MEMO for INF2611 Mock Exam 2017 S1

Question 1

1. The main component for creating a user interface is widgets. Input widgets are used for interacting with the user. Display widgets are used for displaying information or messages to the user. It allows for the creation of an interface quickly and easily
2. Buttons, Menus and scollbars
3. Adding an LCD Widget
    a. Firstly, go to Qt and drag and drop a LCD number Widget Then convert the QT program to python code
    b. Create python code to connect the signal to a function
    c. Create an instance of Qtimer and set its timeout to 1000ms
    d. Whenever the timer is generated it envokes the function
    e. The functions will fetch the system clock, convert it to a string and make it display in the LCDNumber Widget.
4. Displaying the date in a date edit widget when selected on a calendar
    a. Firstly, go to Qt and drag and drop a Calendar and Date Edit Widget Then convert the QT program to python code
    b. Create python code to connect the signal to a function
    c. Whenever the date is changed by clicking in a new date the fuction is envoked
    d. The date selected by the user is retrieved by selectedDate() and displayed in the date edit widget with the setDate() method
5. A comboBox can be used to trigger certain actions and is usefull for saving space on the screen.
    a. Firstly, go to Qt and drag and drop a Calendar and Date Edit Widget Then convert the QT program to python code
    b. Create python code to connect the signal to a function
    c. You can add code to add items into the comboBox so that when selected, together with other widgets like a push button, will generate a result. A function then operates depending on what is true in the programs IF statement.

Question 2

Answers

1. In QtDesigner you can choose Main Window and click the create button. This will then create a window with a “type here” placeholder. You can click on the placeholder and type File for the menu and as you press enter, another placeholder will appear at the bottom and to the right. Choose the placeholder created below and type Copy, similarly choose the placeholder below copy and type Paste
2. You can type the (&) before any charater when naming a placeholder to underline the first letter and the it would be treated as a shortcut key
3. A menubar has several menus which may include submenus. A toolbar may contain a group of buttons with icons to give a hint of what it does.
4. It is a unnamed field within a menu
5. To the right of the menu double click on Add seporator
6. After renaming placeholders, actions can be assigned to them so that when clicked or the shortcut key pressed, the relevant action is called and executed via the QAction class. It can be assigned to the menu or the toolbar button

Creating a toolbar

Questions

1. Represents different tools for tasks the user can perform. They are usually represented as icons. Icons can be picked from disk or resource file.
2. can be picked from disk or resource file.
3. First a toolbar should be created by right clicking and selecting new toolbar. Then a new action should be created in action editor. Drag and drop the action onto the toolbar. Thereafter you can edit the action in action editor by double clicking the action and choosing the options nest to resource to and point to the resource file icon that you want for that button.

DOCK WIDGET

Answers

1. Can be used to create detachable widget panels. They can be closed or docked in an area or floated
2. Left. Right, Top, BottomDockWidgetArea
3. Select the AllDockWidgetFeatures property

TAB WIDGET

Answers

1. It is used to displays information in chunks. Used to split information in small chunks when each tab button is selected
2. Converting a tab to a widget
    - Toolbox (instead of having the tabs from left to right it appears one below the other
    - Stacked widget (only one widget is available at a time via a combo box)
    - Adding items in the combobox → self.ui.combobox.addItem("Food")
    - To convert from tab to toolbox right click → Morph Into → QToolBox
    - Renaming the tabs can be done using the currentItemText property.
3. Settings of the Tab
    - You can change the colour of the tab by adding a resource
    - You can change the location of the tab order by selecting tabPosition

Other Questions on Menus

1. Advantages of Menues
    a. Menus do not require you to remember large commands.
    b. Unlike the command-line interface, the use of menus reduces the number of actions required to perform a specific task
    c. Menus enhance the navigation of an application. In a GUI appliciton you can provide Menu items at every page or form to help the user identify the tasks, and select the appropriet command to navigate further in the appliction
    d. Menus increase the accessibility of an application. Unlike the command line interface, you can add many options to complete tasks

2. Give the necessary steps on how to add a menu to an item (2)
    a. In Qt you can create a Main Window
    b. You will then be left with empty placeholders that you can rename
    c. Name them accordingly and save and conver the Qt File.
    d. Write python code to connect the signal/slot to a function.

3. Explain how to attach code to menu items?. 3

```
#This will connect the Menu item which we named Copy to a function "CopyItem"
Self.connect(self.ui.Copy, QtCore.SIGNAL('triggered()'),self.CopyItem)
#This is the function that will envoke once the Copy menu is triggered/clicked
def CopyItem (self):
#This is just an example but the function can be programmed as you want
self.ui.subwindow.setFocus()
```

1. Explain the different ways that can be used to trigger the same operation. (3)
    1. Menus
    2. Toolbars
    3. Widgets
2. Discuss 3 reasons why menus are required in the development of a database application (2)

    A common use of menus is to provide convenient access to various operations such as saving, opening a file, quitting a program or manupilating data. Most Widget provide some form of pull-down or pop-up menu. A menu offers a convenient way to group commands so that users can access them easily. In GUI applications you choose items and execute programs by pointing and clicking with a mouse. For example consider the interfaces of MS Excel a spreadsheet application. You create and open a spreadsheet by clicking on the menu items in the applications GUI.

Question 3

1. Characteristics of MDI
    a. Consists of a main window which had a menu bar, toolbar and central workspace
    b. One acts as parent and the others a child
    c. Central workspace manages the child widgets
    d. It is represented by the MdiArea
    e. Child windows are also called subwindows
2. benefits for the use of multiple forms
    a. With multiple document interfaces (and also tabbed document interfaces), a single menu bar and/or toolbar is shared between all child windows, reducing clutter and increasing efficient use of screen space. This argument is less relevant on an operating system which uses a common menu bar.
    b. An application's child windows can be hidden/shown/minimized/maximized as a whole.
    c. Features such as "Tile" and "Cascade" can be implemented for the child windows.
3. Modal and Modeless
4. Code

```
def closeAll(self):
 self.ui.mdiArea.closeAllSubWindows()

def cascadeArrange(self):
 self.ui.mdiArea.cascadeSubWindows()

def tileArrange(self):
```

```
        self.ui.mdiArea.tileSubWindows()

    def SubWindowView(self):
        self.ui.mdiArea.setViewMode(0)

    def TabbedView(self):
        self.ui.mdiArea.setViewMode(1)
```

Question 4

1. Database benefits
    a. Multiple users can open, save and update information simultaneously
    b. Databases can be maintained with backups and optimisation
    c. Databases are more reliable and can be setup for high performance and high availability
2. Data is raw/unfiltered unstructured whereas information is structured/filtered data which is somehow usefull to the user.
3. What are the benefits of using MySql
    a. MySql is very popular amongst developers
    b. It is an open source system which means there is no need for licence
    c. It takes less storage and has remarkable performance
    d. Available for Windows, Unix, Linux and Mac OS
    e. It is easy to maintain and upgrade
    f. It has an efficient query engine
4. SQL commands
    a. create database shopping;
    b. show tables;
    c. describe products;
5. a. code – if you using mysql.Connector

```
#InsertRow.py
import sys
import mysql.connector
conn = mysql.connector.connect(host="localhost",user="root",password="test",database=" shopping ")
cursor = conn.cursor()
cursor.execute("create table cars(Name Varchar, Surname Varchar, DateValue DATE, Address Varchar)")
print ("One Row Added")

cursor.close()
conn.commit()
conn.close
```

    b. Code

```
#InsertRow.py
    import sys
import mysql.connector
conn = mysql.connector.connect(host="localhost",user="root",password="test",database="shopping")
cursor = conn.cursor()
cursor.execute("INSERT INTO products VALUES ('Mike', 'Praisley', 23-12-1995, 23 First road)")
print ("One Row Added")

cursor.close()
conn.commit()
conn.close
```

6. Code

```
import sys
from assignment2 import *
from PyQt4 import QtSql, QtGui

def createConnection():
    db = QtSql.QSqlDatabase.addDatabase('QMYSQL')
    db.setHostName('localhost')
    db.setDatabaseName('booking')
    db.setUserName('root')
    db.setPassword('nowin')
    db.open()
    print (db.lastError().text())
    return True

class MyForm(QtGui.QMainWindow):
    def __init__(self, parent=None):
        QtGui.QWidget.__init__(self, parent)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.model = QtSql.QSqlTableModel(self)
        self.model.setTable("players2")
        self.model.setEditStrategy(QtSql.QSqlTableModel.OnManualSubmit)
        self.model.select()
        self.ui.tableView.setModel(self.model)
        #connects the button UpdateButton to the function UpdateRecords.
        QtCore.QObject.connect(self.ui.UpdateButton, QtCore.SIGNAL('clicked()'),
                    self.UpdateRecords)
        QtCore.QObject.connect(self.ui.CancelButton, QtCore.SIGNAL('clicked()'),
                    self.CancelChanges)
        QtCore.QObject.connect(self.ui. DeleteButton, QtCore.SIGNAL('clicked()'),
                    self.DeleteRecords)

    def UpdateRecords(self):
    #submits the current edited row
     self.model.submitAll()

    def CancelChanges(self):
    #cancels and changes
     self.model.revertAll()

    def DeleteRecords(self):
    #deletes a row
     self.model.removeRow(self.ui.tableView.currentIndex().row())
     self.model.submitAll()

if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    if not createConnection():
        sys.exit(1)
    myapp = MyForm()
    myapp.show()
    sys.exit(app.exec_())
```

END of Exam