Tạo giao diện người dùng với các thành phần Qt

Nội dung

Chúng ta sẽ học cách sử dụng các widget sau:

- Hiển thị thông điệp chào mừng
- Sử dụng tiện ích Nút Radio
- Nhóm các nút radio
- Hiển thị tùy chọn ở dạng hộp kiểm
- Hiển thị hai nhóm hộp kiểm

Học tạo ứng dụng GUI với Qt và PyQt5

- **Qt**: bộ công cụ đa nền tảng cho phát triển GUI (Windows, macOS, Linux…).
- **Widget**: nút, nhãn, hộp văn bản, danh sách… để thiết kế giao diện.
- **PyQt5**: ràng buộc Python cho Qt, dùng tạo ứng dụng thời gian thực.

PyQt

- **PyQt** là thư viện cho phép viết ứng dụng GUI bằng **Python** dựa trên **Qt**. Nó giúp Python truy cập mọi tính năng của Qt.
- Khi cài PyQt, Qt cũng được cài tự động.
- **Ứng dụng GUI** có thể là:
  - **Hộp thoại**: nhỏ, chỉ có nút, không có menu hay thanh công cụ.
  - **Cửa sổ chính**: đầy đủ menu, thanh công cụ, thanh trạng thái và widget trung tâm.

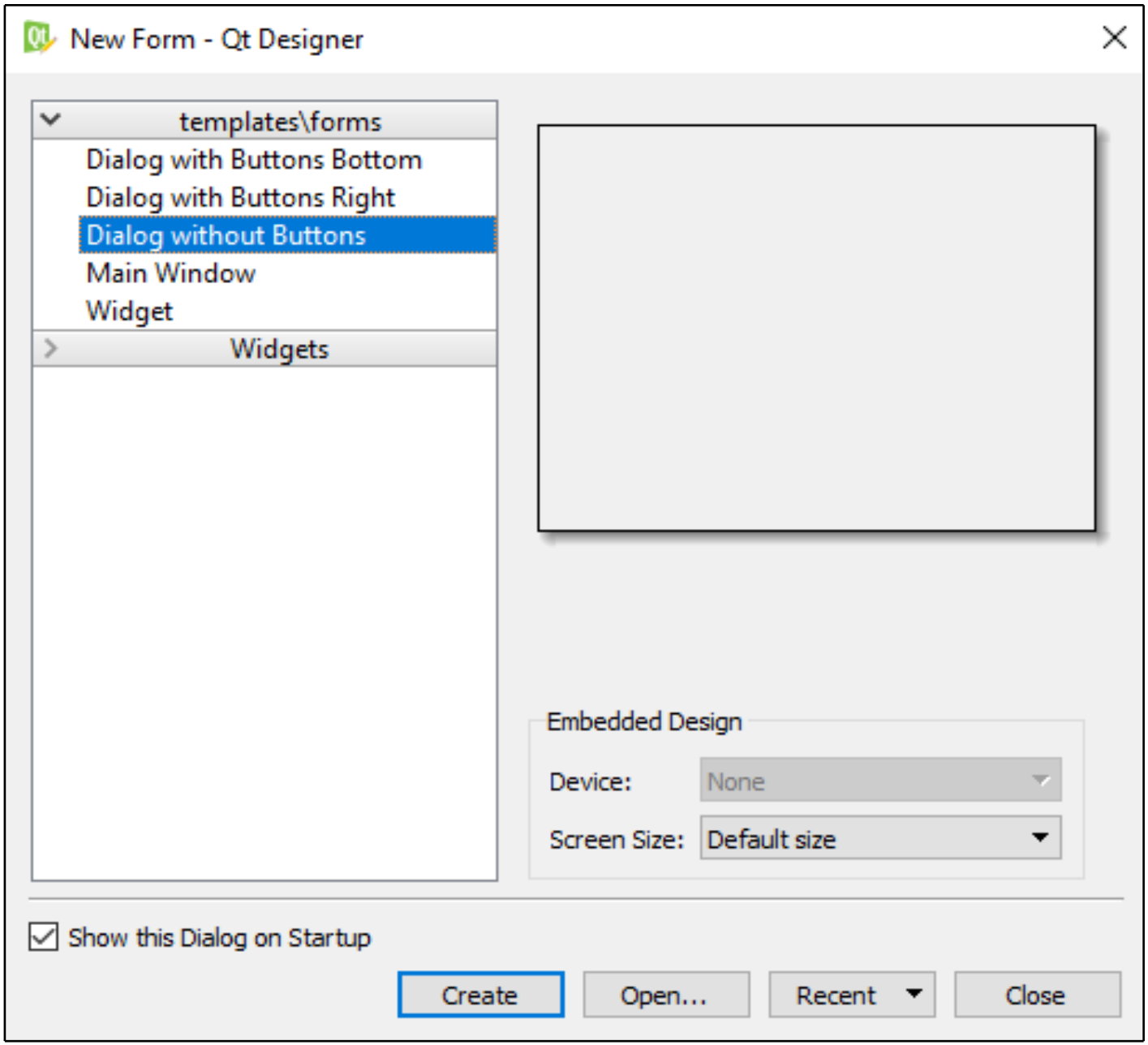
Hộp thoại (dialog) có hai loại sau

- **Modal:** Hộp thoại khóa ứng dụng, người dùng chỉ có thể tương tác với hộp thoại cho đến khi đóng.
- **Modeless:** Hộp thoại cho phép người dùng tương tác đồng thời với hộp thoại và ứng dụng.

Các cách tạo ứng dụng GUI

- Từ đầu bằng trình soạn thảo văn bản
- Hoặc dùng **Qt Designer**: kéo-thả nhanh, dễ thiết kế
- Trong PyQt, **Qt Designer** giúp tạo GUI mà không cần viết mã nhiều

Khi mở Qt Designer yêu cầu chọn một mẫu cho ứng dụng mới



Qt Designer – Các mẫu ứng dụng sẵn có

- **Dialog with Buttons Bottom:** Hộp thoại với nút **OK/Cancel** ở góc dưới phải.
- **Dialog with Buttons Right:** Hộp thoại với nút **OK/Cancel** ở góc trên phải.
- **Dialog without Buttons:** Hộp thoại trống để đặt widget, lớp cha là **QDialog**.
- **Main Window:** Cửa sổ chính có **menu** và **toolbar**, có thể gỡ bỏ nếu không cần.
- **Widget:** Biểu mẫu với lớp cha là **QWidget**, không phải QDialog.

Cấu trúc và thiết kế GUI trong Qt

Mỗi ứng dụng có **widget cấp cao nhất** (QDialog, QWidget, QMainWindow), các widget khác là **widget con**.

Chọn widget cấp cao nhất tùy theo loại ứng dụng:

- QDialog → ứng dụng hộp thoại
- QMainWindow → ứng dụng cửa sổ chính
- QWidget → ứng dụng widget cơ bản

Qt Designer: kéo-thả widget từ hộp widget, sắp xếp bố cục, chỉnh thuộc tính và kết nối **signal → slot**.

Hiển thị thông điệp chào mừng

- Người dùng nhập tên và nhấn nút.
- Hệ thống hiển thị thông báo: "Xin chào [tên]".
- Cần 3 widget: **Label**, **Line Edit**, **Push Button**.

QLabel trong PyQt

- Dùng để hiển thị **văn bản** hoặc **hình ảnh**, chỉ để **thông báo, không nhận input**.
- **Các phương thức chính:**
  - setText(): gán văn bản
  - setPixmap(): gán hình ảnh (QPixmap)
  - setNum(): gán số (nguyên hoặc thập phân)
  - clear(): xóa nội dung
- Văn bản mặc định là **TextLabel**, có thể thay đổi qua **Property Editor** hoặc setText()

QLineEdit (Widget nhập dữ liệu một dòng)

- Nhập/chỉnh sửa, hỗ trợ cắt/dán/hoàn tác
- **Chế độ hiển thị:** Normal, NoEcho, Password, PasswordEchoOnEdit
- **Chính:** maxLength(), setText()/text(), clear()
- **Chỉ đọc / bật-tắt:** setReadOnly(), isReadOnly(), setEnables()
- setFocus() – đặt con trỏ

Push Button widget

- Tạo nút bằng cách khởi tạo

QPushButton.

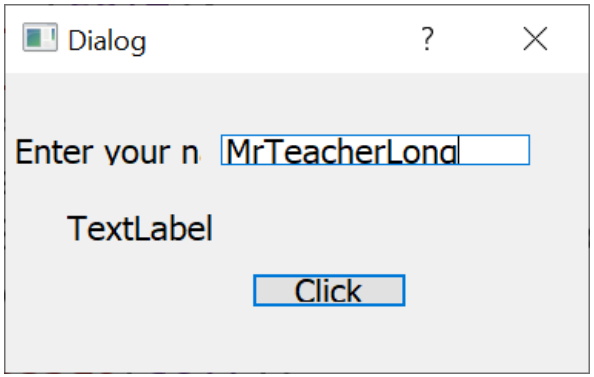
- Gán văn bản với **setText()**; tạo phím tắt

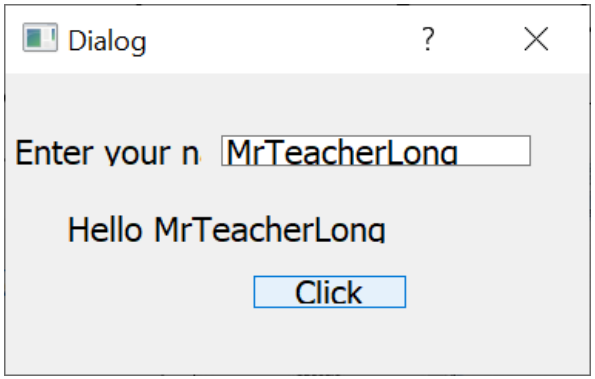
bằng **&** trước ký tự (Alt + ký tự).

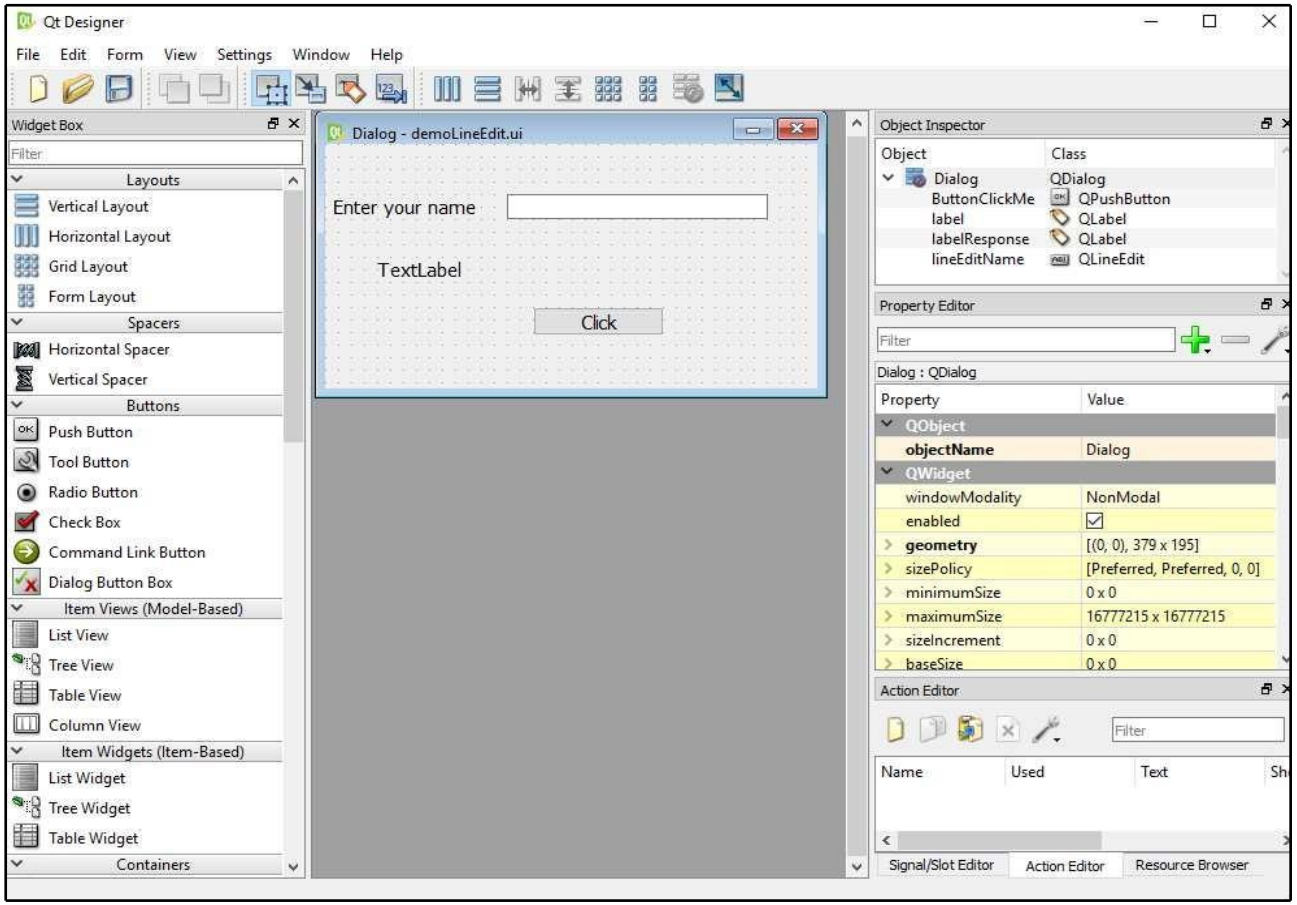
- Gán biểu tượng bằng **setIcon()**.
- Nút phát tín hiệu **clicked()** khi được nhấn.

Ứng dụng demo

Tạo ứng dụng từ mẫu ‘Dialog without Buttons’ yêu cầu người dùng nhập tên, sau đó hiển thị thông báo chào kèm tên đã nhập khi nhấn nút



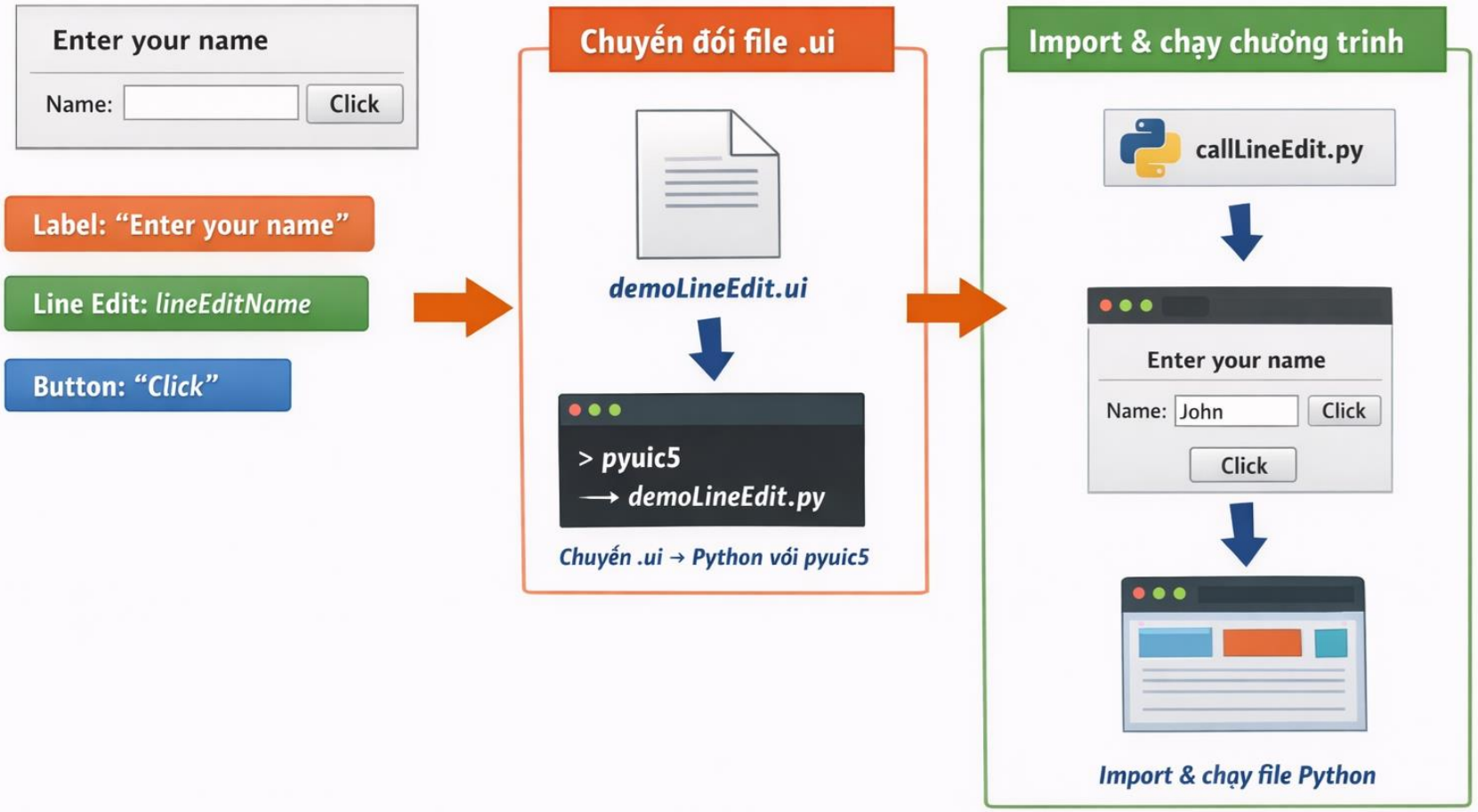




Tạo giao diện PyQt5 với Line Edit và Button

1. Thêm **Label** vào form, đặt text: *Enter your name*, objectName: labelResponse.
2. Thêm thêm một **Label** mặc định.
3. Thêm **Line Edit**, objectName: lineEditName.
4. Thêm **Push Button**, text: *Click*, objectName: ButtonClickMe.
5. Lưu file UI: demoLineEdit.ui.
6. Chuyển .ui → Python bằng **pyuic5** (có sẵn trong PyQt).
7. Xử lý mã Python (demoLineEdit.py) và import vào file gọi giao diện.
8. Tạo file callLineEdit.py để import và sử dụng giao diện.

Tạo giao diện *PyQt5* với *Line Edit* và *Button*



Giao diện người dùng tạo bằng Qt Designer được lưu trữ trong tệp .ui bao gồm tất cả thông tin của biểu mẫu: widget, bố cục, v.v. Tệp .ui là một tệp XML và bạn cần chuyển đổi nó thành mã Python. Bằng cách đó, bạn có thể duy trì một sự tách biệt rõ ràng giữa giao diện trực quan và hành vi được cài đặt trong mã.

```
d:\>pyuic5 demoLineEdit.ui -o demoLineEdit.py
```

```
import sys

from PyQt5.QtWidgets import QDialog, QApplication

from demoLineEdit import *

class MyForm(QDialog):
  def __init__(self):
    super().__init__()
    self.ui = Ui_Dialog()
    self.ui.setupUi(self)
    self.ui.ButtonClickMe.clicked.connect(self.dispmessage)
    self.show()

  def dispmessage(self):
    self.ui.labelResponse.setText(
                          "Hello "+self.ui.lineEditName.text())

if __name__=="__main__":
  app = QApplication(sys.argv)
  w = MyForm()
  w.show()
  sys.exit(app.exec_())
```

Cách chương trình hoạt động...

demoLineEdit.py

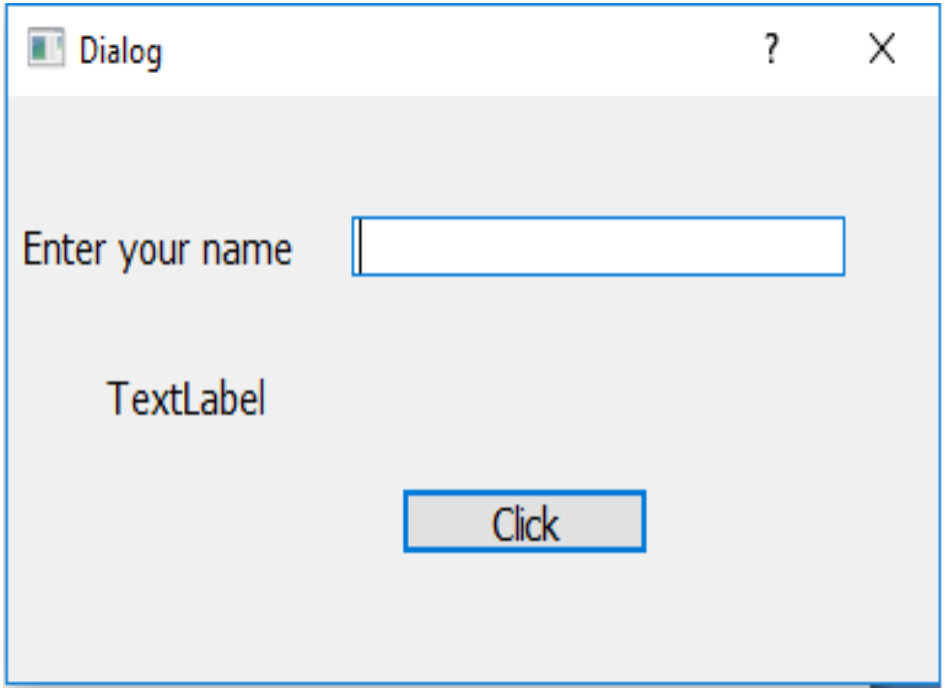
- Lớp Ui_Dialog lưu giao diện chính (widgets) của QDialog.
- setupUi() tạo và thiết lập các widget.
- retranslateUi() dịch/hiển thị văn bản giao diện.

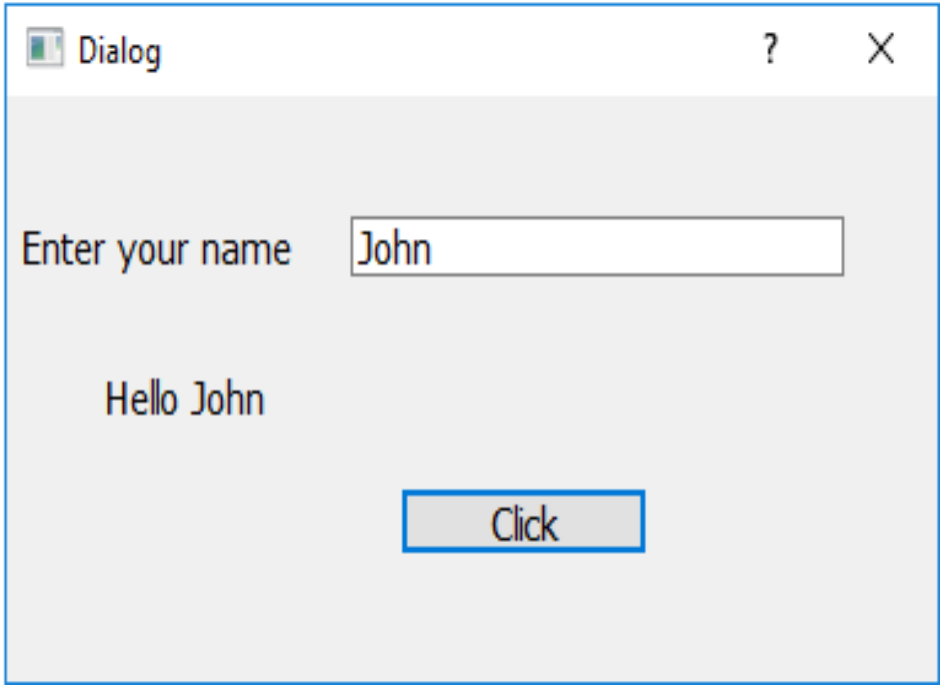
callLineEdit.py

- Tạo lớp MyForm kế thừa QDialog.
- Khởi tạo cửa sổ và widgets.
- Sử dụng **signals & slots**: ví dụ, nhấn nút → gọi dismessage().
- Tạo ứng dụng QApplication và hiển thị widget với show().
- dismessage() hiển thị "Hello" + tên nhập từ Line Edit.
- sys.exit() kết thúc chương trình và giải phóng tài nguyên.

Cách chương trình hoạt động... (t.t)

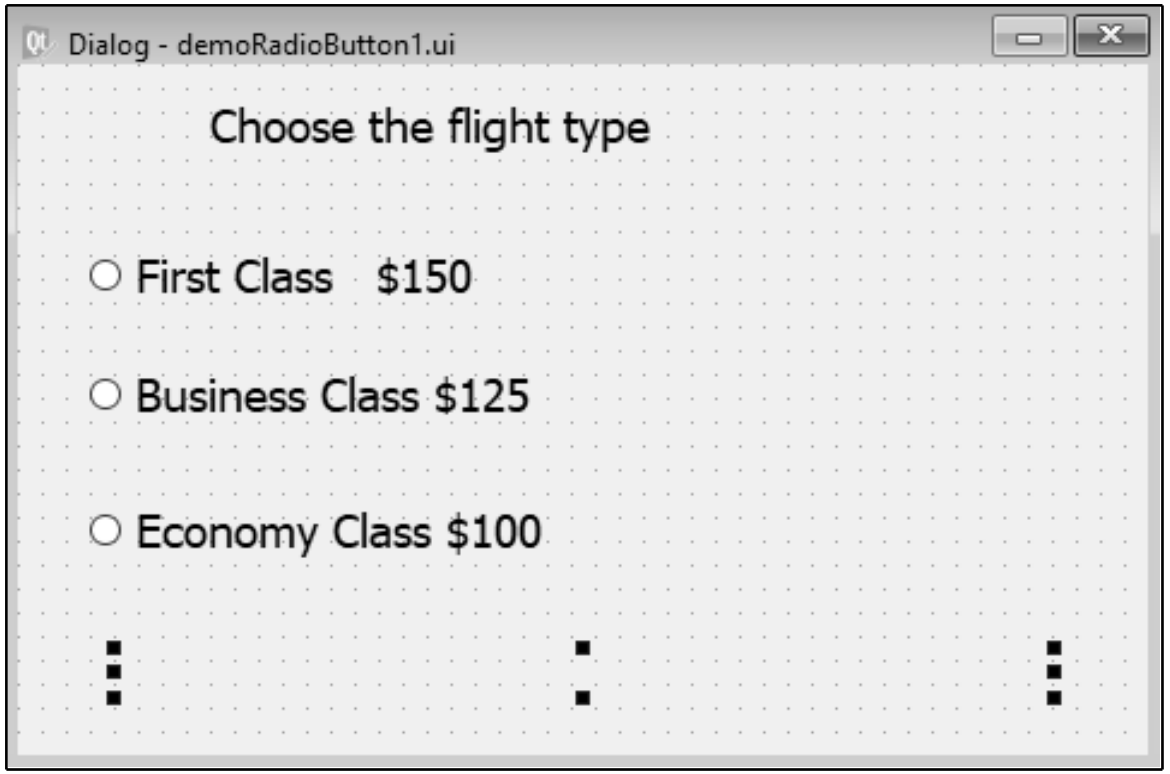
Khi nhấn nút sẽ chạy hàm dismessage(), hiển thị 'Hello' kèm tên nhập từ Line Edit.





Radio Button – Chọn chuyến bay

- Chọn **một tùy chọn duy nhất** (Hạng nhất / Thương gia / Tiết kiệm).
- **Hiển thị giá** tương ứng khi chọn.
- **QRadioButton:** có nhãn, trạng thái chọn/không chọn.
- **Tín hiệu chính:** clicked(), toggled(), stateChanged().



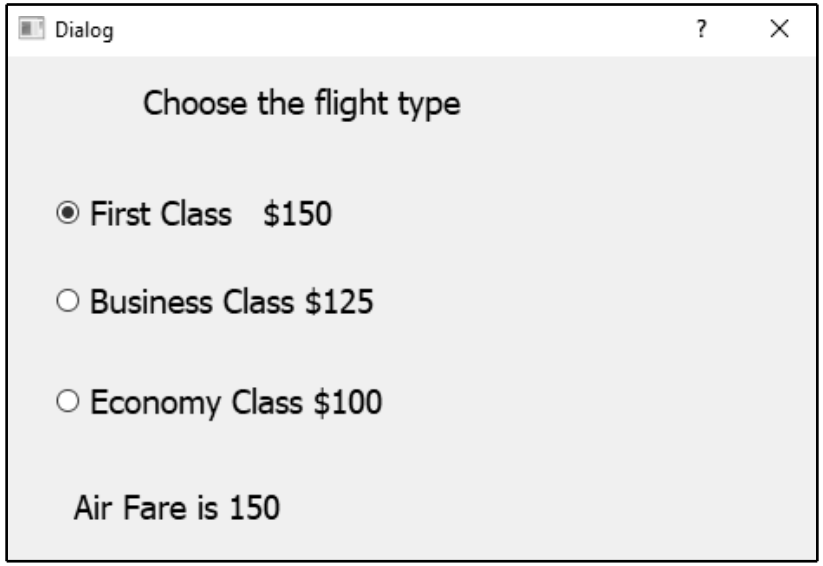
Tạo ứng dụng chọn loại chuyến bay

- Dùng mẫu **Dialog without Buttons**.
- Thêm **2 Label** và **3 Radio Button** vào form.
- Label 1: “Choose the flight type”
- Label 2 (labelFare): hiển thị giá khi chọn chuyến bay.
- Radio Button:
  - First Class $150 → radioButtonFirstClass
  - Business Class $125 → radioButtonBusinessClass
  - Economy Class $100 → radioButtonEconomyClass
- Lưu file: **demoRadioButton1.ui**

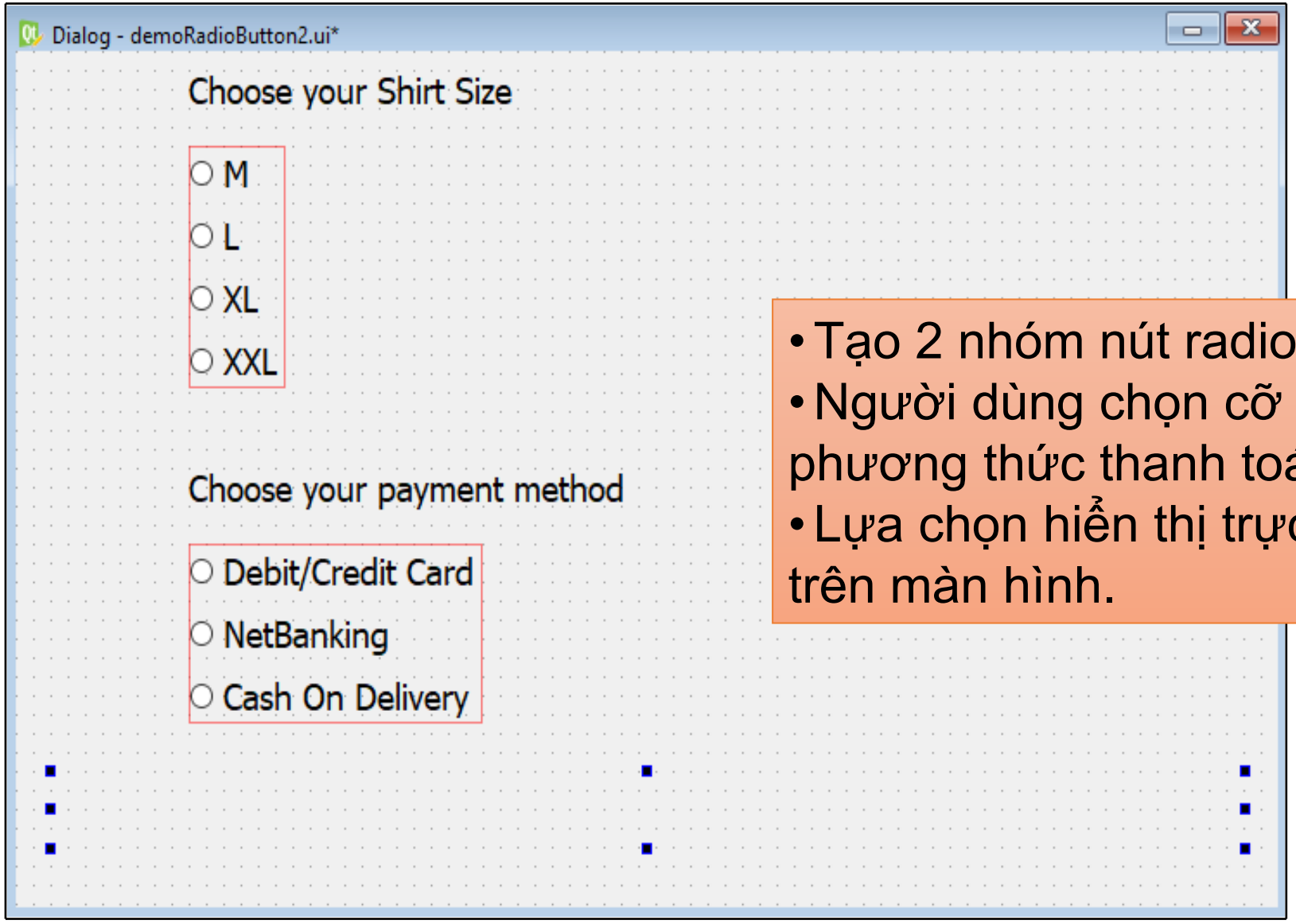
```
import sys
from PyQt5.QtWidgets import QDialog, QApplication
from demoRadioButton1 import *
class MyForm(QDialog):
  def __init__(self):
    super().__init__()
    self.ui = Ui_Dialog()
    self.ui.setupUi(self)
    self.ui.radioButtonFirstClass.toggled.connect(self.dispFare)
    self.ui.radioButtonBusinessClass.toggled.connect(self.dispFare)
    self.ui.radioButtonEconomyClass.toggled.connect(self.dispFare)
    self.show()
  def dispFare(self):
    fare=0
    if self.ui.radioButtonFirstClass.isChecked()==True:
      fare=150
    if self.ui.radioButtonBusinessClass.isChecked()==True:
      fare=125
    if self.ui.radioButtonEconomyClass.isChecked()==True:
      fare=100
      self.ui.labelFare.setText("Air Fare is "+str(fare))
if __name__=="__main__":
  app = QApplication(sys.argv)
  w = MyForm()
  w.show()
  sys.exit(app.exec_())
```

Hiển thị giá vé theo loại chuyến bay

- Người dùng chọn loại vé qua các **Radio Button** (First, Business, Economy).
- Sự kiện toggled() gọi hàm distFare() để xác định giá:
- First Class → 50
- Business Class → 125
- Economy Class → 100
- Giá được hiển thị trên **labelFare**.



Nhóm các Radio Button



- Tạo 2 nhóm nút radio để chọn.
- Người dùng chọn cỡ áo và phương thức thanh toán.
- Lựa chọn hiển thị trực tiếp trên màn hình.

Ứng dụng chọn kích thước áo và phương thức thanh toán với Radio Button (PyQt5)

1. Tạo ứng dụng mới theo mẫu **Dialog without Buttons**.
2. Thêm **Label**: “Choose your Shirt Size” và “Choose your payment method”, tăng kích thước phông chữ.
3. Thêm 4 **Radio Button** cho áo: M, L, XL, XXL → xếp dọc trong **VBoxLayout**.
4. Thêm 3 **Radio Button** cho thanh toán: Debit/Credit Card, NetBanking, Cash On Delivery → xếp dọc trong **VBoxLayout** thứ hai.
5. Đặt **objectName** cho các nút và bố cục để dễ tham chiếu trong mã.
6. Thêm **Label hiển thị lựa chọn** và đặt tên là labelSelected.
7. Lưu tệp **demoRadioButton2.ui** → chuyển thành mã Python bằng **pyuic5**.

callRadioButton2.pyw

```
import sys
from PyQt5.QtWidgets import QDialog, QApplication
from demoRadioButton2 import *

class MyForm(QDialog):
  def init(self):
    super().__init__()
      self.ui = Ui_Dialog()
      self.ui.setupUi(self)
      self.ui.radioButtonMedium.toggled.connect(self.dispSelected)
      self.ui.radioButtonLarge.toggled.connect(self.dispSelected)
      self.ui.radioButtonXL.toggled.connect(self.dispSelected)
      self.ui.radioButtonXXL.toggled.connect(self.dispSelected)
      self.ui.radioButtonDebitCard.toggled.connect(self.dispSelected)
      self.ui.radioButtonNetBanking.toggled.connect(self.dispSelected)
      self.ui.radioButtonCashOnDelivery.toggled.connect(self.dispSelected)
      self.show()
```

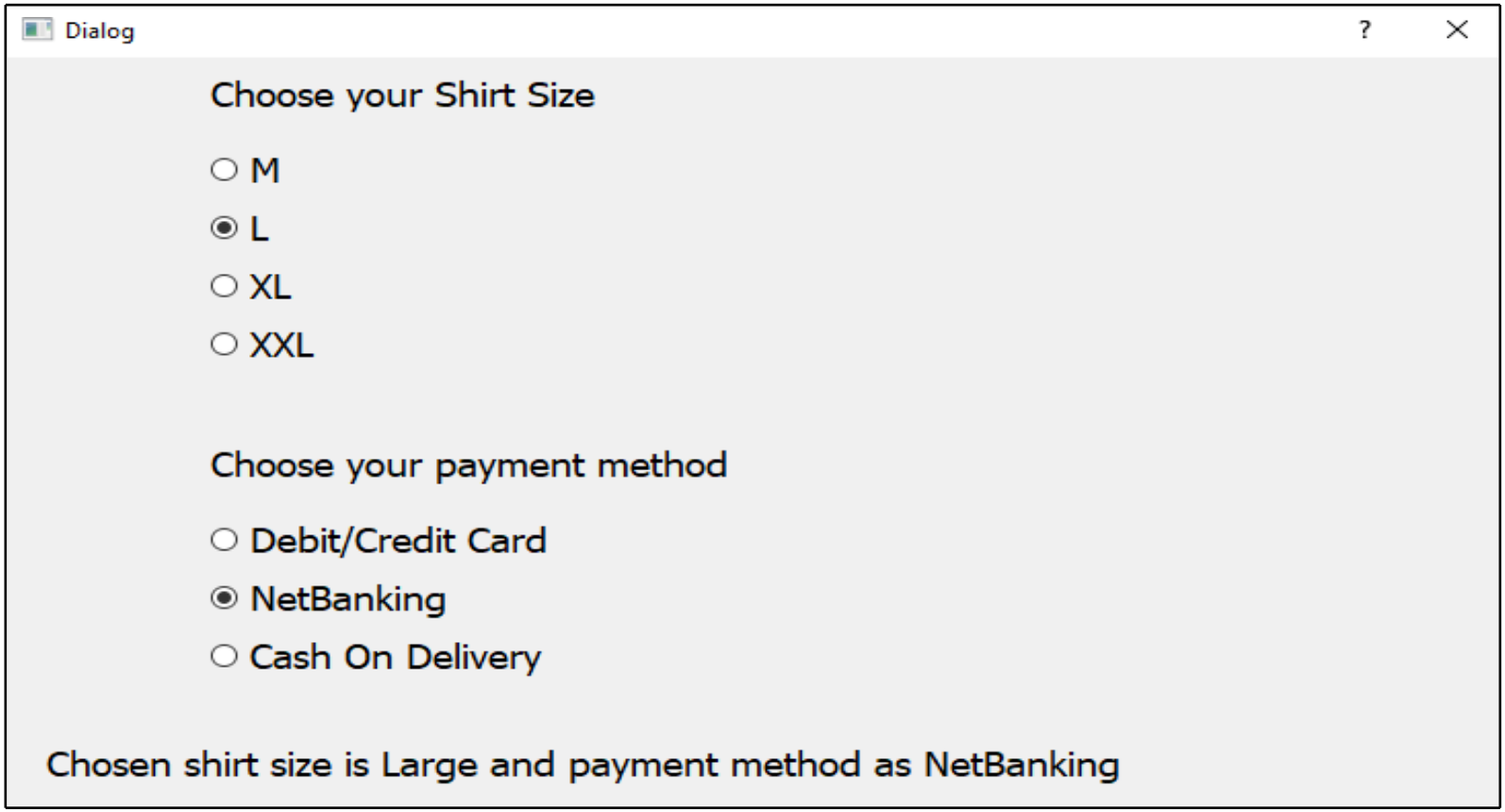
callRadioButton2.pyw (t.t)

```
  def dispSelected(self):
    selected1 = "";
    selected2 = ""
    if self.ui.radioButtonMedium.isChecked() == True:
      selected1 = "Medium"
    if self.ui.radioButtonLarge.isChecked() == True:
      selected1 = "Large"
    if self.ui.radioButtonXL.isChecked() == True:
      selected1 = "Extra Large"
    if self.ui.radioButtonXXL.isChecked() == True:
      selected1 = "Extra Extra Large"
    if self.ui.radioButtonDebitCard.isChecked() == True:
      selected2 = "Debit/Credit Card"
    if self.ui.radioButtonNetBanking.isChecked() == True:
      selected2 = "NetBanking"
    if self.ui.radioButtonCashOnDelivery.isChecked() == True:
      selected2 = "Cash On Delivery"
    self.ui.labelSelected.setText("Chosen shirt size is "+selected1+
                " and payment method as " + selected2)

if __name__ == "__main__":
  app = QApplication(sys.argv)
  w = MyForm()
  w.show()
  sys.exit(app.exec_())
```

Cách chương trình chạy thế nào...

Khi người dùng chọn radio button, hàm dispSelected() hiển thị kích thước áo (Medium, Large, XL, XXL) và phương thức thanh toán (Debit/Credit, NetBanking, COD) lên labelSelected.



Checkbox

Checkbox trong ứng dụng

- Dùng khi muốn người dùng chọn 1 hoặc nhiều tùy chọn.
- Khác radio button: checkbox cho phép chọn nhiều mục cùng lúc.
- Hiển thị với nhãn text qua lớp QCheckBox.
- Có 3 trạng thái:

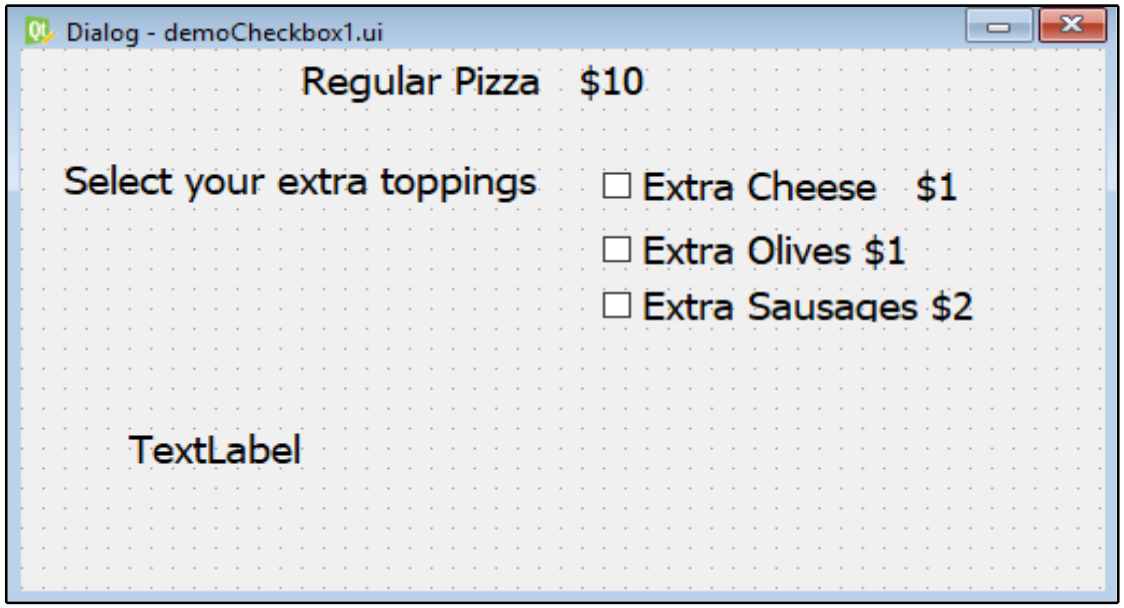
1. Checked (đã chọn)
2. Unchecked (chưa chọn)
3. Tristate (không thay đổi)

Check Box trong PyQt

- **isChecked()**: Kiểm tra hộp có được chọn hay không (True/False).
- **setTristate(True)**: Khóa hộp, người dùng không thể thay đổi trạng thái.
- **setIcon()**: Gắn biểu tượng cho hộp kiểm.
- **setText()**: Gán nhãn văn bản, dùng "&" để tạo phím tắt.
- **setChecked(True)**: Đặt hộp mặc định là đã chọn.
- **clicked()**: Phát tín hiệu khi hộp được nhấn.
- **stateChanged()**: Phát tín hiệu khi trạng thái hộp thay đổi.

Ví dụ minh họa:
Chọn topping cho pizza. Khi người dùng chọn topping, tổng giá pizza cập nhật ngay, giúp dễ quản lý và hiển thị thông tin.

Tạo ứng dụng tính giá pizza với CheckBox



Bắt đầu với ứng dụng Dialog không có nút.

- Thêm Label và 3 CheckBox cho pizza và toppings.
- Đặt tên, text và tăng kích thước phông chữ để dễ nhìn.
- Đổi tên các CheckBox: Cheese, Olives, Sausages.
- Lưu file .ui và chuyển sang Python bằng pyuic5.
- Viết mã Python để cập nhật tổng giá pizza khi người dùng chọn hoặc bỏ chọn toppings.

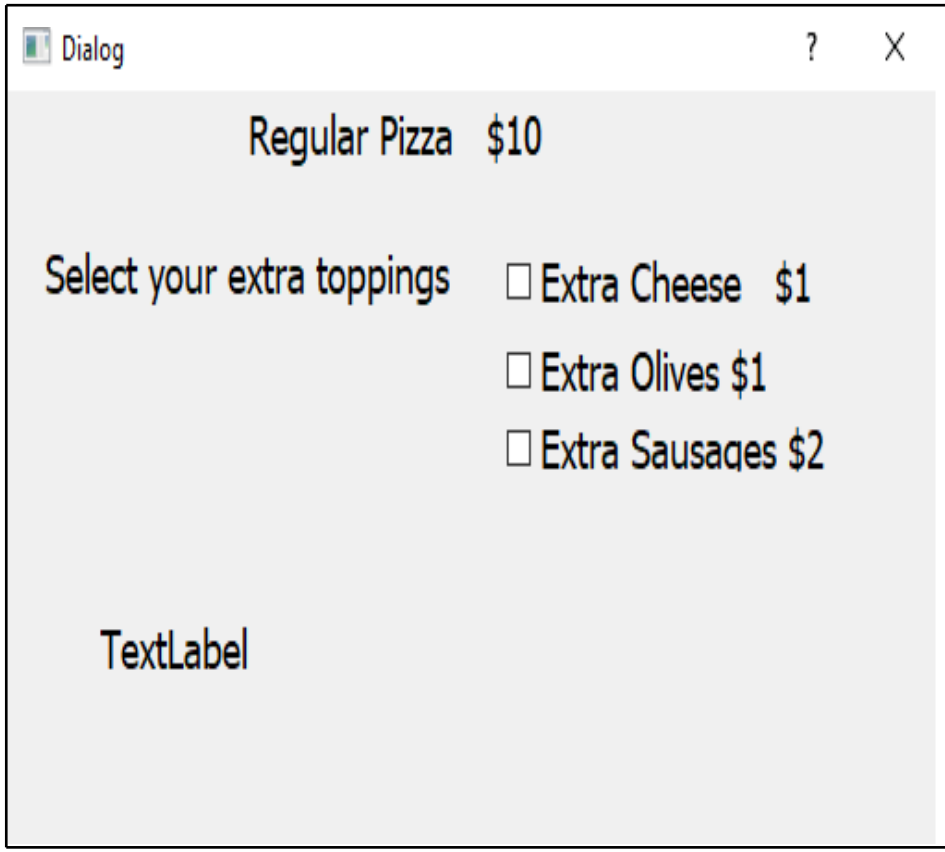
callCheckBox1.pyw

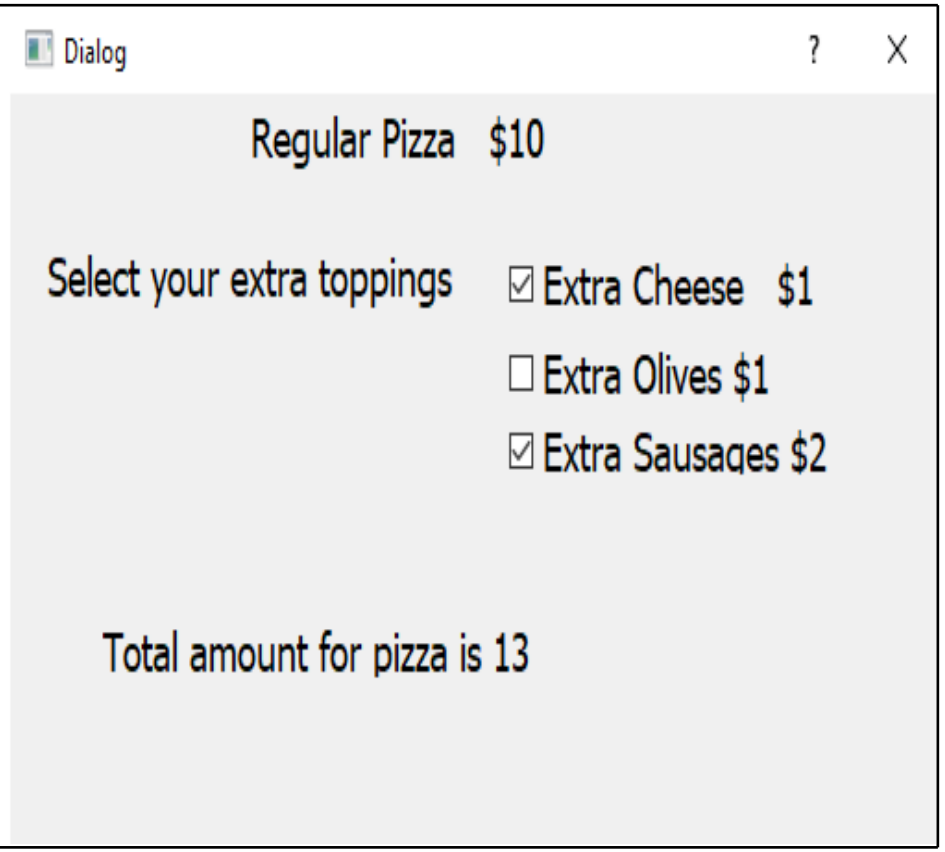
```
import sys
from PyQt5.QtWidgets import QDialog
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton
from demoCheckBox1 import *
class MyForm(QDialog):
  def init(self):
    super().__init__()
    self.ui = Ui_Dialog()
    self.ui.setupUi(self)
    self.ui.checkBoxCheese.stateChanged.connect(self.dispAmount)
    self.ui.checkBoxOlives.stateChanged.connect(self.dispAmount)
    self.ui.checkBoxSausages.stateChanged.connect(self.dispAmount)
    self.show()
  def dispAmount(self):
    amount = 10
    if self.ui.checkBoxCheese.isChecked() == True: amount = amount+1
    if self.ui.checkBoxOlives.isChecked() == True: amount = amount+1
    if self.ui.checkBoxSausages.isChecked() == True: amount = amount+2
    self.ui.labelAmount.setText("Total amount for pizza is " +


                                                        str(amount))
if __name__ == "__main__":
  app = QApplication(sys.argv)
  w = MyForm()
  w.show()
```

Cách chương trình checkbox chạy...

Khi người dùng chọn các hộp kiểm toppings, hàm distAmount() sẽ tính tổng chi phí pizza + toppings và hiển thị giá qua labelAmount. Ứng dụng hiện hộp thoại để chọn toppings muốn thêm.



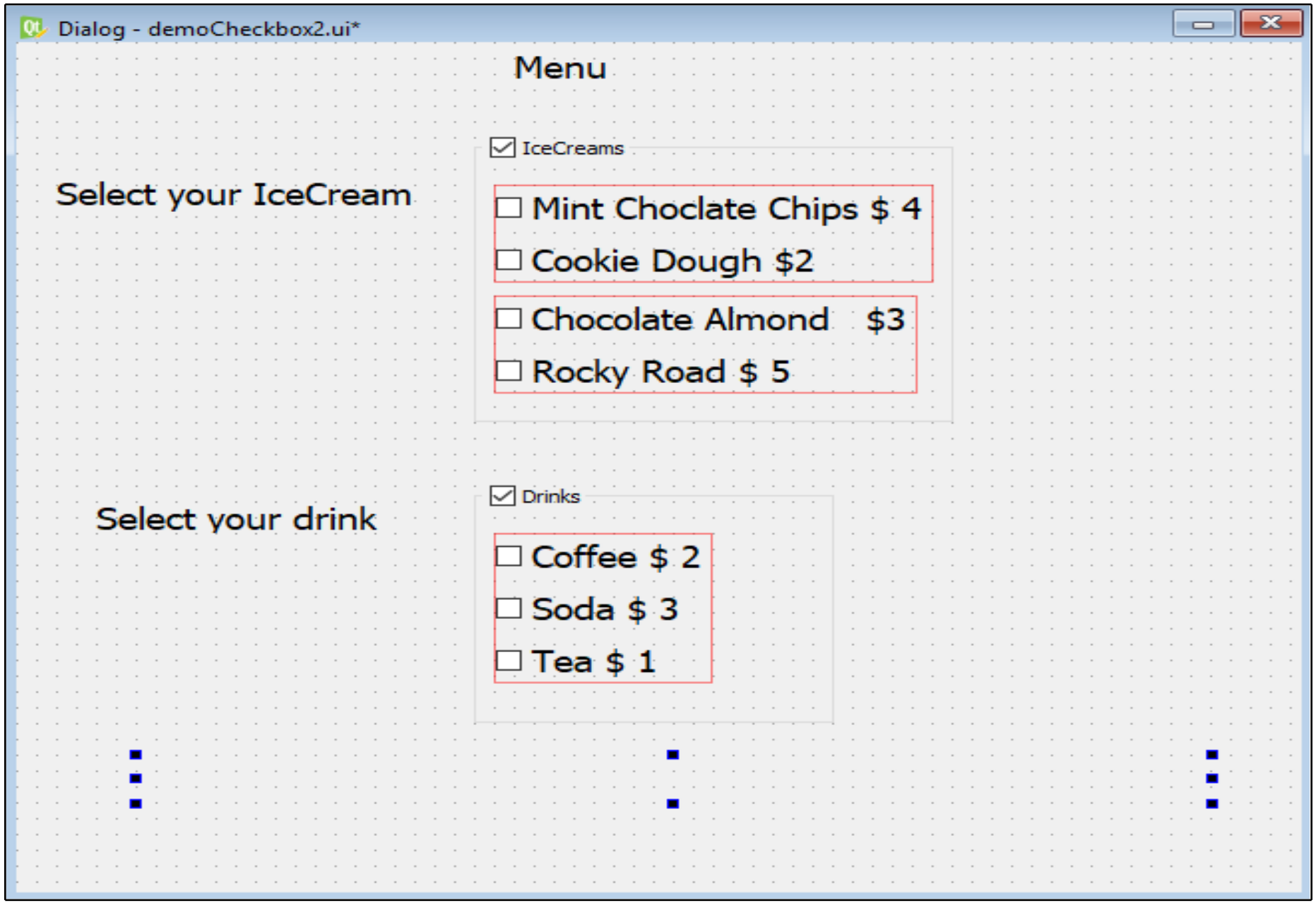


Ứng dụng chọn kem và đồ uống với Check Box

- Ứng dụng hiển thị menu gồm **2 nhóm**:
- **Kem**: 4 loại (Mint Chocolate Chips, Cookie Dough, Chocolate Almond, Rocky Road)
- **Đồ uống**: 3 loại (Coffee, Soda, Tea)
- Mỗi món có **giá riêng**.
- Người dùng có thể **chọn nhiều món** bằng Check Box.
- **Tổng giá** được hiển thị tự động khi chọn/bỏ chọn món.
- Bước thực hiện:

1. Tạo ứng dụng với **Dialog without Buttons**.
2. Thêm **Label, Check Box, Group Box** vào form.
3. Thiết lập text, objectName, font, nhóm các Check Box.
4. Sử dụng Label hiển thị tổng chi phí.
5. Import file .py và viết mã để **tính tổng giá** khi người dùng tương tác.

Ứng dụng chọn kem và đồ uống với Check Box



```
import sys
from PyQt5.QtWidgets import QDialog
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton
from demoCheckBox2 import *

class MyForm(QDialog):
  def init(self):
    super().__init__()
    self.ui = Ui_Dialog()
    self.ui.setupUi(self)
    self.ui.checkBoxChoclateAlmond.stateChanged.connect(self.dispAmount)
    self.ui.checkBoxChoclateChips.stateChanged.connect(self.dispAmount)
    self.ui.checkBoxCookieDough.stateChanged.connect(self.dispAmount)
    self.ui.checkBoxRockyRoad.stateChanged.connect(self.dispAmount)
    self.ui.checkBoxCoffee.stateChanged.connect(self.dispAmount)
    self.ui.checkBoxSoda.stateChanged.connect(self.dispAmount)
    self.ui.checkBoxTea.stateChanged.connect(self.dispAmount)
    self.show()
```

```
def dispAmount(self):
    amount = 0
    if self.ui.checkBoxChoclateAlmond.isChecked() == True: amount = amount+3
    if self.ui.checkBoxChoclateChips.isChecked() == True: amount = amount+4
    if self.ui.checkBoxCookieDough.isChecked() == True: amount = amount+2
    if self.ui.checkBoxRockyRoad.isChecked() == True: amount = amount+5
    if self.ui.checkBoxCoffee.isChecked() == True: amount = amount+2
    if self.ui.checkBoxSoda.isChecked() == True: amount = amount+3
    if self.ui.checkBoxTea.isChecked() == True: amount = amount+1
    self.ui.labelAmount.setText("Total amount is $"+str(amount))


if __name__ =="__main__":
  app=QApplication(sys.argv)
  w=MyForm()
  w.show()
  sys.exit(app.exec_())
```

Ứng dụng chọn kem và đồ uống với Check Box

- Mỗi hộp kiểm kết nối với **hàm distAmount()**.
- Khi chọn món, hàm kiểm tra trạng thái các hộp kiểm.
- Chi phí của các món được chọn được cộng vào **amount**.
- Tổng số tiền hiển thị qua **labelAmount**.
- Khi chạy ứng dụng, người dùng được nhắc chọn kem hoặc đồ uống, và tổng tiền hiện ra ngay khi chọn.

Ứng dụng chọn kem và đồ uống với Check Box