UML 인터랙션 다이어그램 표기법 (UML Interaction Diagram)

Objectives

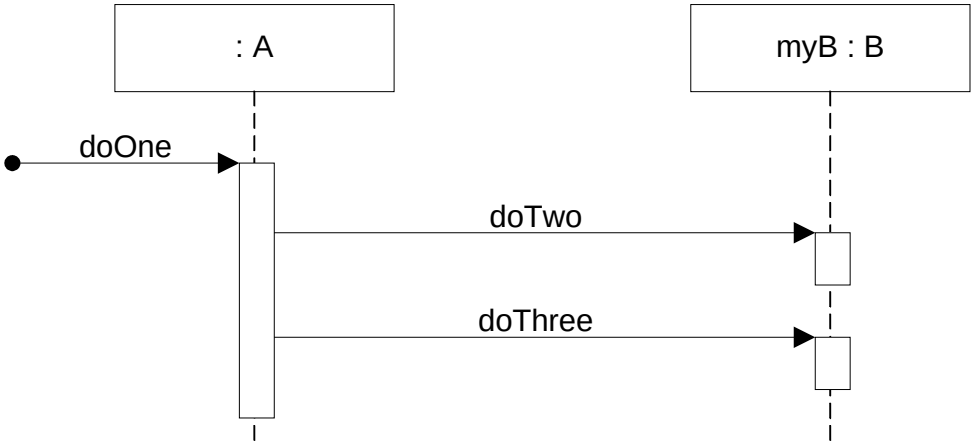
- 자주 사용되는 UML 순차도와 협력도를 그릴 수 있다.

소개

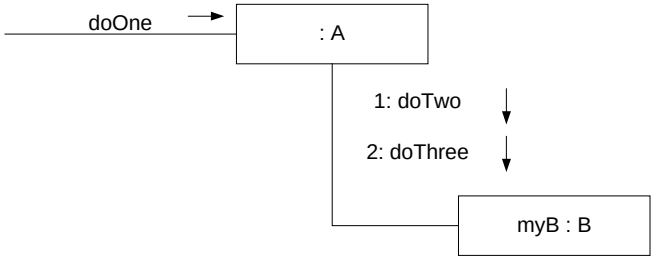
- 동적 객체 모델링은 객체들이 메시지를 주고 받으면서 상호작용하는 것을 표현
- Interaction Diagram 을 사용 – 순차도(sequence diagram)과 협력도(communication diagram)으로 구분.
- 표기법보다는 객체지향 설계의 핵심원칙은 무엇인가가 더 중요!

1. 순차도와 협력도

- 교류도
  - 순차도 (sequence diagram)와 협력도
  - 표현력은 같음



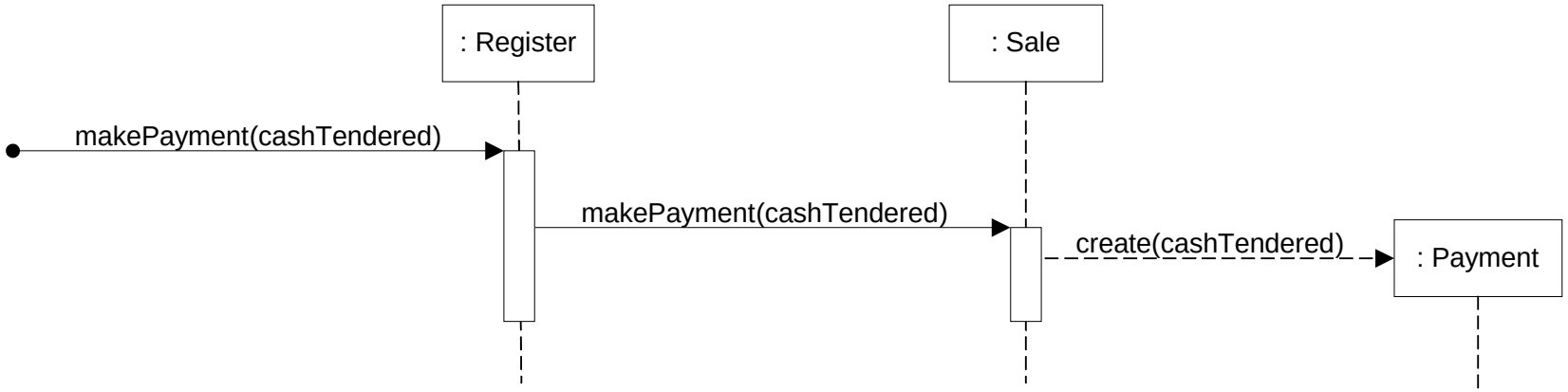

```
public class A
{
  private B myB = new B();

  public void doDone()
  {
     myB.doTwo();
     myB.doThree();
   }
   //…
}
```



순차도 vs. 협력도

- 객체 간의 메시지 통신을 표현하며 표현력은 같다.
- 순차도 장단점
  - 보기에 편하다. 많이 사용한다.
  - 공간을 많이 차지
- 협력도 장단점
  - 공간을 적게 차지하면서 객체간의 협력관계를 볼 수 있다.
  - 순서는 1,2,... 를 따라가야 하므로 순서를 보기에는 어려움

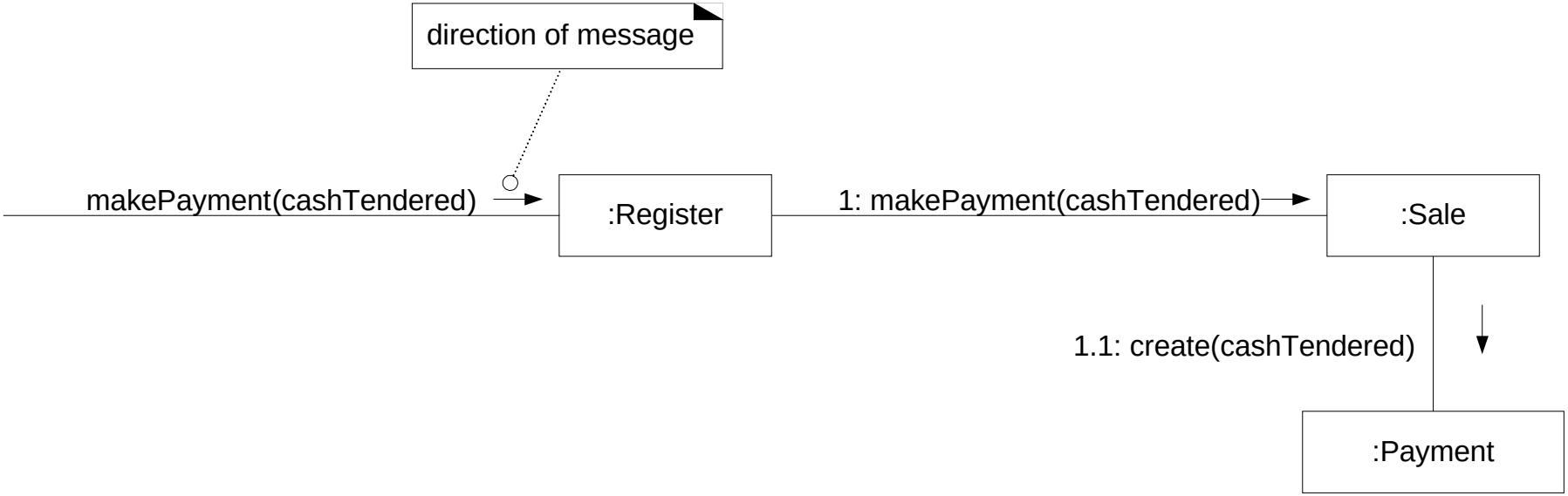
예제: makePayment



- 1. makPayment 메시지가 Register 객체에 도착. 발신자는 모름.
- 2. Register 객체는 Sale 객체에 같은 메시지를 전달.
- 3. Sale 객체는 Payment 객체를 생성

```
public Sale {
    private Payment payment;
    Public void makePayment (Money cashTendered)
    {
      payment = new Payment(cashTendered);
      //…
    }
    }
```

예제: makePayment (협력도)

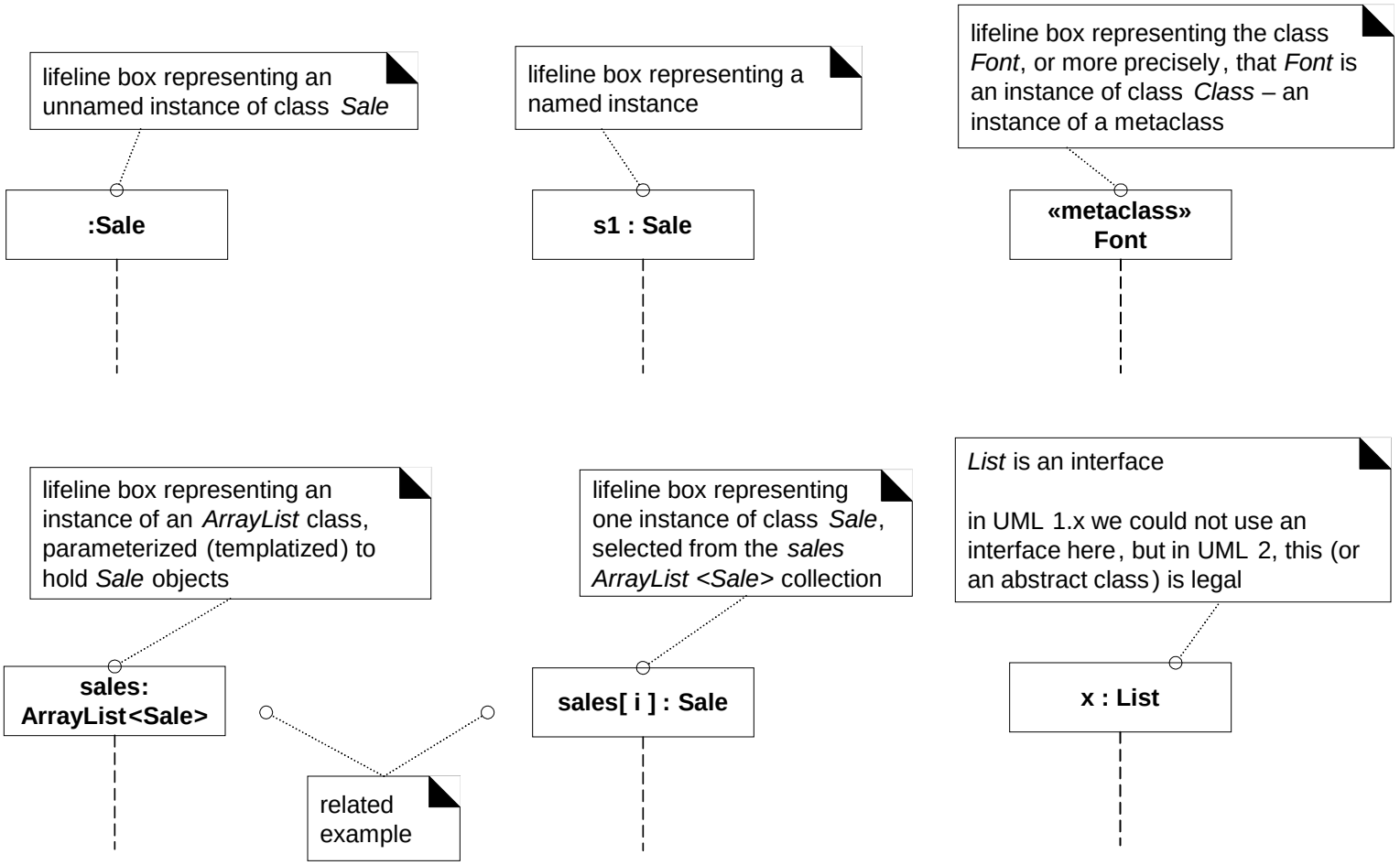


2. 초보 UML모델러는 동적 모델에 충분한 관심이 없다.

- 지침
  - 클래스 다이어그램을 이용한 정적 객체 모델링에만 시간을 소비하지 말고, 인터랙션 다이어그램을 이용하여 동적 객체 모델링을 하는 데도 시간을 할애하라.

3. UML 인터랙션 다이어그램의 일반적 표기법

- 생명선 박스를 이용한 객체 표현



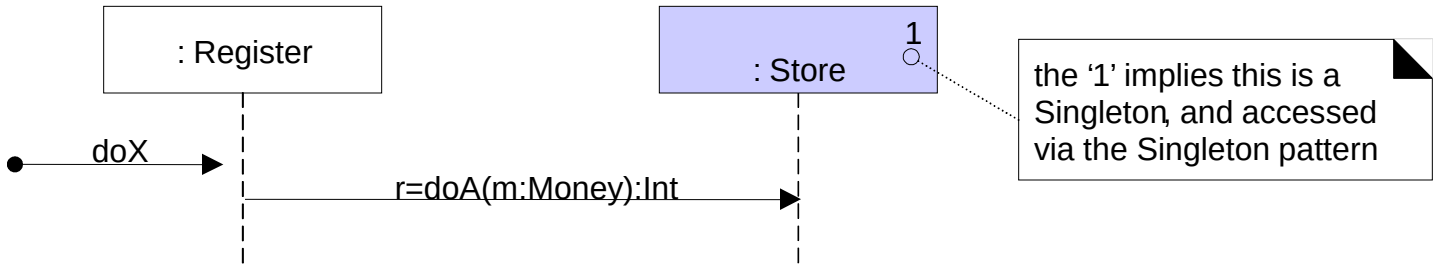
교류도의 일반적 표기법

- 기본 메시지 표현 문법

  return = message (parameter: Type) : returnType

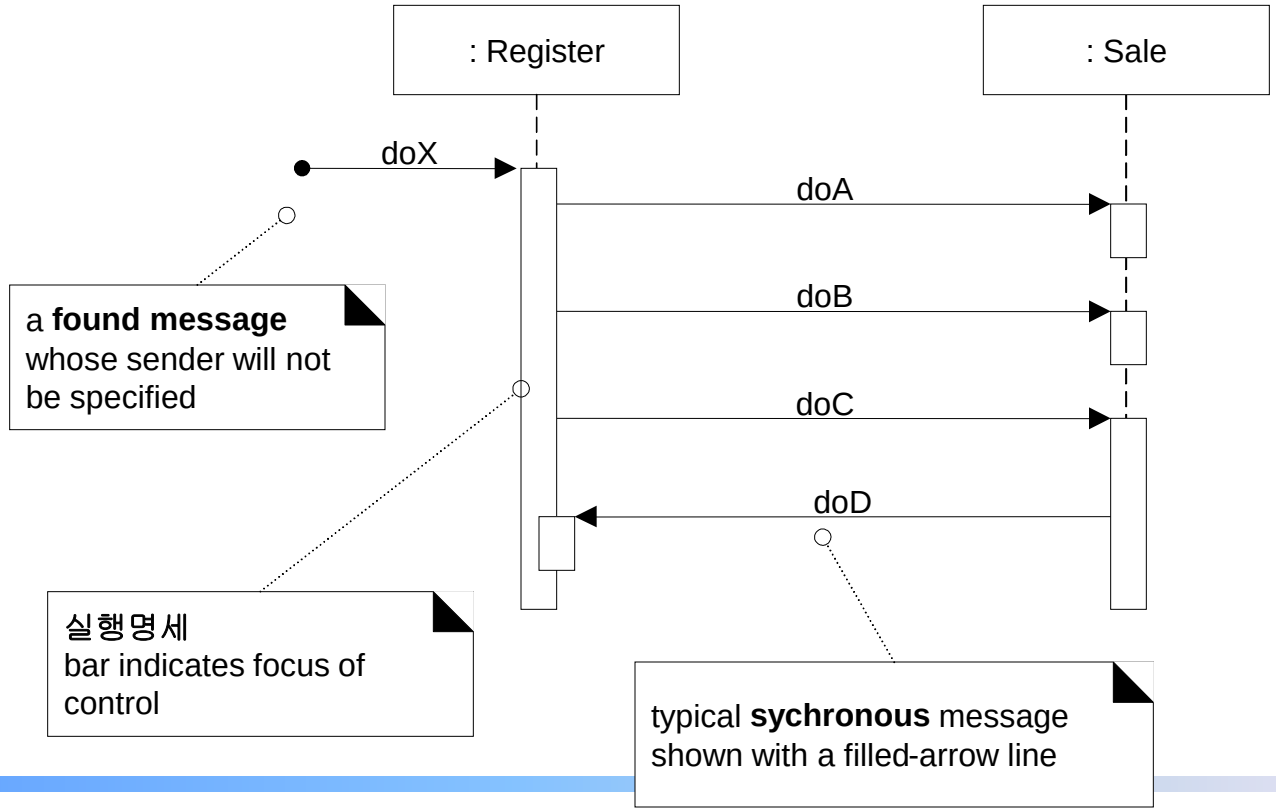
  - 중요하지 않은 부분은 생략할 수 있다.
    - Initialize(cod)
    - Initialize
    - D = getProductDescription(id)
    - D = getProductDescription(id:ItemID)
    - D = getProductDescription(id:ItemID) : ProductDescription

- Singleton 객체



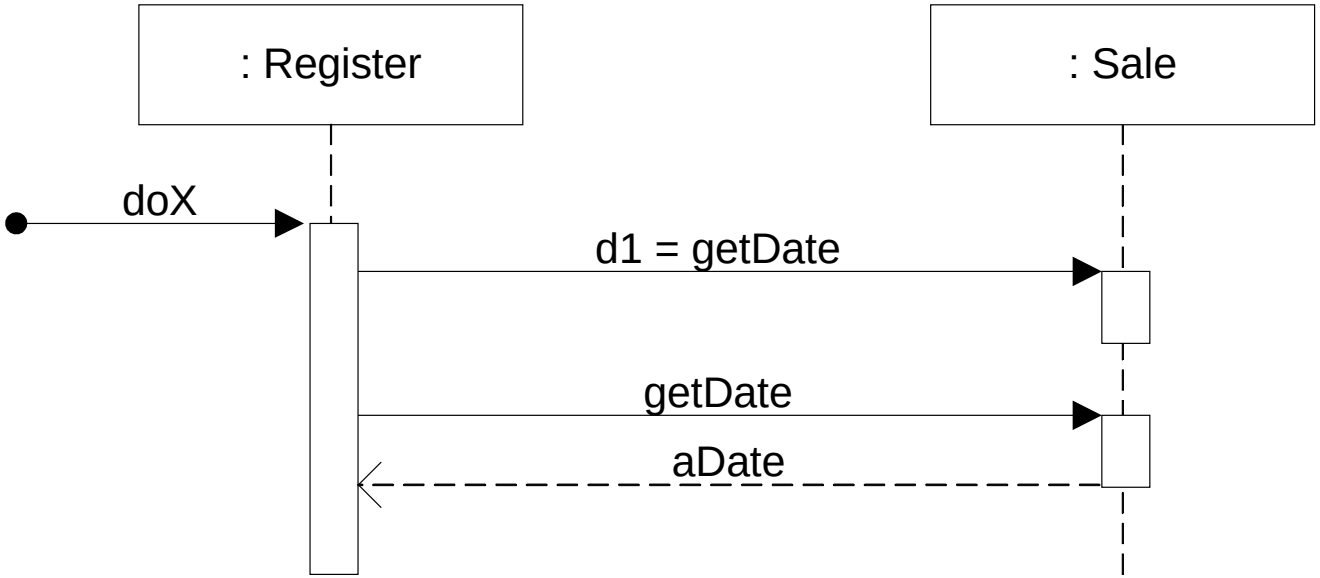
4. 순차도의 기본적인 표기법 1

- 생명선 박스와 생명선
  - UML 1.x : 점선
  - UML 2.x : 점선 또는 실선
- 메시지
  - 생명선 사이를 연결하는 수평 화살표 (대부분 동기 메시지)
  - 메시지 호출 시간 순서는 위에서 아래로.



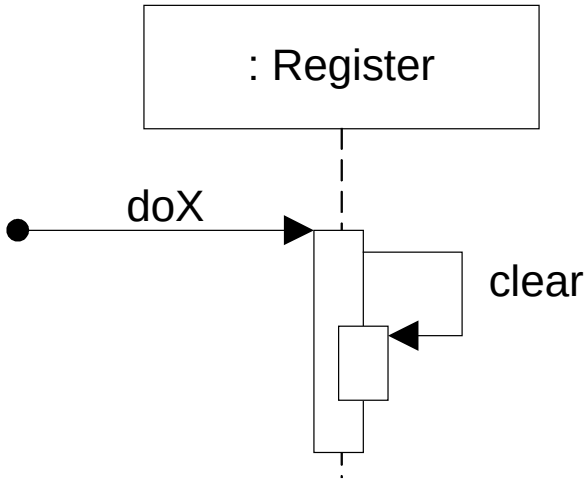
4. 순차도의 기본적인 표기법 2

- 응답 또는 반환 나타내기
  - 1. 메시지 문법을 이용, return = message(params)
  - 2. 응답 (또는 반환) 메시지 선 (점선)을 이용



4. 순차도의 기본적인 표기법 3

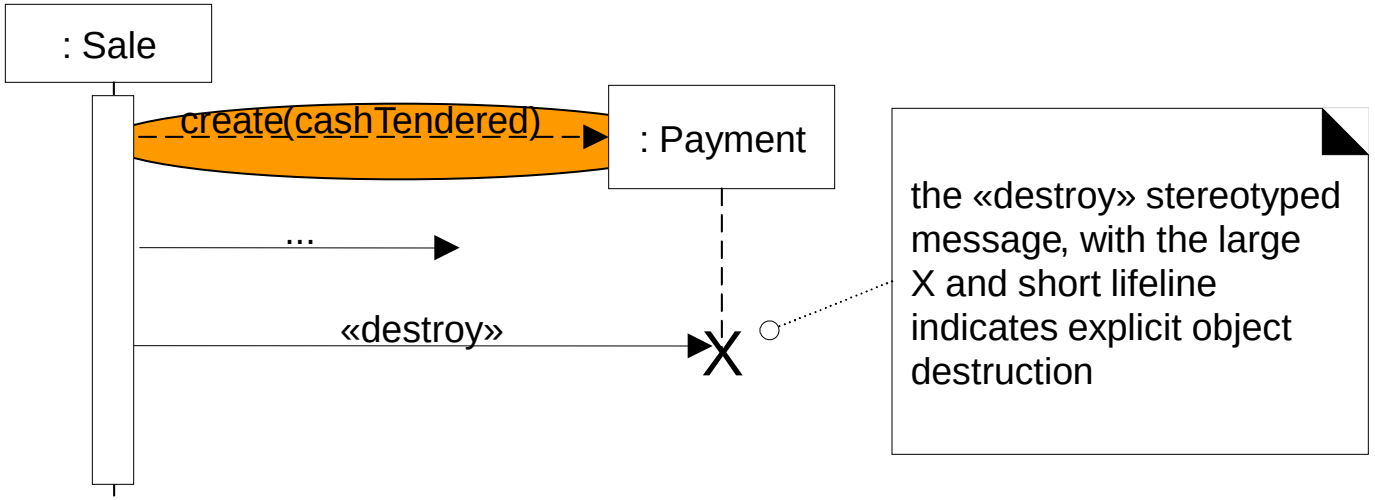
- “self” 또는 “this”로의 메시지
  - 중첩된 활성화 막대 (실행 명세)를 사용




```
public Register {
    public void doX ( )
    {
      clear()
      //…
    }
    private void clear()
    {
    }
```

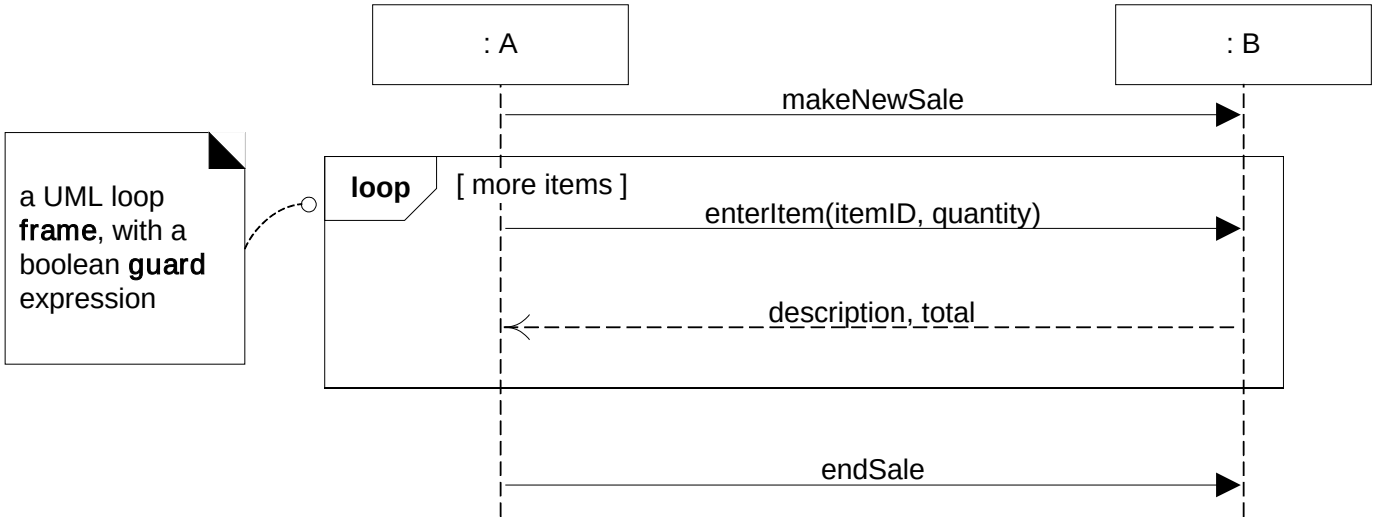
4. 순차도의 기본적인 표기법 4

- 인스턴스의 생성
  - 점선 호출과 create 라는 관용적 함수
  - “new” 연산자를 실행하여 생성자를 호출시켜라는 뜻
- 객체 생명선과 객체 소멸
  - C++ 같이 garbage collection이 안 되는 경우 명시적으로 기술



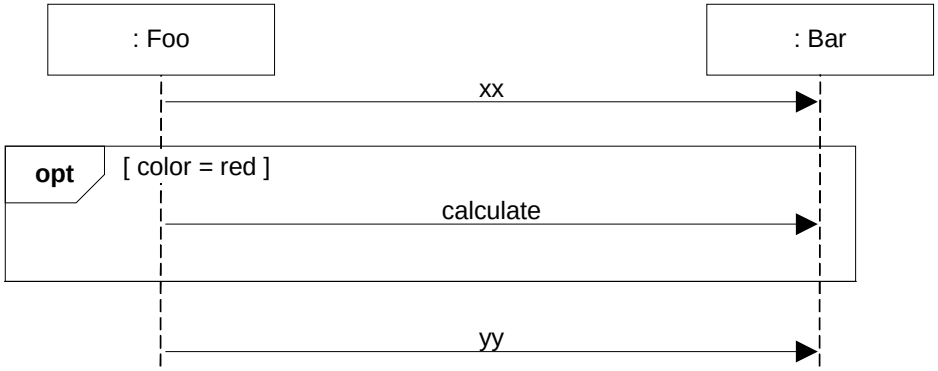
4. 순차도의 기본적인 표기법 5 - Frame

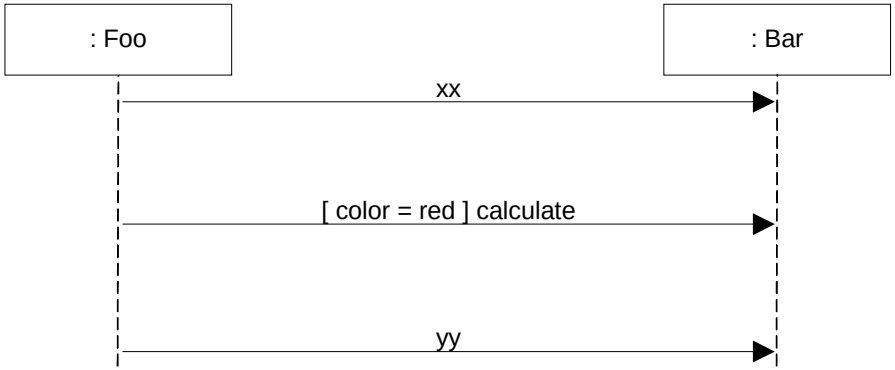
- UML 순차도에서의 프레임(frame)
  - 조건문이나 반복부분에 사용
  - 프레임 연산자나 레이블, 가드로 구성
  - 조건은 생명선 위에 위치
  - 프레임 연산자의 종류
    - alt : 가드 조건에 따라 배타적으로 실행
    - loop : 가드가 참일 때만 반복적으로 실행
    - Opt : 가드가 참일 때만 선택적으로 실행
    - Par : 병렬적으로 수행되는 부분
    - Region : 임계 영역 (오직 하나의 쓰레드만 실행 가능)



4. 순차도의 기본적인 표기법 5 – Frame

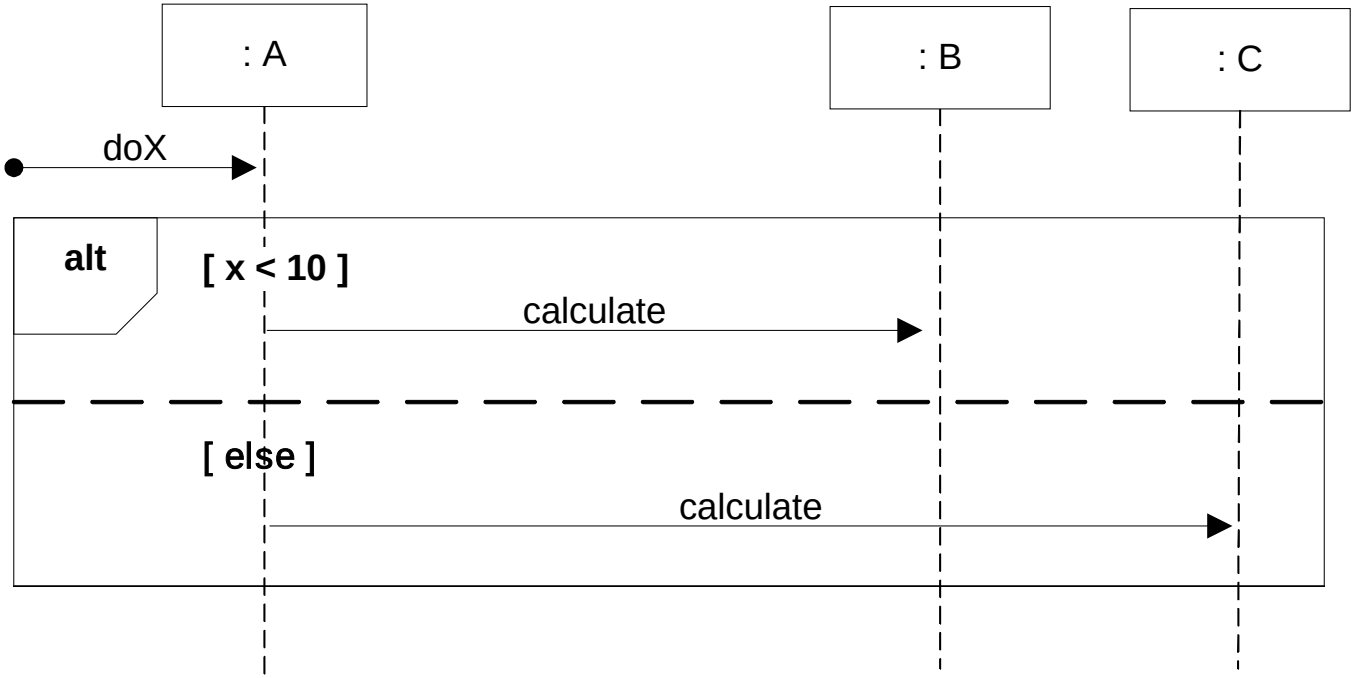
- UML 2.x
  - Option frame 사용
  - Color 가 red 일 때만 수행
- UML 1.x
  - 메지지 앞에 사용
  - 간단할 경우 사용 가능





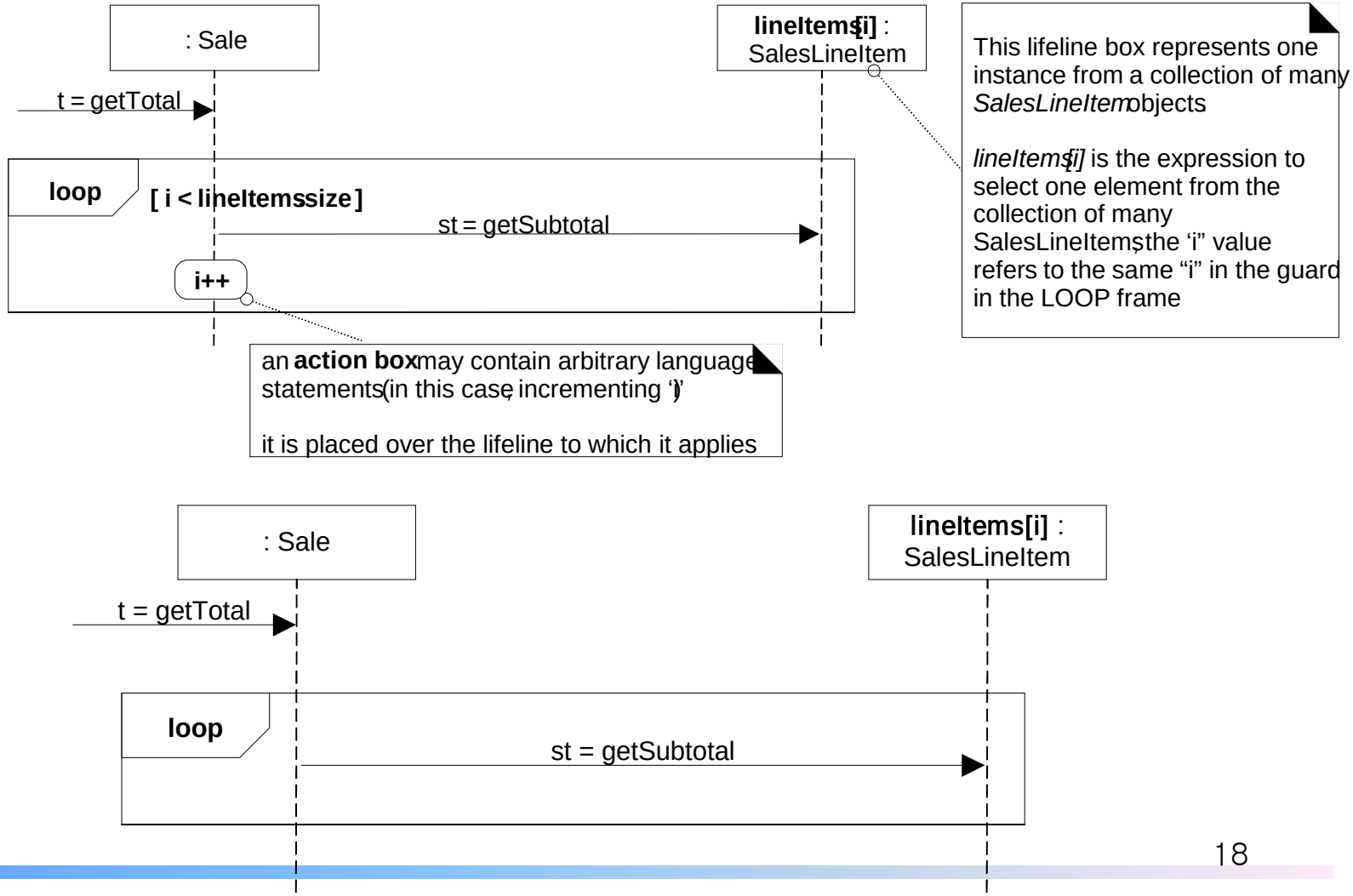
4. 순차도의 기본적인 표기법 5 - Frame

- 상호 배타적 실행
  - 프레임을 점선으로 나눔



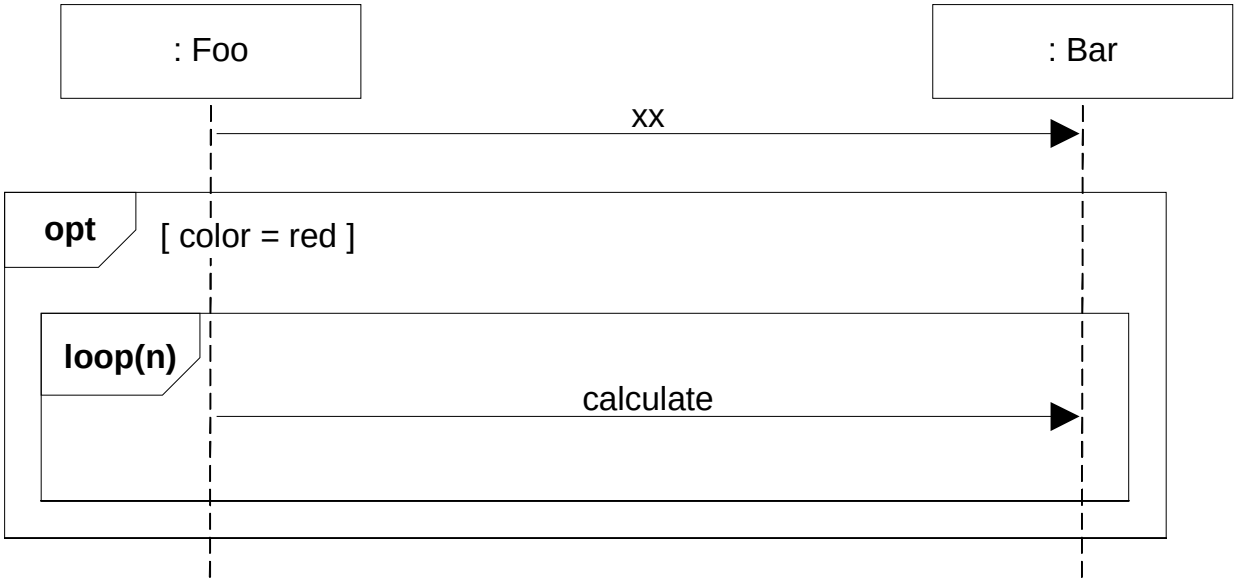
4. 순차도의 기본적인 표기법 5 – Frame

- 컬렉션에 대한 반복



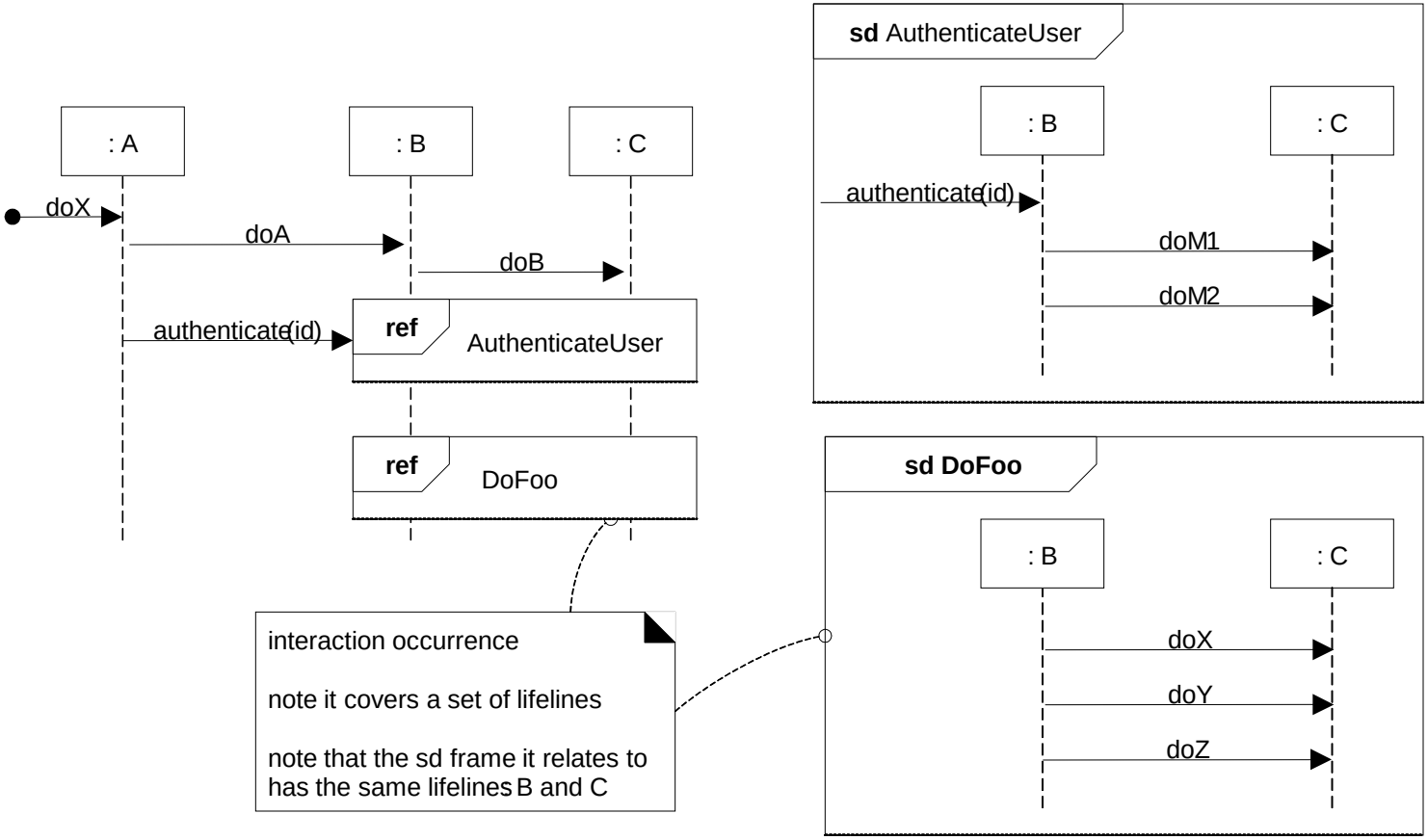
4. 순차도의 기본적인 표기법 5 - Frame

- 프레임의 중첩



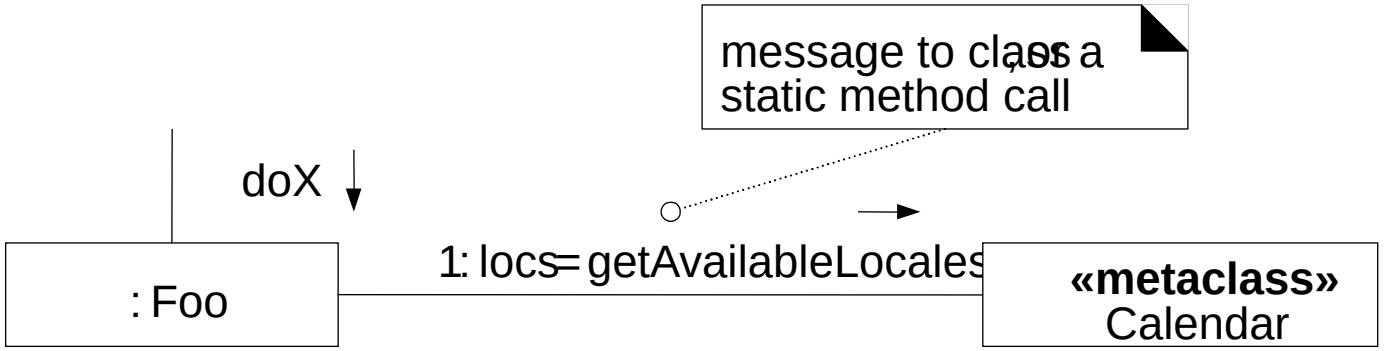
4. 순차도의 기본적인 표기법 6 – 순차도 연관

- 인터렉션의 사용
  - 다른 인터랙션을 참조
  - 단순화하거나, 다른 인터렉션을 포함시키고 싶을 때 사용



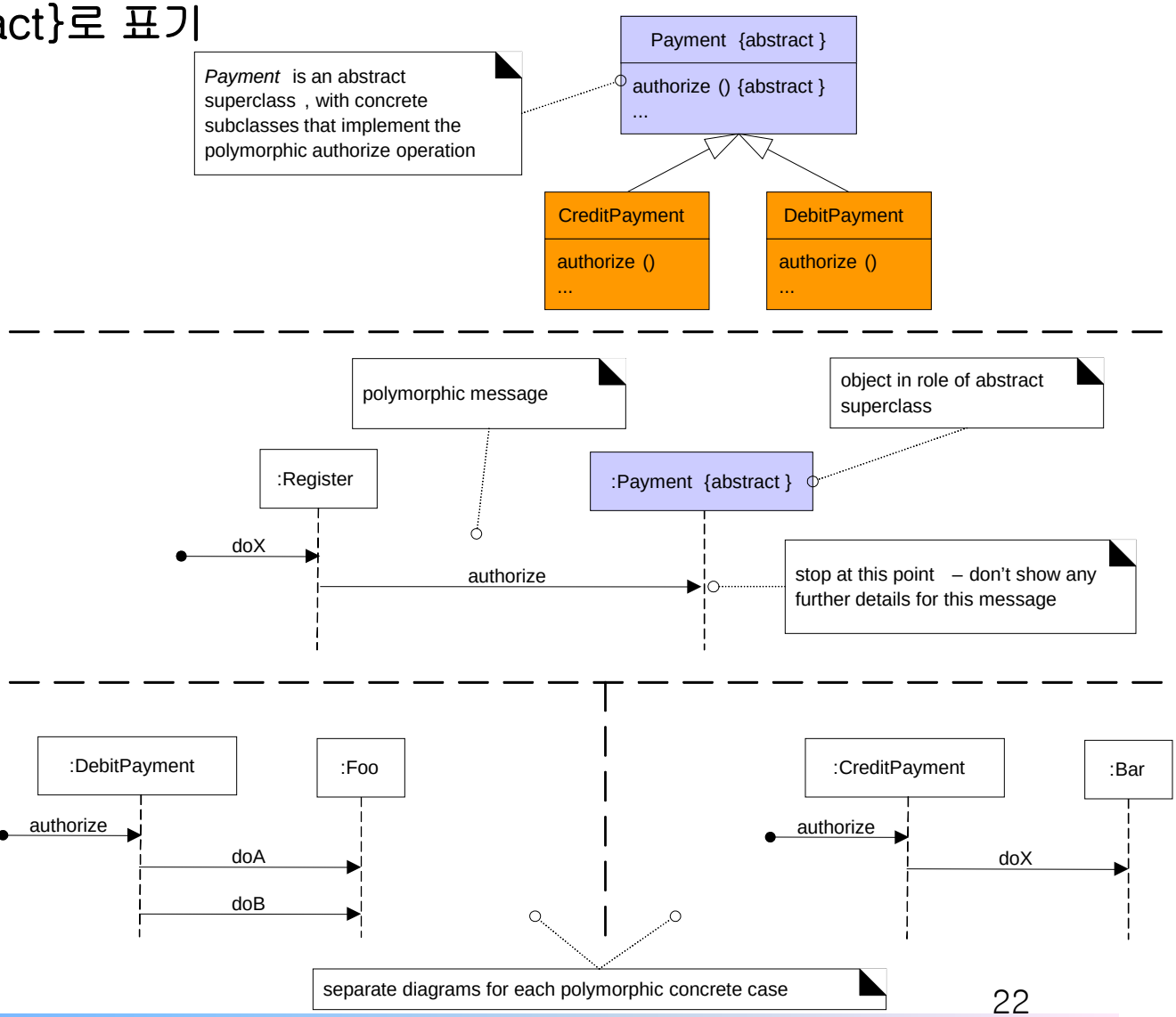
4. 순차도의 기본적인 표기법 6 – 정적 클래스로의 메시지

- 클래스 또는 정적 메소드 호출을 위한 도식
  - 보통은 객체에게 호출하지만, 클래스의 정적 메소드를 호출할 때는 클래스가 필요
  - 특정 클래스는 자바에서 메타클래스인 “Class”의 인스턴스임
  - .NET에서는 “Type”이 모든 클래스의 메타클래스임



4. 순차도의 기본적인 표기법 6 – 다형성 표기

- virtual 함수는 {abstract}로 표기



4. 순차도의 기본적인 표기법 7 – 비동기적 호출과 동기적 호출

- 비동기적 메시지 호출
  - 응답을 기다리지 않는다. 즉 블락(block)되지 않음
  - .NET 또는 java 의 쓰레드 환경에서 사용
  - 실행 도중에 쓰레드를 생성하고 초기화할 수 있다. (Thread.start)
  - 선으로 된 화살촉
- 동기적 메시지 호출
  - 보통의 함수 호출
  - 결과가 올 때까지 호출한 객체는 기다림
  - 속이 꽉 찬 화살촉

4. 순차도의 기본적인 표기법 7 – 액티브 객체

- 액티브 객체
  - 객체 자신만의 쓰레드를 실행시키고 제어함
  - 박스 양쪽에 바가 추가됨
- 비동기적 호출
  - run 메소드

a stick arrow in UML implies an asynchronous call

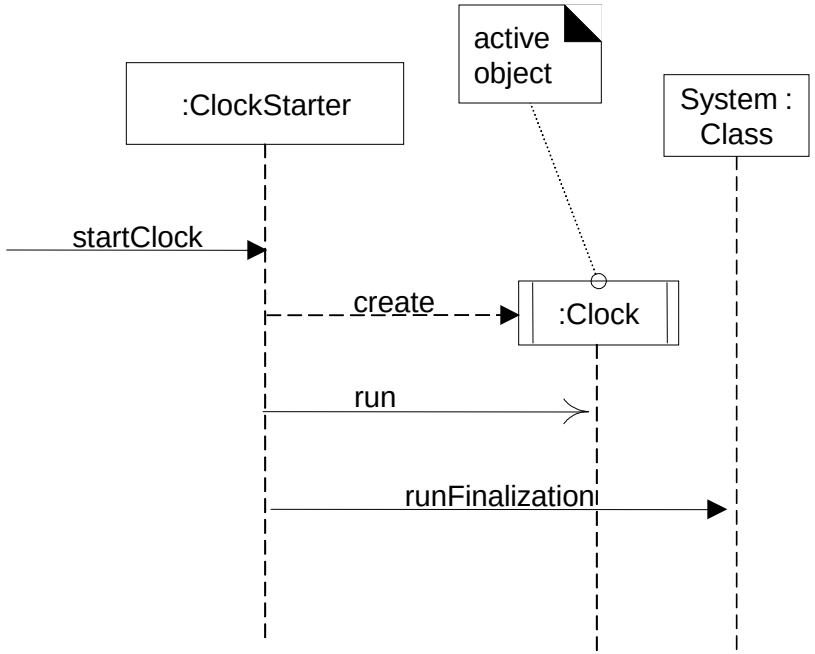
a filled arrow is the more common synchronous call

In Java, for example, an asynchronous call may occur as follows:

```
// Clock implements the Runnable interface
Thread t = new Thread( new Clock() );
t.start();
```

the asynchronous *start* call always invokes the *run* method on the *Runnable* (*Clock*) object

to simplify the UML diagram, the *Thread* object and the *start* message may be avoided (they are standard "overhead"); instead, the essential detail of the *Clock* creation and the *run* message imply the asynchronous call

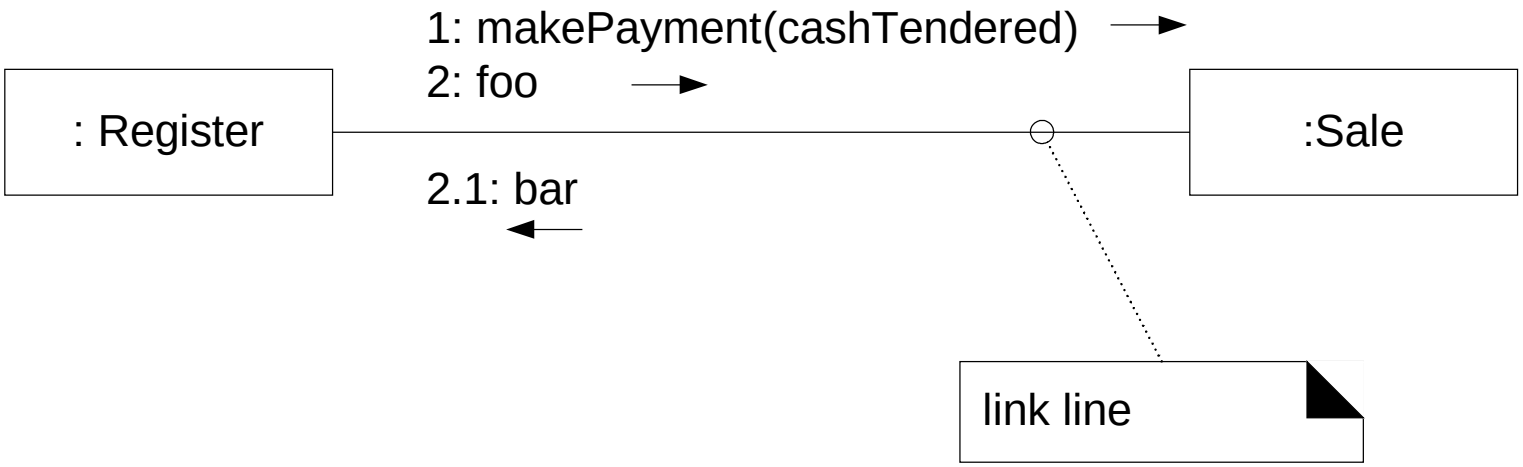


4. 순차도의 기본적인 표기법 7 – 액티브 객체의 java 구현

```
public class Clock implements Runnale
{
public void run() // 쓰레드의 start함수를 부르면 자동 호출되는 함수
{
   while (true) // 쓰레드안에서 무한 반복
   {
     ///
   }

}

Public class ClockStarter
{
   public void startClock()
   {
     Thread t = new Thread(new Clock());
     t.start();  // Clock 쓰레드의 run 메소드의 비동기적 호출
     System.runFinalization();
   }
}
```

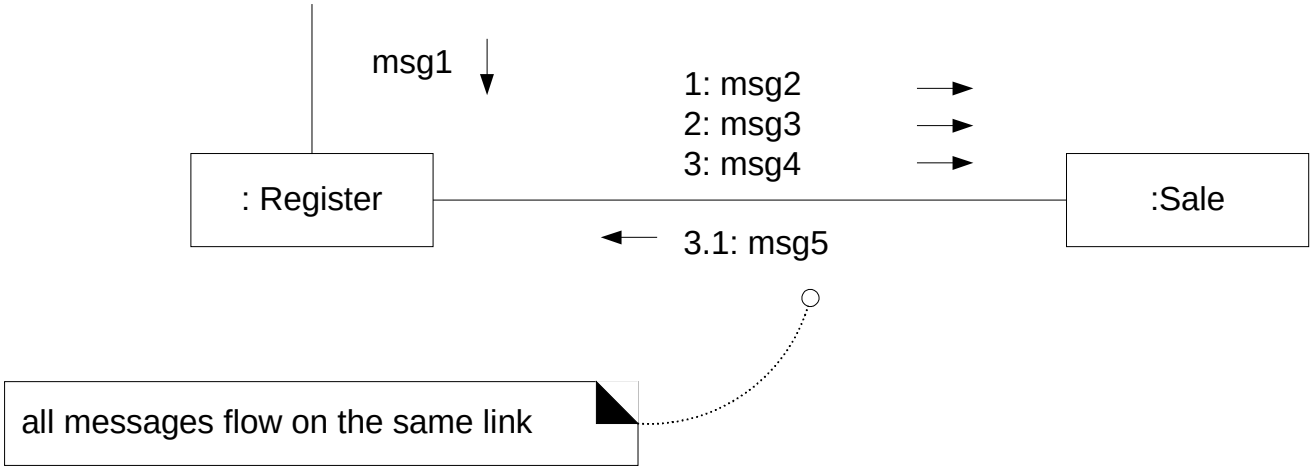
5. 협력도의 기본적인 표기법 - 링크

- 링크
  - 두 객체 사이의 연결 경로로서 연관관계의 인스턴스임
  - 예를 들어 Register 클래스와 Sale 클래스의 연관이 있다면 Register 객체와 Sale 객체 사이에 링크가 존재한다.
  - 링크 하나에는 여러 종류의 메시지가 흘러갈 수 있다.



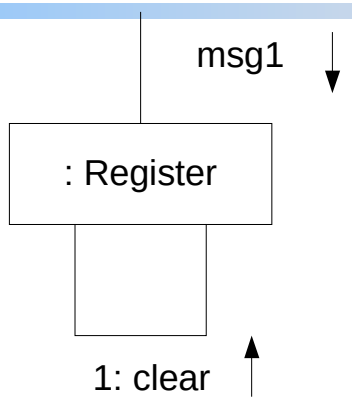
5. 협력도의 기본적인 표기법 – 메시지

- 메시지
  - 링크 위에 메시지 표현식과 작은 화살표로 흐름방향을 표현
  - 순서번호를 사용하여 흐름 순서를 표시
  - 첫 번째 메시지는 순서번호를 붙이지 않는다.

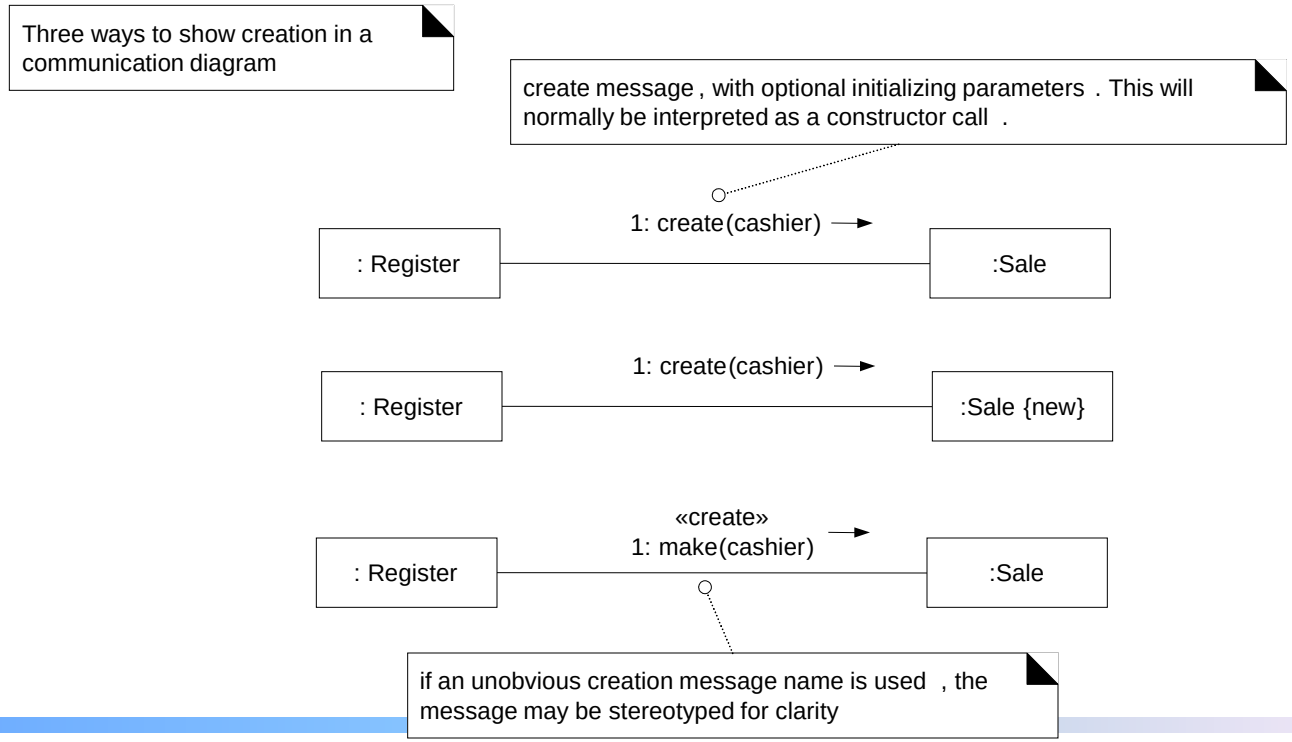


5. 협력도의 기본적인 표기법 – “self” 및 인스턴스 생성

- “self” 또는 “this”로의 메시지
  - 자신으로의 루프

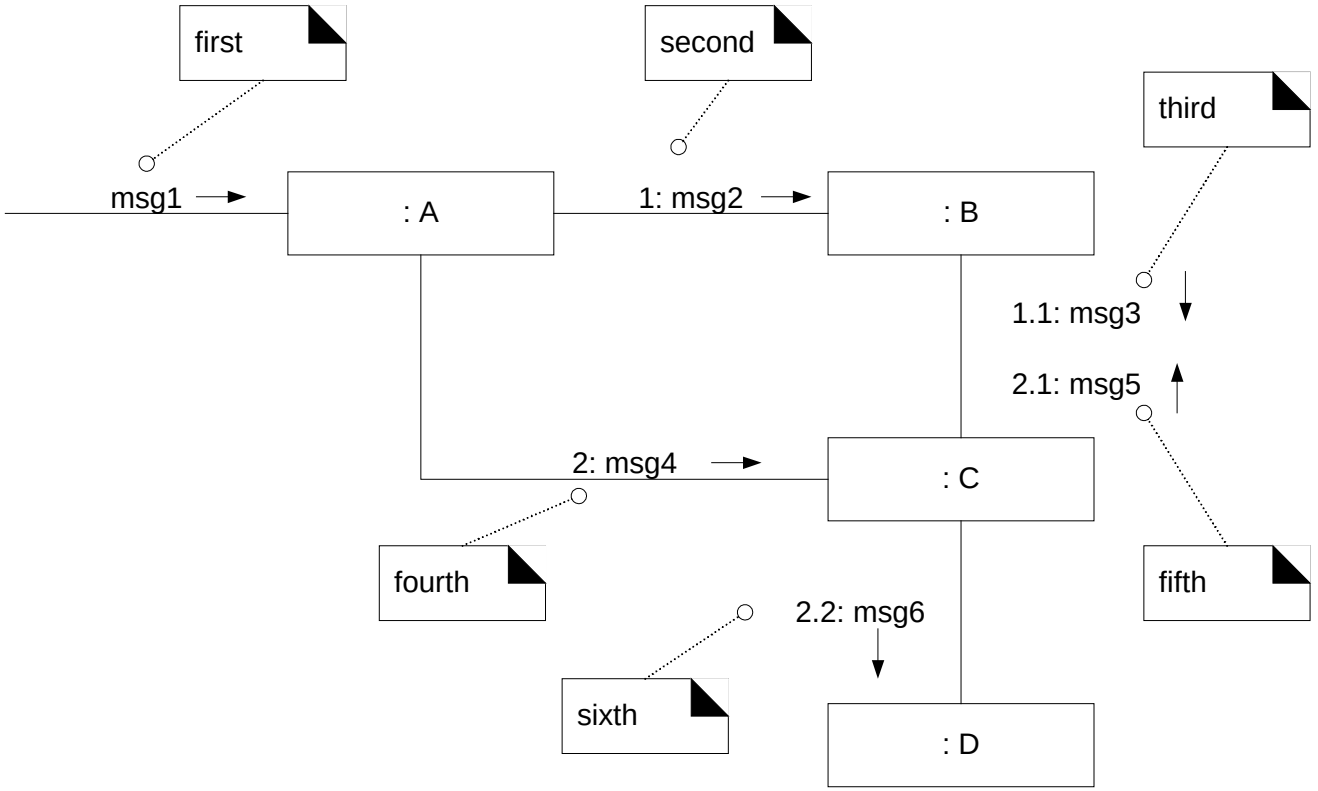


- 인스턴스의 생성
  - <<create>> 스테레오 타입 사용
  - 생성된 인스턴스 {new} 태크 사용



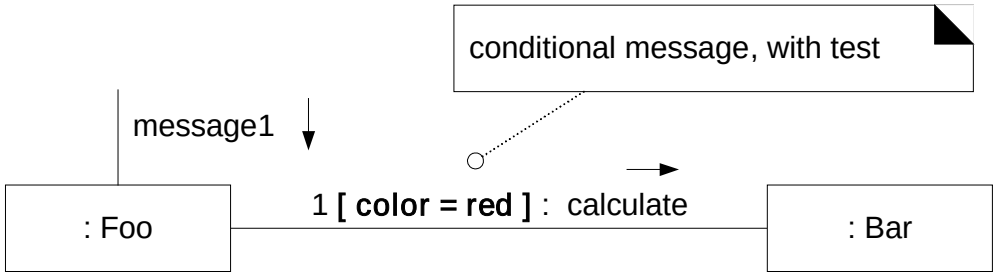
5. 협력도의 기본적인 표기법 –순서번호 매기기

- 순서번호 (sequence number) 규칙
  - 1. 첫 번째 메시지는 번호를 붙이지 않는다.
  - 2. 첫 번째 메시지를 호출 받은 객체가 다른 객체들에게 메시지를 보낼 때는 1,2,3...으로 붙이고, 그 다른 객체들이 또 다시 중첩하여 메시지를 보낼 때는 1.1, 1.2, 등으로 dot 기호를 사용한다.

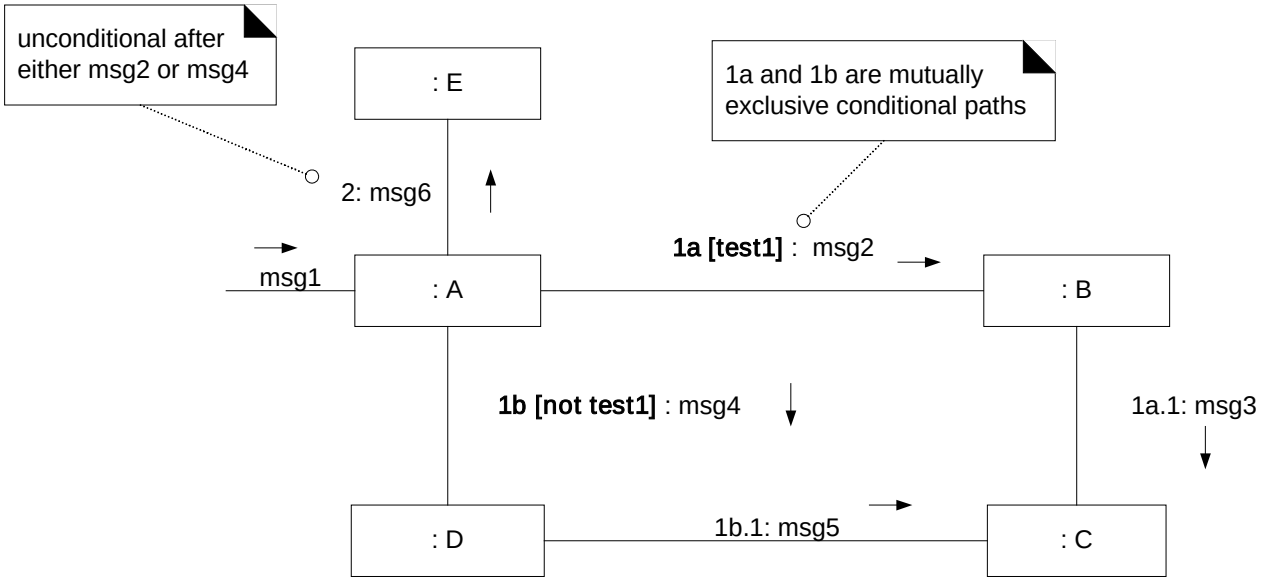


5. 협력도의 기본적인 표기법 – 조건부 메시지 전달

- 조건 메시지
  - 가드를 사용
  - *번호 [조건] : 메시지*

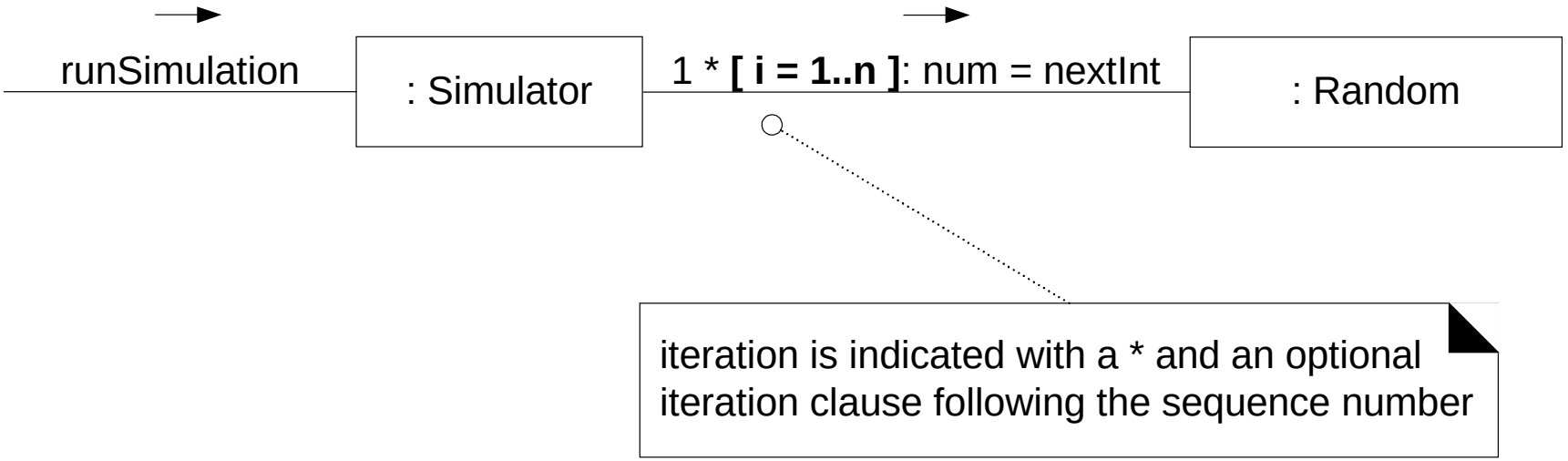


- 상호 배타적인 조건



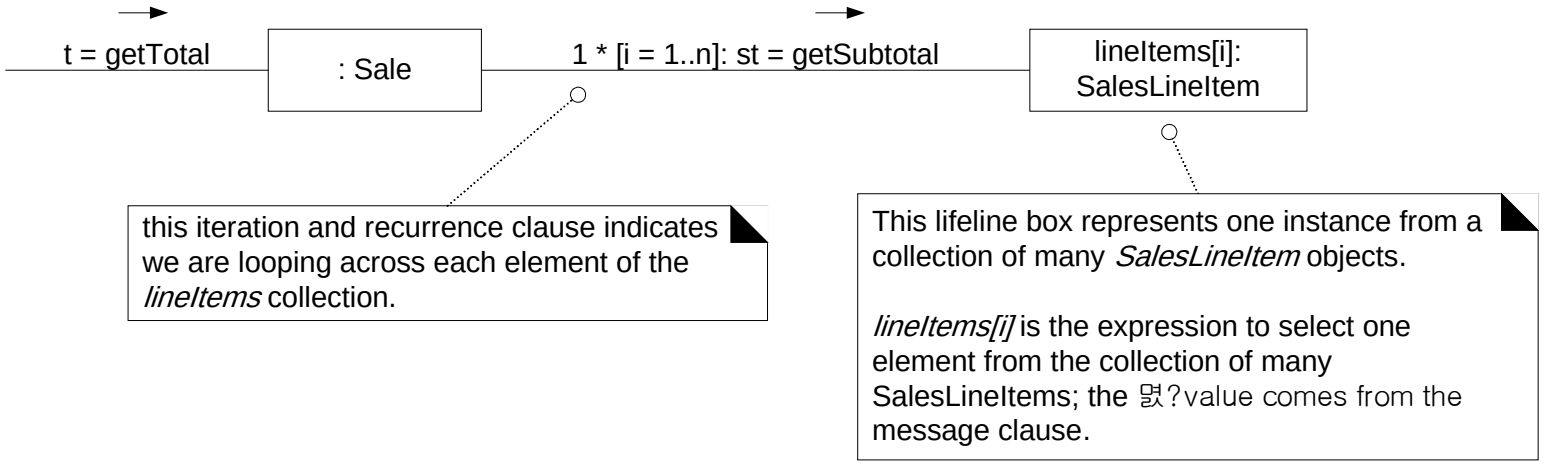
5. 협력도의 기본적인 표기법 – 반복

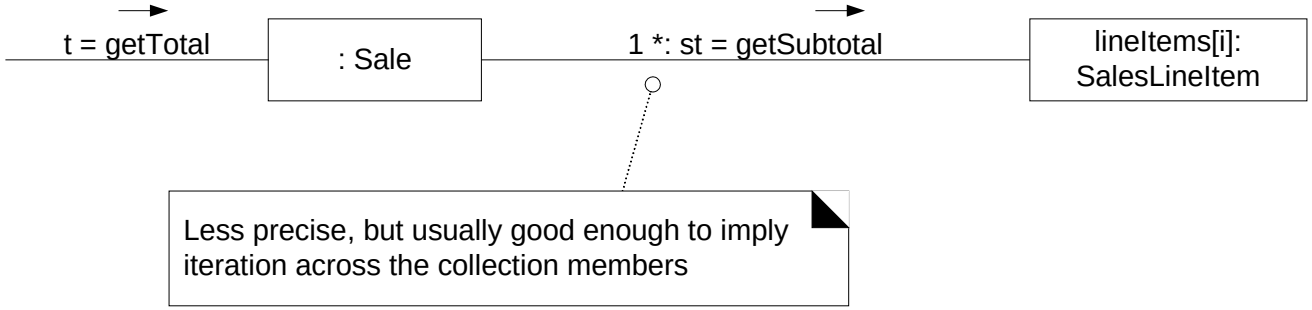
- 반복 혹은 루프
  - 조건 앞에 '*' 를 붙임.



5. 협력도의 기본적인 표기법 – 컬렉션에 대한 반복

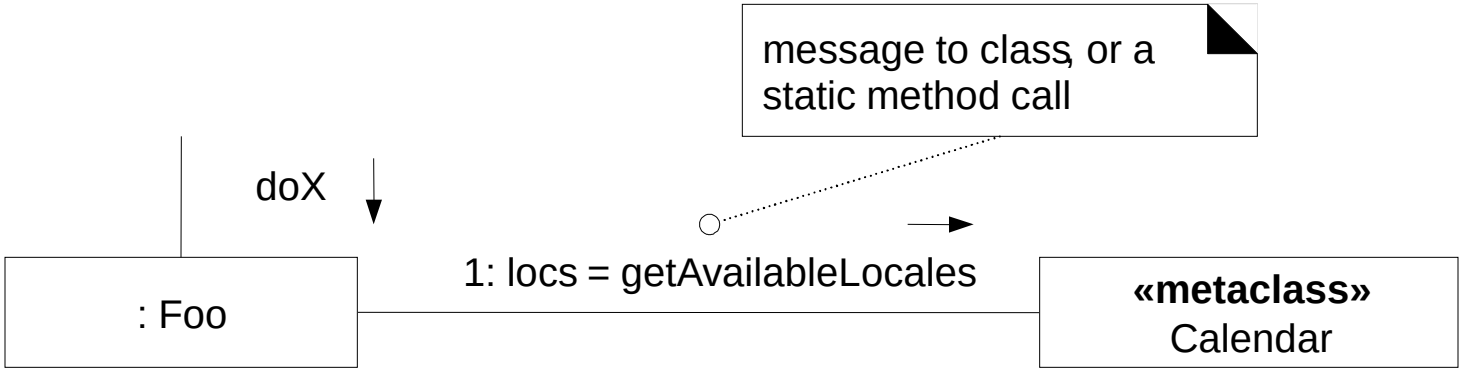
- 컬렉션의 객체들에 대한 똑같은 메시지 호출





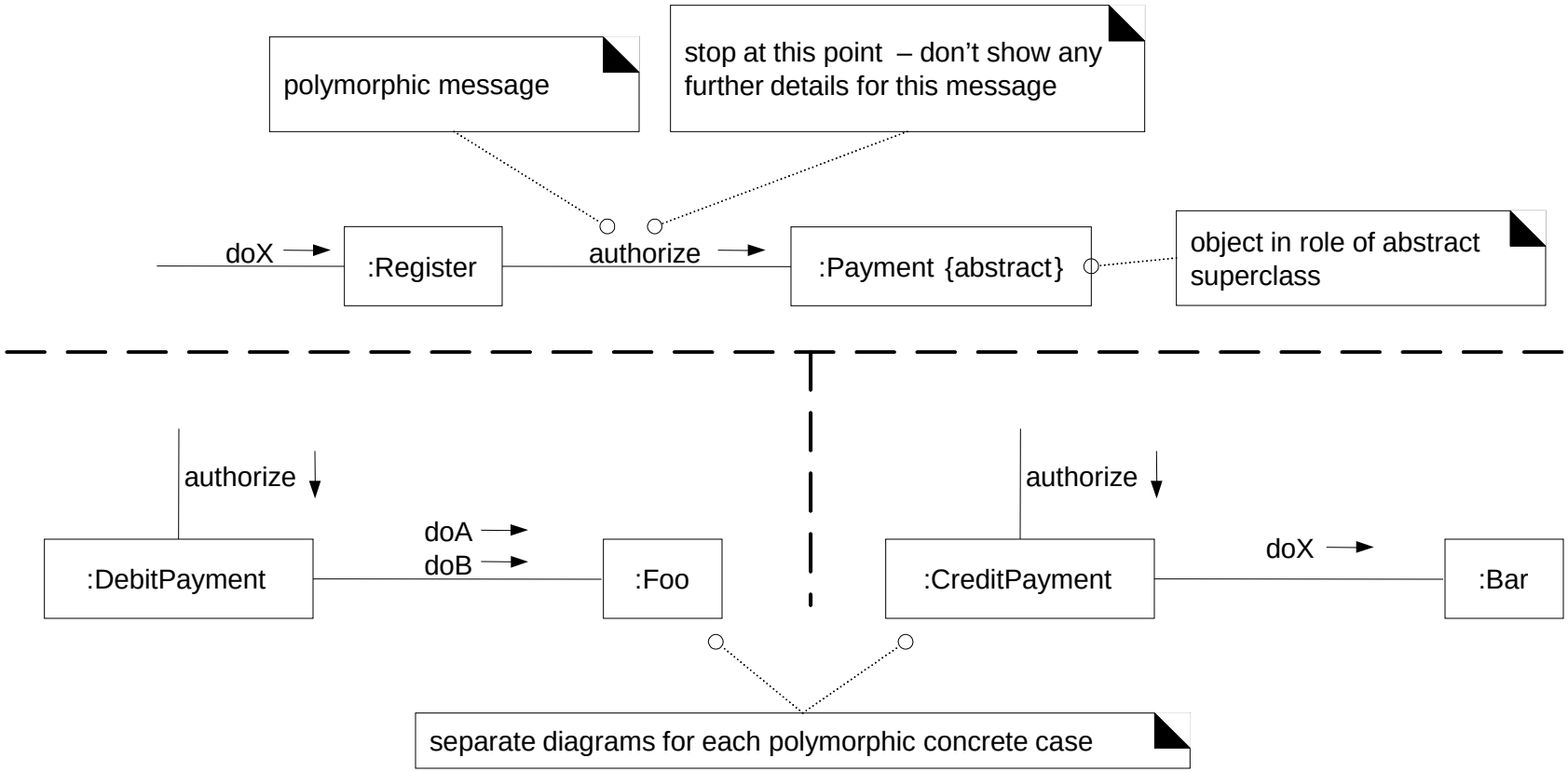
5. 협력도의 기본적인 표기법 – 정적 클래스 또는 메소드

- 순차도 참조하여 이해



5. 협력도의 기본적인 표기법 – 다형적 메시지

- 객체에 {abstract} 태그를 붙여 다형적 메시지임을 표현



5. 협력도의 기본적인 표기법 – 비동기/동기 호출

- 비동기 – 열린 화살표, 대부분 액티브 객체로의 메시지
- 동기적 호출 – 채워진 화살표