Chapter 9: Model Solutions

Below, you'll find sample solutions to the lab exercises in the book.

Lab Exercises 9.1

1. Tkinter is the standard GUI (Graphical User Interface) toolkit for Python. Its role is to provide a framework for creating GUI applications with various widgets and controls.

2. Import the tkinter module. Create an instance of the Tk class to represent the main window. Optionally, set properties such as title, size, and position. Call the mainloop() method to start the event loop.

```
import tkinter as tk

window = tk.Tk()

window.title("My Window")

window.geometry("300x200")

window.resizable(False, False)

label = tk.Label(window, text="Welcome to Tkinter!")

label.pack()

window.mainloop()
```

3. Label: Displays text or an image.
   Button: Executes a command when clicked.
   Entry: Allows user input.
   Text: Multiline text area.
   Canvas: Drawing area.
   Listbox: Displays a list of items.
   Checkbutton: Checkbox for selecting options.
   Radiobutton: Selects one option from multiple choices.
   Menu: Creates menu bars and dropdown menus.

4. Define a function that will be executed when the button is clicked. Then use the command parameter of the Button widget to associate the function with the button.

```
import tkinter as tk

def button_clicked():

    print("Button Clicked!")

window = tk.Tk()

button = tk.Button(window, text="Click Me",
command=button_clicked)

button.pack()

window.mainloop()
```

5. Create a Menu object for the menu bar.

Create Menu objects for each dropdown menu.

Add menu items to each dropdown menu using the add_command() method.

Associate functions with menu items using the command parameter.

6. See paint.py
7. Import the necessary modules: PyQt5.QtWidgets and sys.

   Create an instance of the QApplication class.

   Create an instance of the QMainWindow class.

   Set the window title and size.

   Show the window using the show() method.

   Start the application event loop using sys.exit(app.exec_()).
8. Creating a menu bar, menu, and menu items.

```
from PyQt5.QtWidgets import QApplication, QMainWindow,
QMenuBar, QMenu, QAction

app = QApplication([])

window = QMainWindow()

window.setWindowTitle("Menu Example")

menu_bar = QMenuBar(window)

file_menu = QMenu("File", window)

edit_menu = QMenu("Edit", window)

help_menu = QMenu("Help", window)

menu_bar.addMenu(file_menu)

menu_bar.addMenu(edit_menu)

menu_bar.addMenu(help_menu)

open_action = QAction("Open", window)

save_action = QAction("Save", window)

exit_action = QAction("Exit", window)

file_menu.addAction(open_action)

file_menu.addAction(save_action)

file_menu.addAction(exit_action)

window.setMenuBar(menu_bar)

window.show()

app.exec_()
```

9. Use the QLabel widget to display an image. Create an instance of QPixmap with the path to the image file. Set the QPixmap instance as the pixmap of the QLabel. Add the QLabel to the window layout. Show the window.

```
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel

from PyQt5.QtGui import QPixmap

app = QApplication([])

window = QMainWindow()

window.setWindowTitle("Image Example")

label = QLabel(window)

label.setGeometry(50, 50, 300, 300)  # set size

image_path = "test.jpg"              # replace test.jpg

pixmap = QPixmap(image_path)

label.setPixmap(pixmap)

window.show()

app.exec_()
```

10. Create an instance of the QPushButton class.

    Set the button text using the setText() method.

    Set the button position using the move() method.

    Set the button size using the resize() method.

    Connect a function to the button click event using the clicked signal.

11. Message box

```
from PyQt5.QtWidgets import QApplication, QMessageBox

app = QApplication([])


# Information Box

QMessageBox.information(None, "Information", "This is an
information message.")

# Warning Box

QMessageBox.warning(None, "Warning", "This is a warning
message.")

# Critical Box

QMessageBox.critical(None, "Critical", "This is a critical
message.")

# Question Box
```

```
result = QMessageBox.question(None, "Question", "Do you want
to proceed?")

if result == QMessageBox.Yes:

    print("User clicked Yes")

else:

    print("User clicked No")

app.exec_()
```

12. Create an instance of the QLineEdit class.

    Set the position using the move() method.

    Set the size using the resize() method.

    Add the QLineEdit to the window layout.

13. Create an instance of the QListWidget class.

    Add items to the list box using the addItem() method.

    Connect a function to the itemSelectionChanged signal to handle item selection.

14. Create an instance of the QCheckBox class.

    Set the position using the move() method.

    Set the size using the resize() method.

    Connect a function to the stateChanged signal to handle state changes.

15. Create an instance of the QLabel class.

    Set the position using the move() method.

    Set the size using the resize() method.

    Add text to the label using the setText() method.

    Add the QLabel to the window layout.

16. See fileexp.py