Hiểu các khái niệm OOP

Nội dung

- Lập trình hướng đối tượng
- Sử dụng các lớp trong GUI
- Sử dụng thừa kế đơn
- Sử dụng thừa kế đa cấp
- Sử dụng nhiều kế thừa

Lập Trình Hướng Đối Tượng

- Python hỗ trợ lập trình hướng đối tượng (OOP), giúp **tái sử dụng mã nguồn** khi xây dựng ứng dụng lớn.
  Trong OOP, **lớp (class)** là bản thiết kế gồm **thuộc tính (data members)** và **phương thức (member functions)**, còn **đối tượng (object)** là thể hiện của lớp và có thể sử dụng trực tiếp các thuộc tính, phương thức đó.

Lập Trình Hướng Đối Tượng

- Sau đây là cú pháp để tạo một lớp:

class class_name(base_classes):

statement(s)

Ví dụ:

```
class Student:
    name = ""
    def __init__(self, name):
        self.name = name
        def printName(self):
        return self.name
```

Sử Dụng Các Thuộc Tính Lớp Tích Hợp

- Khi khai báo một lớp, Python tự động gán các **thuộc tính lớp** giúp lấy thông tin về lớp. Các thuộc tính quan trọng gồm:
  - __name__: tên của lớp
  - __base__: các lớp cơ sở (lớp cha)
  - __dict__: chứa các thuộc tính và phương thức của lớp
  - __module__: mô-đun nơi lớp được định nghĩa

Sử Dụng Các Thuộc Tính Lớp Tích Hợp

- Một lớp có thể có nhiều phương thức, mỗi phương thức có nhiều tham số.
- Mỗi phương thức **bắt buộc có tham số đầu tiên** (thường đặt tên là self).
- self đại diện cho **đối tượng (instance)** đang gọi phương thức.
- Phương thức trong lớp được định nghĩa theo cú pháp chuẩn của Python.

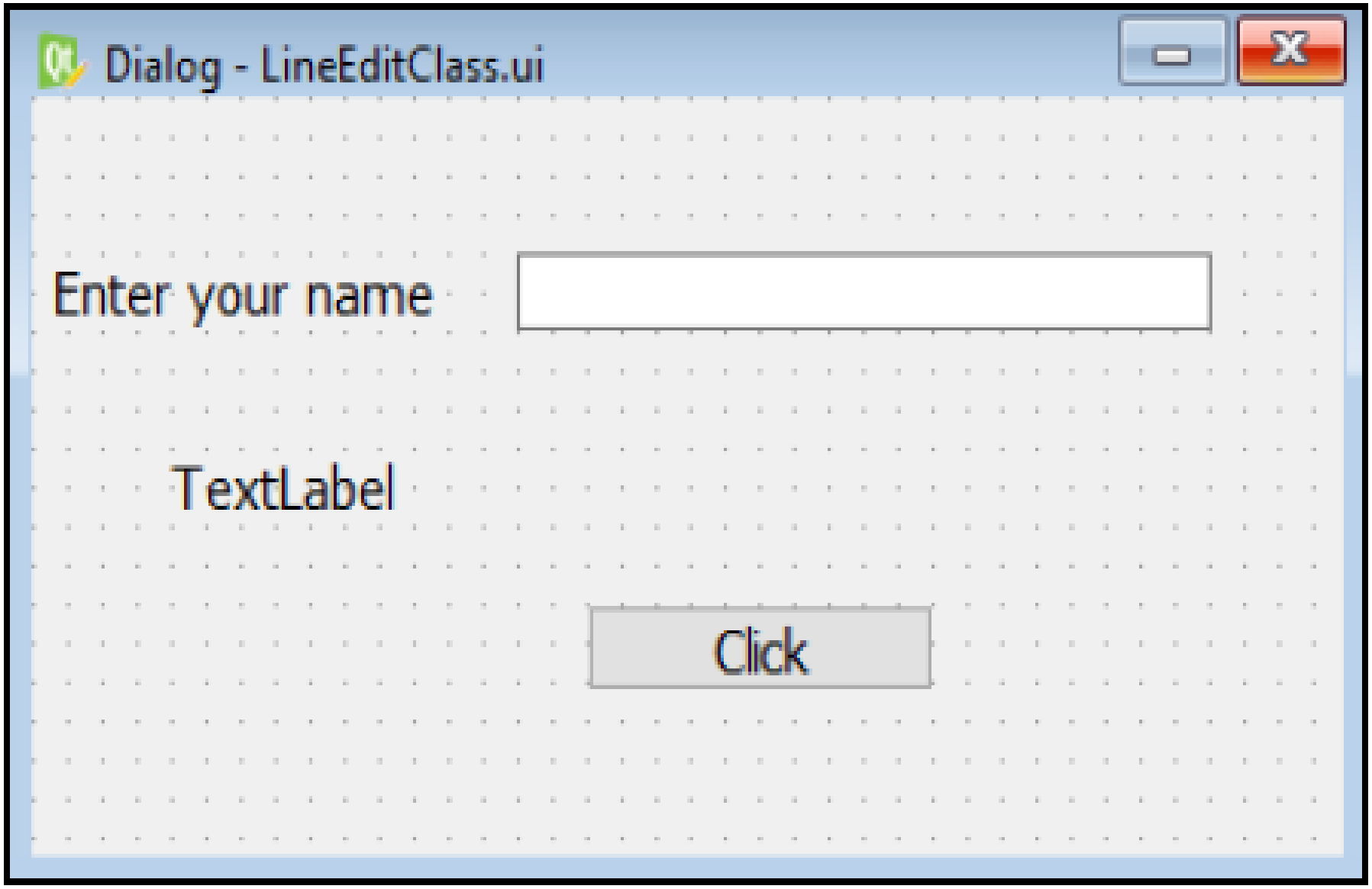
```
class class_name(base_classes):
      Syntax:
            variable(s)
      def method 1(self):
            statement(s)
      [def method n(self):
            statement(s)]
```

Sử Dụng Các Thuộc Tính Lớp Tích Hợp

- **Class variable (biến lớp):** Dùng chung cho tất cả các đối tượng của lớp. Thay đổi ở một đối tượng sẽ ảnh hưởng đến các đối tượng khác.
- **Instance variable (biến thể hiện):** Chỉ thuộc về từng đối tượng riêng lẻ. Thay đổi ở đối tượng nào chỉ ảnh hưởng đến đối tượng đó.

Xử lý dữ liệu GUI bằng Class trong PyQt5

- Dữ liệu nhập từ GUI có thể xử lý bằng biến thường, nhưng để có cấu trúc và tận dụng OOP, dữ liệu nên được lưu trong **lớp (class)**.
- Dữ liệu người dùng nhập sẽ được gán vào **biến lớp**, xử lý và hiển thị thông qua **phương thức lớp**.
- Ví dụ: tạo ứng dụng yêu cầu nhập tên; khi nhấn nút, chương trình hiển thị lời chào kèm tên đã nhập.
- Giao diện được thiết kế bằng **Qt Designer**, lưu dưới dạng file **.ui (XML)**.
- File .ui cần được chuyển sang **mã Python** bằng công cụ pyuic5 để sử dụng trong chương trình.
- Sau đó, nhập file giao diện đã chuyển đổi vào file Python khác để xử lý logic ứng dụng.


Dialog - LineEditClass.ui
Enter your name
TextLabel
Click

Sử Dụng Các Lớp Trong GUI

```
import sys
from PyQt5.QtWidgets import QDialog, QApplication
from LineEditClass import *
class Student:
    name = ""
    def __init__(self, name):
        self.name = name
    def printName(self):
```

Sử Dụng Các Lớp Trong GUI

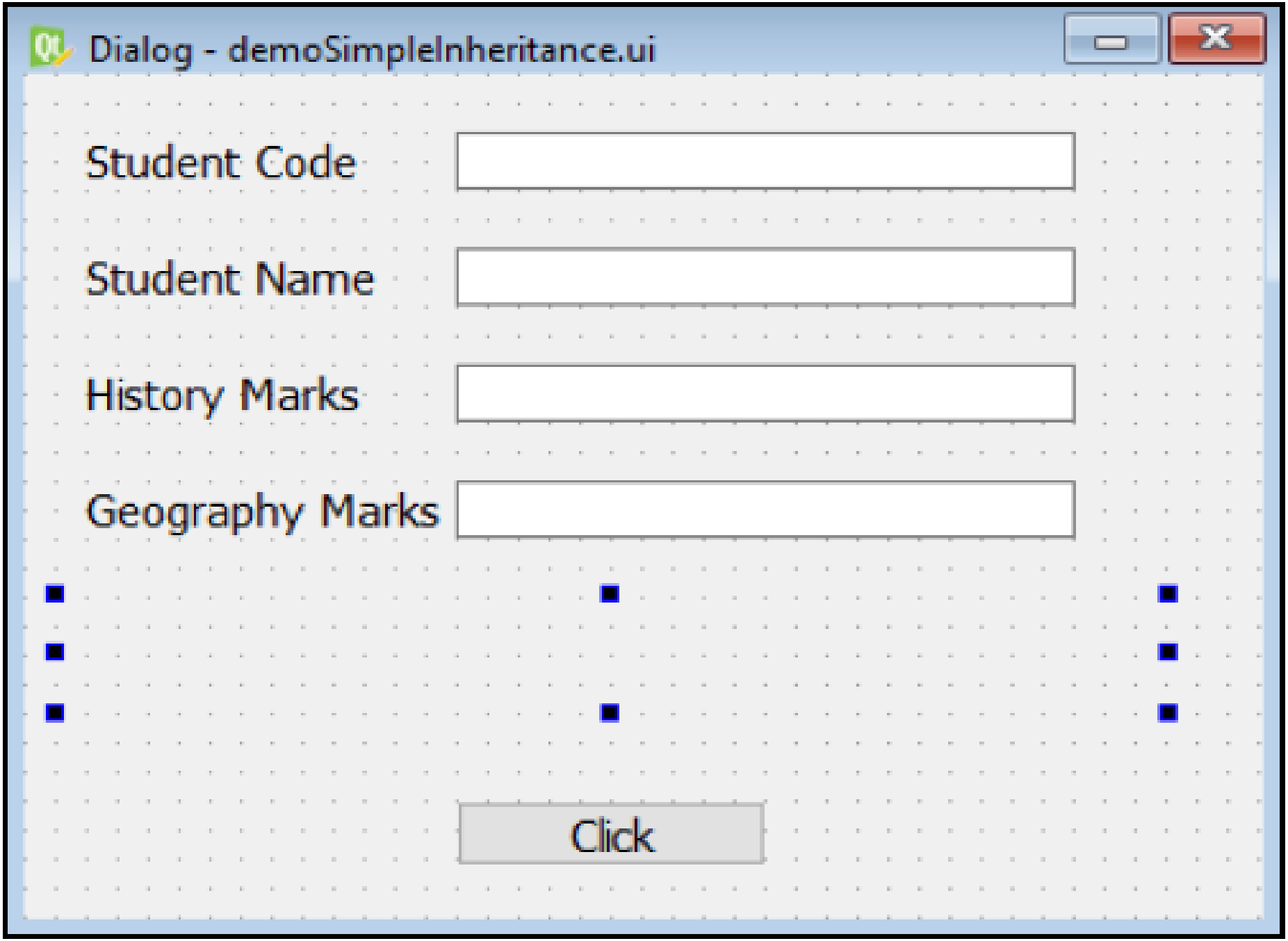
```
        return self.name
class MyForm(QDialog):
    def __init__(self):
        super().__init__()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.ButtonClickMe.clicked.connect(self.dispmessage)
        self.show()
    def dispmessage(self):
        studentObj=Student(self.ui.lineEditName.text())
        self.ui.labelResponse.setText("Hello
        "+studentObj.printName())
if __name__=="__main__":
    app = QApplication(sys.argv)
    w = MyForm()
    w.show()
    sys.exit(app.exec_())
```

Sử Dụng Thừa Kế Đơn

- **Kế thừa đơn** là kiểu kế thừa đơn giản nhất: **một lớp con kế thừa từ một lớp cha**.
- Lớp cha gọi là **siêu lớp (lớp cơ sở)**, lớp con gọi là **lớp dẫn xuất (lớp con)**.
- Ví dụ: class Marks(Student) → lớp **Marks** kế thừa lớp **Student**.
- Nhờ kế thừa, lớp **Marks** có thể **sử dụng lại thuộc tính** code, name của lớp **Student**, đồng thời **thêm dữ liệu riêng** như điểm history, geography.
- Đối tượng của lớp **Marks** có thể **truy cập và hiển thị dữ liệu của cả hai lớp**.

Sử Dụng Thừa Kế Đơn

1. Tạo giao diện mới bằng **Qt Designer** với mẫu *Dialog without Buttons*.
2. Thêm **5 Label, 4 Line Edit và 1 Push Button** vào form.
3. Đặt nội dung Label: *Student Code, Student Name, History Marks, Geography Marks*; một Label để trống (sẽ gán bằng code).
4. Đặt text cho nút bấm là **Click** và gán **objectName** cho các Line Edit, Label, Button theo yêu cầu.
5. Lưu giao diện với tên **demoSimpleInherribution.ui** (file XML) và chuyển sang Python bằng **pyuic5**.
6. Tạo file **callSimpleInherribution.pyw** để import và sử dụng giao diện đã thiết kế.

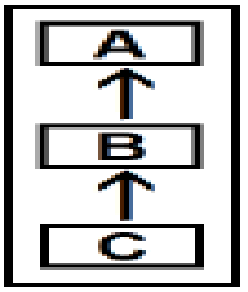

Dialog - demoSimpleInheritance.ui
Student Code
Student Name
History Marks
Geography Marks
Click

```
import sys
from PyQt5.QtWidgets import QDialog, QApplication
from demoSimpleInheritance import *
class Student:
    name = ""
    code = ""
    def __init__(self, code, name):
        self.code = code
        self.name = name
    def getCode(self):
        return self.code
    def getName(self):
        return self.name
class Marks(Student):
    historyMarks = 0
    geographyMarks = 0
    def __init__(self, code, name, historyMarks,
    geographyMarks):
        Student.__init__(self,code,name)
        self.historyMarks = historyMarks
        self.geographyMarks = geographyMarks
    def getHistoryMarks(self):
        return self.historyMarks
    def getGeographyMarks(self):
        return self.geographyMarks
```

```
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):
        marksObj=Marks(self.ui.lineEditCode.text(),
        self.ui.lineEditName.text(),

        self.ui.lineEditHistoryMarks.text(),
        self.ui.lineEditGeographyMarks.text())
        self.ui.labelResponse.setText("Code:
        "+marksObj.getCode()+", Name:"+marksObj.getName()+"
        nHistory Marks:"+marksObj.getHistoryMarks()+", Geography
        Marks:"+marksObj.getGeographyMarks())
if __name__=="__main__":
    app = QApplication(sys.argv)
    w = MyForm()
    w.show()
    sys.exit(app.exec_())
```

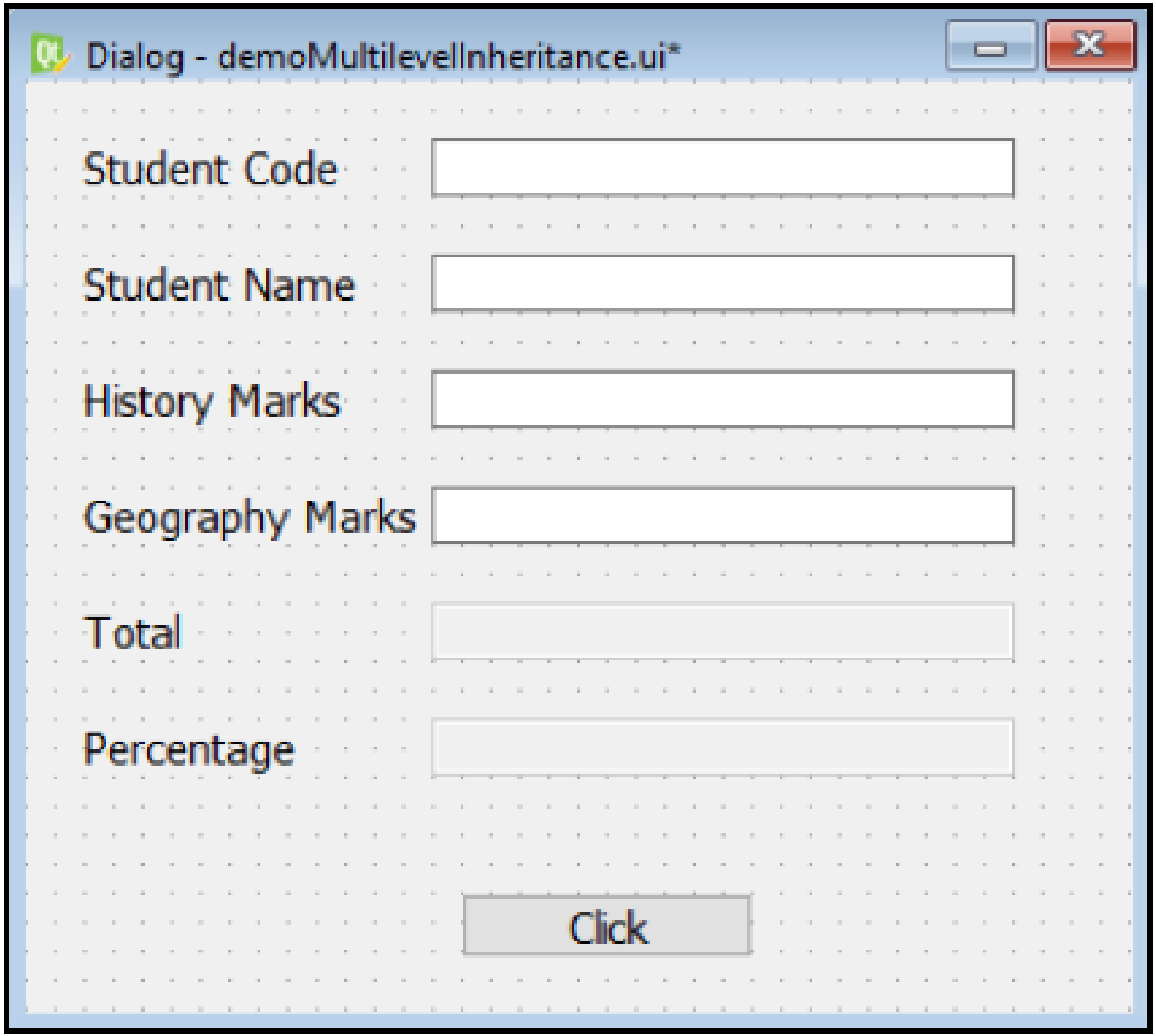
Sử Dụng Thừa Kế Đa Cấp



- **Kế thừa đa cấp** là khi một lớp kế thừa từ lớp khác, và lớp đó lại tiếp tục được kế thừa bởi lớp thứ ba (A → B → C).
- Ví dụ gồm **3 lớp**:
  - Student: lưu **code** và **name**
  - Marks: kế thừa Student, lưu **history** và **geography**
  - Result: kế thừa Marks, tính **tổng điểm** và **phần trăm**
- Nhờ kế thừa đa cấp, lớp Result có thể truy cập dữ liệu của cả Marks và Student.
- **Tổng điểm** = history + geography
- **Phần trăm** = (tổng điểm / 200) × 100

Cài đặt ứng dụng Thừa Kế Đa Cấp

1. Tạo giao diện bằng **Qt Designer** với mẫu *Dialog without Buttons*.
2. Thêm **6 Label**, **6 Line Edit** và **1 Push Button**; đặt nội dung và objectName theo yêu cầu.
3. Nút bấm có tên **Click**, objectName: ButtonClickMe.
4. **Vô hiệu hóa** ô *Total* và *Percentage* vì giá trị được gán bằng code.
5. Lưu giao diện thành **demoMultilevelInherribution.ui**.
6. Chuyển file .ui sang Python bằng **pyuic5** → **demoMultilevelInherribution.py**.
7. Tạo file **callMultilevelInherribution.pyw** để import và sử dụng giao diện đã tạo.

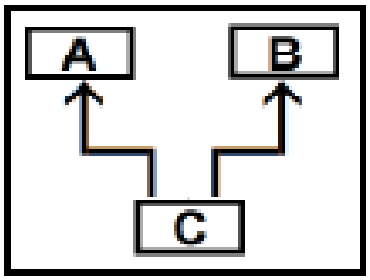

Dialog - demoMultilevelInheritance.ui*
Student Code
Student Name
History Marks
Geography Marks
Total
Percentage
Click

```
import sys
from PyQt5.QtWidgets import QDialog, QApplication
from demoMultilevelInheritance import *
class Student:
    name = ""
    code = ""
    def __init__(self, code, name):
        self.code = code
        self.name = name
    def getCode(self):
        return self.code
    def getName(self):
```

```
        return self.name
class Marks(Student):
    historyMarks = 0
    geographyMarks = 0
    def __init__(self, code, name, historyMarks,
    geographyMarks):
        Student.__init__(self,code,name)
        self.historyMarks = historyMarks
        self.geographyMarks = geographyMarks
    def getHistoryMarks(self):
        return self.historyMarks
    def getGeographyMarks(self):
        return self.geographyMarks
class Result(Marks):
    totalMarks = 0
    percentage = 0
    def __init__(self, code, name, historyMarks,
    geographyMarks):
        Marks.__init__(self, code, name, historyMarks,
        geographyMarks)
        self.totalMarks = historyMarks + geographyMarks
        self.percentage = (historyMarks +
        geographyMarks) / 200 * 100
    def getTotalMarks(self):
        return self.totalMarks
    def getPercentage(self):
        return self.percentage
```

```
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):
        resultObj=Result(self.ui.lineEditCode.text(),
        self.ui.lineEditName.text(),
        int(self.ui.lineEditHistoryMarks.text()),
        int(self.ui.lineEditGeographyMarks.text()))
        self.ui.lineEditTotal.setText(str(resultObj.
        getTotalMarks()))
        self.ui.lineEditPercentage.setText(str(resultObj.
        getPercentage()))
if __name__=="__main__":
    app = QApplication(sys.argv)
    w = MyForm()
    w.show()
    sys.exit(app.exec_())
```

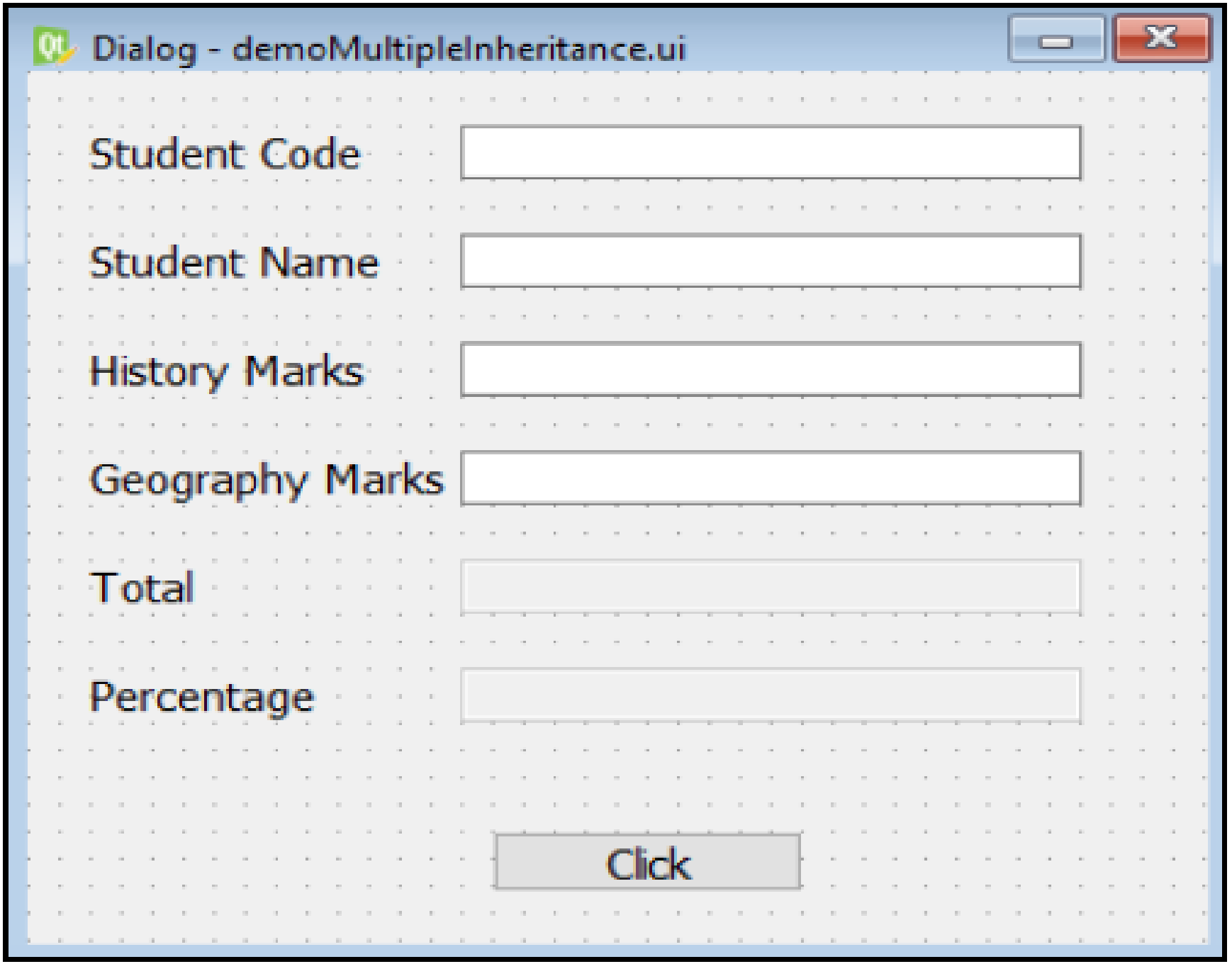
Sử Dụng Nhiều Kế Thừa



- **Đa kế thừa (Multiple Inheritance)**: một lớp có thể kế thừa **từ hai hoặc nhiều lớp** cùng lúc.
- Ví dụ: **Result** kế thừa **Student** (code, name) và **Marks** (history, geography).
- Nhờ đa kế thừa, **Result** truy cập được dữ liệu của cả hai lớp cha.
- **TotalMarks** = history + geography.
- **Percentage** = (TotalMarks / 200) × 100.
- Ứng dụng minh họa: nhập thông tin sinh viên → tính tổng điểm và phần trăm.

Cài đặt ứng dụng đa Kế Thừa

- Tạo giao diện bằng **Qt Designer** với mẫu *Dialog without Buttons*.
- Thêm **6 Label, 6 Line Edit và 1 Push Button** vào form.
- Đặt nội dung Label (mã SV, tên SV, điểm, tổng, phần trăm) và nút **Click**.
- Gán **objectName** cho các Line Edit và nút để dùng trong code.
- **Vô hiệu hóa** ô *Total* và *Percentage* vì giá trị được tính bằng code.
- Lưu file giao diện **.ui** → chuyển sang Python bằng **pyuic5**.
- Import file giao diện vào file Python khác để **chạy ứng dụng GUI**.


Dialog - demoMultipleInheritance.ui
Student Code
Student Name
History Marks
Geography Marks
Total
Percentage
Click

```
import sys
from PyQt5.QtWidgets import QDialog, QApplication
from demoMultipleInheritance import *
class Student:
    name = ""
    code = ""
    def __init__(self, code, name):
        self.code = code
        self.name = name
    def getCode(self):
        return self.code
    def getName(self):
        return self.name
class Marks:
    historyMarks = 0
    geographyMarks = 0
    def __init__(self, historyMarks, geographyMarks):
        self.historyMarks = historyMarks
        self.geographyMarks = geographyMarks
    def getHistoryMarks(self):
        return self.historyMarks
    def getGeographyMarks(self):
        return self.geographyMarks
```

```
class Result(Student, Marks):
    totalMarks = 0
    percentage = 0
    def __init__(self, code, name, historyMarks,
    geographyMarks):
        Student.__init__(self, code, name)
        Marks.__init__(self, historyMarks, geographyMarks)
        self.totalMarks = historyMarks + geographyMarks
        self.percentage = (historyMarks +
        geographyMarks) / 200 * 100
    def getTotalMarks(self):
        return self.totalMarks
    def getPercentage(self):
        return self.percentage
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):
```

```
        resultObj=Result(self.ui.lineEditCode.text(),
        self.ui.lineEditName.text(),
        int(self.ui.lineEditHistoryMarks.text()),
        int(self.ui.lineEditGeographyMarks.text()))
        self.ui.lineEditTotal.setText(str(resultObj.
        getTotalMarks()))
        self.ui.lineEditPercentage.setText(str(resultObj.
        getPercentage()))
if __name__=="__main__":
    app = QApplication(sys.argv)
    w = MyForm()
    w.show()
    sys.exit(app.exec_())
```