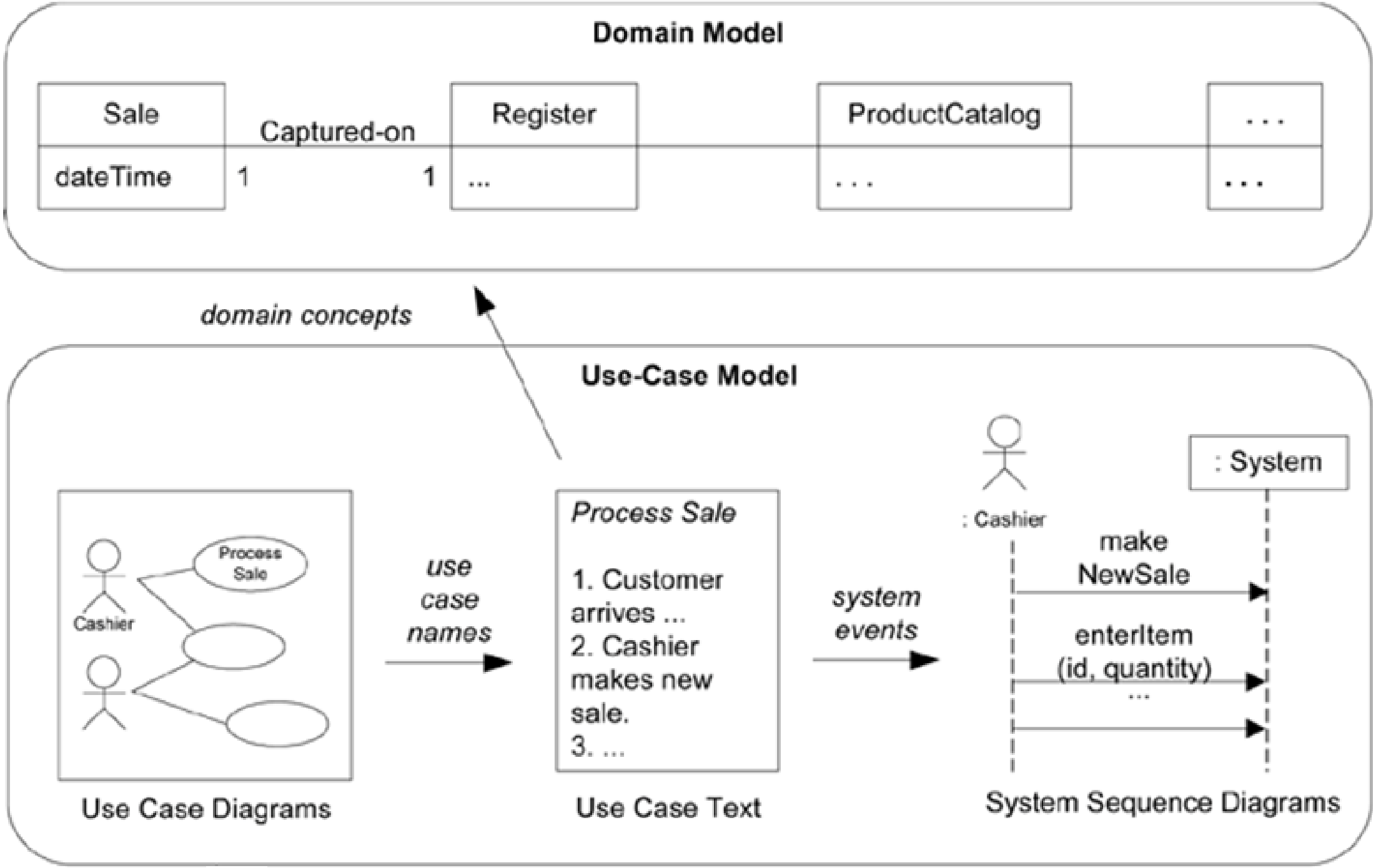
THIẾT KẾ PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

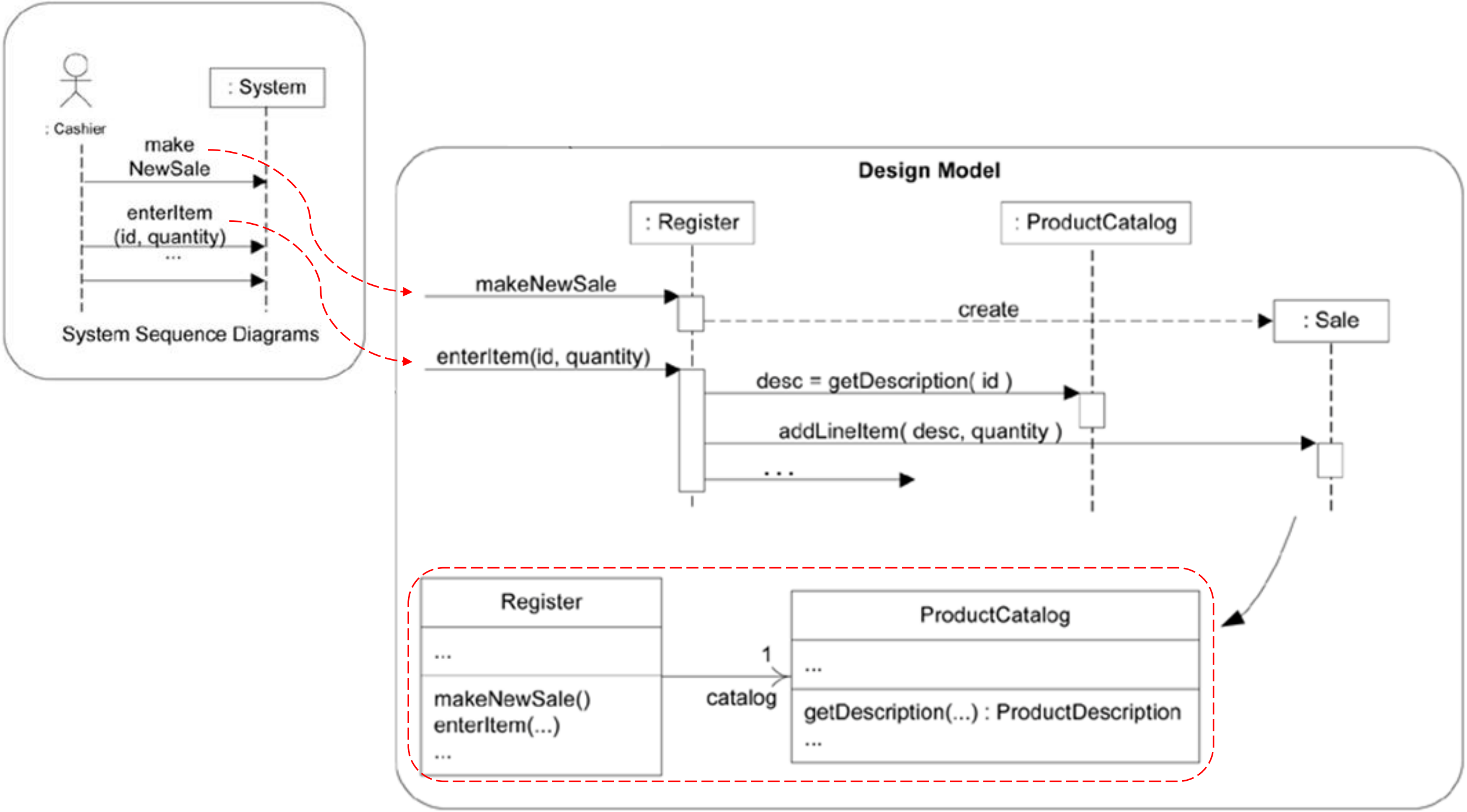
Design Class Diagrams

Gv: Nguyễn Thị Thanh

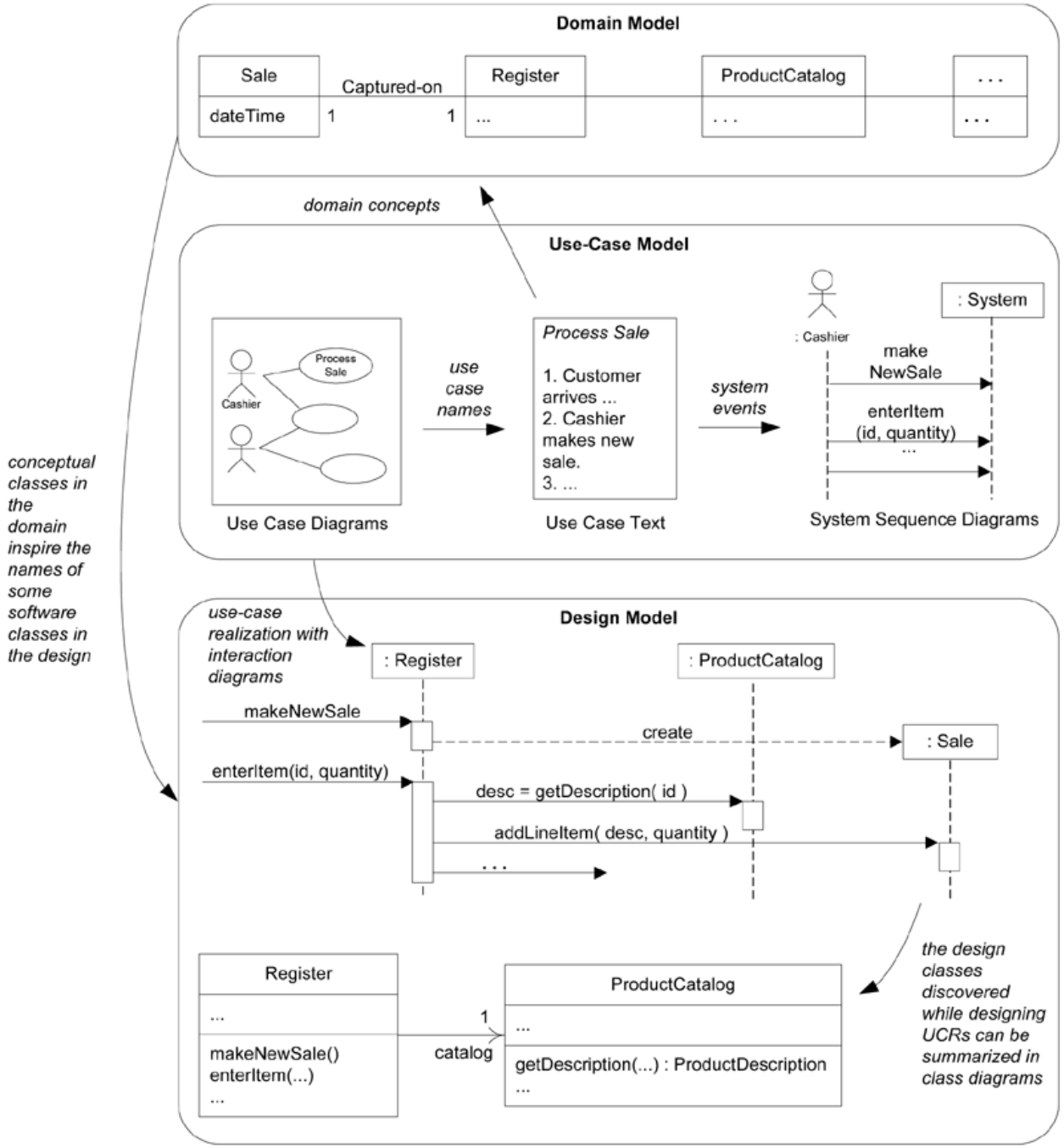
Nội dung

1. So sánh Domain Model và Design Class Diagrams (DCD)
2. Các bước xây dựng DCD
3. Sử dụng công cụ StarUML để vẽ DCD
4. Phát sinh source code tự động từ lược đồ lớp

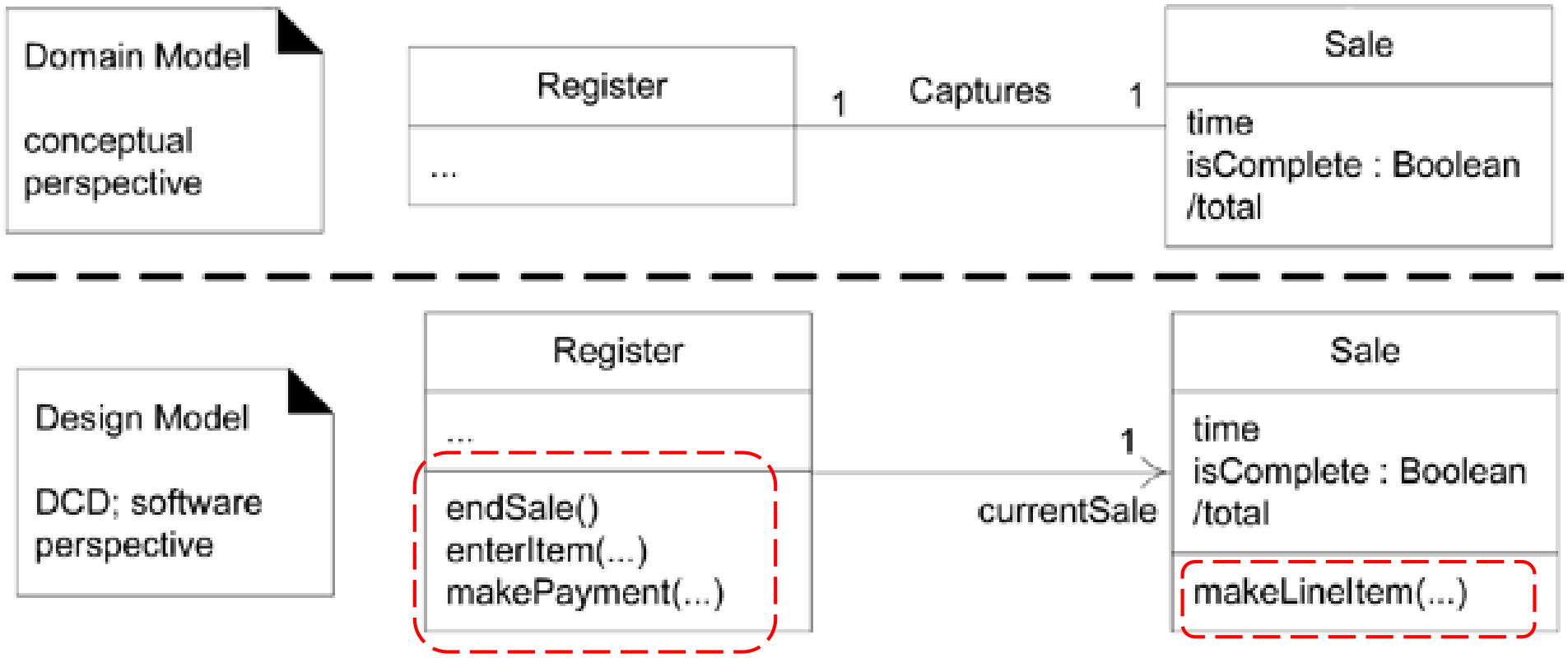

Domain Model
Sale
dateTime
Captured-on
1
1
Register
...
ProductCatalog
. . .
. . .
. . .
domain concepts
Use-Case Model
Process Sale
Cashier
use case names
Process Sale
1. Customer arrives ...
2. Cashier makes new sale.
3. ...
system events
: Cashier
: System
make NewSale
enterItem (id, quantity)
...
Use Case Diagrams
Use Case Text
System Sequence Diagrams


: Cashier
: System
make
NewSale
enterItem
(id, quantity)
...
System Sequence Diagrams
Design Model
: Register
: ProductCatalog
makeNewSale
create
: Sale
enterItem(id, quantity)
desc = getDescription(id)
addLineItem(desc, quantity)
. . .
Register
...
makeNewSale()
enterItem(...)
...
1
catalog
ProductCatalog
...
getDescription(...) : ProductDescription
...

Sample Unified Process Artifact Relationships



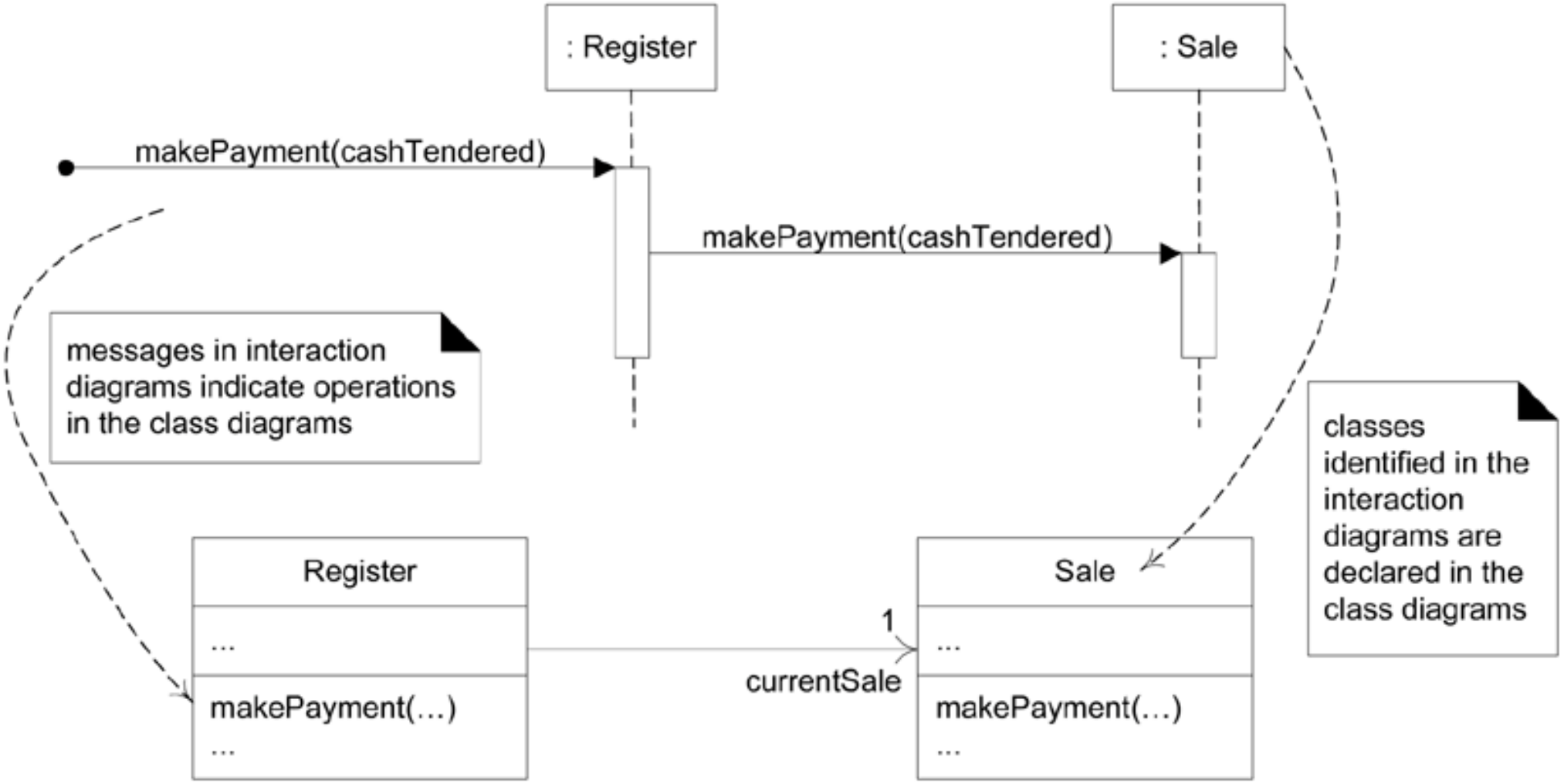
Domain Model và DCD



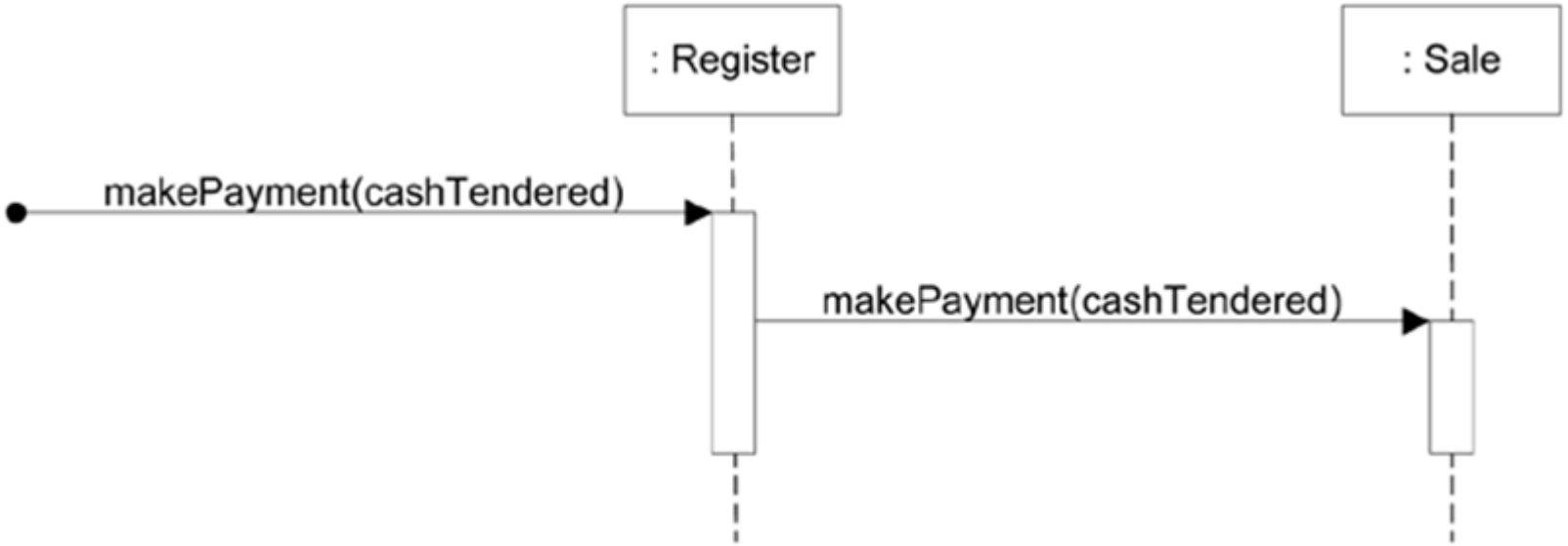
Vẽ DCD

1. Tìm Software Classes trong tất cả các lược đồ trình tự và lược đồ cộng tác.
2. Thêm các thuộc tính đã định nghĩa trong Domain Model.
3. Thêm phương thức từ lược đồ tương tác.
4. Thêm mối quan hệ giữa 2 lớp.

Ví dụ


: Register
: Sale
makePayment(cashTendered)
makePayment(cashTendered)
messages in interaction diagrams indicate operations in the class diagrams
classes identified in the interaction diagrams are declared in the class diagrams
Register
...
makePayment(...)
...
1
currentSale
Sale
...
makePayment(...)
...

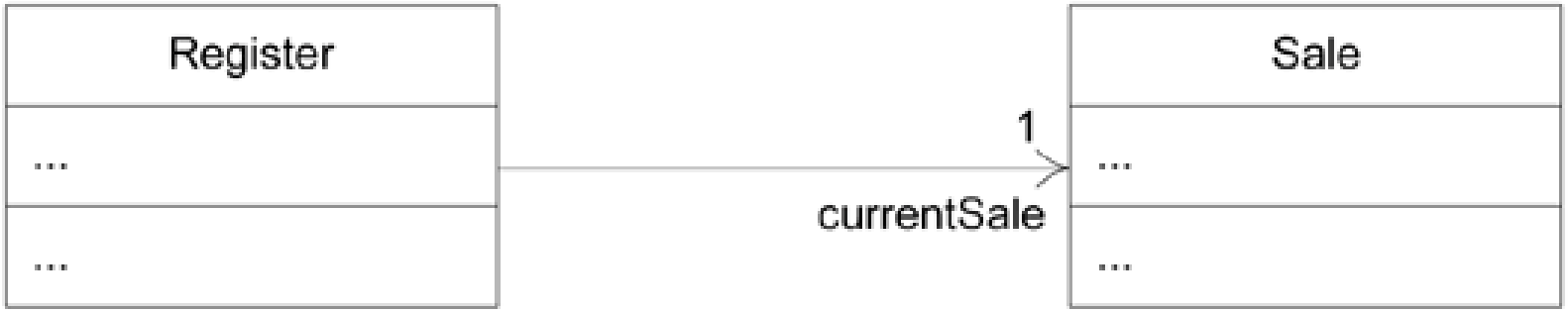
Quan hệ giữa 2 lớp




```
public class Register {
    Sale currentSale = new Sale();

    public void makePayment(cashTendered){
        currentSale.makePayment(cashTendered);
    }
}
```

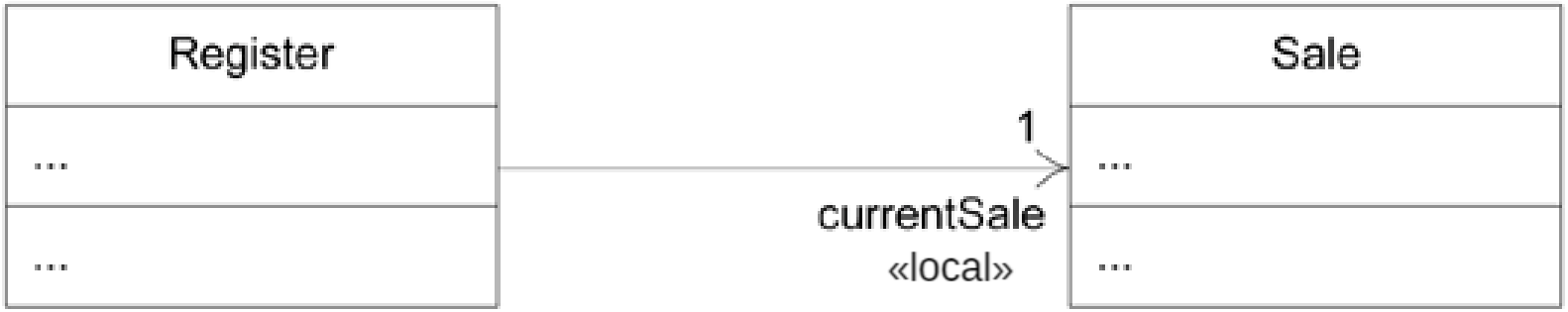
```
public class Register {
    Sale currentSale = new Sale();

    public void makePayment(cashTendered){
        currentSale.makePayment(cashTendered);
    }
}
```

Register có chứa 1 tham khảo (reference) của Sale

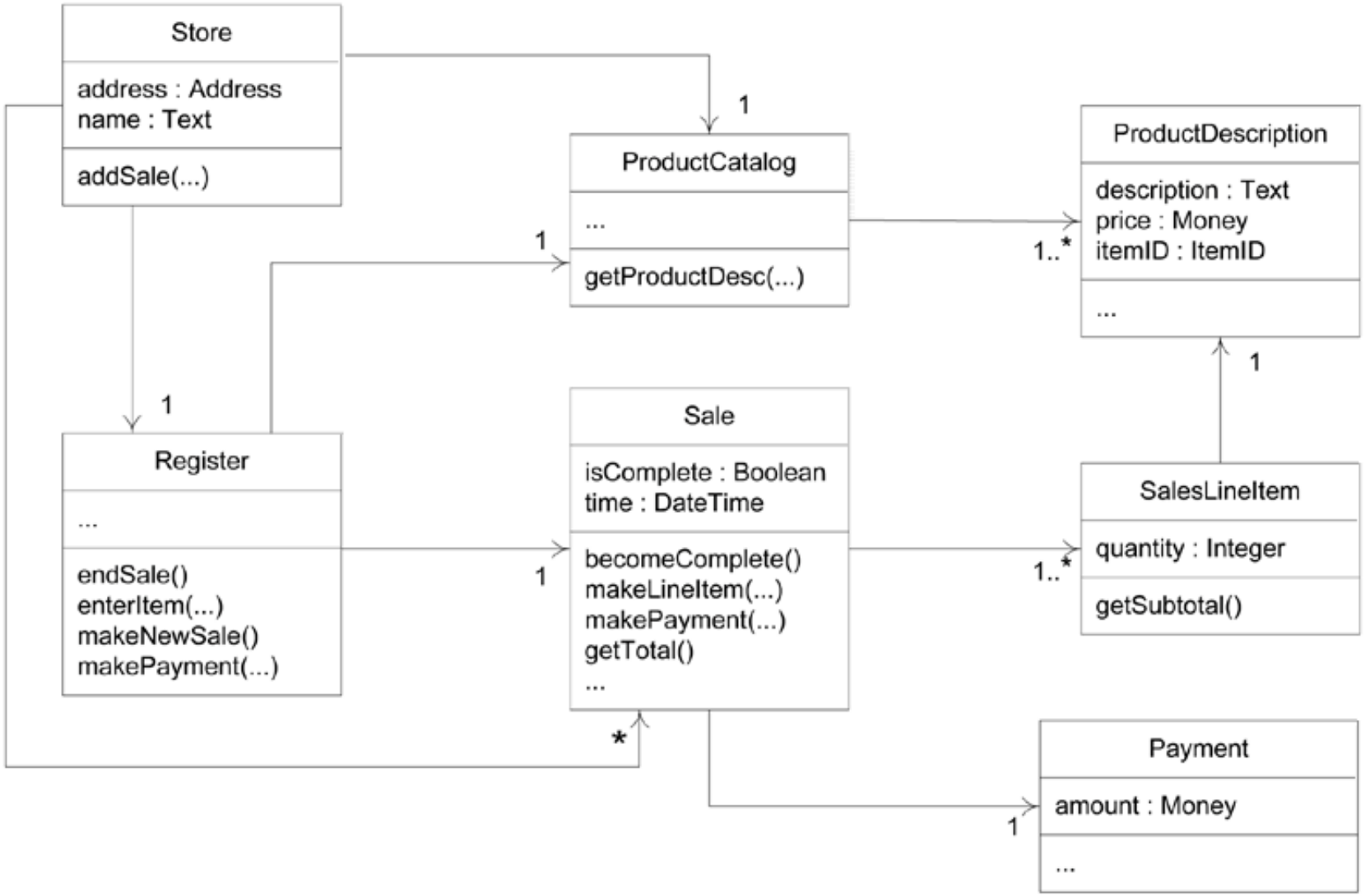


```
public class Register {
    ~~Sale currentSale = new Sale();~~
    public void makePayment(cashTendered){
        Sale currentSale = new Sale();
        currentSale.makePayment(cashTendered);
    }
}
```

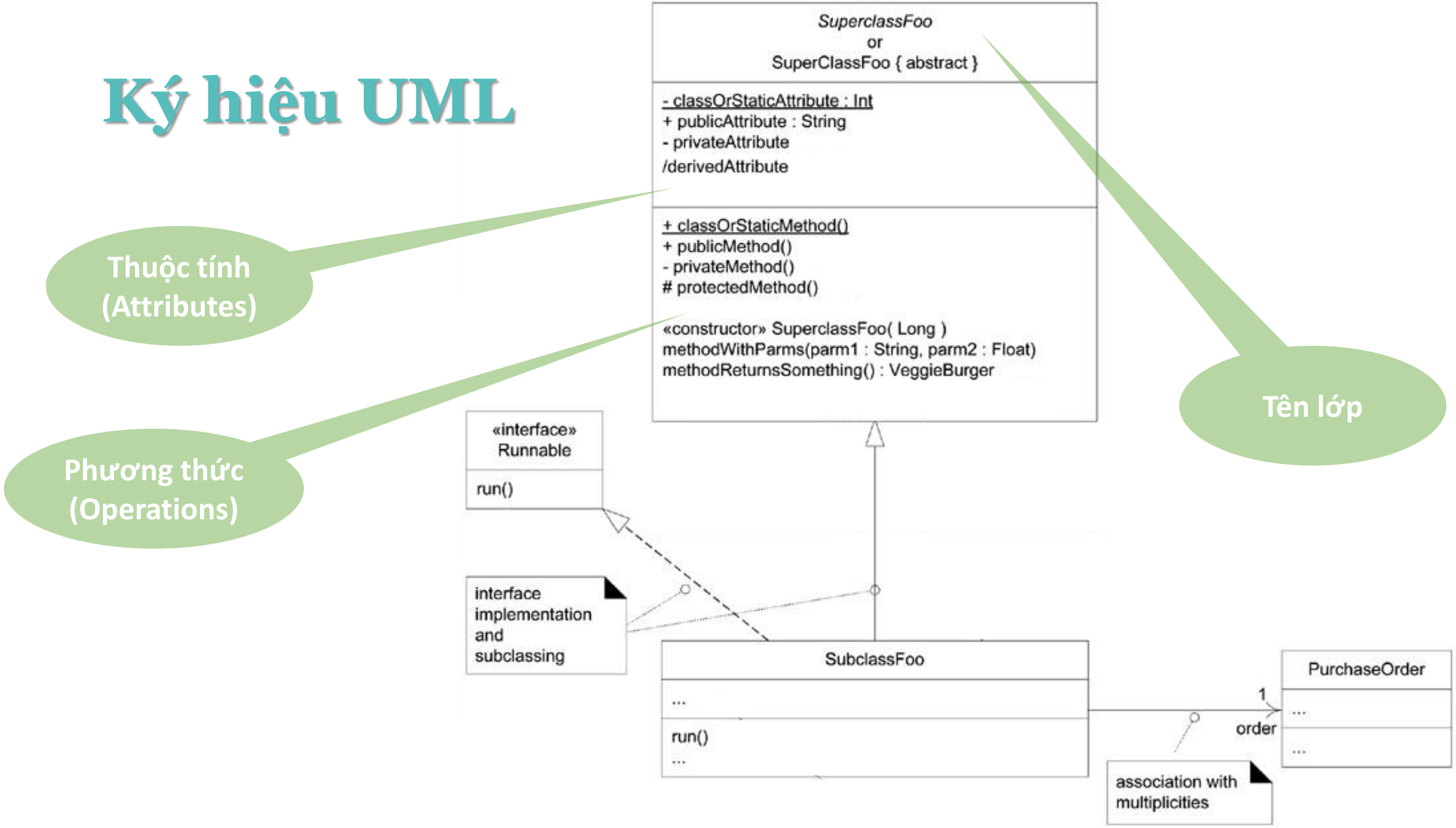
Register có chứa 1 tham khảo (reference) của Sale



DCD - POS system

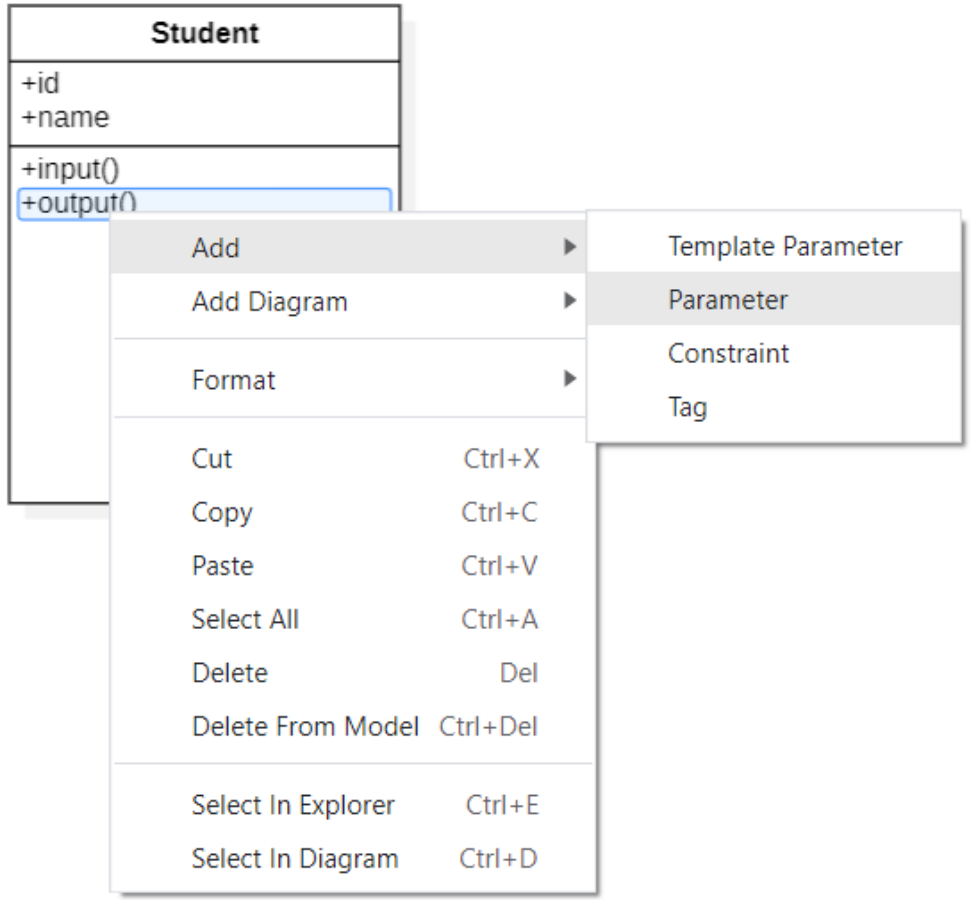


Ký hiệu UML


Thuộc tính (Attributes)
Phương thức (Operations)
Tên lớp
SuperclassFoo
or
SuperClassFoo { abstract }
- classOrStaticAttribute : Int
+ publicAttribute : String
- privateAttribute
/derivedAttribute
+ classOrStaticMethod()
+ publicMethod()
- privateMethod()
protectedMethod()
«constructor» SuperclassFoo(Long)
methodWithParms(parm1 : String, parm2 : Float)
methodReturnsSomething() : VeggieBurger
«interface»
Runnable
run()
interface implementation and subclassing
SubclassFoo
...
run()
...
PurchaseOrder
...
...
1
order
association with multiplicities

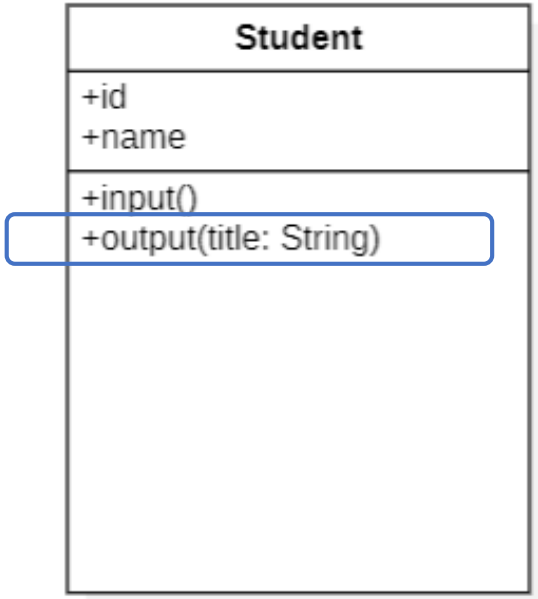
StarUML Guidelines (1)

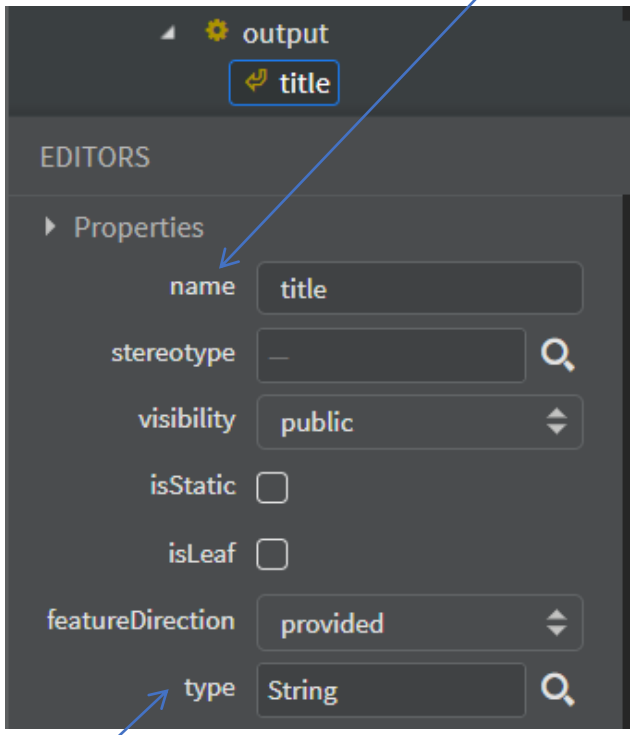
- Add parameter of method



StarUML Guidelines (2)

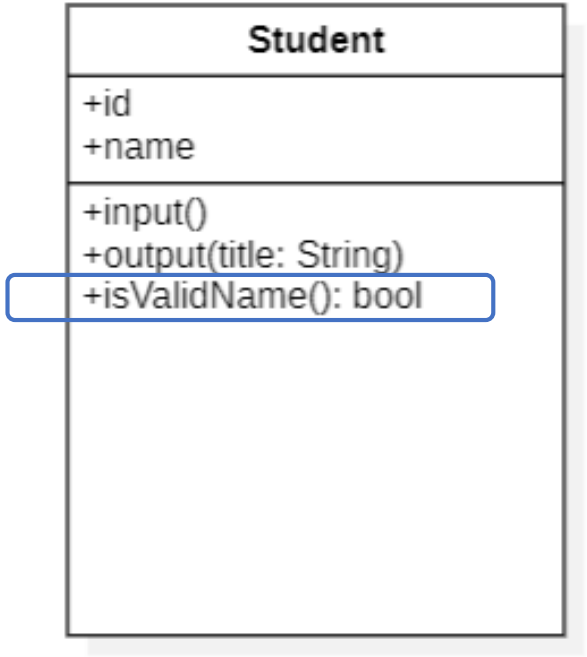
- Add type of parameter

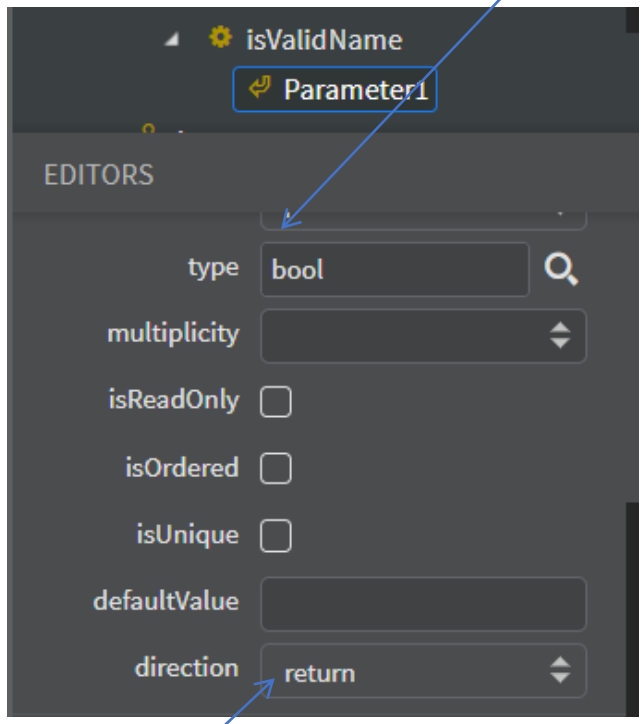




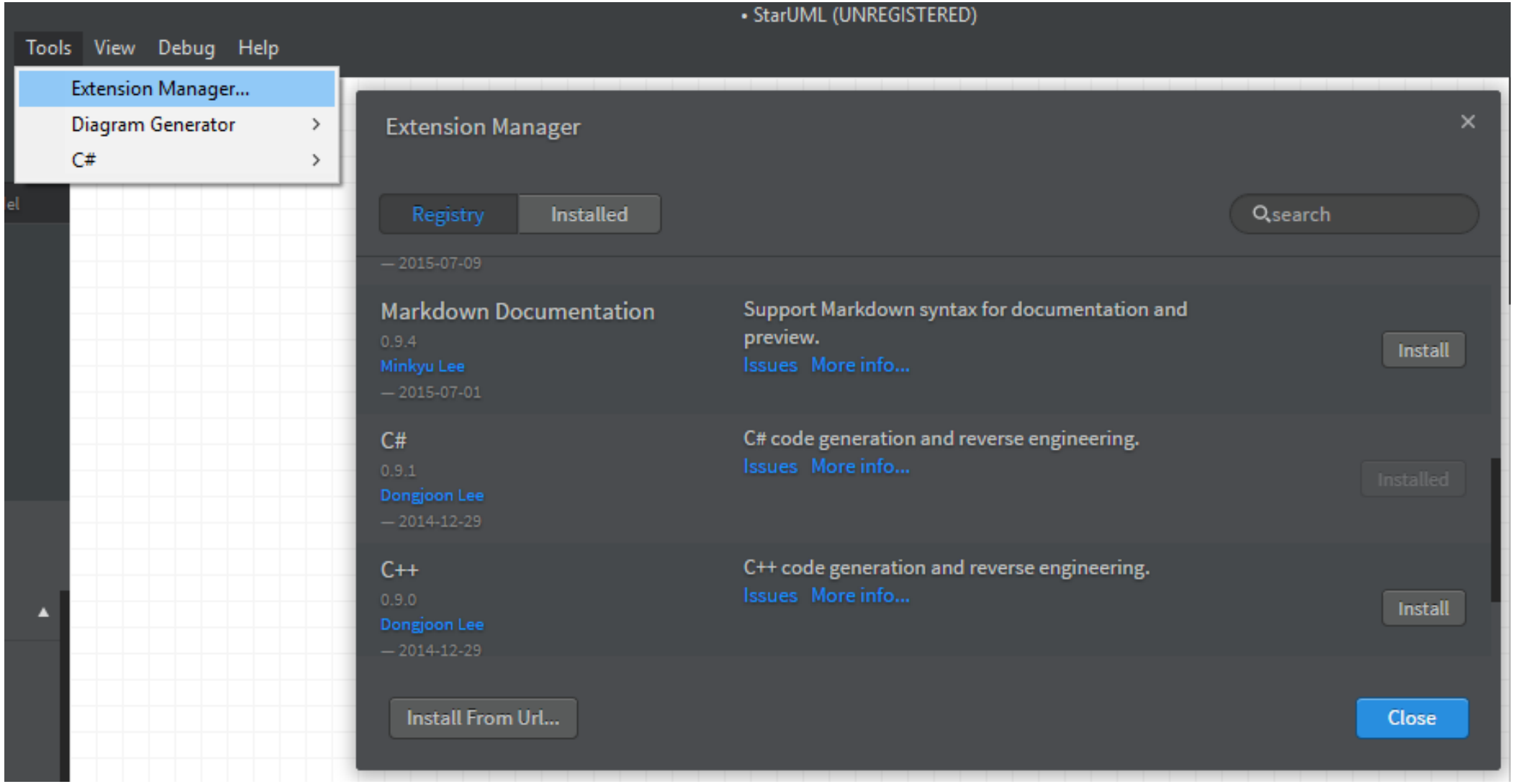
StarUML Guidelines (3)

- Add return type of method

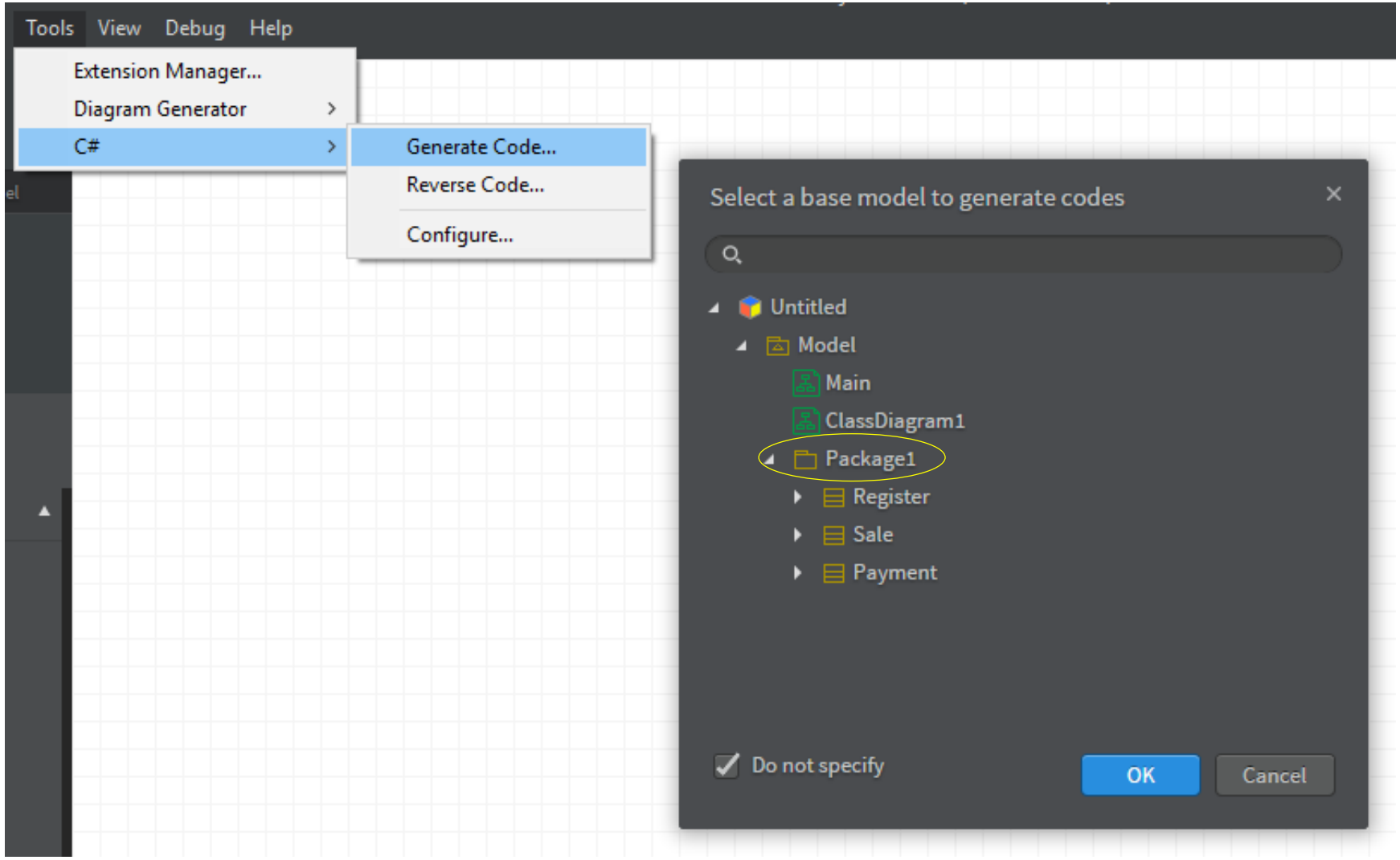




Cài đặt Extension



Tạo code tự động



Kết quả


```java
import java.util.*;

public class Register {

    /**
     * Default constructor
     */
    public Register() {
    }

    private Sale currentSale;

    public void endSale() {
        // TODO implement here
    }

    public void enterItem() {
        // TODO implement here
    }

    public void makeNewSale() {
        // TODO implement here
    }
}
```

Q&A

- Visibility between objects
  - Visibility is the ability of one object to see or have reference to another
  - An object A to send a message to an object B, B must be visible to A.

- *Kinds of visibility?*