

Chương 3

NHỮNG KHÁI NIỆM CƠ BẢN CỦA LẬP TRÌNH HẾT

Mục tiêu



- Phân biệt đối tượng, lớp đối tượng (nhắc lại)
- Mô hình hóa được lớp đối tượng
- Thể hiện được tính đóng gói trong LT HĐT
- Tạo ra một lớp bằng ngôn ngữ lập trình Java từ mô tả cho trước hoặc từ mô hình lớp đơn giản
- Cài đặt các lớp có mối quan hệ với nhau

Nội dung



3.1. Sơ đồ lớp UML

3.2. Định nghĩa một lớp trong Java

3.3. Một số mối quan hệ giữa các lớp đối tượng (association)

3.1. Sơ đồ lớp UML

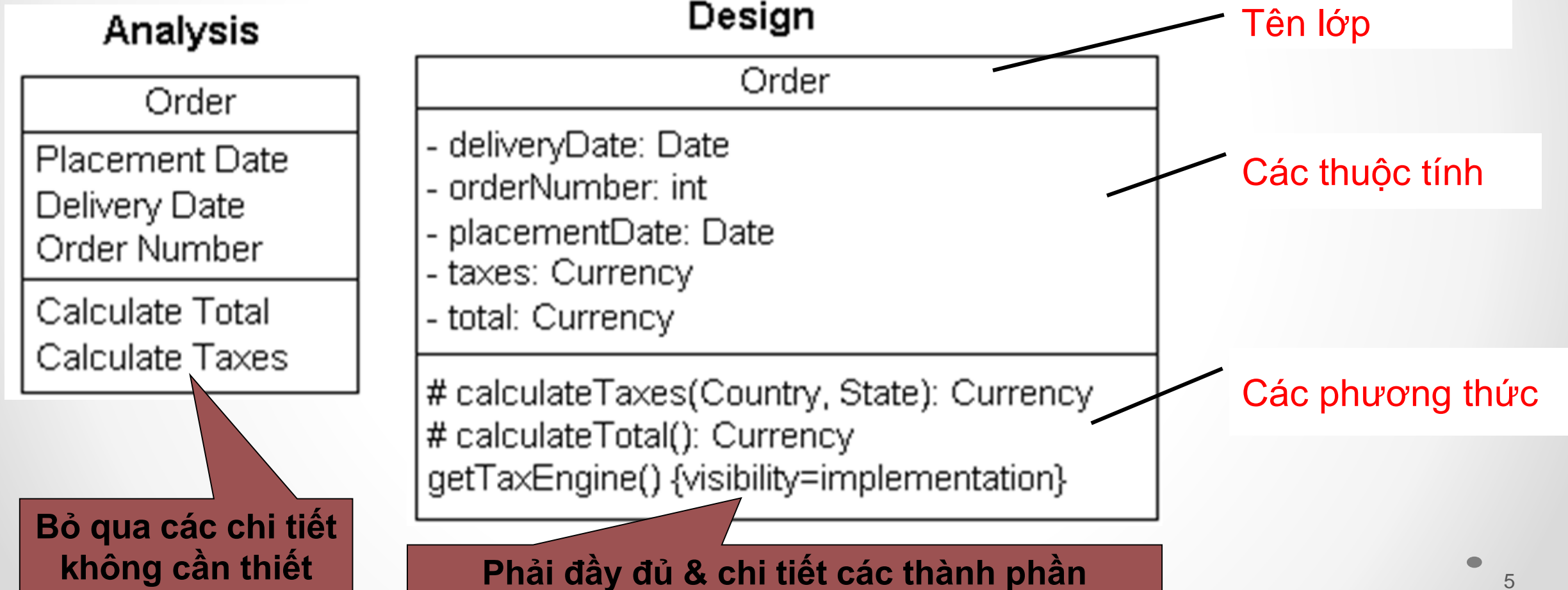


- UML (Unified Model Language) là một ngôn ngữ mô hình hóa, được dùng để đặc tả, trực quan hóa, xây dựng và làm tài liệu cho các hệ thống phần mềm
- UML không lệ thuộc ngôn ngữ lập trình
- Dùng UML để biểu diễn một lớp trong Java
 - Biểu diễn ở mức phân tích (analysis)
 - Biểu diễn ở mức thiết kế chi tiết (design)

3.1. Sơ đồ lớp UML



- Ví dụ UML để biểu diễn một lớp trong Java



3.1. Sơ đồ lớp UML

Lớp – Đối tượng



- ĐỐI TƯỢNG là một thực thể cụ thể mà ta có thể sờ, nhìn thấy hay cảm nhận được. Mỗi đối tượng có thuộc tính và hành vi riêng của nó.
- LỚP là một khuôn mẫu để tạo ra đối tượng. **Lớp tạo ra đối tượng bằng cách gán giá trị cụ thể cho các thuộc tính tương ứng.**



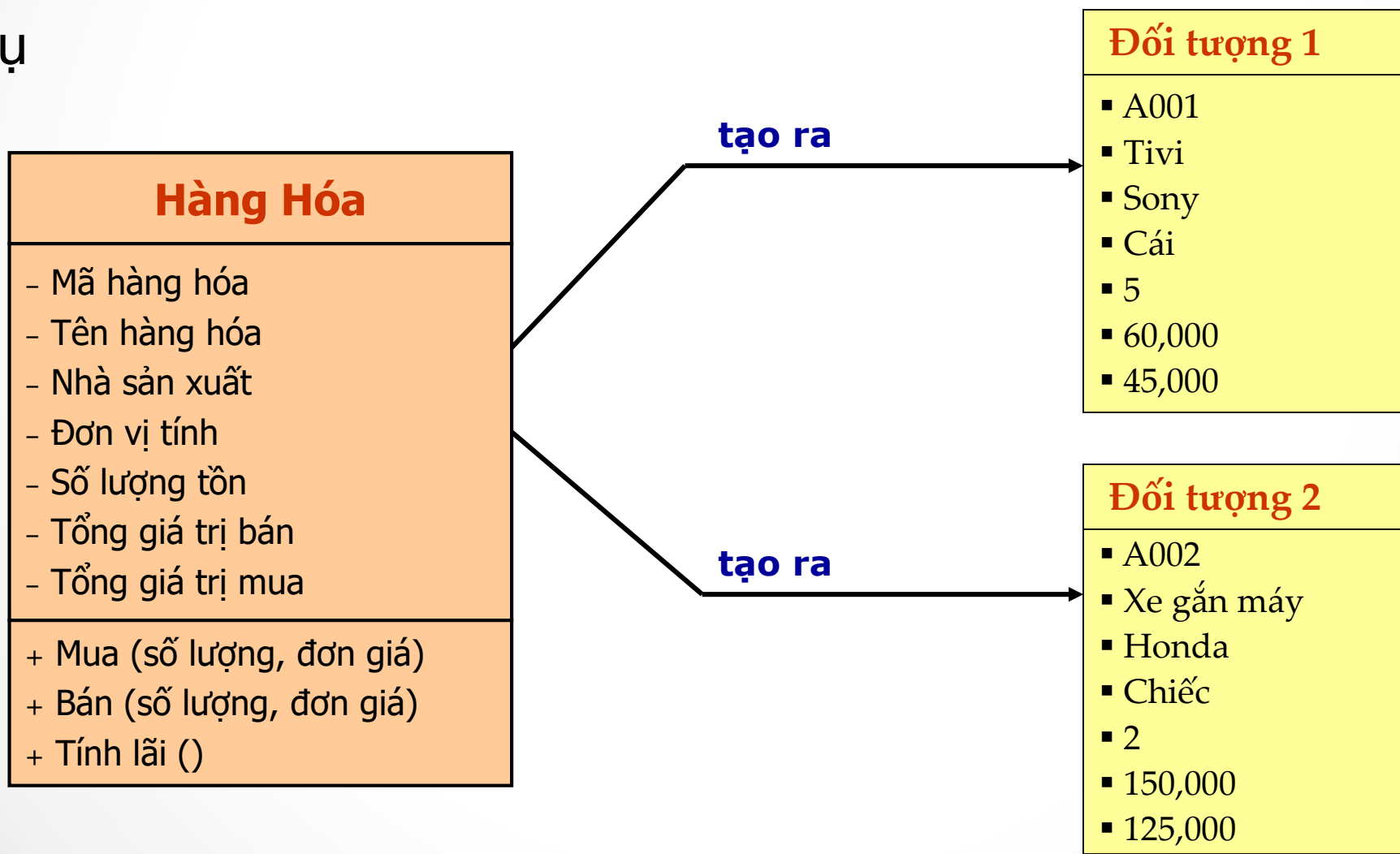
Đối tượng là một thể hiện (instance) của một lớp

3.1. Sơ đồ lớp UML

Lớp – Đối tượng



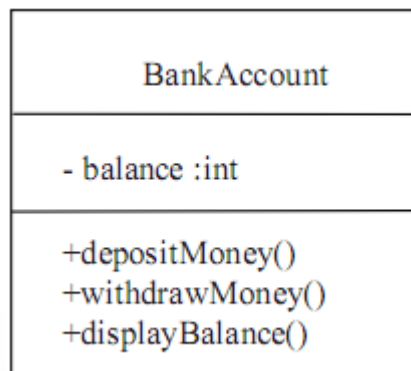
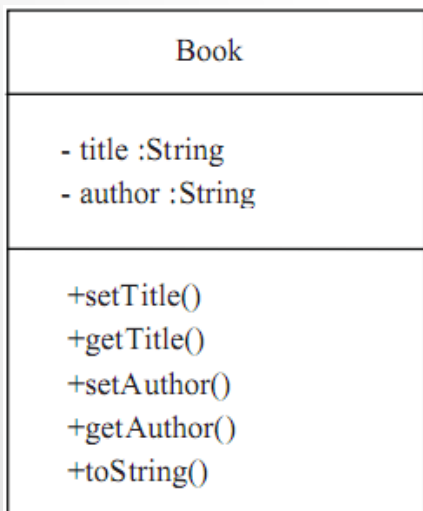
- Ví dụ



3.1. Sơ đồ lớp UML



- Một class có:
 - Tên gọi
 - Tập các thuộc tính
 - Tập các phương thức
- Ví dụ



Vẽ sơ đồ lớp cho lớp **HìnhChuNhat** gồm có:
Các thuộc tính: chiều dài, chiều rộng.
Các phương thức: tính diện tích, tính chu vi, lấy thông tin (trả về các thông tin của một hình chữ nhật)

3.2. Định nghĩa lớp trong Java



- Khai báo lớp
- Phương thức khởi tạo (constructor) và tạo đối tượng
- Thể hiện tính đóng gói – Phạm vi truy cập
- Từ khóa *this*
- Lập chú thích cho tài liệu (documentation comment)
- Kiểm tra ràng buộc dữ liệu
- Kiểm thử

3.2. Định nghĩa lớp trong Java

Khai báo lớp



- Cú pháp:

```
class TenLop
{
    Các thuộc tính của lớp (khai báo biến)
    Các phương thức của lớp
}
```

Tên Lớp
Các Thuộc tính
Các Phương thức

- Trong đó:
 - **class**: là từ khóa cho biết đang khai báo một lớp
 - **TenLop**: là tên của lớp (*phải liên quan đến đối tượng, gọi nhớ đối tượng*), quy luật đặt tên lớp cũng giống như quy luật đặt tên cho biến. *Lưu ý, nên viết hoa chữ đầu của mỗi từ, ví dụ HìnhTron, SinhVien, HoaDon, ...*

3.2. Định nghĩa lớp trong Java

Khai báo lớp: Ví dụ 1



- Vẽ mô hình lớp và cài đặt lớp hình chữ nhật, có:
 - *Các thuộc tính: chiều dài, chiều rộng.*
 - *Các phương thức: tính diện tích, tính chu vi, lấy thông tin (trả về thông tin của một hình chữ nhật)*

HìnhChuNhat	
~	cDai: float
~	cRong: float
~	tinhChuVi(): float
~	tinhDienTich(): float
~	layThongTin(): String

3.2. Định nghĩa lớp trong Java

Khai báo lớp: Ví dụ 1



- Solution

3.2. Định nghĩa lớp trong Java

Khai báo lớp: Ví dụ 2



- Cài đặt lớp sinh viên như sau:

SinhVien	
~	maSV: int
~	hoTen: String
~	diemLiThuyet: float
~	diemThucHanh: float
~	tinhDiemTrungBinh(): float
~	toString(): String

Điểm tb = (điểm lí thuyết + điểm thực hành)/2

Phương thức toString trả về thông tin của SV

3.2. Định nghĩa lớp trong Java

Khai báo lớp: Ví dụ 2



- Solution

3.2. Định nghĩa lớp trong Java

Phương thức khởi tạo (constructor)



- *Phương thức khởi tạo* là phương thức dùng để tạo đối tượng của lớp, nhằm **gán giá trị ban đầu** cho đối tượng, là phương thức đặc biệt, vì được gọi thông qua toán tử **new**.
- Khi viết hàm khởi tạo, phải thỏa 2 điều kiện:
 - *tên phương thức trùng với tên class,*
 - *không trả về giá trị, cũng không phải void*
- Nếu lớp không có constructor nào, trình biên dịch tạo constructor mặc định (*default constructor*), là constructor không có tham số và tự gán các giá trị mặc định (*số $\rightarrow 0$, chuỗi $\rightarrow null$, boolean $\rightarrow false$...*)
- Một lớp có thể không có constructor nào hoặc có thể có một hay nhiều constructor, khác nhau ở danh sách tham số.

3.2. Định nghĩa lớp trong Java

Phương thức khởi tạo (constructor)



- Vd1. Viết phương thức khởi tạo gán giá trị cho cả chiều dài và chiều rộng của hình chữ nhật.

```
public class HìnhChuNhat {  
    private float cDai, cRong;  
    public HìnhChuNhat (float cd, float cr) {  
        cDai = cd;  
        cRong = cr;  
    }  
    //...  
}
```

Phương thức khởi tạo

3.2. Định nghĩa lớp trong Java

Phương thức khởi tạo (constructor)



- Vd2. Viết 4 constructor cho lớp SinhVien:
 - 1 constructor mặc định
 - 1 constructor có một tham số là mã sv
 - 1 constructor có hai tham số là mã sv và họ tên
 - 1 constructor có đầy đủ tham số

```
public class SinhVien {  
    private int maSV;  
    private String hoTen;  
    private float diemLiThuyet;  
    private float diemThucHanh;  
    public SinhVien() {  
    }  
    public SinhVien(int ma) {  
        maSV = ma;  
    }  
    public SinhVien(int ma, String hoten) {  
        maSV = ma;  
        hoTen = hoten;  
    }  
    //...  
}
```

3.2. Định nghĩa lớp trong Java

Tạo đối tượng



- Để tạo một đối tượng: dùng toán tử **new** :

```
// gọi hàm khởi tạo mặc định
```

```
ClassName objectName = new ClassName();
```

```
// HOẶC gọi hàm khởi tạo có tham số
```

```
ClassName objectName = new ClassName(các giá trị khởi tạo);
```

- Để truy xuất các thuộc tính và phương thức của đối tượng: dùng **toán tử chấm** (dot operator)

3.2. Định nghĩa lớp trong Java

Tạo đối tượng: Ví dụ



- Vd1. Tạo đối tượng hình chữ nhật có chiều dài 10, chiều rộng 5 từ lớp HìnhChuNhat:

```
public class HìnhChuNhat {  
    private float cDai, cRong;  
    public HìnhChuNhat(float cd, float cr) {  
        cDai = cd;  
        cRong = cr;  
    }  
    //...  
}
```

```
public class Test {  
    public static void main(String[] args) {  
        HìnhChuNhat h = new HìnhChuNhat(10,5);  
        System.out.println(h.LayThongTin());  
    }  
}
```

Console

```
<terminated> Test (1) [Java Application] C:\Program Files\Java\jre1.8.0_141  
Hình chu nhật, cdai=10.00, crong= 5.00
```

3.2. Định nghĩa lớp trong Java

Tạo đối tượng: Ví dụ



- Vd2. Tạo một đối tượng hình chữ nhật (dùng hàm khởi tạo mặc định):

```
public class Test {  
    public static void main(String[] args) {  
        HìnhChuNhat h = new HìnhChuNhat();  
  
        System.out.println(h.layThongTin());  
    }  
}
```

- Vd3. Tạo một đối tượng sinh viên (dùng hàm khởi tạo mặc định):

```
SinhVien sv = new SinhVien();  
System.out.println(sv.toString());
```

3.2. Định nghĩa lớp trong Java

Thể hiện tính đóng gói



- Khai báo các thuộc tính là **private**, để các lớp khác không thể truy cập trực tiếp được.
- Tạo ra các phương thức **get/set** là **public**, để lấy/gán giá trị của thuộc tính, nếu cần.

3.2. Định nghĩa lớp trong Java

Thể hiện tính đóng gói: Ví dụ



HìnhChuNhat
- cDai: float
- cRong: float
+ tinhChuVi(): float
+ tinhDienTich(): float
+ layThongTin(): String

HìnhChuNhat
- cDai: float
- cRong: float
+ getChieuDai(): float
+ setChieuDai(float): void
+ getChieuRong(): float
+ setChieuRong(float): void
+ tinhChuVi(): float
+ tinhDienTich(): float
+ layThongTin(): String

Trước khi viết các phương thức get/set

```
public class Test {  
    public static void main(String[] args) {  
        HìnhChuNhat h = new HìnhChuNhat();  
        h.cDai = 10;  
        System.out.println(h.layThongTin());  
    }  
}
```

The field HìnhChuNhat.cDai is not visible Test.java

Sau khi viết các phương thức get/set

```
public class Test {  
    public static void main(String[] args) {  
        HìnhChuNhat h = new HìnhChuNhat();  
        h.setChieuDai(10);  
        System.out.println(h.layThongTin());  
    }  
}
```

LÀM SAO GÁN GIÁ TRỊ ĐƯỢC RÕ RÀNG, NHANH CHÓNG?

3.2. Định nghĩa lớp trong Java

Thể hiện tính đóng gói: Ví dụ



```
public class HìnhChuNhat {  
    private float cDai, cRong;  
  
    public float getChieuDai() {  
        return cDai;  
    }  
    public void setChieuDai(float chieuDai) {  
        this.cDai = chieuDai;  
    }  
    public float getChieuRong() {  
        return cRong;  
    }  
    public void setChieuRong(float chieuRong){  
        this.cRong = chieuRong;  
    }  
}
```

```
    public float tinhChuVi() {  
        return (cDai + cRong) * 2;  
    }  
    public float tinhDienTich() {  
        return cDai * cRong;  
    }  
    public String layThongTin() {  
        return String.format("Hình chu nhat,  
                                cdai=%5.2f, crong=%5.2f", cDai,  
                                cRong);  
    }  
}
```

3.2. Định nghĩa lớp trong Java

Phạm vi truy cập



- Các quyền truy xuất trong Java: public, private, protected, default
 - Thành phần được khai báo **public (+)** sẽ được truy cập ở bất cứ đâu.
 - Thành phần được khai báo **private (-)** chỉ được truy cập bên trong class, bên ngoài bản thân class đó thì không truy xuất được.
 - Thành phần được khai báo **protected (#)** sẽ được truy cập bởi bất cứ class nào trong cùng một gói (package) hoặc ở lớp con của nó (có thể khác gói).
 - Thành phần được khai báo **mặc định (~)** (không ghi gì) sẽ được truy cập bởi bất cứ class nào bên trong cùng một package.

3.2. Định nghĩa lớp trong Java

Từ khóa *this*



- Từ khóa **this** được dùng để gọi constructor:
 - Để tránh lặp lại mã, một constructor có thể gọi một constructor khác trong cùng một lớp.
 - Nếu dùng từ khóa *this* để gọi constructor, thì phải để ở dòng lệnh đầu tiên trong constructor đó (nếu không, sẽ bị lỗi biên dịch).
- Từ khóa **this** được dùng như biến đại diện cho đối tượng hiện tại:
 - Dùng để truy xuất một thành phần của đối tượng: ***this.tênThànhPhần***.
 - Khi tham số trùng với tên thuộc tính thì nhờ từ khóa *this* để phân biệt rõ thuộc tính với tham số.

3.2. Định nghĩa lớp trong Java

Từ khóa *this*: Ví dụ



```
public class SinhVien {  
    private int maSV;  
    private String hoTen;  
    private float diemLiThuyet;  
    private float diemThucHanh;  
    public SinhVien() {  
    }  
    public SinhVien(int maSV) {  
        this.maSV = maSV;  
    }  
    public SinhVien(int maSV, String hoTen) {  
        this(maSV);  
        this.hoTen = hoTen;  
    } //...  
}
```

thuộc tính của lớp

giá trị gán

Dùng *this* để gọi constructor

3.2. Định nghĩa lớp trong Java

Documentation comment



- Công dụng: ghi chú cho mã nguồn Java
- Cú pháp:
- Ví dụ:

```
/**  
 * <nội dung ghi chú>  
 * @param args  
 */
```

```
15- /**  
16   * This is an example of a documentation of a method with a return type.  
17   * Note that there isn't a type on the `@return` tag  
18   *  
19   * @return the someAttribute  
20   */  
21- public String getSomeAttribute() {  
22   return someAttribute;  
23 }  
24  
25- /**  
26   * @param someAttribute  
27   */  
28- public void setSomeAttribute(String someAttribute) {  
29   this.someAttribute = someAttribute;  
30 }  
31  
32 }
```

String Foo.getSomeAttribute()

This is an example of a documentation of a method with a return type. Note that there isn't a type on the `@return` tag

Returns:

the someAttribute

Press 'F2' for focus

3.2. Định nghĩa lớp trong Java



- Cài đặt bài hoàn chỉnh cho lớp HìnhChuNhat theo mô hình sau:

class Class Model

HinhChuNhat
- cDai: float - cRong: float
+ tinhChuVi(): float + tinhDienTich(): float + toSring(): String «constructor» + HinhChuNhat() + HinhChuNhat(float, float) «property get» + getChieuDai(): float + getChieuRong(): float «property set» + setChieuDai(float): void + setChieuRong(float): void

3.2. Định nghĩa lớp trong Java

Review questions 1



1. Quy tắc đặt tên lớp, thuộc tính, phương thức.
2. Dùng từ khóa gì để khai báo đối tượng.
3. Thể hiện tính đóng gói trong OOP bằng cách nào.
4. Phương thức khởi tạo dùng để làm gì, có đặc điểm gì. Một lớp có thể có bao nhiêu phương thức khởi tạo.
5. Có thể gán giá trị cho một đối tượng bằng các cách nào.
6. Từ khóa *this* dùng để làm gì.

3.2. Định nghĩa lớp trong Java

Case study 1



1. Vẽ mô hình và nêu các ràng buộc cho lớp hình tròn (**Circle**), biết:
 - Các thuộc tính: **radius** (kiểu double, **mặc định** là 1), **color** (kiểu String, **mặc định** là "red").
 - Các phương thức get/set với các **ràng buộc dữ liệu**:
 - radius phải ≥ 0 (không thì gán giá trị mặc định)
 - color không được rỗng (nếu rỗng thì gán giá trị mặc định).
 - Các phương thức khởi tạo: 1 constructor mặc định, 1 constructor khởi tạo cho radius, 1 constructor đầy đủ tham số. Có **kiểm tra ràng buộc** trong constructor.
 - Phương thức toString() trả về thông tin của hình tròn.
 - Phương thức tính: diện tích hình tròn.
2. Cài đặt cho các phương thức khởi tạo, các getter/setter, phương thức toString, phương thức tính diện tích hình tròn.

3.2. Định nghĩa lớp trong Java

Kiểm tra ràng buộc dữ liệu



- Trong một class, khi gán giá trị cho thuộc tính (trong các **constructor**, các **setter**), cần phải **kiểm tra dữ liệu gán có hợp lệ không** (ràng buộc dữ liệu).
- Nếu dữ liệu muốn gán không thỏa thì giải quyết bằng 1 trong 2 cách:
 - Gán giá trị mặc định HOẶC
 - Ném lỗi ra bên ngoài (phát sinh ngoại lệ)

3.2. Định nghĩa lớp trong Java

Kiểm tra RB dữ liệu – Gán default [1]



- Ví dụ 1: Kiểm tra ràng buộc cho lớp Circle:
 - radius phải ≥ 0 , nếu radius < 0 thì gán giá trị mặc định (bằng 1).
 - color không được rỗng, nếu rỗng thì gán giá trị mặc định (là “red”).

3.2. Định nghĩa lớp trong Java

Kiểm tra RB dữ liệu – Gán default [2]



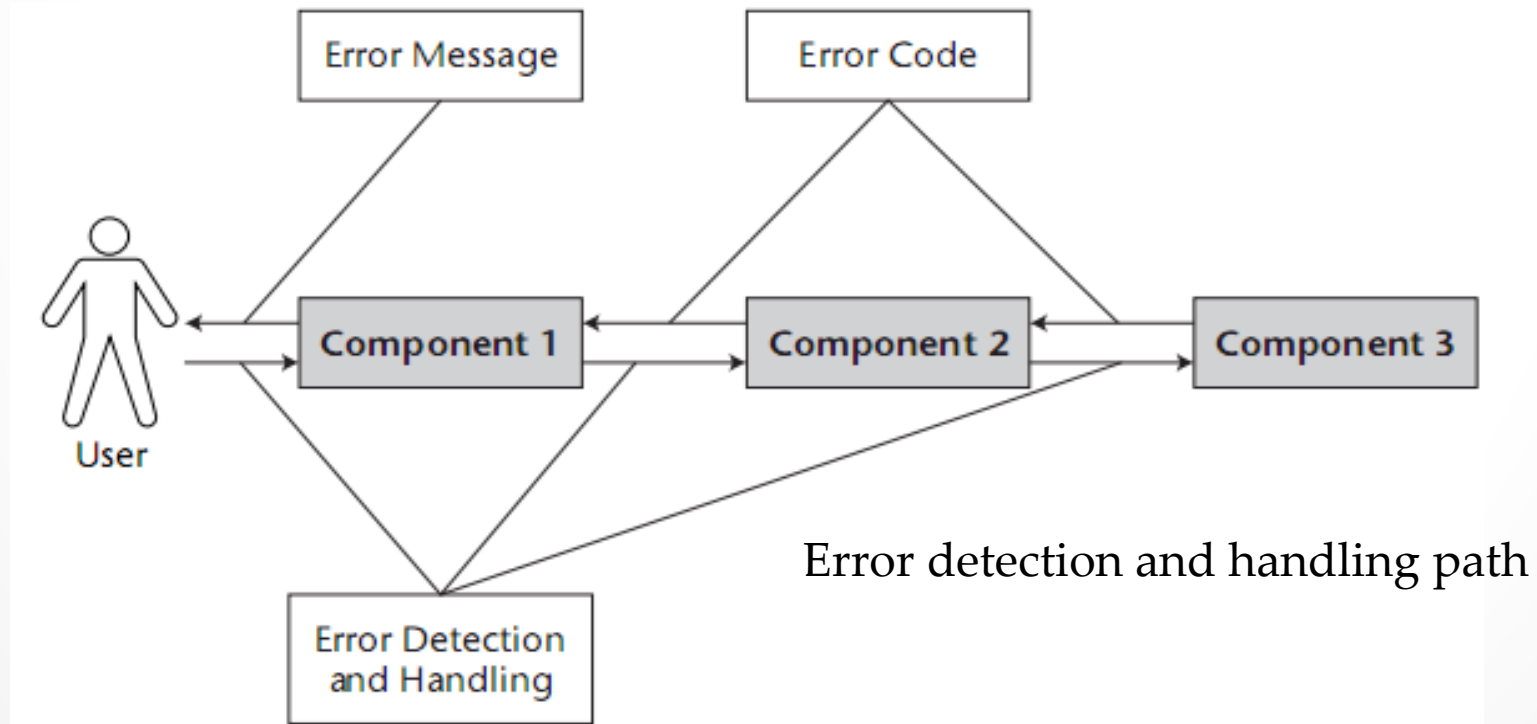
- Ví dụ 1: Solution

3.2. Định nghĩa lớp trong Java

Kiểm tra RB dữ liệu – Ném lỗi [1]



- Bước 1. Dùng `throws` để ném lỗi
- Bước 2. Dùng `try/catch` để bắt lỗi



3.2. Định nghĩa lớp trong Java

Kiểm tra RB dữ liệu – Ném lỗi [2]



- Ví dụ 2: Dùng Exception để kiểm tra ràng buộc cho lớp Circle:
 - Nếu $\text{radius} < 0$ thì ném lỗi “Radius phải > 0 ”
 - Nếu color là rỗng thì ném lỗi “Color không được rỗng”

3.2. Định nghĩa lớp trong Java

Kiểm tra RB dữ liệu – Ném lỗi [3]



- Ví dụ 2: Solution

3.2. Định nghĩa lớp trong Java

Kiểm thử



- Bước 1. Lập bảng các dữ liệu cần kiểm thử (Xác định các test case)
- Bước 2. Cài đặt và thực thi với các dữ liệu này
- Bước 3. Báo cáo kết quả kiểm thử

3.2. Định nghĩa lớp trong Java

Kiểm thử - B1. Xác định các test case



- Lập bảng

STT	Tên test case	Tên phương thức	Giá trị nhập	Kết quả mong đợi

Chú ý: Các test case không nên trùng nhau

Ví dụ: Lập bảng test case để test lớp Circle cho trường hợp thỏa ràng buộc (tất cả thuộc tính) và không thỏa ràng buộc (cho mỗi thuộc tính)

#	Tên phương thức	Tên test case	Giá trị nhập	Kết quả mong đợi
1	Circle(radius,color)	radius và color thỏa rb	radius=5, color="blue"	radius=5, color="blue"
2	Circle(radius,color)	radius không thỏa rb	radius=-5, color="blue"	radius=1, color="blue"
3	Circle(radius,color)	color không thỏa rb	radius=5, color=""	radius=5, color="red"
...				

3.2. Định nghĩa lớp trong Java

Kiểm thử - B2. Cài đặt và thực thi



- Ví dụ

//Dữ liệu trong từng hàm test phải khớp với dữ liệu trong bảng test case (file excel)
//Ví dụ: Test1 là để test cho test case #1 thì phải lấy dữ liệu của test case #1

```
public static boolean Test1() {  
    Circle c = new Circle(5, "blue");  
    double radius_exp = 5;  
    String color_exp = "blue";  
    if (c.getRadius() == radius_exp && c.getColor() == color_exp) {  
        return true;  
    } else {  
        System.out.println("Expected: radius = " + radius_exp +  
                             ", color = " + color_exp);  
        System.out.println("Actual: radius = " + c.getRadius() +  
                             ", color = " + c.getColor());  
        return false;  
    }  
}
```

3.2. Định nghĩa lớp trong Java

Kiểm thử - Báo cáo kết quả test



- Điền kết quả thực thi vào bảng test case

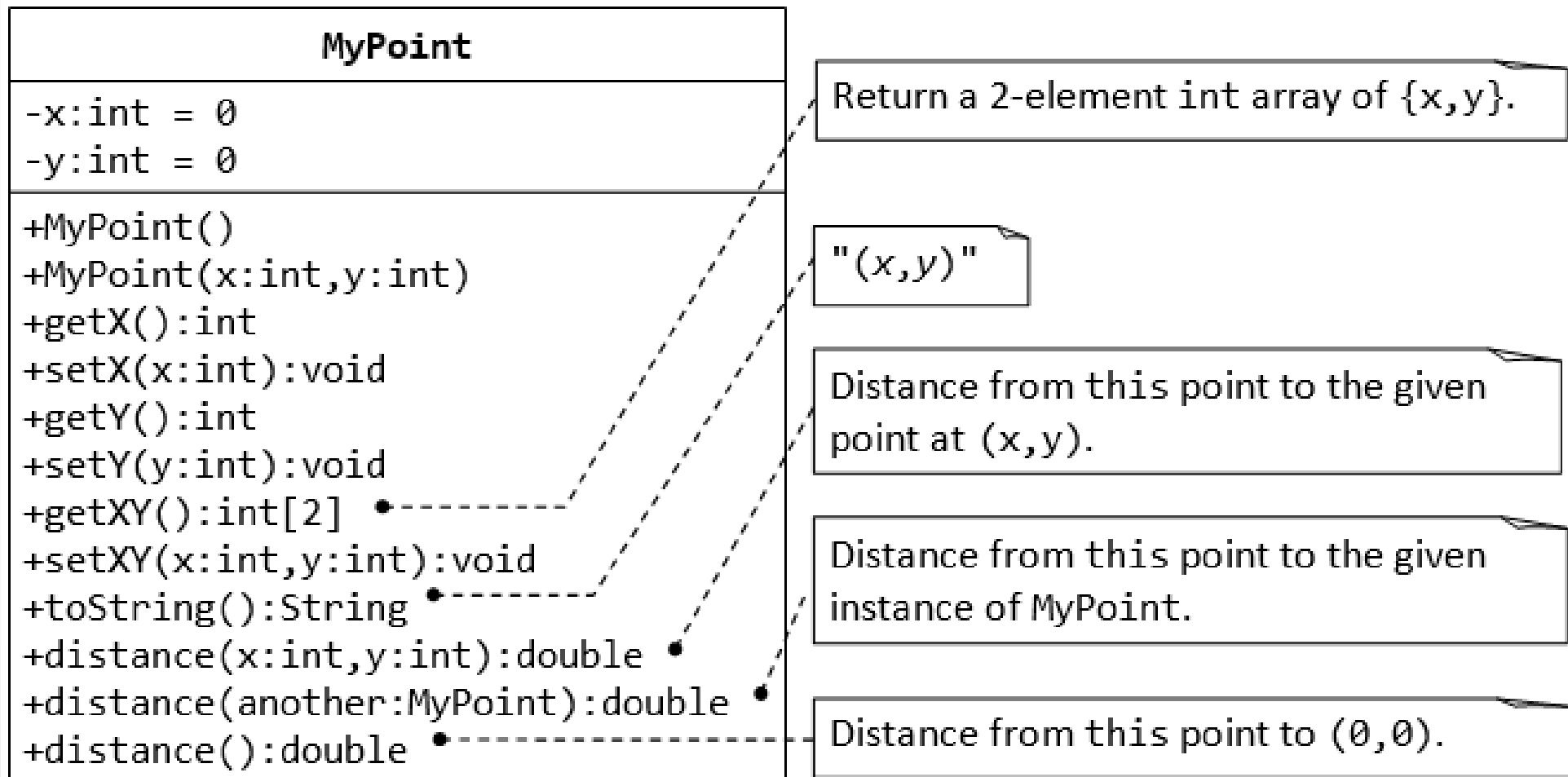
#	Tên test case	Tên phương thức	Giá trị nhập	Kết quả mong đợi	Kết quả thực tế	Trạng thái	Ghi chú
1	Circle(radius,color)	radius và color thỏa rb	radius=5, color="blue"	radius=5, color="blue"			
2	Circle(radius,color)	radius không thỏa rb	radius=-5, color="blue"	radius=1, color="blue"			
3	Circle(radius,color)	color không thỏa rb	radius=5, color=""	radius=5, color="red"			

3.2. Định nghĩa lớp trong Java

Case study 2



- Cài đặt cho mô hình lớp sau:



3.2. Định nghĩa lớp trong Java

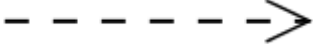



Bài tập



- Module 2

3.3. Một số mối quan hệ giữa các lớp



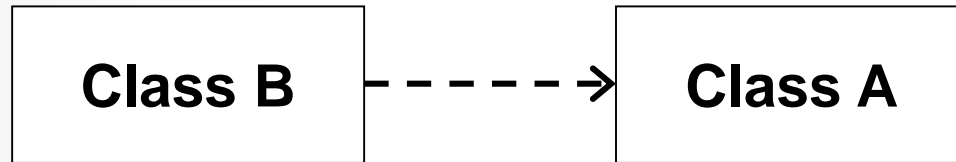
- Dependency 
- Simple association 
- Aggregation 
- Composition 

3.3. Một số mối quan hệ giữa các lớp

Dependency [1]



- Là mối quan hệ “uses”:



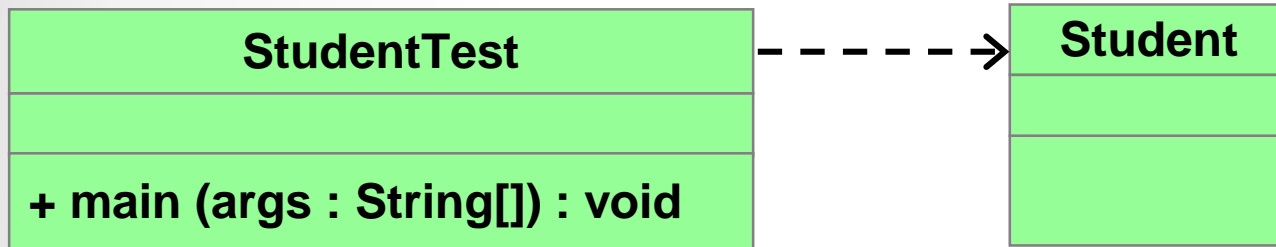
- Lớp B sử dụng lớp A như biến cục bộ
 - Lớp B có một phương thức mà tham số là đối tượng lớp A
 - Lớp B gọi các phương thức “static” của lớp A
- Lớp A thay đổi có thể ảnh hưởng đến lớp B.

3.3. Một số mối quan hệ giữa các lớp

Dependency [2]



- Ví dụ 1: Lớp **StudentTest** sử dụng lớp **Student** như biến cục bộ



```
public class Student {  
    //...  
}
```

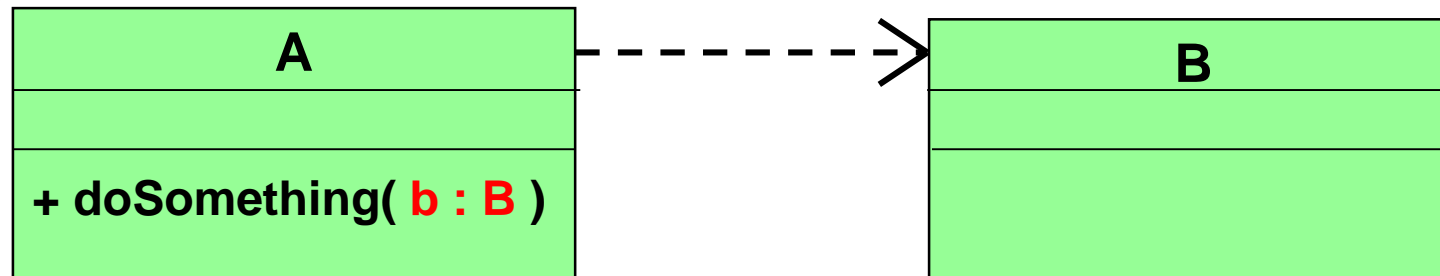
```
public class StudentTest {  
    public static void main(String[] args) {  
        Student student = new Student();  
        //...  
    }  
}
```

3.3. Một số mối quan hệ giữa các lớp

Dependency [3]



- Ví dụ 2: Lớp A có một phương thức mà tham số là đối tượng của lớp B.



```
public class A {
    public void doSomething(B b) {
    }
}
```

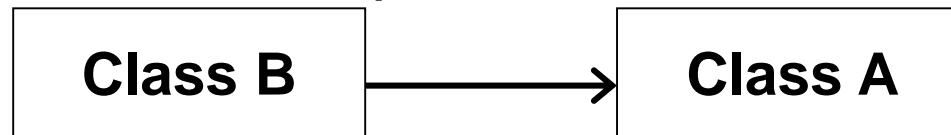
```
public class B {
}
```

3.3. Một số mối quan hệ giữa các lớp

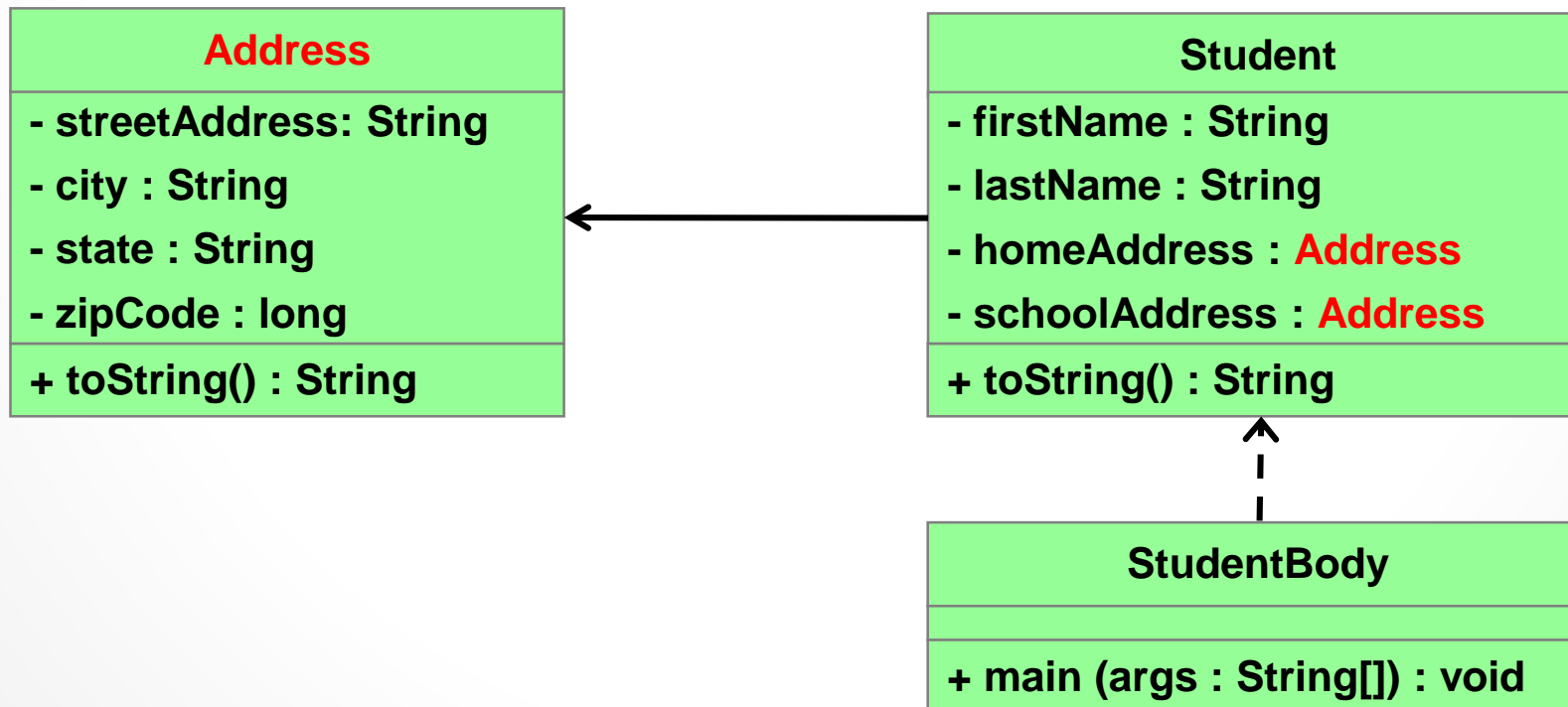
Simple association



- Là mối quan hệ “uses”: Lớp B có thuộc tính có kiểu là lớp A.



- Ví dụ:

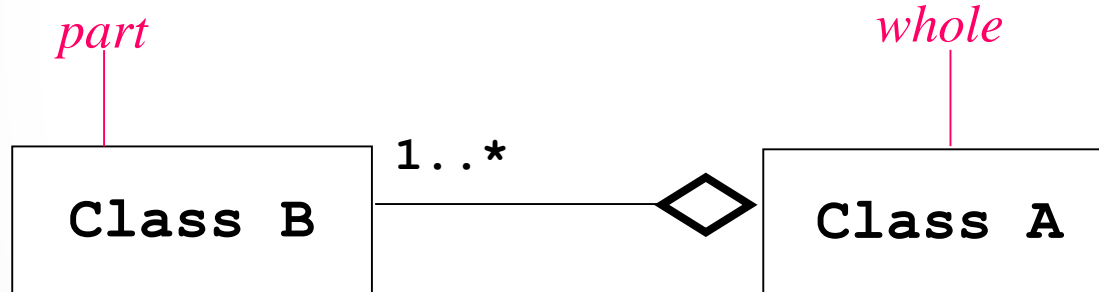


3.3. Một số mối quan hệ giữa các lớp

Aggregation [1]



- Là mối quan hệ “whole/part”, “has-a”.
- Các đối tượng của lớp B thuộc về các đối tượng của lớp A, và các đối tượng của lớp B vẫn tồn tại độc lập với lớp A.



- Ví dụ:



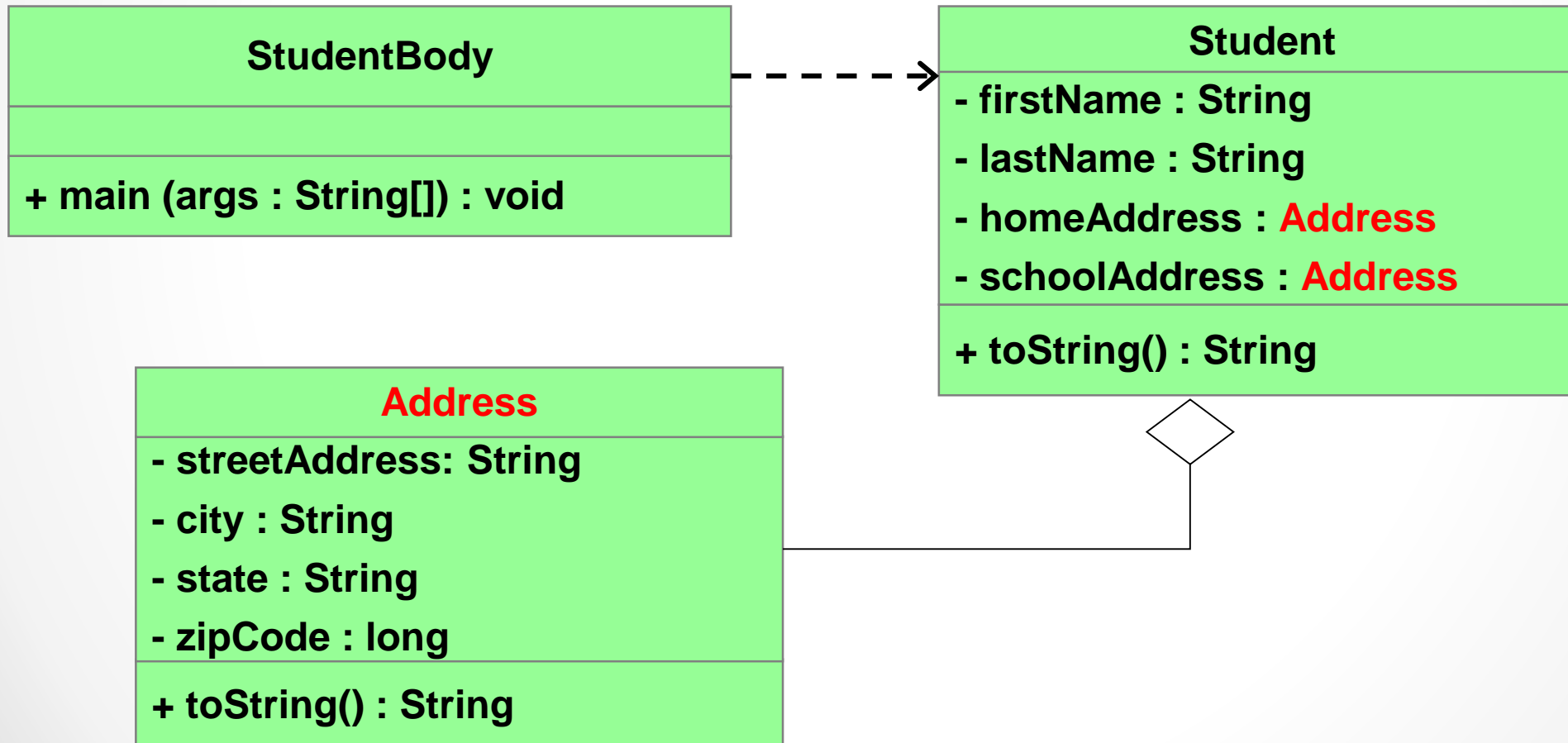
```
public class A {  
    private B bb;  
    public void setB(B b) {  
        bb = b;  
    }  
    ...  
}
```


3.3. Một số mối quan hệ giữa các lớp

Aggregation [2]



- Ví dụ:

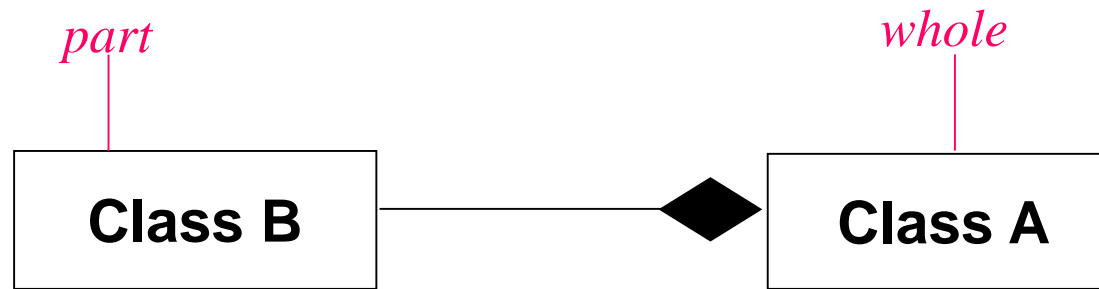


3.3. Một số mối quan hệ giữa các lớp

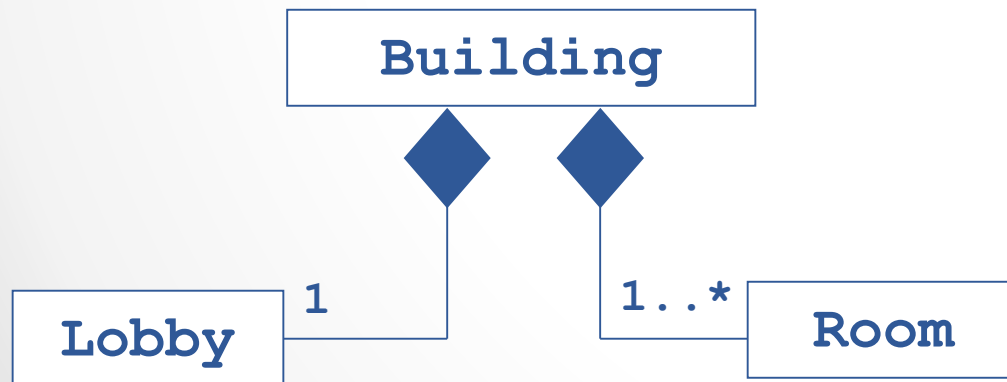
Composition [1]



- Là mối quan hệ “whole/part” giống Aggregation nhưng nếu phần “whole” bị xóa thì tất cả phần “part” của nó cũng sẽ không còn.



- Ví dụ:



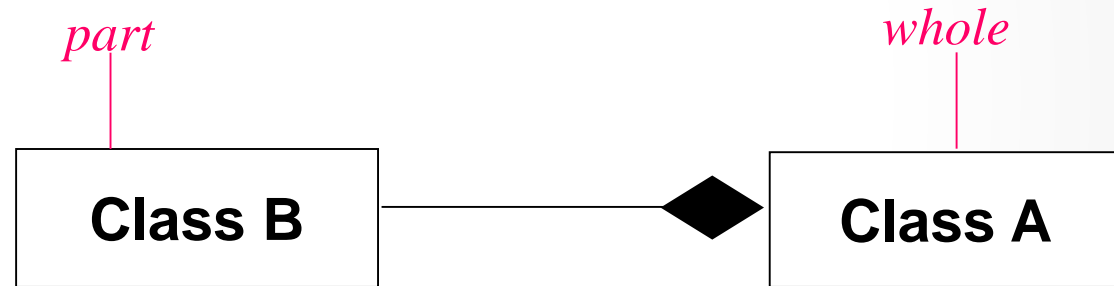
3.3. Một số mối quan hệ giữa các lớp

Composition [2]



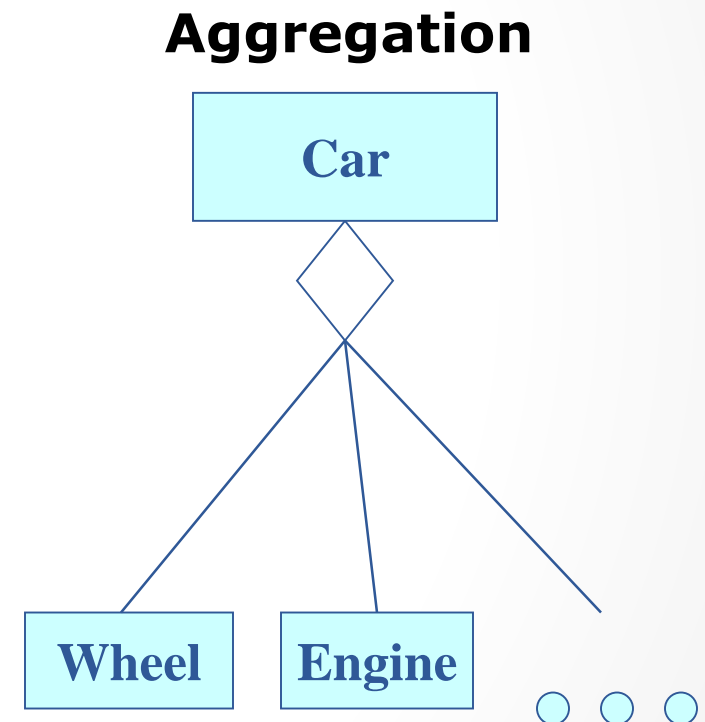
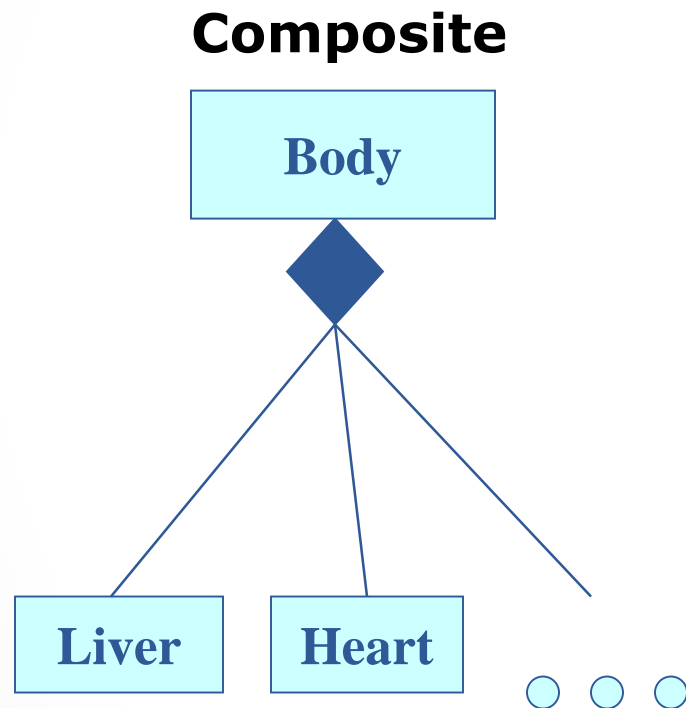
- Lớp “whole” phải chịu trách nhiệm tạo ra các đối tượng “part”.
- Ví dụ:

```
public class A {  
    private B bb;  
    public A() {  
        bb = new B();  
    }  
}
```



3.3. Một số mối quan hệ giữa các lớp

Composition - Aggregation



Composite is a stronger form of aggregation. Composite parts live and die with the whole.

Book

-name:String
-author:Author
-price:double
-qty:int = 0

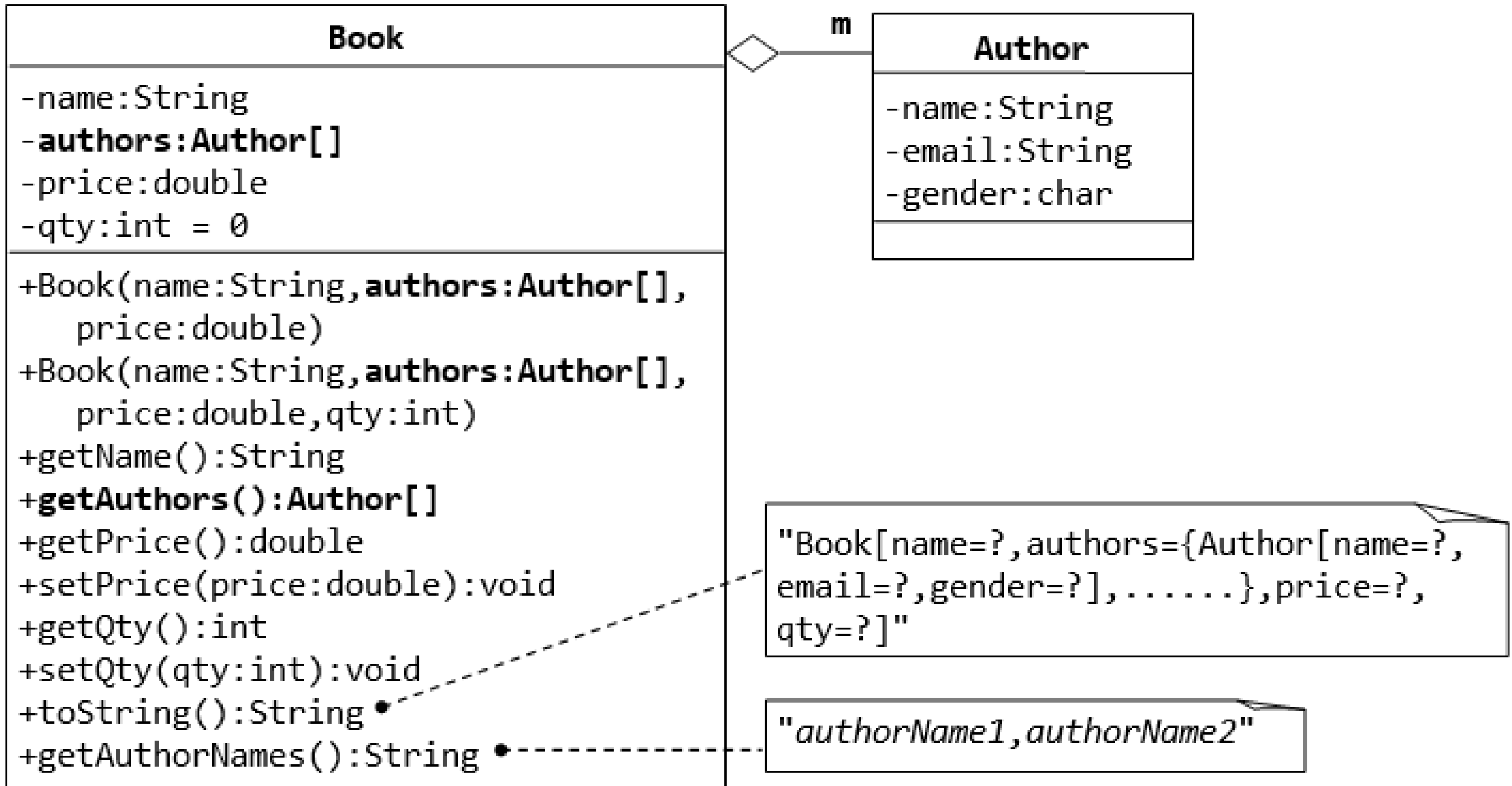
+Book(name:String,author:Author,
price:double)
+Book(name:String,author:Author,
price:double,qty:int)
+getName():String
+getAuthor():Author
+getPrice():double
+setPrice(price:double):void
+getQty():int
+setQty(qty:int):void
+toString():String

Author

-name:String
-email:String
-gender:char



"Book[name=?,Author[name=?,email=?,gender=?],price=?,qty=?]"
You need to reuse Author's toString().

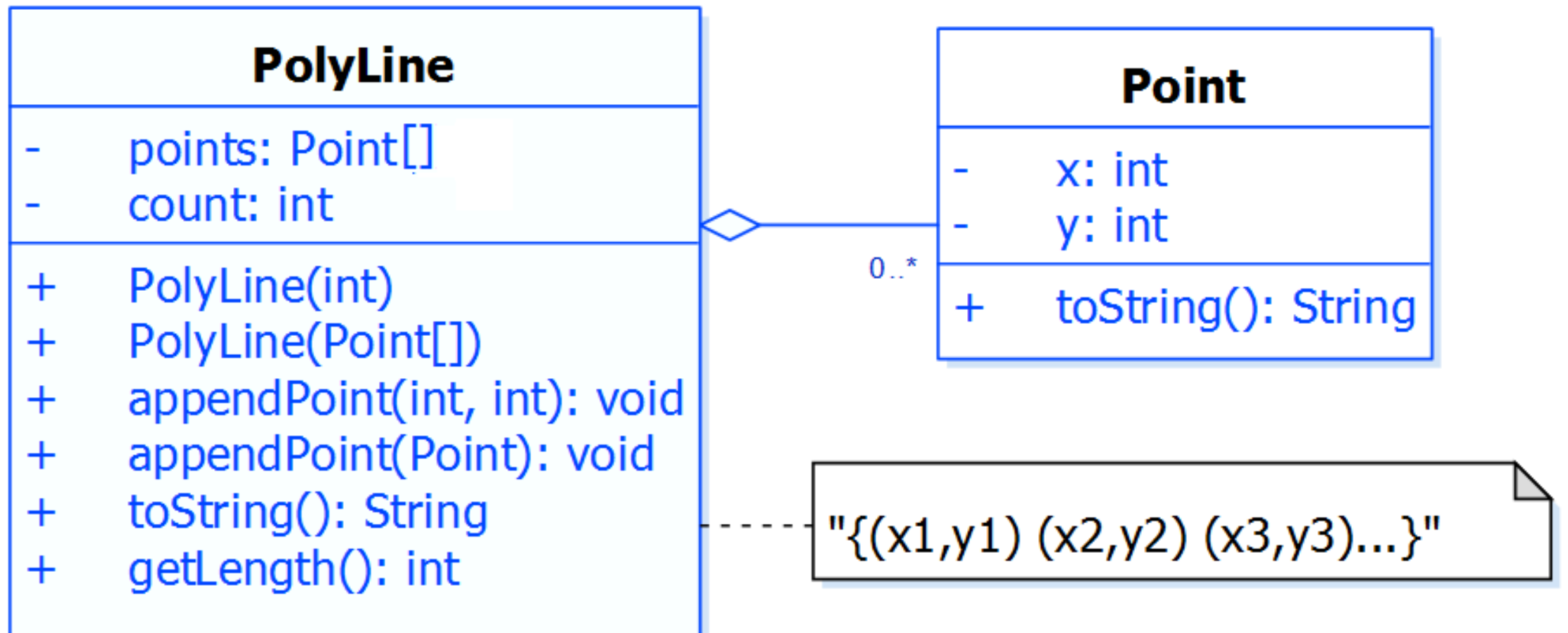


3.3. Một số mối quan hệ giữa các lớp

Case study 3

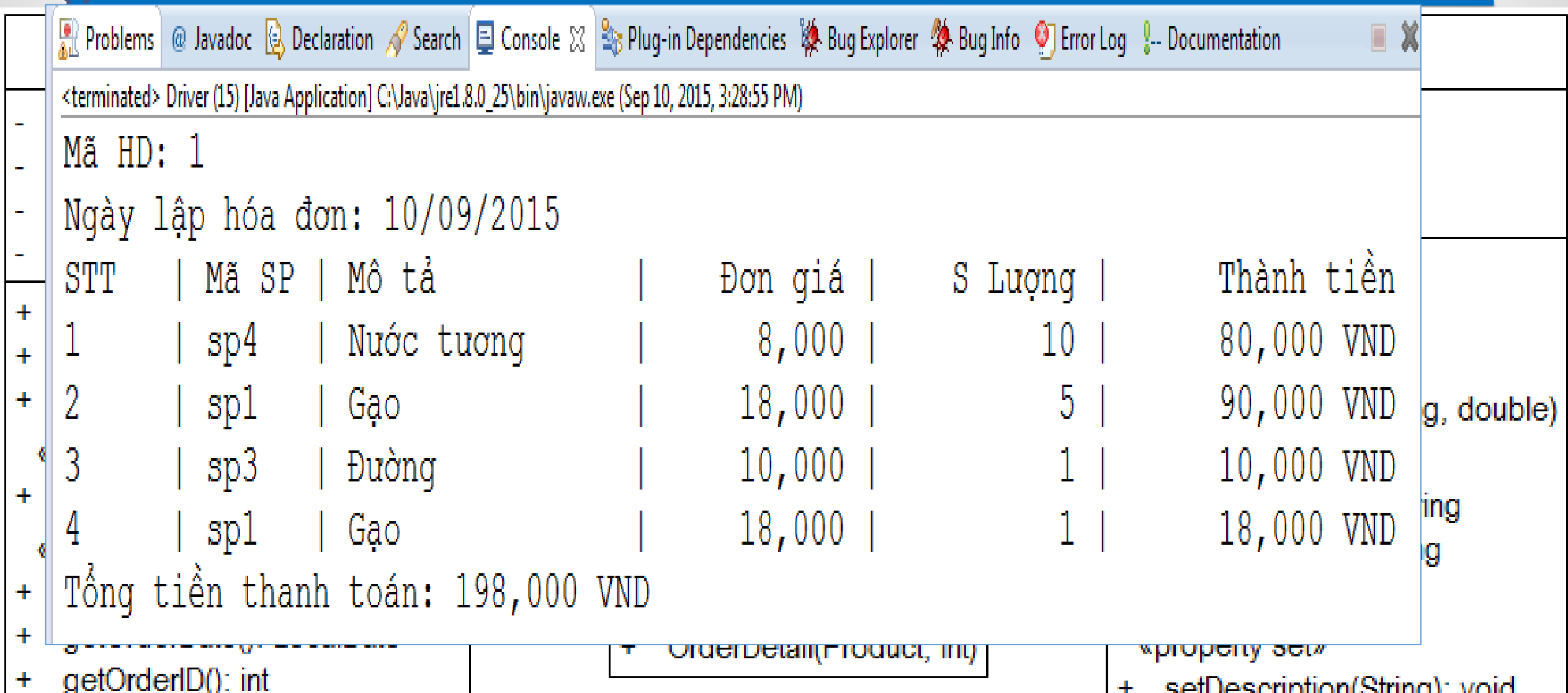


- Cài đặt cho mô hình lớp sau và viết hàm main để test lớp PolyLine:



3.3. Một số mối quan hệ giữa các lớp

Case study 4



The screenshot shows an IDE window with a console output. The console title is "<terminated> Driver (15) [Java Application] C:\Java\jre1.8.0_25\bin\javaw.exe (Sep 10, 2015, 3:28:55 PM)". The output text is as follows:

```
Mã HD: 1
Ngày lập hóa đơn: 10/09/2015
```

STT	Mã SP	Mô tả	Đơn giá	S Lượng	Thành tiền
1	sp4	Nước tương	8,000	10	80,000 VND
2	sp1	Gạo	18,000	5	90,000 VND
3	sp3	Đường	10,000	1	10,000 VND
4	sp1	Gạo	18,000	1	18,000 VND

Tổng tiền thanh toán: 198,000 VND

Below the console output, a snippet of Java code is visible, showing a method signature:

```
+ getOrderID(): int
```

On the right side of the IDE, a partial view of a class structure is visible, showing a method signature:

```
+ setDescription(String): void
```






Review questions 2

1. Làm sao để kiểm tra ràng buộc dữ liệu.
2. Các bước kiểm thử cơ bản. Lập bảng test case cần chú ý gì. Báo cáo kiểm thử có thể gồm những gì.
3. Những trường hợp nào có mối quan hệ dependency.
4. Có những mối quan hệ “whole-parts” nào, chúng khác nhau ra sao.