Memento Pattern
상태를 보존한다

01. Memeto 패턴

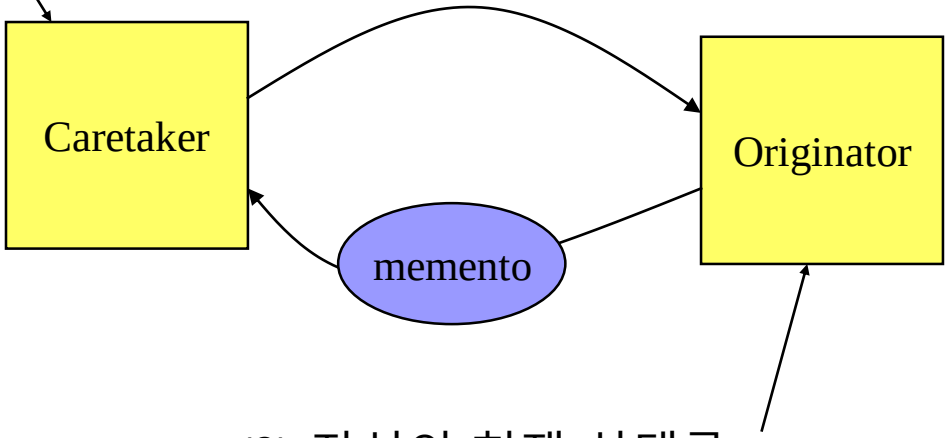
- undo 기능을 제공하려면 이전의 상태를 누군가가 기억하고 있어야 한다.

- Memento 패턴 사용 목적
  - undo(다시 하기)
  - redo(재실행)
  - history(작업 이력의 작성)
  - snapshot(현재 상태의 보존)

- memento의 뜻
  - 기념품, 유품, 추억거리

01. Memeto 패턴

(1) 어떤 시점에 memento를 달라고 요청하면,



(2) 자신의 현재 상태를 memento에 담아서 반환해 준다.

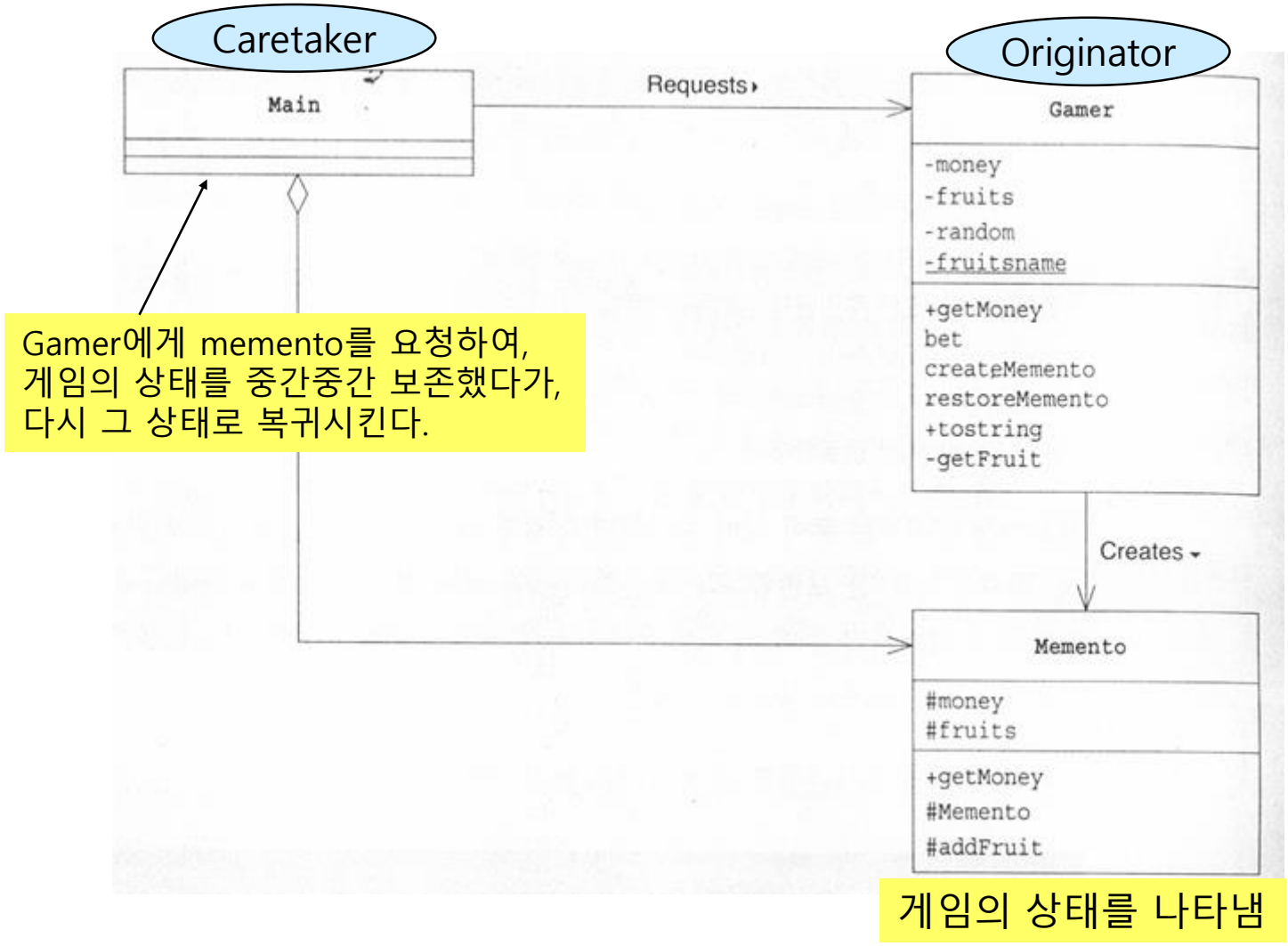
02. 예제 프로그램

- 주사위 게임으로 과일 모으기
  - 게임 규칙
    - 이 게임은 자동으로 진행된다.
    - 게이머가 주사위를 던져 나온 수가 다음 상태를 결정한다.
    - 좋은 수가 나오면 게이머의 돈이 증가한다.
    - 나쁜 수가 나오면 돈이 줄어든다.
    - 특히 좋은 수가 나오면 게이머는 과일을 받는다.
    - 돈이 다 없어지면 종료한다.
  - 돈이 많이 모이면, 장래를 위해 Memento 클래스의 인스턴스를 만들어 '현재의 상태(돈과 과일)'를 보존한다.
  - 계속해서 돈이 줄어들어 돈이 반 미만으로 줄면, Memento의 인스턴스를 사용해서 보존해 두었던 이전 상태로 복귀한다.

02. 예제 프로그램

패키지	이름	해설
game	Memento	Gamer의 상태를 나타내는 클래스
game	Gamer	게임을 하는 주인공의 클래스. Memento의 인스턴스를 만듭니다.
Anonymous	Main	게임을 진행시키는 클래스. Memento의 인스턴스를 보존해 두고 필요에 따라서 Gamer의 상태를 복원합니다.

02. 예제 프로그램


Caretaker
Originator
Main
Requests
Gamer
-money
-fruits
-random
-fruitsname
+getMoney
bet
createMemento
restoreMemento
+tostring
-getFruit
Creates
Memento
#money
#fruits
+getMoney
#Memento
#addFruit
Gamer에게 memento를 요청하여,
게임의 상태를 중간중간 보존했다가,
다시 그 상태로 복귀시킨다.
게임의 상태를 나타냄

02. 예제 프로그램

- Memento 클래스
  - 게이머의 상태를 표현하는 클래스
  - 객체의 상태 == 속성 값들의 집합
    - money 필드: 현재 소유한 돈
    - fruits 필드: 현재 가지고 있는 과일
  - 생성자에 public이 없다 => package 접근 권한
    - 같은 패키지에 속하는 클래스에서만 호출할 수 있다.
  - addFruit(String)
    - 과일을 추가할 때 호출하는 메소드
    - package 접근 권한 => game 패키지 외부에서는 이 클래스의 내부 상태를 변경할 수 없다.
  - getMoney( )

02. 예제 프로그램

- Gamer 클래스
  - 게임을 수행하는 게이머를 표현하는 클래스
  - 돈과 과일, 난수 발생기, 과일 이름 배열을 가진다.
  - bet( )
    - 1이 나오면 돈이 100원 증가한다.
    - 2가 나오면, 돈이 반으로 준다.
    - 6이 나오면, 과일을 받는다.
  - createMemento( )
    - 이 메소드가 호출되면, 현재 상태를 보존하는 Memento 객체가 생성된다.
    - 현재 돈과 맛있는 과일들 만을 Memento 객체에 보존한다
    - 찰칵하고 사진을 찍듯이 현재의 상태를 Memento 인스턴스에 보존해 두는 것이다.

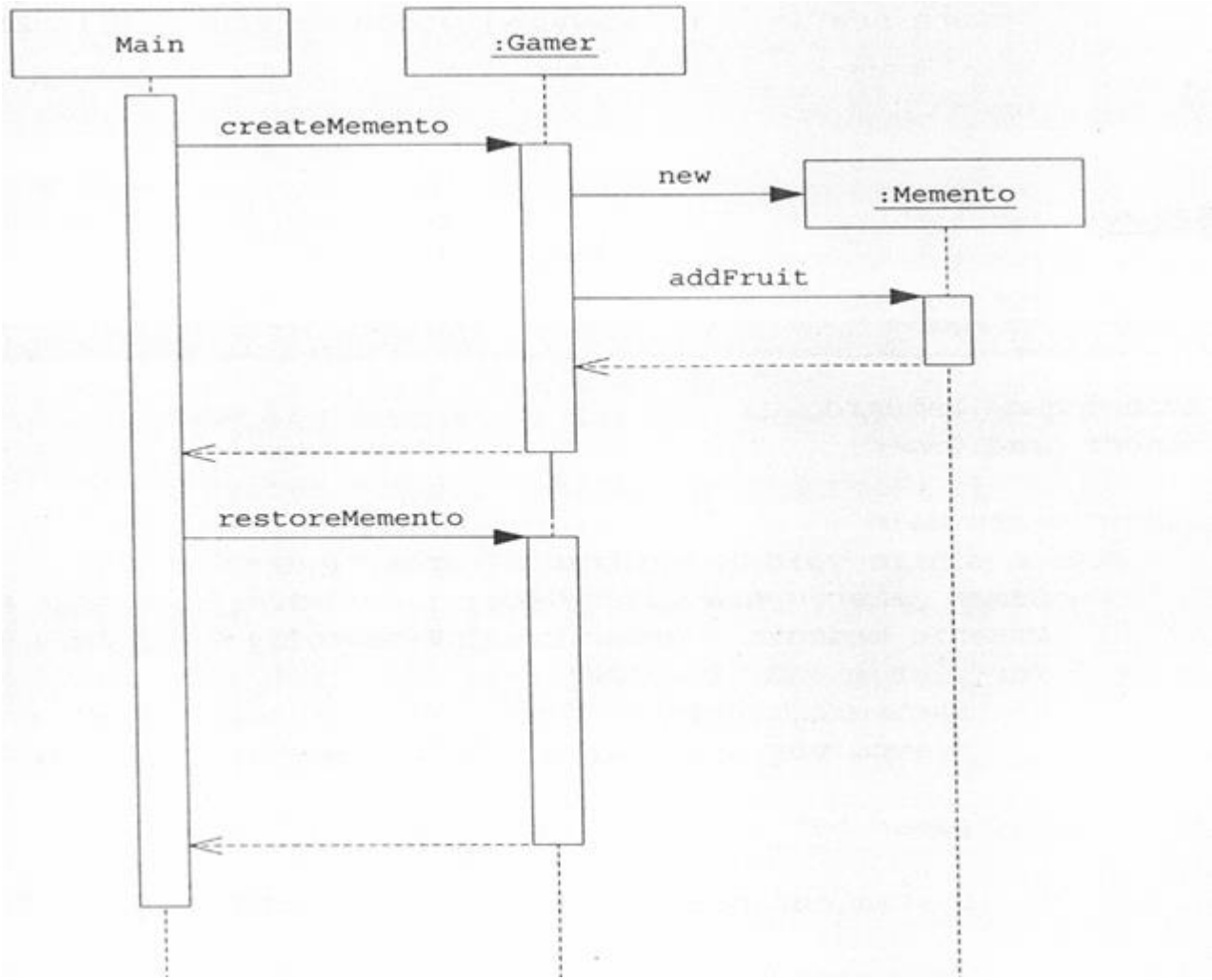
02. 예제 프로그램

- Gamer 클래스(계속)
  - restoreMemento( )
    - undo를 행하는 메소드
    - 입력인자로 들어온 Memento 객체로부터 보존되었던 돈과 과일을 얻어온다.
  - getFruit( )
    - 선택적으로 '맛있다'를 과일이름 앞에 붙인다.

02. 예제 프로그램

- Main 클래스
  - gamer.createMemento( )를 호출하여 현재 gamer의 상태를 얻어와서 보존한다.
  - 현재 게이머의 돈이 Memento 객체의 돈보다 크면
    - 현재 상태를 Memento 객체에 보존한다.
  - 현재 게이머의 돈이 Memento 객체의 돈의 반보다 작으면
    - Memento 객체에 보존된 이전 상태로 복귀한다.

02. 예제 프로그램


Main
:Gamer
createMemento
new
:Memento
addFruit
restoreMemento

03. 등장 역할

- Originator(작성자)의 역할
  - Memento 역할을 만들어 자신의 현재 상태를 반환하는 역할을 한다.
  - 또한, Memento로부터 이전 상태를 받아서 복귀하는 일을 한다.
  - 예제에서는, Gamer 클래스가 해당됨

- Memento(기념품)의 역할
  - Originator 역할의 내부 정보를 보존한다.
  - 예제에서는 Memento 클래스가 해당됨

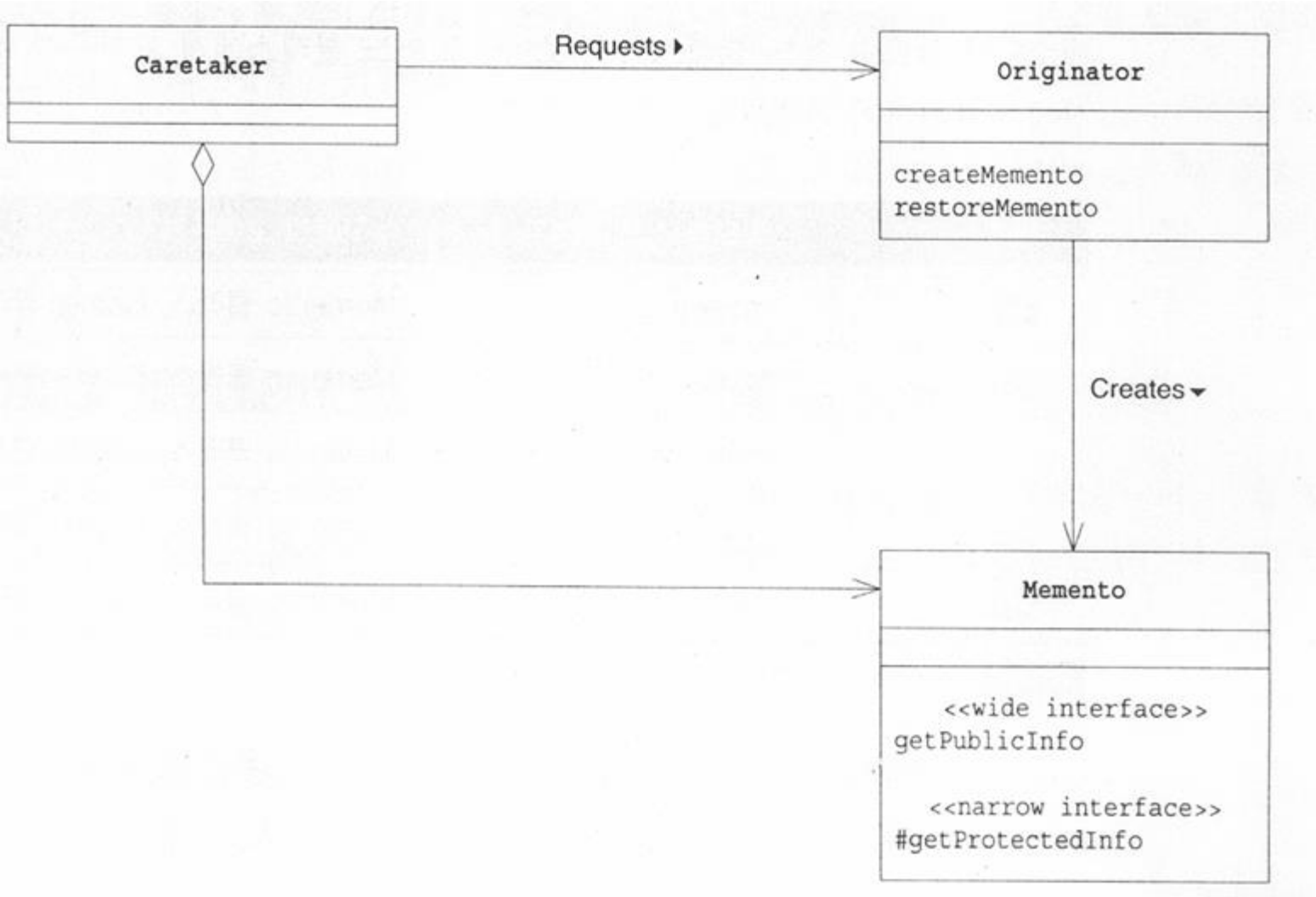
03. 등장 역할

- Caretaker(돌보는 사람)의 역할
  - Originator 역할의 상태를 보존하고 싶을 때 Originator에게 알리는 역할을 담당한다.
  - 예제에서는, Main 클래스가 해당됨

- Originator 역할과 Memento 역할은 견고하게 결속되어 있다.

- Caretaker 역할과 Memento 역할은 느슨하게 연결되어 있다.
  - Caretaker 역할이 Memento 내부의 정보를 쉽게 접근할 수 없다.

03. 등장 역할


Caretaker
Requests
Originator
createMemento
restoreMemento
Creates
Memento
<<wide interface>>
getPublicInfo
<<narrow interface>>
#getProtectedInfo

04. 독자의 사고를 넓혀주는 힌트

- 액세스 제어

액세스 제어	해설
public	모든 클래스에서 보입니다.
protected	그 패키지 및 하위 클래스에서 보입니다.
없음	그 패키지에서만 보입니다.
private	그 클래스에서만 보입니다.

  - Memento 클래스의 생성자는 아무것도 안 붙어 있음
    - Main 클래스는 Memento의 생성자를 호출할 수 없다.
    - Main 클래스 안에서 Memento 객체를 생성할 수 없다.
    - Memento가 필요할 때 마다, Gamer 클래스의 createMemento를 호출한다.

04. 독자의 사고를 넓혀주는 힌트

- Memento를 몇 개 가질까?
  - 예제에서는, Main 클래스가 가지고 있는 Memento는 하나뿐이었다.
  - 배열 등을 사용하면 Main이 여러 개의 Memento 인스턴스를 생성해서 가지고 있을 수 있다.
    - 즉, 여러 시점의 상태를 보존해 둘 수가 있다.

04. 독자의 사고를 넓혀주는 힌트

- Caretaker 역할과 Originator 역할을 분리하는 의미
  - "undo를 하고 싶으면, Originator 역할에게 그 기능을 만들어 넣으면 되지, 일부러 디자인 패턴을 사용할 필요가 있을까?"
  - Caretaker의 역할
    - 어느 시점에서 스냅샷을 찍어, 언제 undo를 실행할 지 결정함
    - Memento 역할을 보관하는 일을 함
  - Originator의 역할
    - Memento 역할을 생성하는 일을 함
    - Memento 역할을 사용해서 자신의 상태를 원래대로 되돌리는 일을 함
  - 분리해 두면 좋은 점
    - 다음과 같은 수정을 할 때, Originator 역할은 변경할 필요가 없다.
      - 여러 단계의 undo를 실행하도록 변경하고 싶다
      - undo만 하는 것이 아니라, 현재의 상태를 파일에 보존하고 싶다.

05. 관련 패턴

- Command 패턴(22장)
- Prototype 패턴(6장)
- State 패턴 (19장)

06. 요약

- 현재 객체의 상태를 기록하여 보존하기 위한 Memento 패턴
  - Caretaker 역할은 중간 중간 Originator 역할에게 부탁하여 '현재의 상태'를 표현하는 Memento 역할을 만든다.
  - Caretaker 역할은 필요할 때 서랍 안에서 Memento 역할을 꺼내서 Originator 역할에게 건네주면, 그 상태로 복원이 된다.