Activity Name #4 - Introduction to GUI Development using Pycharm	
Delavin, Katarina Nicole R.	10/17/24
CPE 009B - CPE21S1	Engr. Maria Rizette Sayo

6. Supplementary Activity

- **Main**

```
import sys
from PyQt5.QtWidgets import QWidget, QLineEdit, QPushButton,
QApplication, QLabel
from PyQt5.QtGui import QIcon
from registration import App

if __name__ == '__main__':
   app = QApplication(sys.argv)
   ex = App()
   ex.show()
   sys.exit(app.exec_())
```

- **Registration**

```
import sys
from PyQt5.QtWidgets import QWidget, QLineEdit, QPushButton,
QApplication, QLabel
from PyQt5.QtGui import QIcon


class App(QWidget):

   def __init__(self):
       super().__init__()
       self.title = "PyQt Registration"
       self.x = 300
       self.y = 300
       self.width = 300
       self.height = 300
       self.initUI()

   def initUI(self):
       self.setWindowTitle(self.title)
       self.setGeometry(self.x, self.y, self.width, self.height)
       self.setWindowIcon(QIcon('pythonico.ico'))

       self.textboxlbl = QLabel("Registration", self)
       self.textboxlbl.move(120, 20)

       self.textboxlbl2 = QLabel("First name:", self)
       self.textboxlbl2.move(20, 70)
       self.textboxentry = QLineEdit(self)
       self.textboxentry.move(80, 70)

       self.textboxlbl3 = QLabel("Last name:", self)
       self.textboxlbl3.move(20, 100)
```

```
        self.textboxentry2 = QLineEdit(self)
        self.textboxentry2.move(80, 100)

        self.textboxlbl4 = QLabel("Username:", self)
        self.textboxlbl4.move(20, 130)
        self.textboxentry3 = QLineEdit(self)
        self.textboxentry3.move(80, 130)

        self.textboxlbl5 = QLabel("Password:", self)
        self.textboxlbl5.move(20, 160)
        self.textboxentry4 = QLineEdit(self)
        self.textboxentry4.move(80, 160)
        self.textboxentry4.setEchoMode(QLineEdit.Password)

        self.textboxlbl6 = QLabel("Email:", self)
        self.textboxlbl6.move(20, 190)
        self.textboxentry5 = QLineEdit(self)
        self.textboxentry5.move(80, 190)

        self.button = QPushButton('Submit', self)
        self.button.clicked.connect(self.submit)
        self.button.move(50, 230)

        self.result_label = QLabel("", self)
        self.result_label.move(80, 280)

        self.button2 = QPushButton('Clear', self)
        self.button2.move(150, 230)
        self.button2.clicked.connect(self.clear)

    def submit(self):
        first_name = self.textboxentry.text()
        last_name = self.textboxentry2.text()
        username = self.textboxentry3.text()
        password = self.textboxentry4.text()
        email = self.textboxentry5.text()
        self.result_label.setText("Registration successful!")
    def clear(self):
        self.textboxentry.clear()
        self.textboxentry2.clear()
        self.textboxentry3.clear()
        self.textboxentry4.clear()
        self.textboxentry5.clear()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = App()
    sys.exit(app.exec_())
```

- **GUI labels**

```
import sys
from PyQt5.QtWidgets import QWidget, QLineEdit, QMainWindow,
QApplication, QLabel
from PyQt5.QtGui import QIcon


class App(QWidget):

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

```
        self.title = "PyQt Line Edit"
        self.x = 200
        self.y = 200
        self.width = 300
        self.height = 300
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.x, self.y, self.width, self.height)
        self.setWindowIcon(QIcon('pythonico.ico'))

        self.textboxlbl = QLabel("Hello world!", self)
        self.textboxlbl.move(100, 150)
        self.textboxlbl = QLabel("This program is written in Pycharm",
self)
        self.textboxlbl.move(100, 170)

        self.show()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = App()
    sys.exit(app.exec_())
```

- **GUI buttons**

```
import sys
from PyQt5.QtWidgets import QWidget, QPushButton, QMainWindow,
QApplication
from PyQt5.QtGui import QIcon


class App(QWidget):

    def __init__(self):
        super().__init__()
        self.title = "PyQt Button"
        self.x = 200
        self.y = 200
        self.width = 300
        self.height = 300
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.x, self.y, self.width, self.height)
        self.setWindowIcon(QIcon('pythonico.ico'))

        self.button = QPushButton('Click me!', self)
        self.button.setToolTip("You've hovered over me")
        self.button.move(100, 70)
        self.button2 = QPushButton('Register', self)
        self.button2.setToolTip('This button does nothing... yet')
        self.button2.move(100, 95)
        self.show()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = App()
```

```
    sys.exit(app.exec_())
```

- **GUI text**

```
import sys
from PyQt5.QtWidgets import QWidget, QLineEdit, QMainWindow,
QApplication
from PyQt5.QtGui import QIcon


class App(QWidget):

    def __init__(self):
        super().__init__()
        self.title = "PyQt Line Edit"
        self.x = 200
        self.y = 200
        self.width = 300
        self.height = 300
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.x, self.y, self.width, self.height)
        self.setWindowIcon(QIcon('pythonico.ico'))

        self.textbox = QLineEdit(self)
        self.textbox.move(20, 20)
        self.textbox.resize(280, 40)
        self.textbox.setText("Set this text value")
        self.show()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = App()
    sys.exit(app.exec_())
```

- **Class app**

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5.QtGui import QIcon


class App(QMainWindow):
    def __init__(self):
        super().__init__()
        self.title = "First OOP GUI"
        self.initUI()

    def initUI(self):
        self.setWindowTitle(self.title)
        self.setGeometry(200, 200, 300, 300)
        self.setWindowIcon(QIcon('pythonico.ico'))
        self.show()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    Main = App()
    sys.exit(app.exec_())
```

Questions

1. What are the common GUI Applications that general end-users such as home users, and office employees use? (give at least 3 and describe each)

- Google docs utilized for text document creation, editing, and formatting. These programs include features including grammar suggestions, spell checkers, text formatting tools, and template designs. They are essential for office workers producing documents, memos, and other written content, as well as for students composing essays, reports, or homework.

- Google Chrome used to access the internet, allowing users to view and interact with websites, stream media, and use web-based applications. These applications offer a graphical user interface for browsing the web, saving bookmarks, managing tabs, and customizing privacy settings. For students and office employees, web browsers are essential for research, communication, and productivity tools like Google Docs

- Email help users send, receive, and organize email communications. They often come with features like contact management, scheduling, and task management. These applications are essential for office employees who rely on email for business communication, scheduling meetings, and managing contacts. Students also use email for corresponding with teachers, peers, and organizing academic schedules.

2. Based from your answer in question 1, why do you think home users, students, and employees use those GUI programs?

- Home Users Generally graphical user interface (GUI) apps are made to be visually simple to use, with easily interpreted buttons, menus, and icons. Complex command-line interfaces are no longer necessary for non-technical users to master thanks to GUI-based software, which makes technology more approachable. and Students Word processors, spreadsheets, and web browsers are among the many tools that students frequently need to quickly access for a range of tasks. They can quickly and easily create, amend, and manage assignments thanks to a graphical user interface (GUI). while the Office Workers These workers have to finish duties fast, like writing emails, analyzing data, and creating documents. GUI applications streamline these processes by offering recognizable user interfaces and lowering the requirement for technical know-how.

3. How does Pycharm help developers in making GUI applications, what would be the difference if developers made GUIprograms without GUI Frameworks such as Pycharm Tkinter?

- With capabilities like code completion, debugging, and version control, PyCharm is an integrated development environment (IDE) that assists developers in creating graphical user interfaces (GUIs). By supporting frameworks like PyQt or Tkinter, it makes the process of creating GUI applications more efficient and provides tools for effectively managing intricate projects and designing interfaces visually.Developers

would have to manually handle lower-level tasks like rendering windows, drawing widgets, and processing user inputs if GUI frameworks like PyCharm or Tkinter weren't available. They would have to handle system-level operations and graphical aspects that frameworks abstract away, which takes more time, knowledge, and work.

-

4. What are the different platforms a GUI program may be created and deployed on? (Three is required then state why might a program be created on tha specific platform)

- Desktop
- Web
- Android

5. What is the purpose of app = QApplication(sys.argv), ex = App(), and sys.exit (app.exec_())?

- These three lines of code are frequently seen in Python bindings for the Qt framework, such as PyQt or PySide, which are used to create GUI applications. Each line serves the following purpose:

1. QApplication(sys.argv) = app
The goal is to construct an instance of the QApplication class, which is necessary for every GUI application based on Qt. It controls the event loop, preferences, and resources for the entire application. Any command-line arguments that are supplied to the application can be handled by the program with the help of the sys.argv argument.
2. ex = App()
Goal: This generates an instance of a custom class (App in this example) that is derived from a Qt widget such as QMainWindow or QWidget. The primary GUI components, such as windows, buttons, and layouts, are developed and shown here. It serves as the foundation for specifying the application's

3. sys.exit(application.exec())
Goal: This initiates the Qt event loop, which watches for and handles user inputs (such keystrokes or clicks). Until the window is dismissed, the program remains open and responsive to events thanks to app.exec_(). When the event loop concludes, sys.exit() guarantees a clean exit for the Python interpreter and returns the application's exit status. Combining these lines ensures that the GUI application is launched and runs correctly, handling user input and ending gracefully.

7. Conclusion:

In a nutshell, Object-Oriented Programming (OOP) GUI development offers a strong and adaptable framework for creating user interfaces that are scalable and intuitive. The fundamental concepts of OOP, such as polymorphism, inheritance, and encapsulation, allow programmers to create modular, reusable, and manageable code for intricate graphical user interfaces. Event-driven programming is made easier by the abstraction provided by classes and objects, which makes it possible to manage user interactions effectively. OOP also encourages the division of responsibilities, which makes architecture

clearer by separating the GUI design from the underlying business logic. All things considered, incorporating OOP into GUI development improves the capacity to create complex, dynamic, and intuitive programs.