Laboratory Activity No. 9	
Introduction to GUI Development using Pycharm	
Course Code: CPE103	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed:**
Section:	**Date Submitted:**
Name:	**Instructor:**

1. Objective(s):

This activity aims to familiarize students with the Pycharm framework for GUI Development

2. Intended Learning Outcomes (ILOs):

The students should be able to:
2.1 Identify the main components in a GUI Application
2.2 Create a simple GUI Application using Pycharm Widgets

3. Discussion:

A Graphical User Interface (GUI) application is a program that the user can interact with through graphics (windows, buttons, text fields, checkboxes, images, icons, etc..) such as the Desktop GUI of Windows OS by using a mouse and keyboard unlike with a Command-line program or Terminal program that support keyboard inputs only.

Pycharm is an integrated development environment used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django.

4. Materials and Equipment:

Desktop Computer with Anaconda Python or Pycharm
Windows Operating System

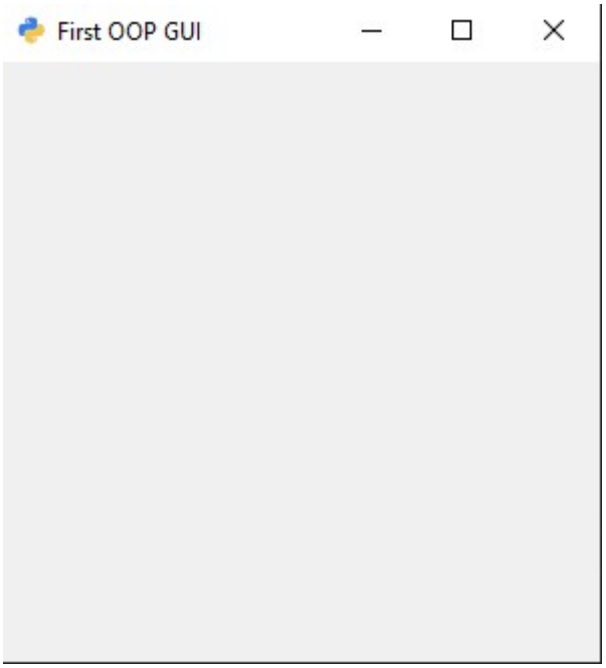
5. Procedure:

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

class App(QMainWindow):

    def __init__(self):
        super().__init__() # initializes the main window like in the previous one
        # window = QMainWindow()
        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')) # sets an icon
        self.show()

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

```

2. Run the program and observe the output.

Adding an icon

3. Download any .ico picture from https://icon-icons.com/ or any similar sites.
4. Place the icon in your folder (ex. Oopfa1<lastname>_lab8)
5. Run the program again, the program should now have an icon similar to the program below.



Creating Buttons

1. Create a new .py file named **gui_buttons.py** then copy the program as shown below:

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

class App(QWidget):

    def __init__(self):
        super().__init__() # initializes the main window like in the previous one
        # window = QMainWindow()
        self.title= "PyQt Button"
        self.x=200 # or left
        self.y=200 # or top
        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'))

        # In GUI Python, these buttons, textboxes, labels are called Widgets
        self.button = QPushButton('Click me!', self)
        self.button.setToolTip("You've hovered over me!")
        self.button.move(100,70)  # button.move(x,y)

        self.show()


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

2. Run the program and observe the output.
3. Add a new button named button2 named Register to the GUI that will display “this button does nothing.. yet..” when it is hovered.

Creating Text Fields

1. Create a new file named **gui_text.py** and copy the code shown below:

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

class App(QWidget):

    def __init__(self):
        super().__init__() # initializes the main window like in the previous one
        # window = QMainWindow()
        self.title= "PyQt Line Edit"
        self.x=200 # or left
        self.y=200 # or top
        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'))

        # Create textbox
        self.textbox = QLineEdit(self)
        self.textbox.move(20, 20)
        self.textbox.resize(280,40)

        self.show()

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

2. Run the program and observe the error.
3. Add an import QLineEdit to the Pycharm.Widgets import
4. Run the program and observe the output.
5. Add the following code below self.textbox.resize()

```python
self.textbox.setText("Set this text value")
```

4. Run the program again then resize the textbox so that it fits in the Window and that its height is just above the written text's height.

Creating Labels

1. Create a new file called **gui_labels.py** and copy the following code below:

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

class App(QWidget):

    def __init__(self):
        super().__init__() # initializes the main window like in the previous one
        # window = QMainWindow()
        self.title= "PyQt Line Edit"
        self.x=200 # or left
        self.y=200 # or top
        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(30,25)

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

```

2. Run the program and observe the output.
3. Add the necessary Widget at the import line to make the program run.
4. Center the label by adjusting the parameters of .move(). This is called Absolute Positioning.
5. Create a new label called “This program is written in Pycharm” and place it at the center and below the Hello World!

6. Supplementary Activity:

Task

Create an Object-Oriented GUI Application for a simple Account Registration System with the following required information: first name, last name, username, password, email address, contact number.

Requirements:

- The GUI must be centered on your screen.
- The GUI Components should be organized according to the order of information required using Absolute Positioning.
- The position of the components should be auomatically computed based on the top component.
- All the text fields should be accompanied with their corresponding label on the left side while the text field is on the right side.
- There should be a program title other than the Window Title.
- There should be a submit button and clear button at the bottom (submit button on the left, clear button on the right).
- The program should be launched on **main.py** while the GUI Codes should be on a separate file called **registration.py**

Questions

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

   -People commonly use Microsoft Word for documents, Web Browsers for internet access, and Microsost Excel for organizing data. These apps make work, study, and daily tasks easier.

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

   -These programs are popular because they're easy to use, boost productivity, support teamwork, and work for different needs.

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

   -It simplifies GUI development with code assistance and built-in support for frameworks like PyQt and Tkinter. Without these, developers would have to manually build UI components, making things harder and slower.

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 that specific platform)

   -GUI apps can run on Windows (for business and personal use), Linux (for open-source development), and web platforms (for cross-device access). The platform depends on who will use the app.

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

   - app = QApplication(sys.argv), ex = App() it creates the window and sys.exit(app.exec_()) keeps it running until the user exits

7. Conclusion:

GUI development makes software easier to use, whether for work, school, or everyday tasks. With tools like PyCharm and frameworks like PyQt or Tkinter, developers can create user-friendly applications without the hassle of building everything from scratch. Choosing the right platform—Windows, Linux, or web—depends on who will use the app and how. Understanding how GUI programs work and the key commands behind them helps create smoother, more efficient experiences. In the end, well-designed GUIs make technology more accessible and help people get things done faster and easier.

8. Assessment Rubric: