Design Model: Determining Visibility

CH-18

Objectives

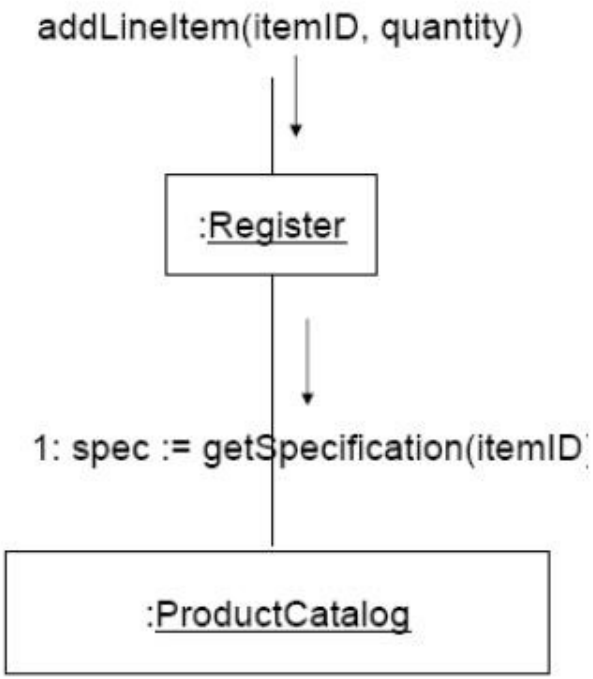
- Identify four kinds of visibility.
- Design to establish visibility.
- Illustrate kinds of visibility in the UML notation.

- Visibility is the ability of one object to see or have reference to another.

Visibility Between Objects

- The designs created for the system events *(enterItem, etc.) illustrate messages* between objects.
- For a sender object to send a message to a receiver object, the sender must be visible to the receiver
- the sender must have some kind of reference or pointer to the receiver object.

Visibility Between Objects



- For example, the *getSpecification message sent from a Register to a ProductCatalog implies that the ProductCatalog instance is visible to the Register instance*

Visibility Between Objects

- The UML has special notation for illustrating visibility
- When creating a design of interacting objects, it is necessary to ensure that the necessary visibility is present to support message interaction

Visibility

- There are **four** common ways that visibility can be achieved from object **A** to object ***B:***

- **Attribute visibility**—B is an attribute of A**.**
- **Parameter visibility**—B is a parameter of a method of A.
- **Local visibility**—B is a (non-parameter) local object in a method of A.
- **Global visibility**—B is in some way globally visible.

Attribute Visibility

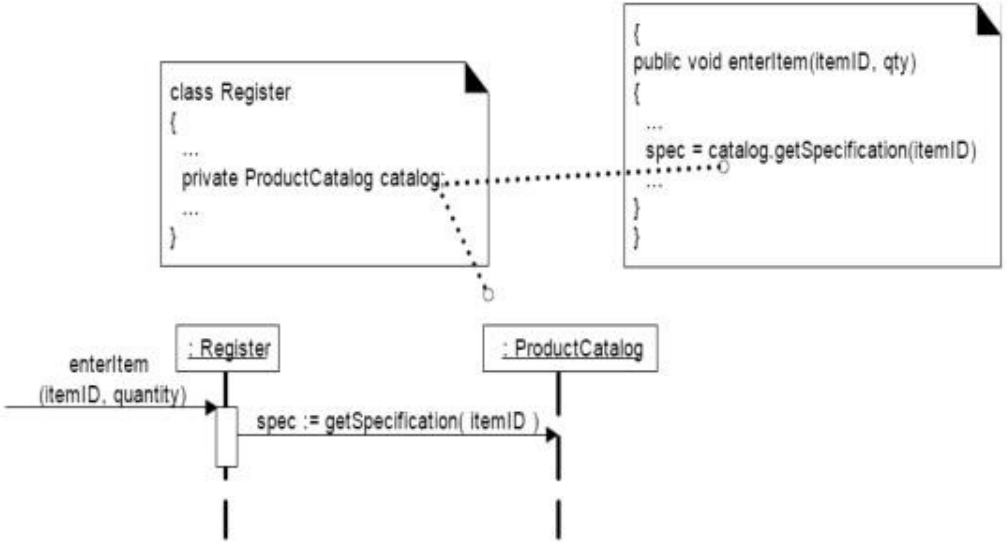
- **Attribute visibility** from A to B exists when B is an attribute of A.
- It is a relatively permanent visibility because it persists as long as A and B exist.
- This is a very common form of visibility in object-oriented systems.

Attribute Visibility

For ex. in a Java class definition for *Register, a Register instance may have* attribute visibility to a *ProductCatalog, since it is an attribute (Java instance* variable) of the *Register.*

```
public class Register
{
...
private ProductCatalog catalog;
...
}
```

This visibility is required because in the *enterItem diagram* , a *Register needs to send the getSpecification message to a ProductCatalog:*



Parameter Visibility

- **Parameter visibility** from A to B exists when B is passed as a parameter to a method of A.
- It is a relatively temporary visibility because it persists only within the scope of the method.
- After attribute visibility, it is the second most common form of visibility in object-oriented systems.

Parameter Visibility

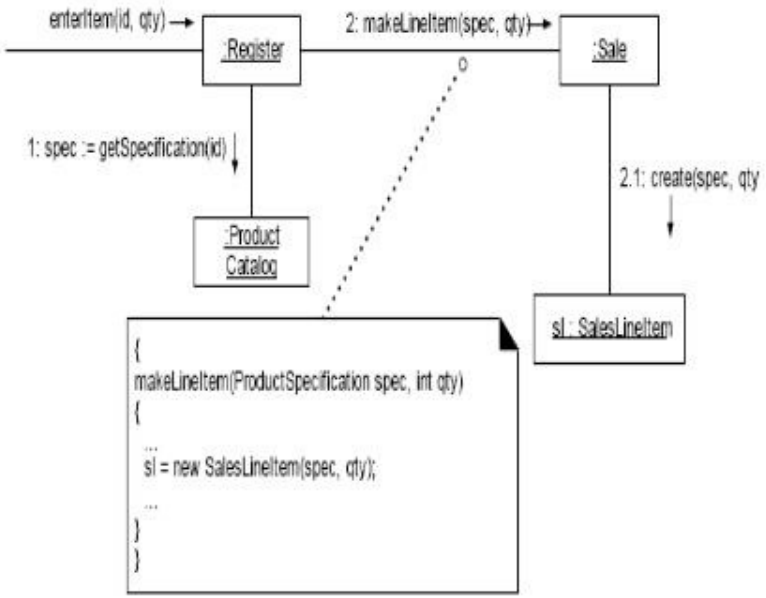


Figure 18.3 Parameter visibility.

- when the *makeLineItem* message is sent to a Sale instance, a *ProductSpecification* instance is passed as a parameter. Within the scope of the makeLineItem method, the Sale has parameter visibility to a *ProductSpecification* (see Figure 18.3).

Parameter Visibility

- It is common to transform parameter visibility into attribute visibility.

Parameter to attributeVisibility

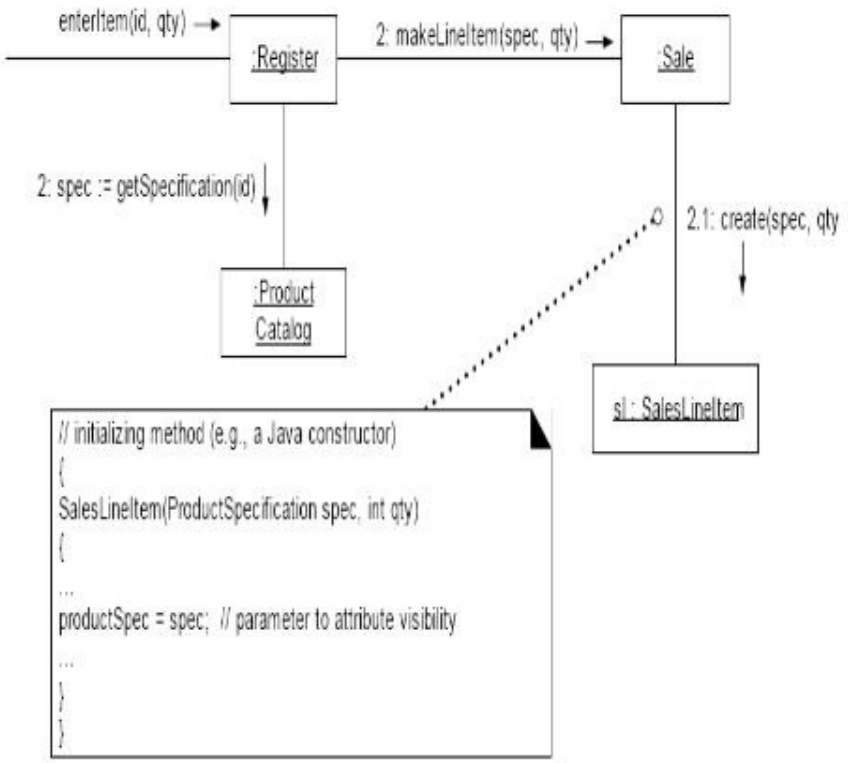


Figure 18.4 Parameter to attribute visibility.

when the *Sale creates a new SalesLineItem, it passes a*

ProductSpecifi-cation in to its initializing method (in C++ or Java, this would

be its constructor). Within the initializing method, the parameter is assigned

to an attribute, thus establishing attribute visibility (Figure 18.4).

Local Visibility

- **Local visibility** from A to B exists when B is declared as a local object within a method of A.
- It is a relatively temporary visibility because it persists only within the scope of the method.
- After parameter visibility, it is the third most common form of visibility in object-oriented systems.

Local Visibility

- Two common means by which local visibility is achieved are:
- Create a new local instance and assign it to a local variable.
- Assign the returning object from a method invocation to a local variable.

Local Visibility

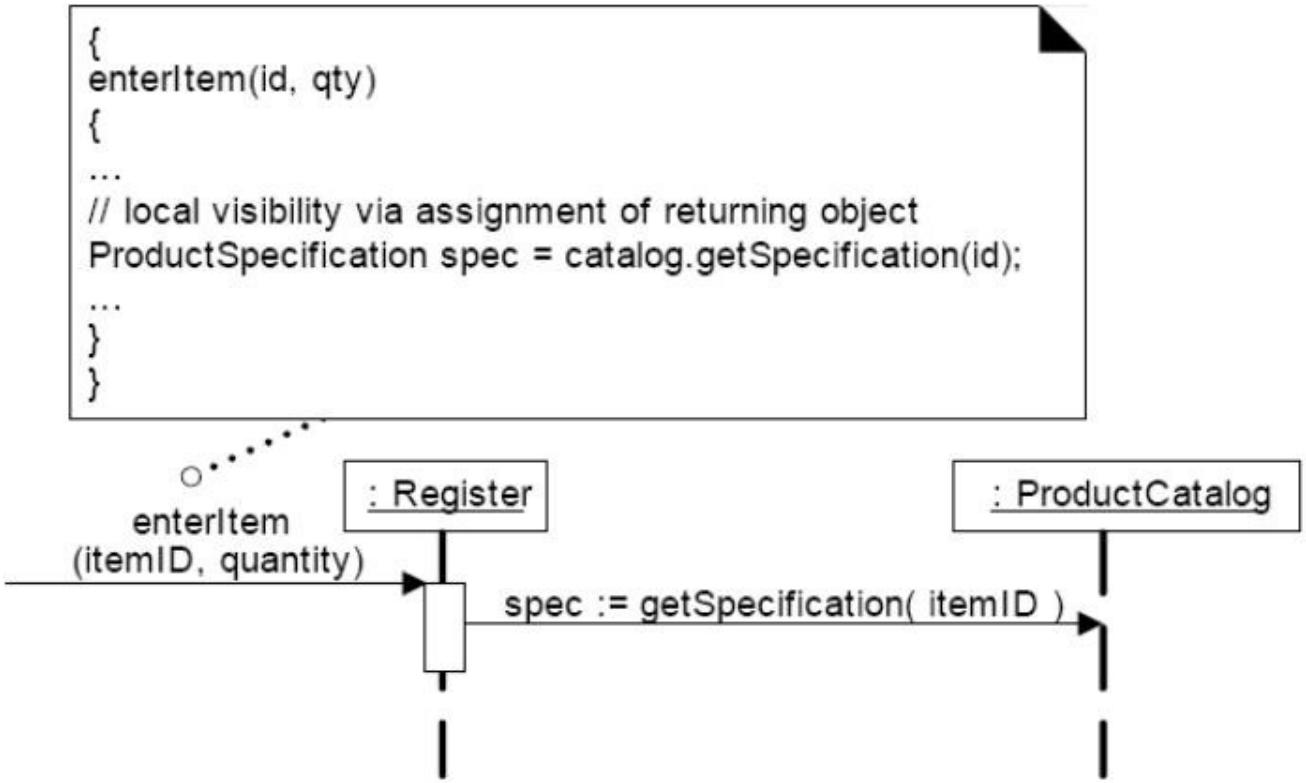


Figure 18.5 Local visibility.

Global Visibility

- **Global visibility** from A to B exists when B is global to A.
- It is a relatively permanent visibility because it persists as long as A and B exist.
- It is the least common form of visibility in object-oriented systems.

Global Visibility

- One way to achieve global visibility is to assign an instance to a global variable, which is possible in some languages, such as C++, but not others, such as Java.

Illustrating Visibility in the UML Collaboration diagram (optional)

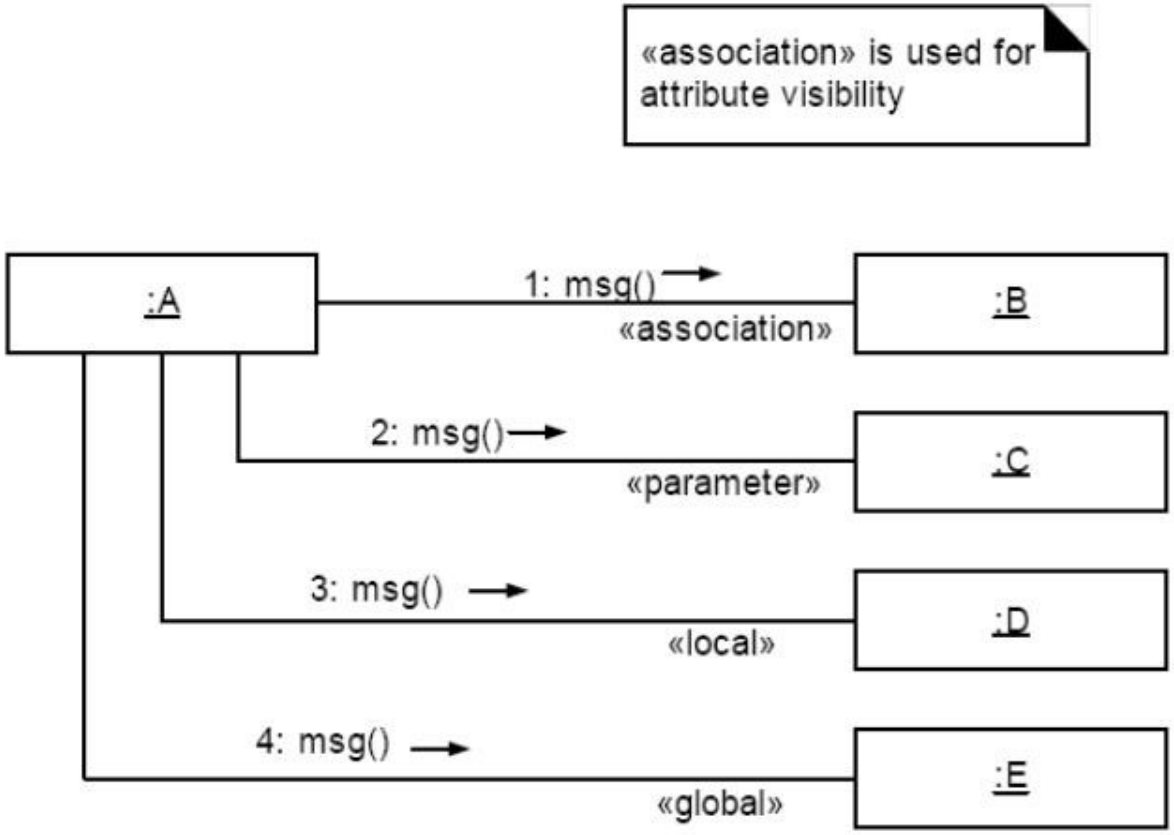


Figure 18.6 Implementation stereotypes for visibility.