461         elif fishList[count][6]=="Koi":
462             hunger=fishList[count][2]+foodNeed[3][1]
463         elif fishList[count][6]=="Rudd":
464             hunger=fishList[count][2]+foodNeed[4][1]
465         elif fishList[count][6]=="Shark":
466             hunger=fishList[count][2]+foodNeed[5][1]
467         if fishList[count][3]=="Dead":
468             hunger=0
469             daysAlive=0
470         elif fishList[count][3]=="good":
471             daysAlive=fishList[count][4]+1
472         tup=(hunger, daysAlive, fishList[count][0])
473         updateFish(tup)
474
475     for count in range(0, len(plantList)):
476         daysAlive=plantList[count][4]+1
477         if plantList[count][1]=="Rooted Floater":
478             growth=plantList[count][5]+plantType[0][1]
479         elif plantList[count][1]=="True Floater":
480             growth=plantList[count][5]+plantType[1][1]
481         elif plantList[count][1]=="Emergent":
482             growth=plantList[count][5]+plantType[2][1]
483         elif plantList[count][1]=="submersed":
484             growth=plantList[count][5]+plantType[3][1]
```

```
485         if plantList[count][2]=="Dead":
486             daysAlive=0
487         else:
488             daysAlive=plantList[count][4]+1
489         tup=(daysAlive,plantList[count][5]+1,plantList[count][0])
490         updatePlants(tup)
491
492     randomNo=random.randint(0,20)
493     if randomNo==1:
494         try:
495             pump=pondSimDatabase.fetchPump(db,cursor,mainPond.getPump())
#fetching pump data
496             pump=Pump(pump[0][1], pump[0][0]) #creating pump object from pump
data
497             pumpStatus=pump.getPumpStatus()
498             pump.setPumpStatus("Broken")
499             pump.updatePump(db,cursor)
500             mainPond.setPump(pump.getPumpID())
501         except:
502             pass
503
504
505
506     randomNo=random.randint(0,20)
507     if randomNo==1:
508         try:
509             filter=pondSimDatabase.fetchFilter(db,cursor,self.mainPond.getFilter()) #fetching
pump data
510             filter=Filter(filter[0][1], filter[0][0]) #creating pump object from
pump data
511             filterStatus=filter.getFilterStatus()
512             filter.setFilterStatus("Broken")
513             filter.updateFilter(db,cursor)
514             mainPond.setFilter(filter.getFilterID())
515         except:
516             pass
517
518
519     mainPond.setDays(int(mainPond.getDays()+1)
520     mainPond.updatePond(db,cursor)
521
522 #manage pond menu - displays when a pond is opened or created
523 def managePond(pond, db, cursor):
524     def displayMenu():
525         print('')
526         print("1. Check pond status")
527         print("2. Manage pump and filter")
528         print("3. Edit pond settings")
529         print("4. Manage pond life")
530         print("5. Increment Day")
531         print("6. Exit to main menu")
532         print('')
533
534     def getMenuChoice():
535         displayMenu()
536         choice=0
537         while choice==0:
538             try:
539                 choice=int(input("What would you like to do? "))
540                 if choice>6 or choice<1:
541                     print("You must enter an option between 1 and 6")
542             except:
543                 print("You must enter a number between 1 and 6")
544             return choice
545
546     def showPondStatus():
```

```

547         #displaying pond information
548         print('')
549         print('Day',pond.getDays())
550         print('')
551         print("Pond depth:",pond.getPondDepth())
552         print("Pond Length:", pond.getPondLength())
553         print("Pond Width:", pond.getPondWidth())
554         print("Water Level:",pond.getWaterLevel())
555         print("Water Temperature:",pond.getWaterTemp())
556         print('')
557
558         pump=pond.getPump()
559         filter=pond.getFilter()
560         pumpStatus=pump.getPumpStatus()
561         filterStatus=filter.getFilterStatus()
562         print("Filter:",filterStatus)
563         print("Pump:",pumpStatus)
564         print("")
565
566         fishTup=pondSimDatabase.fetchFish(db, cursor, pond.getPondID())
567         plantsTup=pondSimDatabase.fetchPlants(db, cursor, pond.getPondID())
568
569         #displaying list of fish in pond
570
571         print('{0:^10}|{1:^10}|{2:^15}|{3:^15}|{4:^15}|{5:^15}|{6:^10}|'.format('FishID','
fishName','Hunger','Status','daysAlive','growth','FishType'))
572         for each in range(0,len(fishTup)):
573             print('{0:^12} {1:^10} {2:^10} {3:^20} {4:^10} {5:^20}
{6:^5}'.format(fishTup[each][0],fishTup[each][1],fishTup[each][2],fishTup[each][3],
fishTup[each][4],fishTup[each][5],fishTup[each][6]))
574         print('')
575         #displaying lists of plants in the pond
576
577         print('{0:^10}|{1:^10}|{2:^15}|{3:^15}|{4:^15}|{5:^15}|'.format('PlantID','PlantTy
pe','Status','LightNeed','daysAlive','growth'))
578         for each in range(0,len(plantsTup)):
579             print('{0:^12} {1:^10} {2:^10} {3:^20} {4:^10}
{5:^20}'.format(plantsTup[each][0],plantsTup[each][1],plantsTup[each][2],plantsTup[
each][3],plantsTup[each][4],plantsTup[each][5]))
580         print('')
581
582         def managePumpAndFilter():
583             def displayMenu():
584                 print('1. Add pump and filter')
585                 print('2. Remove pump and filter')
586                 print('3. Edit pump and filter')
587
588             def getMenuChoice():
589                 displayMenu()
590                 choice=0
591                 while choice==0:
592                     try:
593                         choice=int(input("What would you like to do? "))
594                         if choice>3 or choice<1:
595                             print("You must enter an option between 1 and 3")
596                     except:
597                         print("You must enter a number between 1 and 3")
598                 return choice
599
600         def addPaF():
601             newFilter=Filter("good",None)
602             newPump=Pump("good",None)
603             newFilter.saveFilter()
604             newPump.savePump()
605             pond.setFilter(newFilter)
606             pond.setPump(newPump)
607             pond.updatePond()

```

```

606
607     def removePaF():
608         filter=pond.getFilter()
609         filterID=filter.getFilterID()
610         pump=pond.getPump()
611         pumpID=pump.getPumpID()
612         pondSimDatabase.deletePump(db,cursor,pumpID)
613         pondSimDatabase.deleteFilter(db,cursor,filterID)
614         pond.setFilter(None)
615         pond.setPump(None)
616         pond.updatePond()
617
618     def editPaF():
619         pump=pond.getPump()
620         filter=pond.getFilter()
621         pumpStatus=pump.getPumpStatus()
622         filterStatus=filter.getFilterStatus()
623         print("Filter:",filterStatus)
624         print("Pump:",pumpStatus)
625         if filterStatus=="Broken":
626             choice=int(input("Would you like to fix the filter? (yes = 1, no
= 0) "))
627             if choice==1:
628                 filter.setFilterStatus("good")
629             else:
630                 pass
631         if pumpStatus=="Broken":
632             choice=int(input("Would you like to fix the pump? (yes = 1, no =
0) "))
633             if choice==1:
634                 pump.setPumpStatus("good")
635             else:
636                 pass
637
638         choice=getMenuChoice()
639         if choice==1:
640             addPaF()
641         if choice==2:
642             removePaF()
643         if choice==3:
644             editPaF()
645
646
647
648 def manualGrow(amount,pond,db,cursor):
649     def updateFish(tup):
650         sql = """update fish
651             set hunger=?, growth=?, status=?
652             where fishID=?"""
653         cursor.execute(sql,tup)
654         db.commit()
655
656     hunger=pondSimDatabase.fetchFishInfo(db,cursor,pond.getPondID())
657
658     for count in range(0,len(hunger)):
659         amount2=amount
660         amount=amount-hunger[count][2]
661         hunger2=hunger[count][2]-amount2
662         if hunger2<0:
663             hunger2=0
664
665         growthList=pondSimDatabase.fetchGrowthRate(db,cursor)
666         if hunger2==0 and hunger[count][3]!="Dead":
667             if hunger[count][6]=="Goldfish":
668                 growth=growthList[0][0]+hunger[count][5]
669             elif hunger[count][6]=="Tench":
670                 growth=growthList[1][0]+hunger[count][5]

```

```

671         elif hunger[count][6]=="Orfe":
672             growth=growthList[2][0]+hunger[count][5]
673         elif hunger[count][6]=="Koi":
674             growth=growthList[3][0]+hunger[count][5]
675         elif hunger[count][6]=="Rudd":
676             growth=growthList[4][0]+hunger[count][5]
677         elif hunger[count][6]=="Shark":
678             growth=growthList[5][0]+hunger[count][5]
679     else:
680         growth=hunger[count][5]
681     fishID=hunger[count][0]
682
683     if hunger[count][2]>20:
684         status="Dead"
685         growth=hunger[count][5]
686     else:
687         status=hunger[count][3]
688         #hunger2=0
689         fishID=hunger[count][0]
690         tup=(hunger2,growth,status,fishID)
691         updateFish(tup)
692     incrementDay(pond,db,cursor)
693
694
695 def automaticGrow(db,cursor,pond):
696     #feeds fish the required food amount for 7 days.
697     def updateFish(tup):
698         sql = """update fish
699             set hunger=?, growth=?, status=?
700             where fishID=?"""
701         cursor.execute(sql,tup)
702         db.commit()
703
704     fishList=pondSimDatabase.fetchFishInfo(db,cursor,pond.getPondID())
705     growthList=pondSimDatabase.fetchGrowthRate(db,cursor)
706     for count in range(0,7):
707         for count in range(0,len(fishList)):
708             if fishList[count][3]!="Dead":
709                 if fishList[count][6]=="Goldfish":
710                     growth=growthList[0][0]+fishList[count][5]
711                 elif fishList[count][6]=="Tench":
712                     growth=growthList[1][0]+fishList[count][5]
713                 elif fishList[count][6]=="Orfe":
714                     growth=growthList[2][0]+fishList[count][5]
715                 elif fishList[count][6]=="Koi":
716                     growth=growthList[3][0]+fishList[count][5]
717                 elif fishList[count][6]=="Rudd":
718                     growth=growthList[4][0]+fishList[count][5]
719                 elif fishList[count][6]=="Shark":
720                     growth=growthList[5][0]+fishList[count][5]
721             else:
722                 growth=fishList[count][5]
723             if fishList[count][2]>20:
724                 status="Dead"
725                 growth=fishList[count][5]
726             else:
727                 status=fishList[count][3]
728                 hunger=0
729                 fishID=fishList[count][0]
730                 tup=(hunger,growth,status,fishID)
731                 updateFish(tup)
732             incrementDay(pond,db,cursor)
733
734
735 #main program
736 if __name__ == "__main__":
737     db=sqlite3.connect('pond.db') #connecting to database

```

```
738     cursor=db.cursor()
739     databaseCreation.createDatabases()
740     fish.saveFishTypes(db,cursor)
741     plant.savePlantTypes(db,cursor)
742
743     Exit=False
744
745     while Exit==False:
746         choice=mainMenu()
747         if choice==1:
748             pond=createPond()
749             managePond(pond, db, cursor)
750         if choice==2:
751             pond=openPond()
752             managePond(pond, db, cursor)
753         if choice==3:
754             ID=None
755             deletePond(ID)
756         if choice==4:
757             Exit=True
758     cursor.close()
```

10.6 pondSimDatabase Module

```
1  import sqlite3
2
3  #database queries
4  def getPonds(db,cursor):
5      sql="""select *
6          from pond"""
7      cursor.execute(sql)
8      results=cursor.fetchall()
9      return results
10
11 def fetchPond(db,cursor,IDchoice):
12     IDchoice=(IDchoice,)
13     sql="""select *
14         from pond
15         where pondID=?"""
16     cursor.execute(sql,IDchoice)
17     results=cursor.fetchall()
18     return results
19
20 def fetchPump(db,cursor,ID):
21     ID=(ID,)
22     sql="""select *
23         from pump
24         where pumpID=?"""
25     cursor.execute(sql,ID)
26     pumpResults=cursor.fetchall()
27     return pumpResults
28
29 def fetchFilter(db,cursor,ID):
30     ID=(ID,)
31     sql="""select *
32         from filter
33         where filterID=?"""
34     cursor.execute(sql,ID)
35     filterResults=cursor.fetchall()
36     return filterResults
37
38 def fetchFish(db,cursor,ID):
39     ID=(ID,)
40     sql="""select *
41         from fish
```



```
42         where pondID=?""
43     cursor.execute(sql,ID)
44     fishResults=cursor.fetchall()
45     return fishResults
46
47 def fetchPlants(db,cursor,ID):
48     ID=(ID,)
49     sql="""select *
50         from plant
51         where pondID=?""
52     cursor.execute(sql,ID)
53     plantResults=cursor.fetchall()
54     return plantResults
55
56 def fetchPlantInfo(db,cursor,ID):
57     ID=(ID,)
58     sql="""select *
59         from plant
60         where pondID=?""
61     cursor.execute(sql,ID)
62     plantResults=cursor.fetchall()
63     return plantResults
64
65 def fetchPlantType(db,cursor):
66     sql="""select *
67         from plantType""
68     cursor.execute(sql)
69     plantResults=cursor.fetchall()
70     return plantResults
71
72 def fetchFishType(db,cursor):
73     sql="""select *
74         from fishType""
75     cursor.execute(sql)
76     fishResults=cursor.fetchall()
77     return fishResults
78
79 def fetchGrowthRate(db,cursor):
80     sql="""select growthRate
81         from fishType""
82     cursor.execute(sql)
83     foodNeed=cursor.fetchall()
84     return foodNeed
85
86 def fetchFoodNeed(db,cursor):
87     sql="""select *
88         from fishType""
89     cursor.execute(sql)
90     foodNeed=cursor.fetchall()
91     return foodNeed
92
93 def fetchFishInfo(db,cursor,ID):
94     ID=(ID,)
95     sql="""select *
96         from fish
97         where pondID=?""
98     cursor.execute(sql,ID)
99     hunger=cursor.fetchall()
100    return hunger
101
102 def deletePond(db,cursor,ID):
103     ID=(ID,)
104     sql="""delete
105         from pond
106         where pondID=?""
107     cursor.execute(sql,ID)
108     db.commit()
```

```
109
110def fetchPondID(db,cursor):
111     sql="""select pondID
112           from pond"""
113     cursor.execute(sql)
114     IDList=cursor.fetchall()
115     return IDList
116
117def deleteFish(db,cursor,ID):
118     ID=(ID,)
119     sql="""delete
120           from fish
121           where fishID=?"""
122     cursor.execute(sql,ID)
123     db.commit()
124
125def deleteFishFromPond(db,cursor,ID):
126     ID=(ID,)
127     sql="""delete
128           from fish
129           where pondID=?"""
130     cursor.execute(sql,ID)
131     db.commit()
132
133def deletePlantFromPond(db,cursor,ID):
134     ID=(ID,)
135     sql="""delete
136           from plant
137           where pondID=?"""
138     cursor.execute(sql,ID)
139     db.commit()
140
141def deletePlant(db,cursor,ID):
142     ID=(ID,)
143     sql="""delete
144           from plant
145           where plantID=?"""
146     cursor.execute(sql,ID)
147     db.commit()
148
149def deletePump(db,cursor,ID):
150     ID=(ID,)
151     sql="""delete
152           from pump
153           where pumpID=?"""
154     cursor.execute(sql,ID)
155     db.commit()
156
157def deleteFilter(db,cursor,ID):
158     ID=(ID,)
159     sql="""delete
160           from filter
161           where filterID=?"""
162     cursor.execute(sql,ID)
163     db.commit()
```