LỚP & ĐỐI TƯỢNG

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

GVHD: Trương Toàn Thịnh

NỘI DUNG

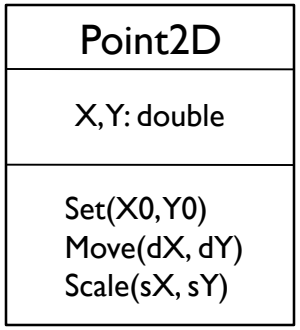
- Khái niệm
- Chu kì sống đối tượng
- Bao bọc dữ liệu
- Tạo lập & hủy đối tượng
- Sơ lược nhập xuất
- Tổ chức mã nguồn
- Một số kí hiệu UML

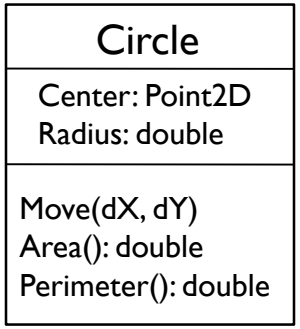
KHÁI NIỆM

- Lớp: dùng để mô hình hóa một nhóm các thực thể cùng loại trong thế giới thực
- Đối tượng: dùng để chỉ một thực thể cụ thể của một lớp nào đó
- Các đối tượng trong cùng một lớp sẽ có cùng một hành vi
- Ví dụ:
  - Giáo viên và học sinh là 2 lớp
  - Trong một lớp học có 2 giáo viên và 20 học sinh (2 đối tượng giáo viên & 20 đối tượng học sinh)
  - Trong lớp học xuất hiện một con mèo, lúc này có thể xem có hai lớp Người & Thú (Vậy có tổng cộng 23 đối tượng)

KHÁI NIỆM

- Lớp (Class):
  - Có một tên lớp: tên duy nhất giúp phân biệt với các lớp khác trong cùng một phạm vi
  - Có các thành phần dữ liệu (data members) chứa dữ liệu mô tả đối tượng
  - Có hệ thống các phương thức (methods) mô tả hành vi của đối tượng





KHÁI NIỆM

- Lớp (Class): đoạn mã **khai báo** lớp:
  - Từ khóa public: các phương thức có thể được truy xuất từ bên ngoài

```
//File Figures.h
#ifndef _FIGURE_H_
#define _FIGURE_H_
class Point2D{
  double X, Y;
  public:
    void Set(double, double);
    void Move(double, double);
    void Scale(double, double);
}; //Ket thuc khai bao
```

```
class Circle{
  Point2D Center;
  double Radius;
  public:
    void Set(double, double, double);
    void Move(double, double);
    double Area();
    double Perimeter();
}; //Ket thuc khai bao
#endif
```

KHÁI NIỆM

- Lớp (Class): đoạn mã **cài đặt** lớp

```
//File Figures.cpp
#include "Figures.h"
void Point2D::Set(double X0, double Y0)
{
  X = X0; Y = Y0;
}
void Point2D::Move(double dX, double dY)
{
  X+=dX; Y+=dY;
}
void Point2D::Scale(double sX, double sY)
{
  X*=sX; Y*=sY;
}
```

```
const double PI = 3.14159;
void Circle::Set(double X0, double Y0, double r)
{
  Center.Set(X0, Y0);
  if(r < 0) r = 0;
  this->Radius = r;
}
void Circle::Move(double dX, double dY){
  Center.Move(dX, dY);
}
double Circle::Area(){
  return PI*Radius*Radius;
}
double Circle::Perimeter(){
  return 2*PI*Radius;
}
```

NỘI DUNG

- Khái niệm
- Chu kì sống đối tượng
- Bao bọc dữ liệu
- Tạo lập & hủy đối tượng
- Sơ lược nhập xuất
- Tổ chức mã nguồn
- Một số kí hiệu UML

CHU KÌ SỐNG ĐỐI TƯỢNG

- Mỗi đối tượng là một thể hiện cụ thể của lớp
  - Bắt đầu tồn tại khi vừa tạo lập
  - Nhận thông điệp để xử lý trong quá trình sống
  - Bị hủy khi không còn cần thiết

```
//File main.cpp
#include "Figures.h"
void main(){
  Point2D P;
  P.Set(1, 2);
  P.Move(3, 4);
  Circle Cir;
}
```

P được khai báo & tạo lập

P nhận thông điệp & xử lý

P bị hủy & Cir được khai báo và tạo lập

BAO BỌC DỮ LIỆU

- Xét hàm main sau:

```
//File main.cpp
#include "Figures.h"
void main(){
  Circle Cir;
  Cir.Set(0, 10, 20);
  Cir.Radius = -5
}
```

  - Câu lệnh Cir.Set là hợp lệ vì đây là một phương thức public (được phép truy xuất từ bên ngoài), vậy hàm main có thể truy xuất phương thức "Set(double, double, double)".
  - Câu lệnh 'Cir.Radius = -5' bất hợp lệ do là trường dữ liệu private (không được phép truy xuất từ bên ngoài), vậy hàm main không thể gán dữ liệu trực tiếp cho trường Radius của đối tượng Cir.

BAO BỌC DỮ LIỆU

- Xét hàm main sau:
  - Cần bổ sung thêm một phương thức "Set(double)" trong khai báo lớp Circle.

```
void Circle::Set(double r){
  if(r >= 0) Radius = r;
}
```

  - Vậy hàm main sau hoàn toàn hợp lệ

```
//File main.cpp
#include "Figures.h"
void main(){
  Circle Cir;
  Cir.Set(0, 10, 20);
  Cir.Set(-5);
  Cir.Set(25)
}
```

NỘI DUNG

- Khái niệm
- Chu kì sống đối tượng
- Bao bọc dữ liệu
- Tạo lập & hủy đối tượng
- Sơ lược nhập xuất
- Tổ chức mã nguồn
- Một số kí hiệu UML

TẠO LẬP & HỦY ĐỐI TƯỢNG

- Toán tử new:
  - Dùng toán tử này để tạo lập đối tượng

```
Circle* mycir = new Circle();
if(mycir != NULL){
  mycir->Set(20, 20, 100);
  double S = mycir->Area();
}
```

- Toán tử delete
  - Dùng toán tử này để hủy đối tượng đã tạo

```
//…
if(mycir != NULL){
  delete mycir;
  mycir = NULL;
}
```

SƠ LƯỢC NHẬP XUẤT

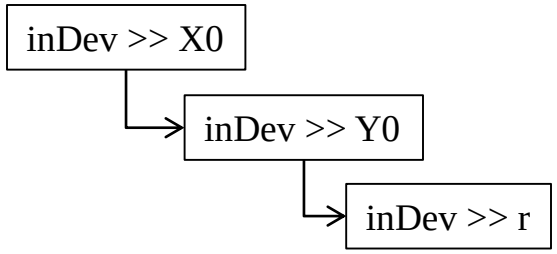
- Đối tượng nhập/xuất trong C++
  - cin: là một đối tượng dựng sẵn kiểu istream
  - cout: là một đối tượng dựng sẵn kiểu ostream
- Cách dùng
  - Cần thêm 2 dòng lệnh sau:
    - #include <iostream>
    - using namespace std;

SƠ LƯỢC NHẬP XUẤT

- Sử dụng cin & cout xây dựng hàm nhập/xuất cho lớp Circle: Do nhập/xuất là 2 thao tác cơ bản cho mọi lớp nên ta đặt 2 phương thức nhập/xuất trong tập tin chứa hàm main.

```
#include "Figures.h"
#include <iostream>
using namespace std;
void inputCircleData(istream& inDev, Circle& cir){
  double X0, Y0, r;
  inDev >> X0 >> Y0 >> r;
  cir.Set(X0, Y0, r);
}
void outputCircleData(ostream& outDev, const Circle& cir){
  outDev << " Area: " << cir.Area() << endl;
  outDev << " Perimeter: " << cir.Perimeter() << endl;
}
```

```
void main(){
  Circle mycir;
  cout << "Input center and radius: ";
  inputCircleData(cin, mycir);
  outputCirleData(cout, mycir);
}
```



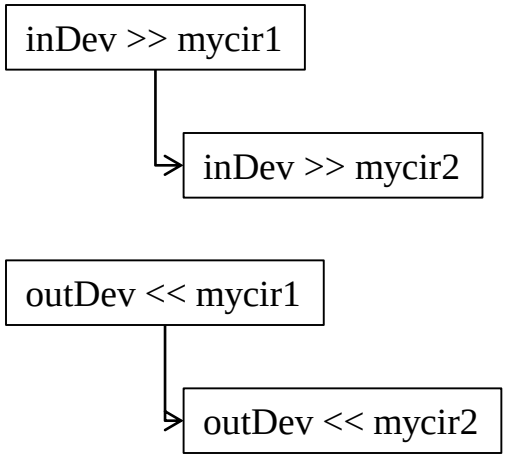
SƠ LƯỢC NHẬP XUẤT

- Mở rộng cin & cout cho lớp Circle

  cin >> mycir; cout << mycir;

```
#include "Figures.h"
#include <iostream>
using namespace std;
istream& operator>>(istream& inDev, Circle& cir){
  double X0, Y0, r;
  inDev >> X0 >> Y0 >> r;
  cir.Set(X0, Y0, r);
  return inDev;
}
ostream& operator<<(ostream& outDev, const Circle& cir){
  outDev << " Area: " << cir.Area() << endl;
  outDev << " Perimeter: " << cir.Perimeter() << endl;
  return outDev;
}
```

```
void main(){
  Circle mycir1, mycir2;
  cout << "Input center and radius: ";
  cin >> mycir1 >> mycir2;
  cout << mycir1 << mycir2;
}
```



NỘI DUNG

- Khái niệm
- Chu kì sống đối tượng
- Bao bọc dữ liệu
- Tạo lập & hủy đối tượng
- Sơ lược nhập xuất
- Tổ chức mã nguồn
- Một số kí hiệu UML

TỔ CHỨC MÃ NGUỒN

- Chương trình C++ gồm:
  - MỘT hàm main()
  - Vài tập tin .h
  - Vài tập tin .cpp tương ứng
- Với từng cặp .h/.cpp cho ra tập tin .o (hay .obj)
- Cuối cùng trình Linker sẽ liên kết tất cả các tập tin .o (hay .obj) cho ra MỘT tập tin thực thi duy nhất (ví dụ .exe)

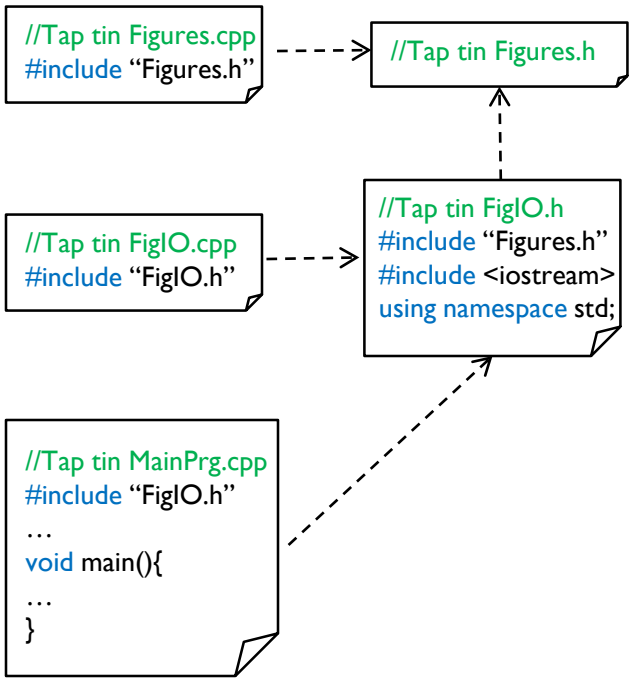
TỔ CHỨC MÃ NGUỒN

- Ví dụ: tạo thêm FigIO.h/FigIO.cpp

```
#ifndef _FIGURES_IO_H_
#define _FIGURES_IO_H_

#include “Figures.h”
#include <iostream>
using namespace std;
istream& operator>>(istream& inDev, Circle& cir);
ostream& operator<<(ostream& outDev, const Circle& cir);
#endif
```

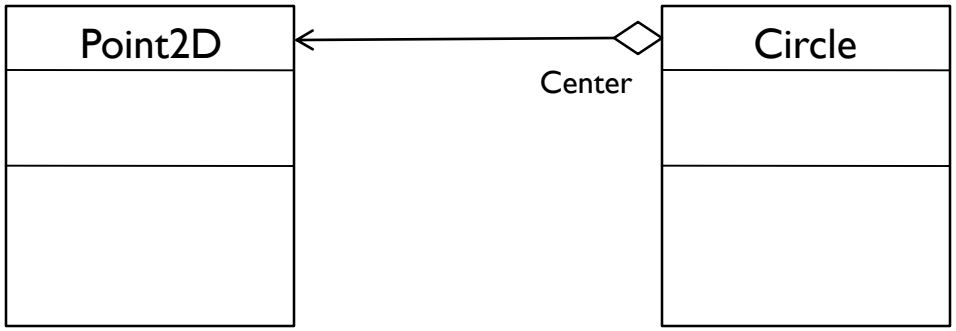
```
#include “FigIO.h”
istream& operator>>(istream& inDev, Circle& cir){
  double X0, Y0, r;
  inDev >> X0 >> Y0 >> r;
  cir.Set(X0, Y0, r);
  return inDev;
}
ostream& operator<<(ostream& outDev, const Circle& cir){
  outDev << “ Area: ” << cir.Area() << endl;
  outDev << “ Perimeter: ” << cir.Perimeter() << endl;
  return outDev;
}
```



MỘT SỐ KÍ HIỆU UML

- Kí hiệu lớp đối tượng:

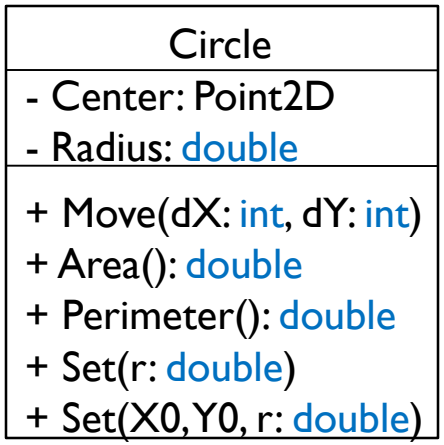
- Kí hiệu mối liên hệ giữa hai lớp



MỘT SỐ KÍ HIỆU UML

- Kí hiệu đối tượng:

- Kí hiệu thuộc tính & phương thức



BÀI TẬP

- Bài tập 1
  - Xây dựng lớp phân số cho phép thực hiện:
    - Nhập, xuất
    - Lấy tử số, lấy mẫu số
    - Gán giá trị cho tử số, mẫu số
    - Nghịch đảo, rút gọn
    - Cộng trừ nhân chia với phân số khác

BÀI TẬP

- Bài tập 1
  - Gợi ý

```
#ifndef _PHANSO_H_
#define _PHANSO_H_

#include <iostream>
using namespace std;
class PhanSo{
  private:
    int tu;
    int mau;
  public:
    void nhap(int, int);
    PhanSo operator+(const PhanSo&);
    …
    friend ostream& operator<<(ostream&, const PhanSo&);
};
#endif
```

```
#include "PhanSo.h"

void PhanSo::nhap(PhanSo& ps){
   …
}
PhanSo PhanSo::operator+(const PhanSo& ps){
   PhanSo temp;
   …
   return temp;
}
ostream& operator<<(ostream& os, const PhanSo& ps)
   os << ps.tu < < "/" << ps.mau;
   return os;
}
…
```