Chương 1a:
Ôn tập về hướng đối tượng

TS. Nguyễn Sơn Hoàng Quốc

Phân tích và thiết kế phần mềm

Tham khảo

“Mastering Object-Oriented Analysis and Design with UML 2.0”

IBM Software Group

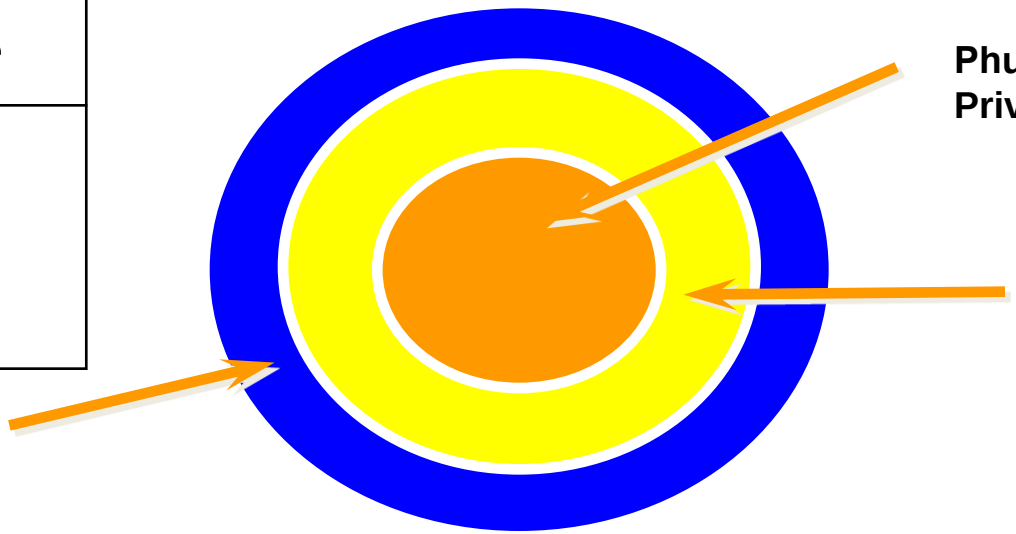
Nhắc lại về hướng đối tượng

Một số ký hiệu

Public/Protected/Private

+	Thuộc tính/Phương thức public
#	Thuộc tính/Phương thức protected
-	Thuộc tính/Phương thức private

Class
- privateAttribute # protectedAttribute
+publicOp() # protectedOp() - privateOp()

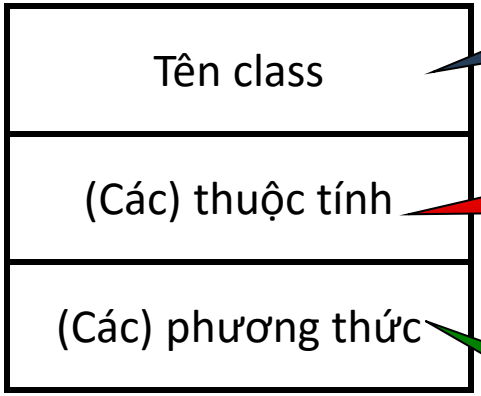


Static (tầm vực)

- Xác định số lượng thể hiện của thuộc tính / phương thức
- Sử dụng chung cho tất cả các đối tượng

Student
- name - address - studentID - nextAvailID : int
+ addSchedule(theSchedule : Schedule, forSemester : Semester) + getSchedule(forSemester : Semester) : Schedule + hasPrerequisites(forCourseOffering : CourseOffering) : boolean # passed(theCourseOffering : CourseOffering) : boolean + getNextAvailID() : int

Quy ước

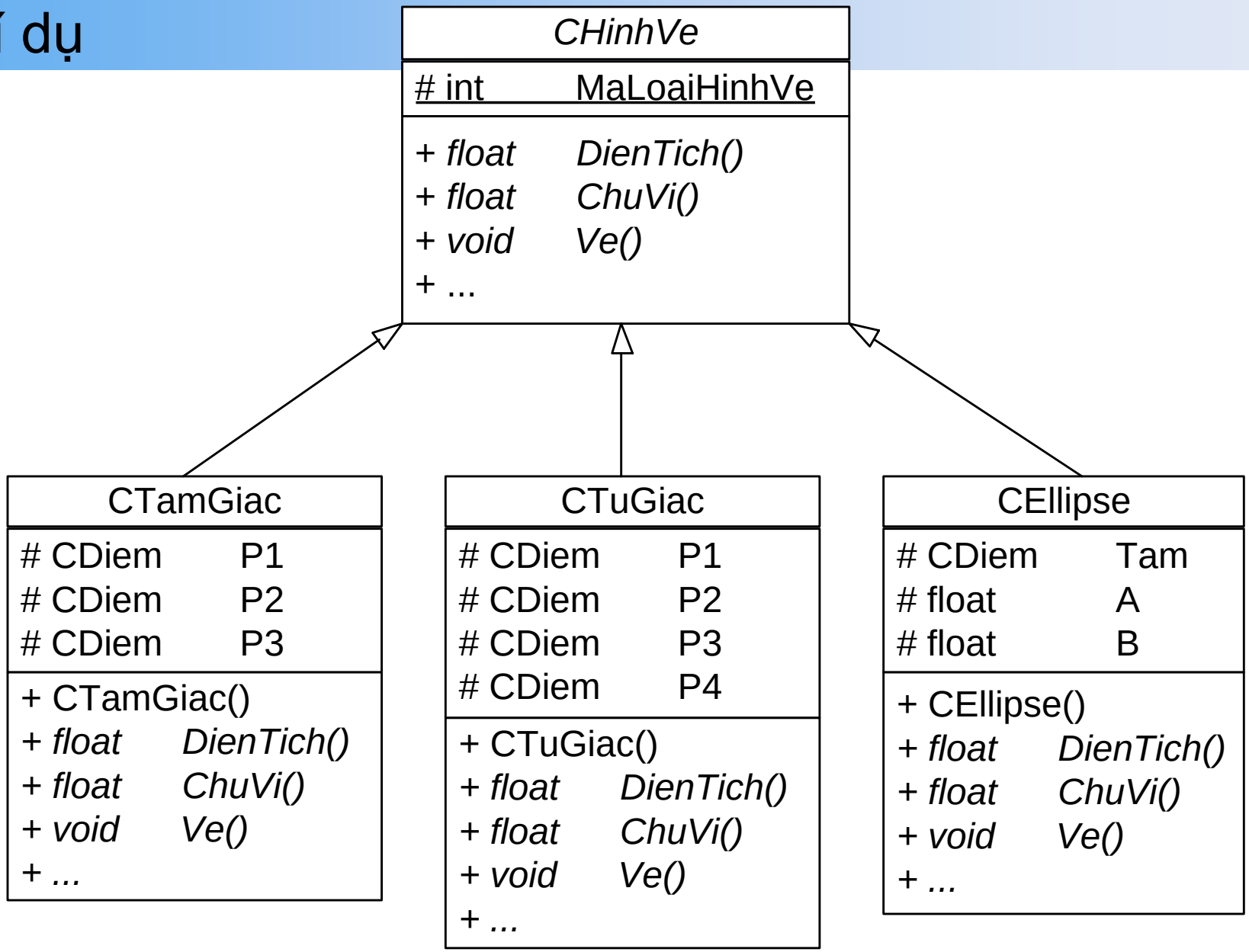


Bình thường: Class bình thường
In nghiêng: Class thuần ảo/Interface
Gạch dưới: Object (không phải class)

Bình thường: Thuộc tính bình thường
In nghiêng: không sử dụng
Gạch dưới: Thuộc tính static

Bình thường: Phương thức bình thường
In nghiêng: Phương thức virtual
Gạch dưới: Phương thức static

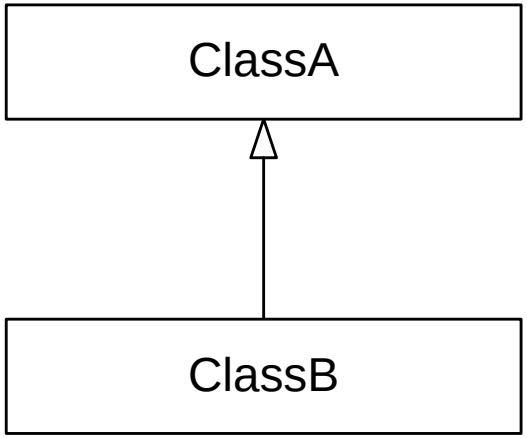
Ví dụ


CHinhVe
int MaLoaiHinhVe
+ float DienTich()
+ float ChuVi()
+ void Ve()
+ ...
CTamGiac
CDiem P1
CDiem P2
CDiem P3
+ CTamGiac()
+ float DienTich()
+ float ChuVi()
+ void Ve()
+ ...
CTuGiac
CDiem P1
CDiem P2
CDiem P3
CDiem P4
+ CTuGiac()
+ float DienTich()
+ float ChuVi()
+ void Ve()
+ ...
CEllipse
CDiem Tam
float A
float B
+ CEllipse()
+ float DienTich()
+ float ChuVi()
+ void Ve()
+ ...

QUAN HỆ

Quan hệ giữa các lớp đối tượng

- Quan hệ kế thừa



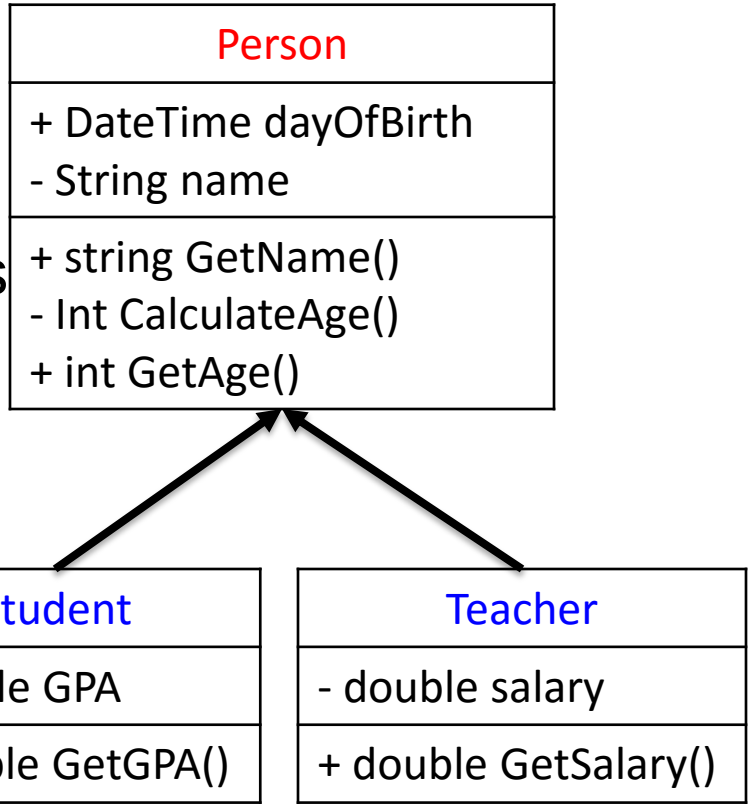
  - ClassB kế thừa từ ClassA
  - ClassB là một trường hợp đặc biệt của ClassA
  - ClassA là trường hợp tổng quát của ClassB

HOW TO DECIDE INHERITANCE

"is-a" relationship

- A Student "is-a" Person
  - all students are persons
  - some persons are not students

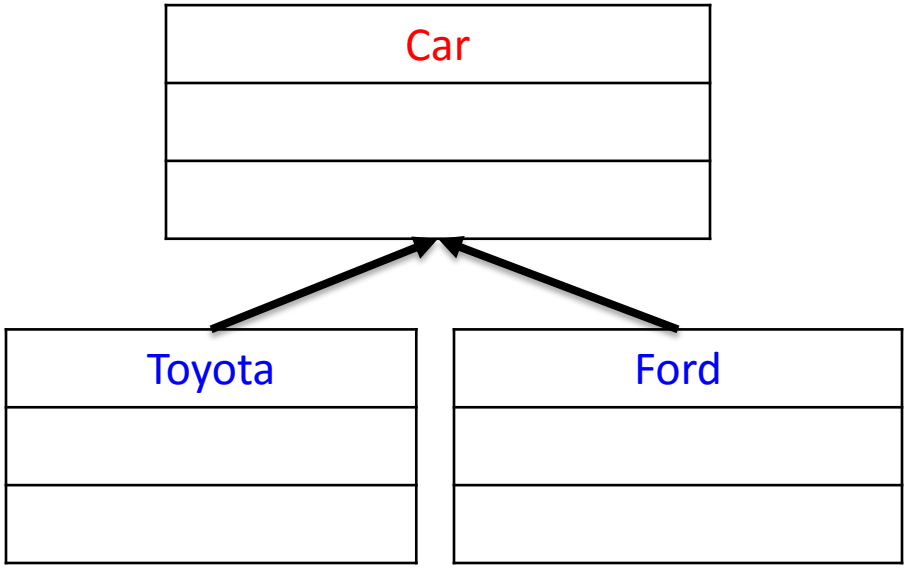
Student class should be "inherited from" Person class



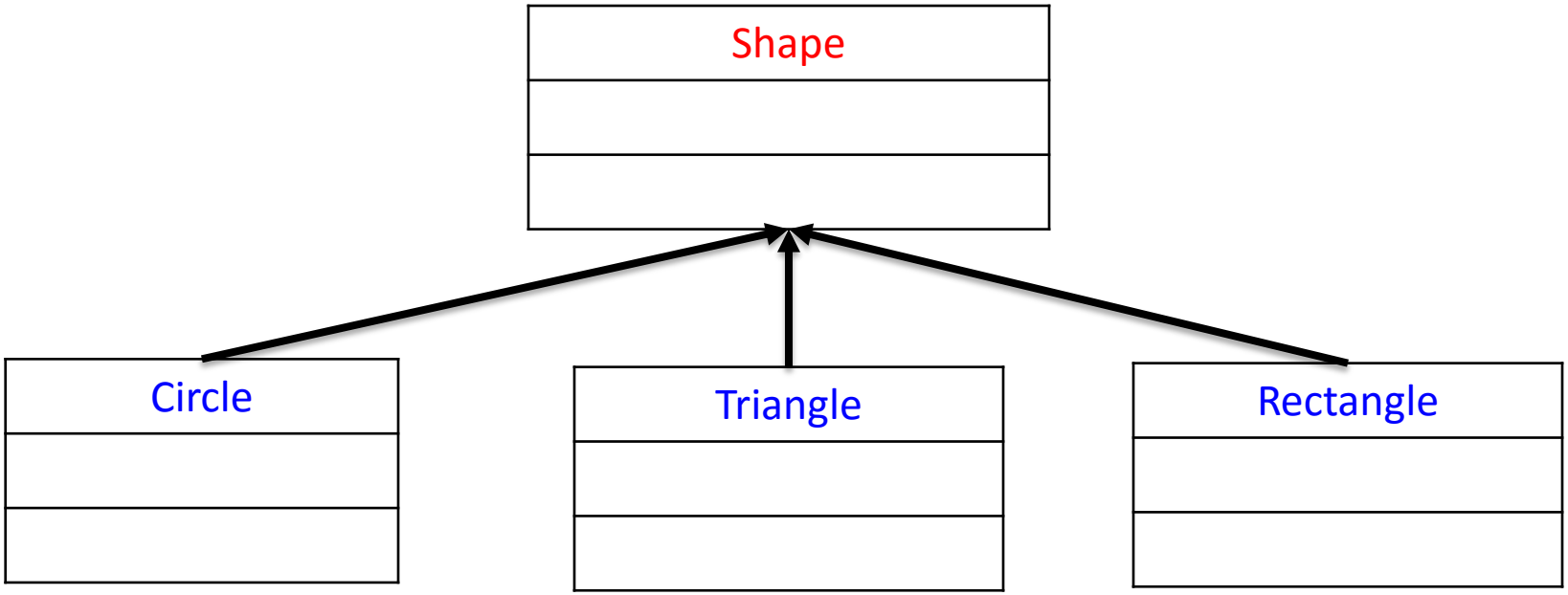
- A Teacher "is-a" Person
  - all teachers are persons
  - some persons are not teachers

Teacher class should be "inherited from" Person class

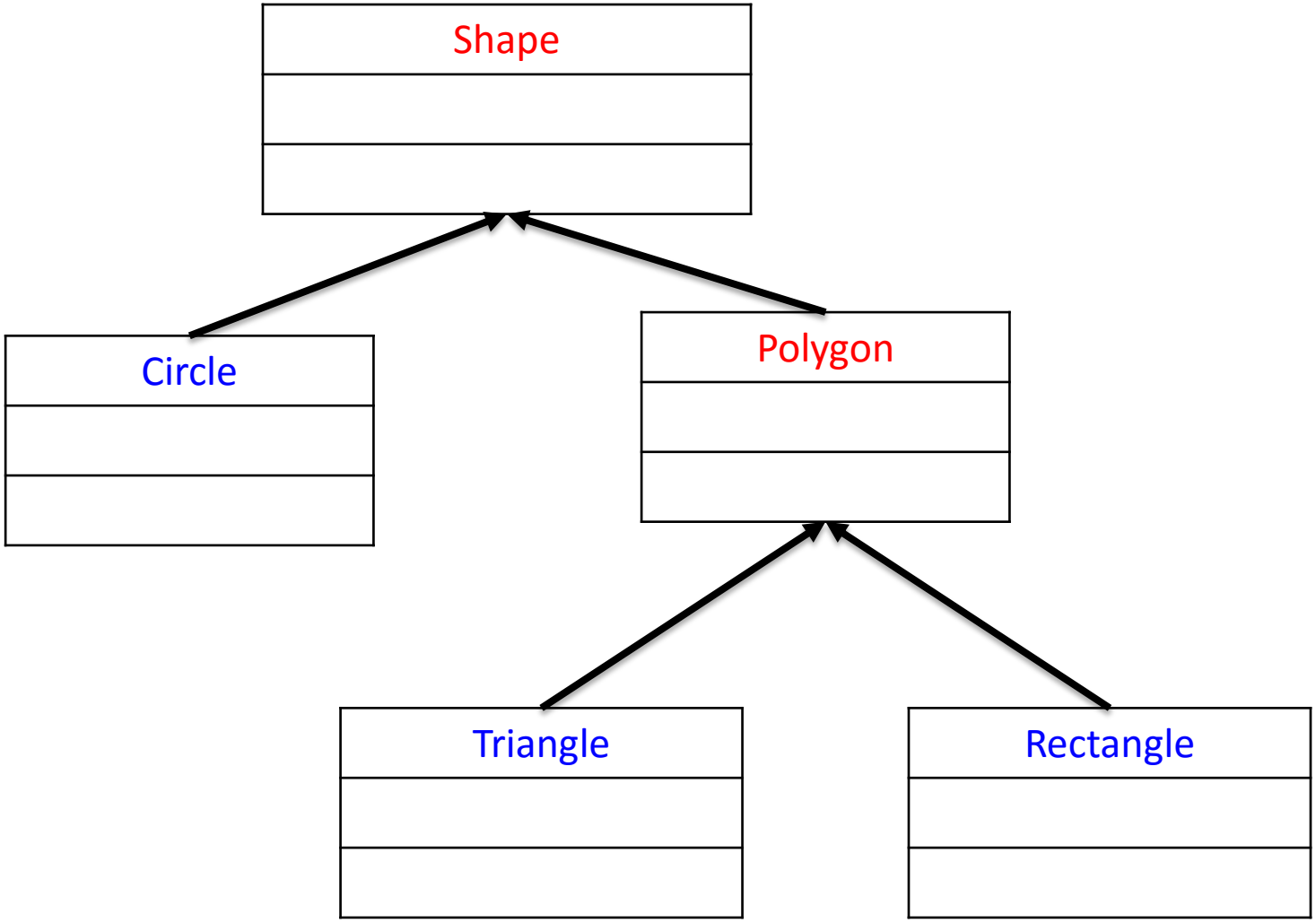
"is-a" relationship



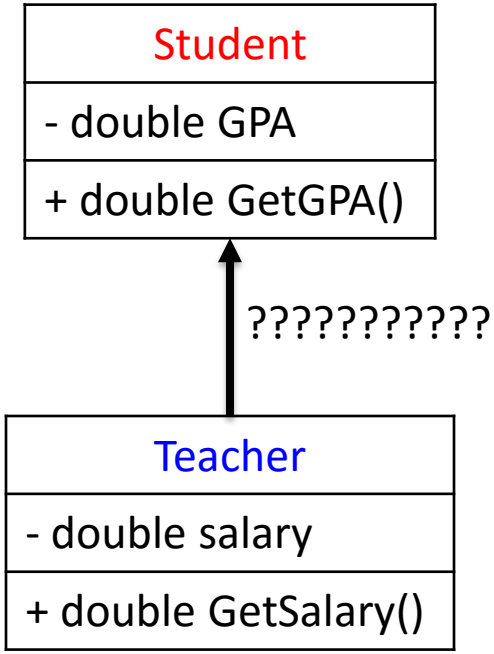
"is-a" relationship



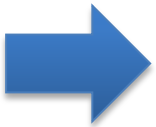
"is-a" relationship



"is-a" relationship



- A Student "is-a" Teacher?
  - all students are teachers ➔ NOT CORRECT



Teacher class should not be "inherited from" Student class

Luyện tập

- Xét các quan hệ dưới đây có phải là quan hệ kế thừa
  - Xe ba bánh và Xe bốn bánh
  - Lớp học và sinh viên
  - Sinh viên và lớp trưởng
  - Giáo vụ và giáo viên
  - Hình vuông và Hình tròn
  - Tam giác cân và tam giác đều
  - Tam giác cân và tam giác vuông
  - Tam giác cân và tam giác vuông cân

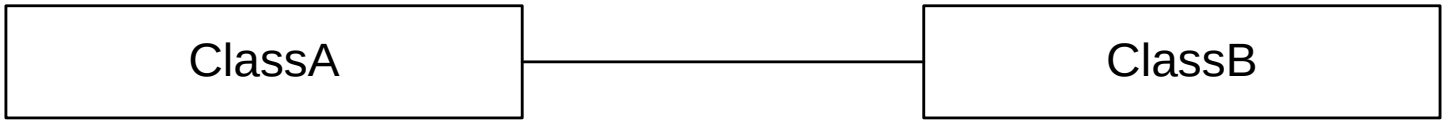
Thuộc tính của kế thừa

- Hình vuông và Hình chữ nhật
- Hình tròn và Hình Ellipse
- Hình đa giác và Hình chữ nhật

ASSOCIATION

Quan hệ giữa các lớp đối tượng

- Quan hệ Association



  - Hoặc
    - Trong ClassA có thuộc tính có kiểu là ClassB
  - Hoặc
    - Trong ClassB có thuộc tính có kiểu là ClassA

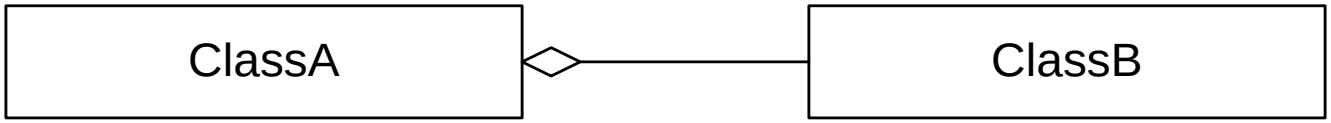
  - Nhận xét: Về mặt lập trình, thuộc tính có thể được lưu trữ dạng biến đơn, biến mảng, hay biến con trỏ
  - Ví dụ:?

Luyện tập

- Xét các cặp lớp sau đây có quan hệ association
  - Lớp học và Môn học
  - Lớp học và Thời khóa biểu
  - Laptop và Desktop

Quan hệ giữa các lớp đối tượng

- Quan hệ Aggregation



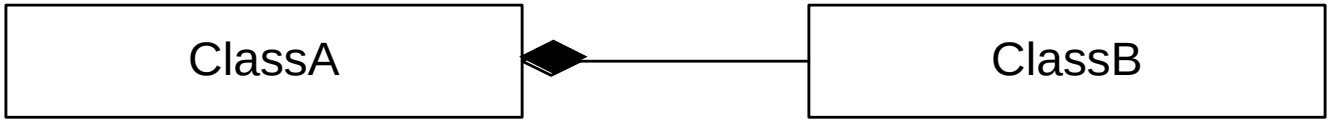
  - Đã xác định được ClassA và ClassB có quan hệ Association với nhau
  - Xác định rõ hơn:
    - Trong object của ClassA có chứa (trong phần thuộc tính) object của ClassB
    - ObjectX của ClassA bị hủy thì ObjectY của ClassB (bên trong ObjectX) vẫn có thể còn tồn tại
  - Ví dụ:?

Luyện tập

- Xét các cặp lớp sau đây có quan hệ aggregation
  - Lớp học và Môn học
  - Lớp học và Thời khóa biểu
  - Sinh viên và Mắt kiếng
  - Laptop và Con người
  - Giáo viên và Bộ môn

Quan hệ giữa các lớp đối tượng

- Quan hệ Composition



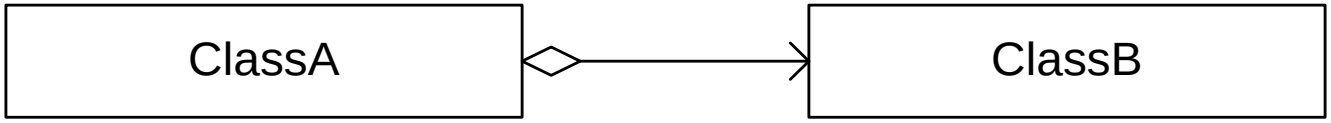
  - Đã xác định được ClassA và ClassB có quan hệ Association với nhau
  - Xác định rõ hơn:
    - Trong object của ClassA có chứa (trong phần thuộc tính) object của ClassB
    - ObjectX của ClassA bị hủy thì ObjectY của ClassB (bên trong ObjectX) không thể còn tồn tại
  - Ví dụ:?

Luyện tập

- Xét các cặp lớp sau đây có quan hệ composition
  - Xe hơi và Bánh xe
  - Iphone và Pin iphone
  - Thư mục và Tập tin
  - Mainboard và Card màn hình onboard
  - Người và Trái tim
  - Người và Đầu
  - Playlist và File mp3

Quan hệ giữa các lớp đối tượng

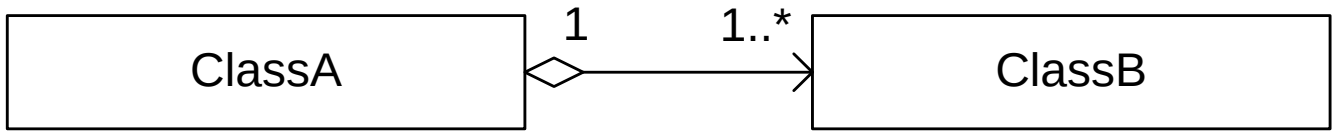
- Chiều của quan hệ (Association, Aggregation, Composition)



  - Nếu quan hệ là 1 chiều: đa số các lời gọi hàm được gọi theo đúng chiều của quan hệ
  - Nếu quan hệ là 2 chiều: không vẽ mũi tên
- Ví dụ: Hình tròn và Tâm

Quan hệ giữa các lớp đối tượng

- Bản số - Multiplicity (Association, Aggregation, Composition)



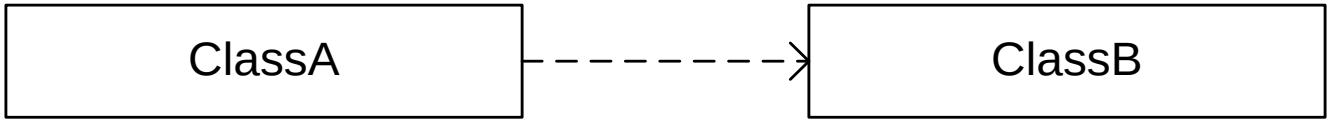
  - Ý nghĩa
  - Ví dụ:
    - 1
    - 2
    - 1..*
    - 0..*
    - *
    - 1, 3, 5..9

Luyện tập

- Vẽ bản số cho các trường hợp sau
  - Lớp học và Giáo viên
  - Lớp học và Giáo viên chủ nhiệm
  - Vợ và Chồng
  - Cha và Con
  - Họa sĩ và Tác phẩm
  - Bác sĩ và Bệnh nhân

Quan hệ giữa các lớp đối tượng

- Quan hệ Dependency



  - ClassA và ClassB không có quan hệ Association
  - ClassA "phụ thuộc" vào ClassB

Tham số truyền vào

```
class A
{
   void   F(B   x)
   {
    …
   }
};
```

Kết quả trả ra

```
class A
{
   B   F()
   {
    …
   }
};
```

Biến cục bộ

```
class A
{
   void   F()
   {
      B   x;
   }
};
```

Trong ClassA có sử dụng phương thức/thuộc tính static của ClassB

PHỤ LỤC

THAM CHIẾU TRONG C#

Tham chiếu trong C#

- Tất cả các kiểu dữ liệu nguyên thủy (int, float, char, double, string…) sử dụng tham trị
- Tất cả các kiểu dữ liệu mảng, đối tượng đều sử dụng tham chiếu

VÍ DỤ THAM TRỊ

Example of memory model

```
int operand1;
operand1 = 2;
int operand2 = 5;
int sum = operand1 + operand2;
Console.WriteLine(operand1 + " + " +
        operand2 + " = " + sum);
```

Variable declaration

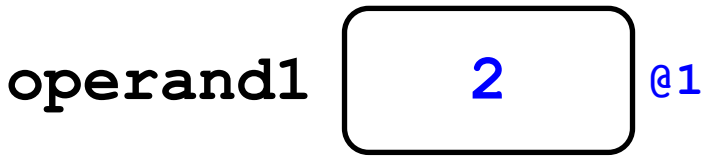
STACK

operand1  @1

Example of memory model

```
int operand1;
operand1 = 2;
int operand2 = 5;
int sum = operand1 + operand2;
Console.WriteLine(operand1 + " + " +
        operand2 + " = " + sum);
```

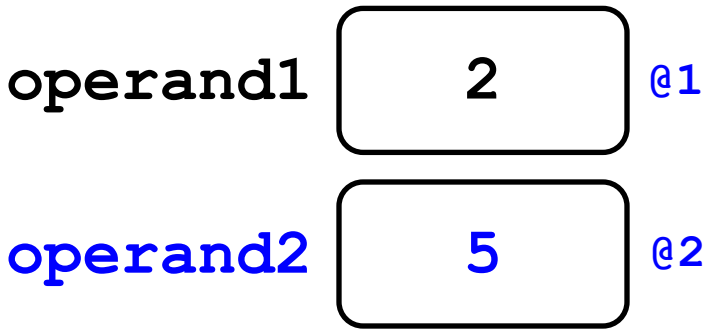
Assignment



Example of memory model

```
int operand1;
operand1 = 2;
int operand2 = 5;
int sum = operand1 + operand2;
Console.WriteLine(operand1 + " + " +
        operand2 + " = " + sum);
```

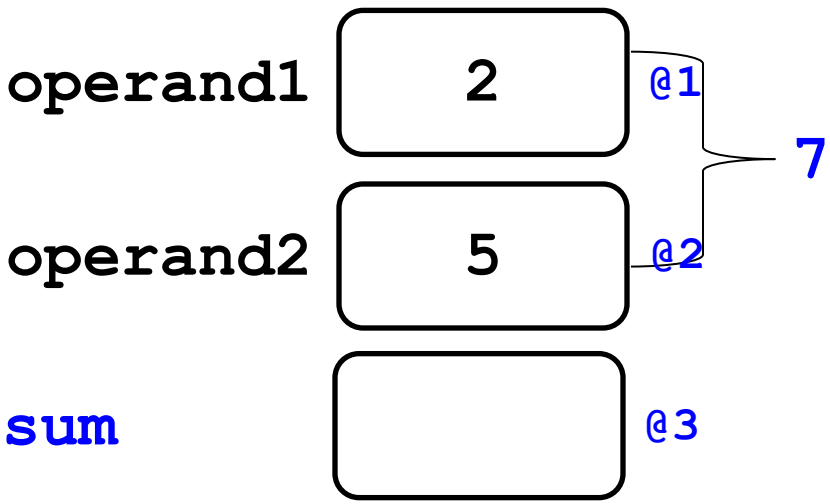
Declaration and Assignment



Example of memory model

```
int operand1;
operand1 = 2;
int operand2 = 5;
int sum = operand1 + operand2;
Console.WriteLine(operand1 + " + " +
        operand2 + " = " + sum);
```

Declaration and Assignment



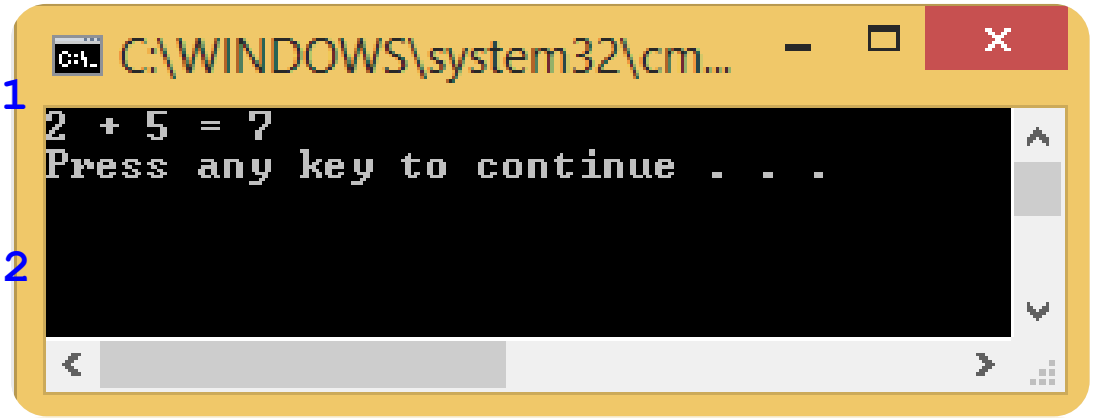
Example of memory model

```
int operand1;
operand1 = 2;
int operand2 = 5;
int sum = operand1 + operand2;
Console.WriteLine(operand1 + " + " +
        operand2 + " = " + sum);
```

output

STACK

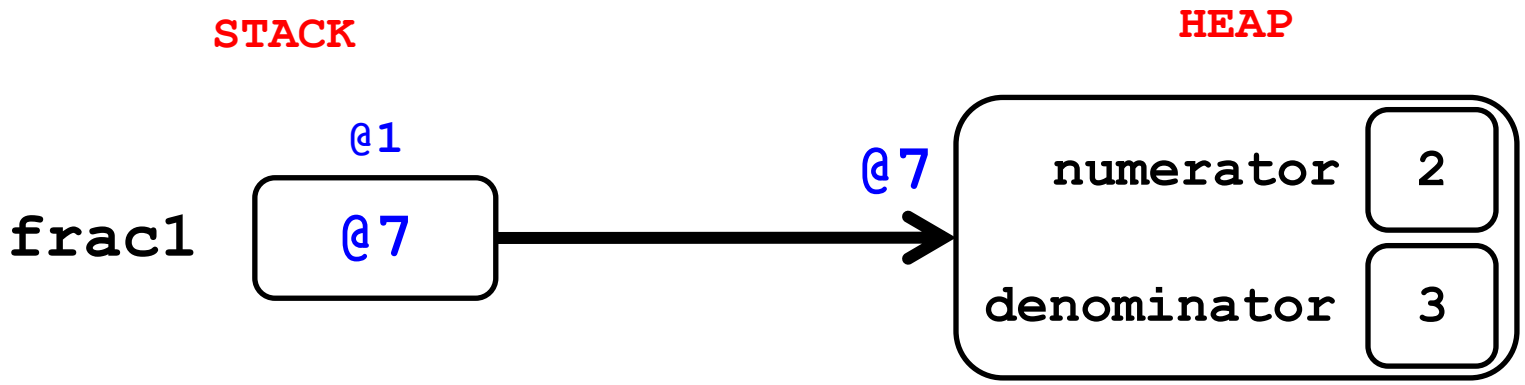
operand1	2	@1
operand2	5	@2
sum	7	@3



VÍ DỤ THAM CHIẾU

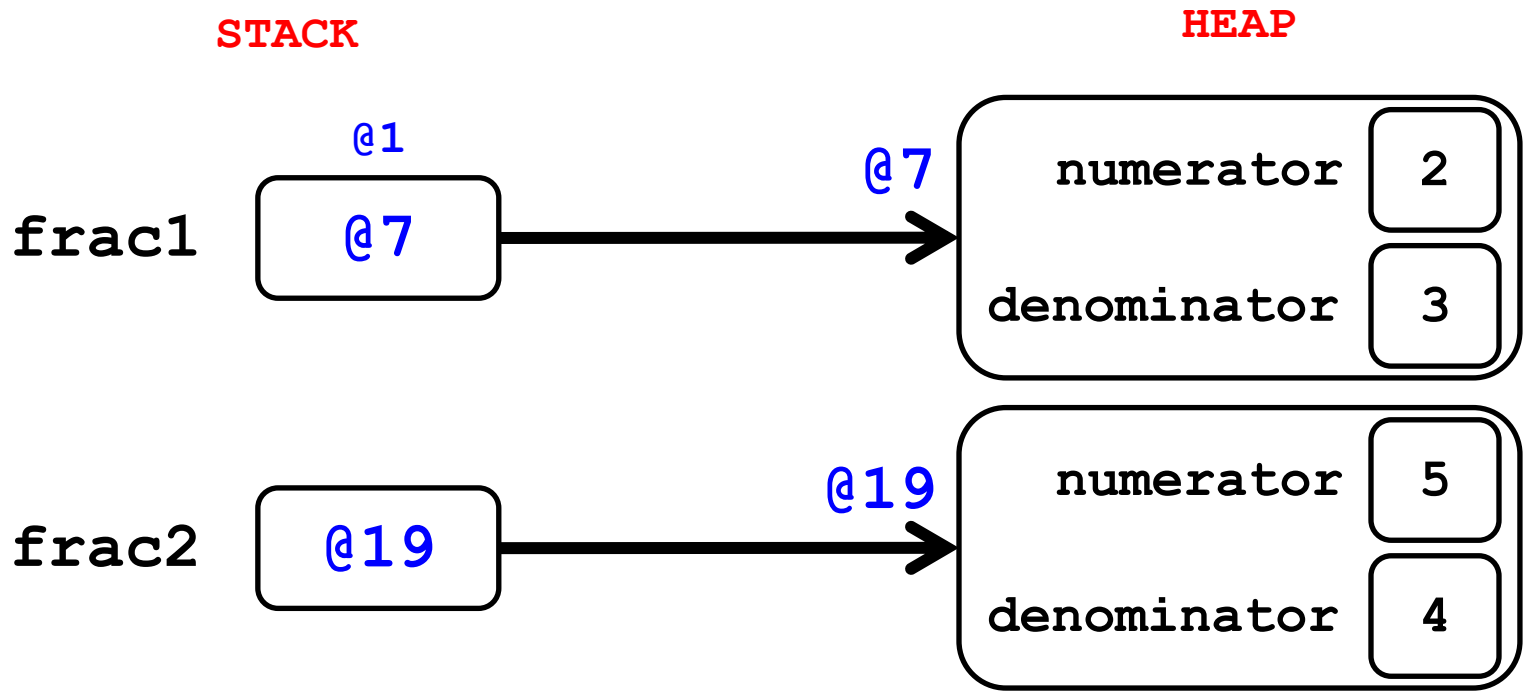
Object assignment

```
Fraction frac1 = new Fraction(2, 3);
Fraction frac2 = new Fraction (5, 4);
frac2 = frac1;
frac1.numerator = 9;
Console.WriteLine(frac2.numerator+"/"+
                  frac2.denominator
```



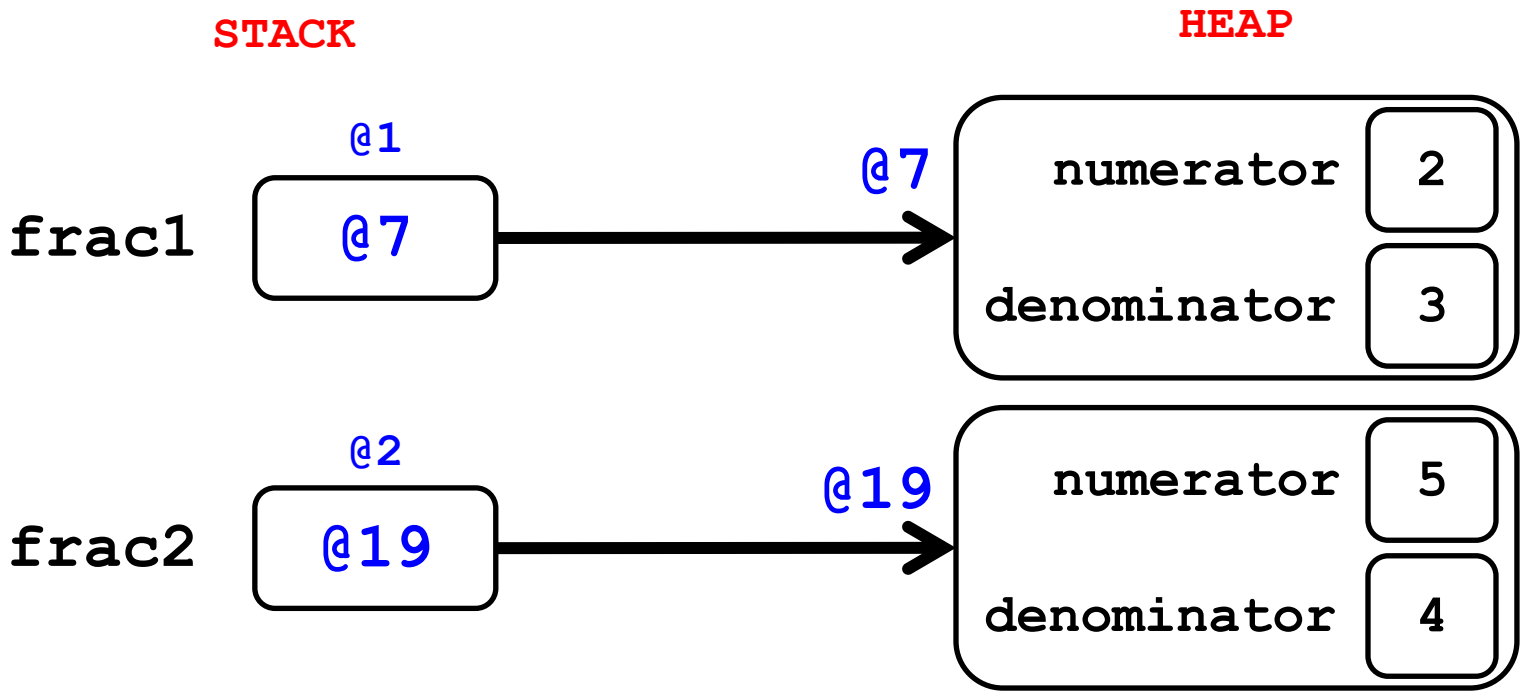
Object assignment

```
Fraction frac1 = new Fraction(2, 3);
Fraction frac2 = new Fraction (5, 4);
frac2 = frac1;
frac1.numerator = 9;
Console.WriteLine(frac2.numerator+"/"+
                  frac2.denominator
```



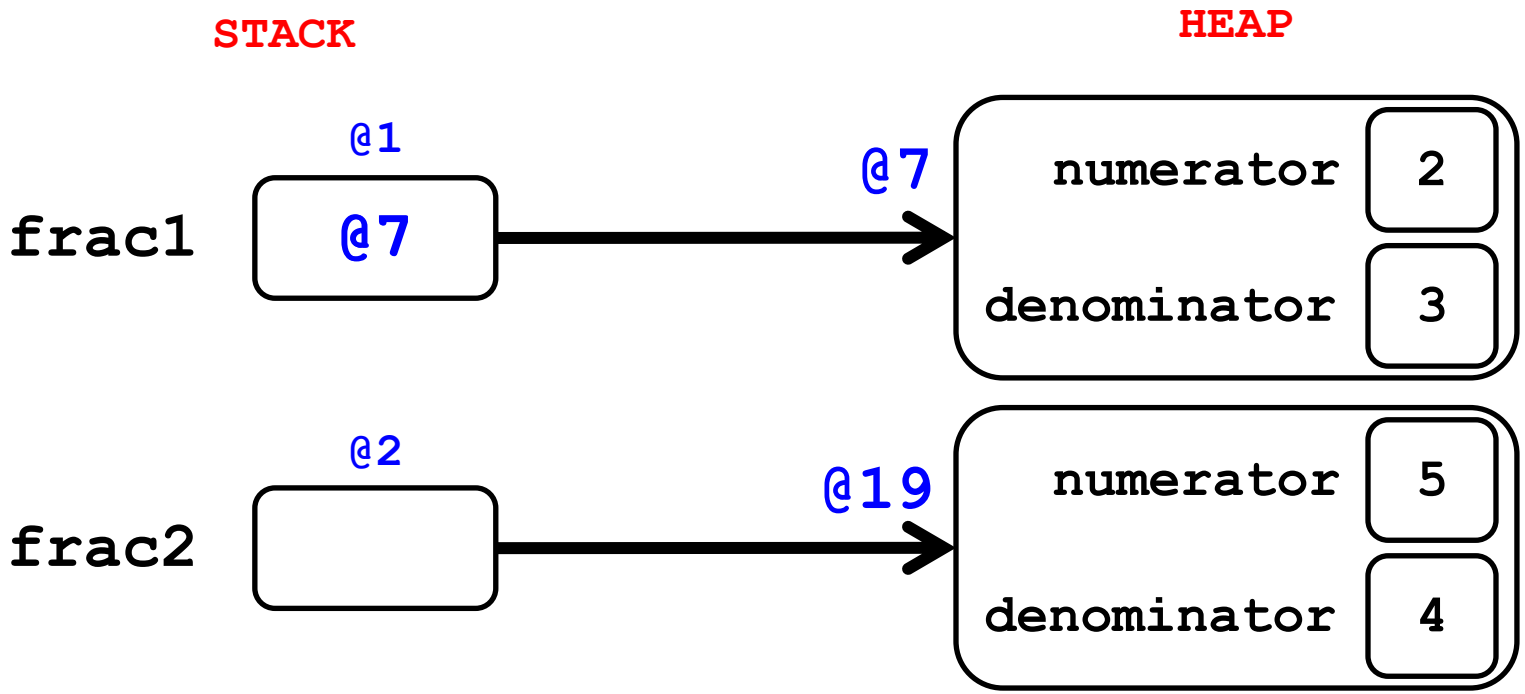
Object assignment

```
Fraction frac1 = new Fraction(2, 3);
Fraction frac2 = new Fraction (5, 4);
frac2 = frac1;
frac1.numerator = 9;
Console.WriteLine(frac2.numerator+"/"+
                  frac2.denominator
```



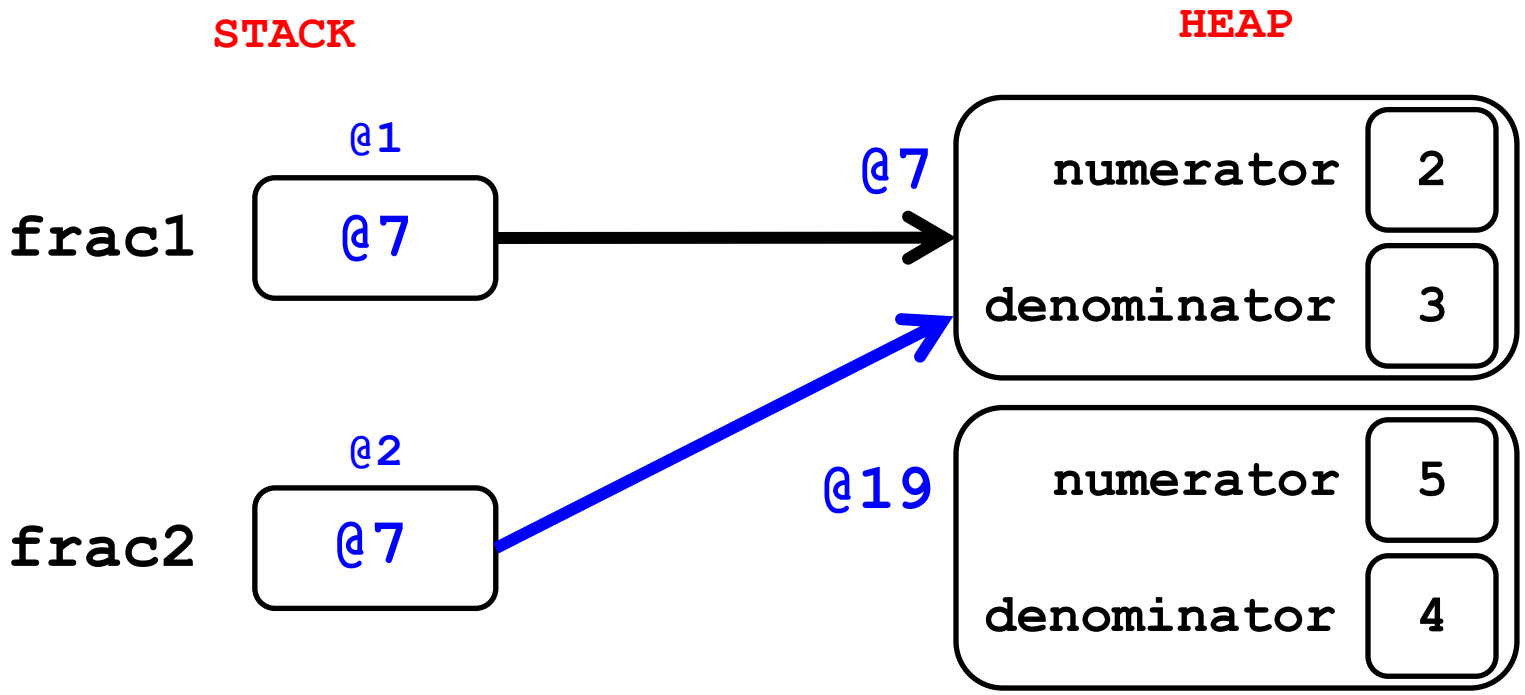
Object assignment

```
Fraction frac1 = new Fraction(2, 3);
Fraction frac2 = new Fraction (5, 4);
frac2 = frac1;
frac1.numerator = 9;
Console.WriteLine(frac2.numerator+"/"+
                  frac2.denominator
```



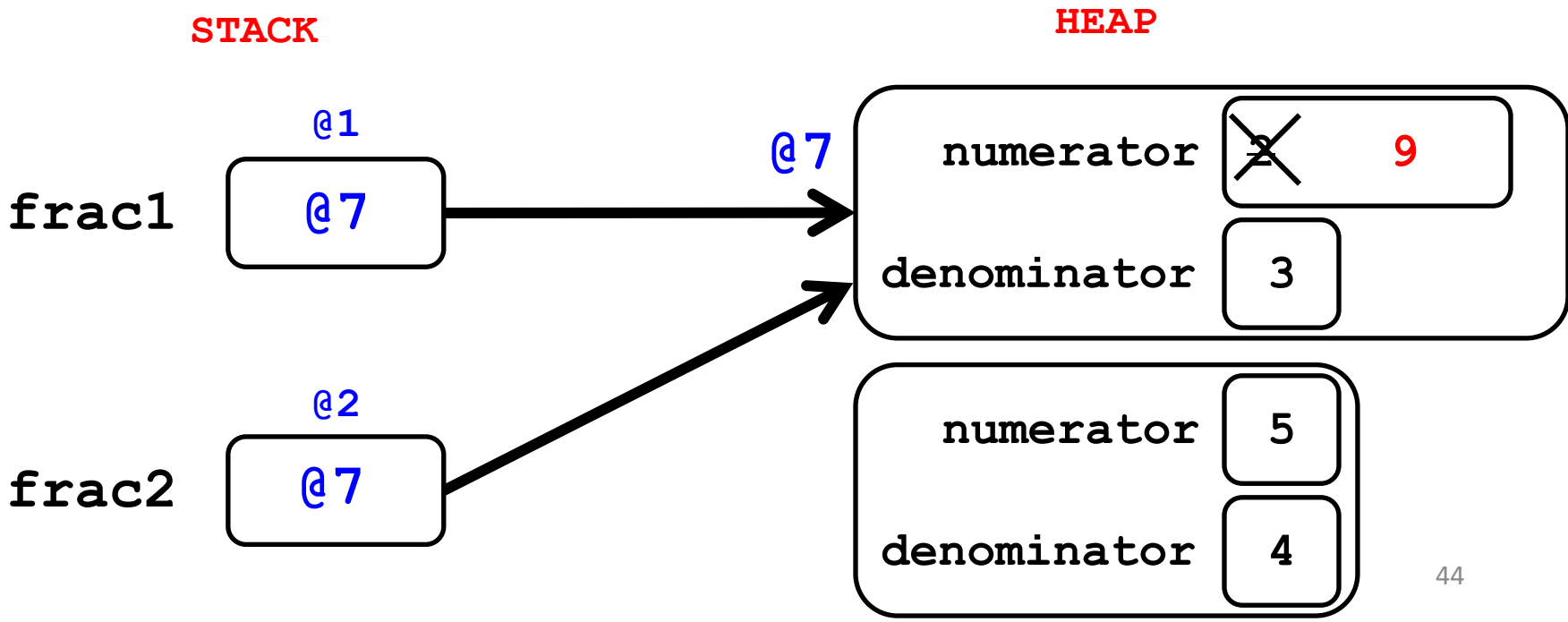
Object assignment

```
Fraction frac1 = new Fraction(2, 3);
Fraction frac2 = new Fraction (5, 4);
frac2 = frac1;
frac1.numerator = 9;
Console.WriteLine(frac2.numerator+"/"+
                  frac2.denominator
```



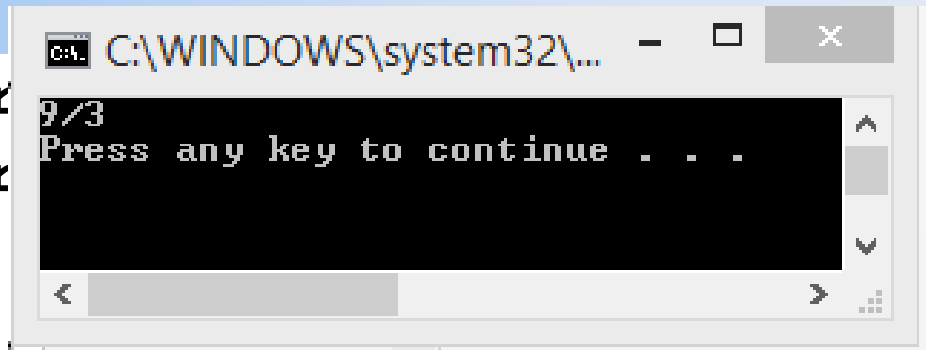
Object assignment

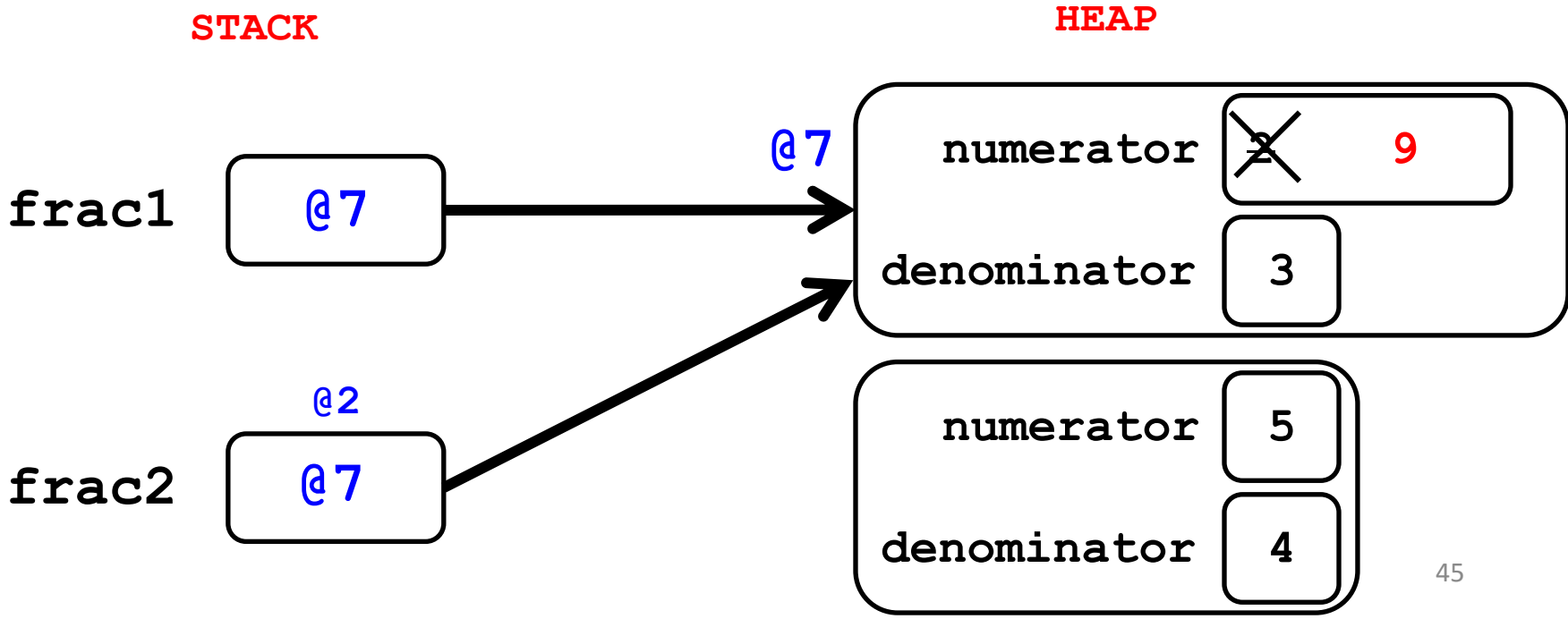
```
Fraction frac1 = new Fraction(2, 3);
Fraction frac2 = new Fraction (5, 4);
frac2 = frac1;
frac1.numerator = 9;
Console.WriteLine(frac2.numerator+"/"+
                  frac2.denominator
```



Object assignment

```
Fraction frac1 = new Fr
Fraction frac2 = new Fr
frac2 = frac1;
frac1.numerator = 9;
Console.WriteLine(frac2.numerator+"/"+
                  frac2.denominator);
```





CON TRỎ THIS

Con trỏ this

- Mỗi lớp đều có con trỏ this.
- Đại diện cho đối tượng đang gọi phương thức.
- Hữu dụng trong một số trường hợp.

```
class PhanSo
{
private:
        int         iTuSo;
        int         iMauSo;
public:
        void ganTuSo(int iTuSo) { this->iTuSo = iTuSo; }
};
```

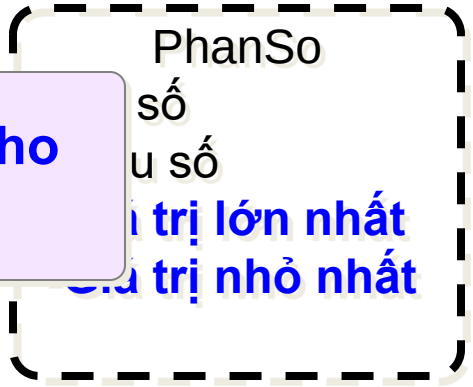
```
void main()
{
    PhanSo   p1;
    p1.ganTuSo(3);

    PhanSo   p2;
    p2.ganTuSo(5);
}
```

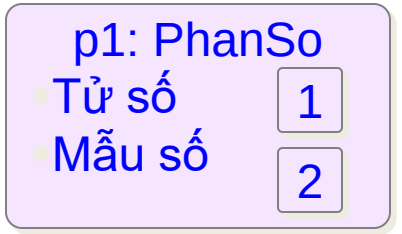
STATIC

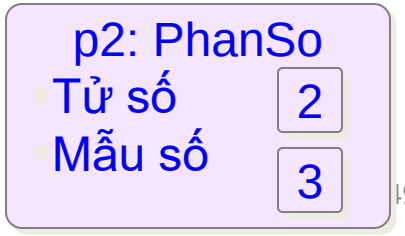
Thành phần tĩnh

- Thành phần của lớp (class members):
  - Thành phần đối tượng (instance members).
    - Thuộc tính và phương thức thông thường.
    - Mỗi đối tượng có bản sao riêng.
  - Thành phần tĩnh (static members).
    - Thuộc tính và phương thức tĩnh.
    - Các đối tượng dùng chung.

Thành phần dùng chung cho MỌI đối tượng của lớp!!







Thành phần tĩnh

- Khai báo và sử dụng:
  - Dùng từ khóa static.
  - Truy xuất bằng toán tử ::.

```
class PhanSo
{
private:
        static int m_iGiaTriLN;
public:
        static int layGiaTriLN();
public:
        PhanSo(int tu, int mau)
        {
                m_iTu= tu;
                m_iMau = mau:
        }
private:
        int     m_iTu;
        int     m_iMau;
};
```

```
int PhanSo::m_iGiaTriLN = 10000;

int PhanSo::layGiaTriLN()
{
    return m_iGiaTriLN;
}

void main()
{
    PhanSo   p1(1, 2);
    PhanSo   p2(2, 3);

    int x1 = PhanSo::layGiaTriLN();
    int x2 = p1.layGiaTriLN();
}
```