Prototype Pattern
복사해서 인스턴스를 만든다

01. Prototype 패턴

- 일반적인 인스턴스 생성은, 클래스의 생성자를 호출한다.

```
new Something( );
```

  - 인스턴스 생성시에 반드시 클래스 이름을 지정해야 한다.

- 의도
  - 클래스로부터 인스턴스를 새로 만드는 것이 아니라, 현재 존재하는 인스턴스를 복사(복제)해서 새로운 인스턴스를 만들 필요가 있을 때, 이 작업을 편하게 하기 위해 사용한다.

01. Prototype 패턴

- Prototype
  - 원형, 규범
  - "규범이 되는 인스턴스를 근본으로 해서 똑같은 새로운 인스턴스를 만든다"
  - 클래스 안에, 자신을 복사하는 메소드를 두자.
    - 복제(복사): 모든 필드 값이 동일한 인스턴스를 생성함

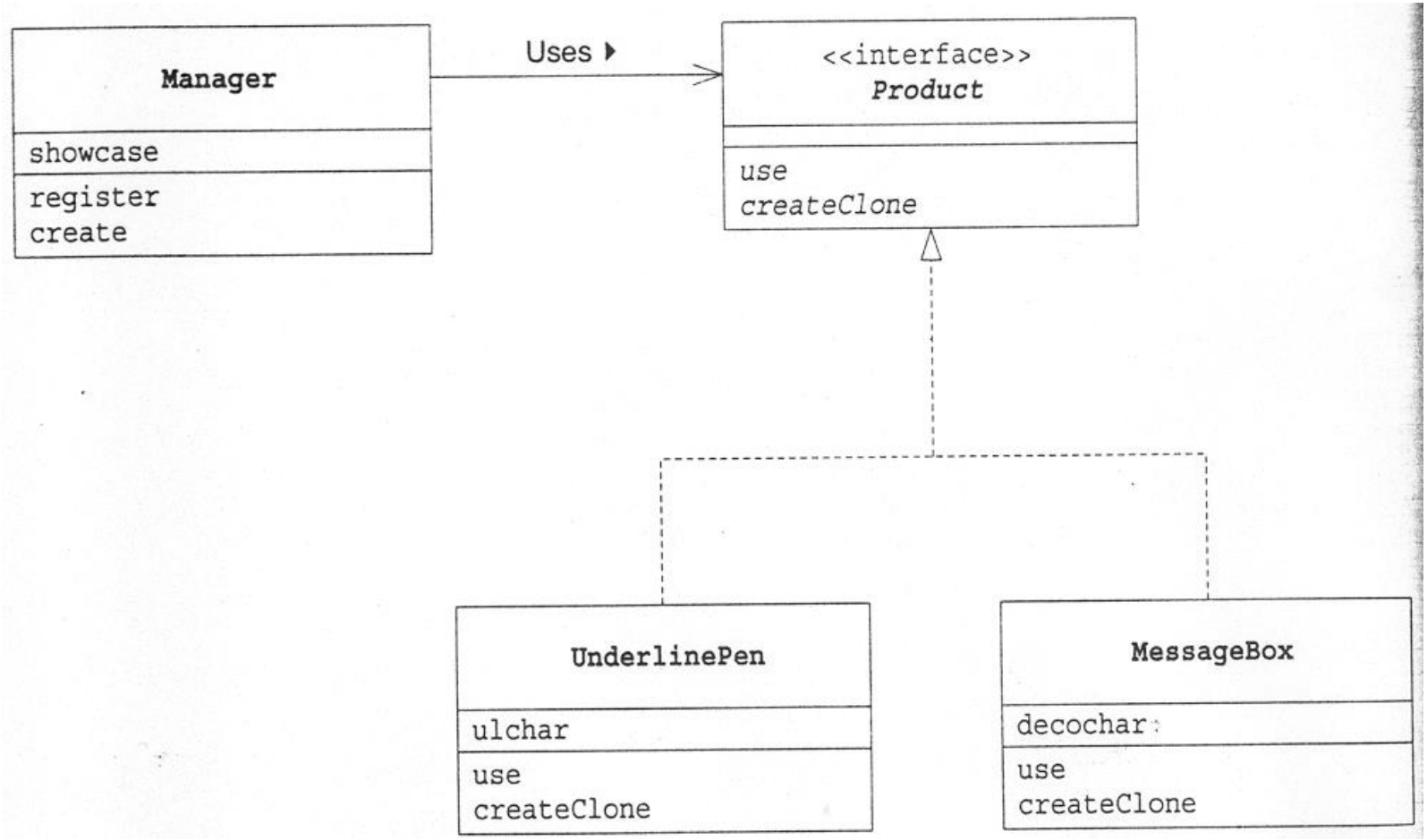
02. 예제 프로그램

- 문자열을
  - 틀에 넣어서 표시하거나
  - 밑줄을 그어 표시

패키지	이름	해설
framework	Product	추상 메소드 use와 createClone이 선언되어 있는 인터페이스
framework	Manager	createClone을 사용해서 인스턴스를 복제하는 클래스
Anonymous	MessageBox	문자열을 틀에 넣어 표시하는 클래스. use와 createClone을 구현하고 있습니다.
Anonymous	UnderLinePen	문자열에 밑줄을 그어 표시하는 클래스. use와 createClone을 구현하고 있습니다.
Anonymous	Main	동작 테스트용 클래스

02. 예제 프로그램

- 클래스다이어그램



02. 예제 프로그램

- Product 인터페이스
  - java.lang.Cloneable 인터페이스를 상속함
    - 이를 구현한 클래스는, clone( ) 메소드를 이용해서 복제를 만들어 낼 수 있다.
  - use( )
    - '사용'을 위한 메소드
    - '사용'이 실제 무엇을 의미하는지는 하위 클래스의 구현이 결정함

02. 예제 프로그램

- Manager 클래스
  - Product 인터페이스를 이용해서 인스턴스를 복제하는 일을 함
  - showcase 필드
    - java.util.Hashtable (name-value 쌍들을 저장하고 관리하는 자료구조)
    - Product 의 '이름'과 '인스턴스'를 저장함
  - register( ) 메소드
    - 제품의 '이름'과 제품의 '인스턴스'를 showcase 에 저장함
  - create( ) 메소드
    - 등록된 제품의 creatClone( )을 호출하여 복사본을 만든 다음, 이것을 반환한다.

02. 예제 프로그램

- Product 인터페이스나 Manager 클래스의 소스에, 구체적인 제품인 MessagBox나 UnderlinePen 클래스의 이름이 전혀 등장하지 않는다
  - Manager 클래스는, 구체적인 클래스 이름을 사용하지 않고, Product 인터페이스 이름만을 사용한다.
  - framework와 구체적인 클래스를 분리함
  - Product나 Manager를, 구체적인 클래스와 상관없이 수정할 수 있다.

02. 예제 프로그램

- MessageBox 클래스
  - Product 인터페이스를 구현
  - decochar 필드
    - 문자열을 둘러쌀 장식 문자
  - use( ) 메소드
    - 주어진 문자열을 decochar로 둘러싸서 출력하는 일을 한다.

02. 예제 프로그램

- MessageBox 클래스
  - createClone( ) 메소드
    - 자기 자신을 복제하는 메소드 (자신의 clone( )을 호출함)
    - clone( ): 인스턴스가 가지고 있는 필드 값이 그대로 복사된 복제 인스턴스를 반환한다.
      - java.lang.Cloneable 인터페이스를 구현한 클래스만이 이 clone( ) 메소드를 가진다.
      - 이 인터페이스를 구현하지 않는 경우에는, CloneNotSupportedException이 발생한다.
    - clone( ) 메소드는, 자신의 클래스(및 하위 클래스)에서만 호출할 수 있다.
      - 다른 클래스의 요청으로 복제하는 경우에는, createClone과 같은 다른 메소드로 clone을 감싸주어야 한다.

02. 예제 프로그램

- UnderlinePen 클래스
  - MessageBox와 거의 같은 동작을 함
  - 문자열에 밑줄을 그어 준다.
  - ulchar 필드
    - 밑줄 그을 때 사용할 문자를 가지고 있음

02. 예제 프로그램

- Main 클래스
  - 1. Manager의 인스턴스를 생성
  - 2. UnderlinePen 인스턴스와 MessageBox 인스턴스를 이름을 붙여서 등록한다.
  - 3. Manage의 create( ) 메소드를 호출해서, 원하는 '이름'의 제품을 얻어서, 그것의 use( )를 실행한다.

이름	클래스와 인스턴스의 내용
"strong message"	UnderlinePen에서 ulchar는 '~'
"warning"	MessageBox에서 decochar는 '*'
"slash box"	MessageBox에서 decochar는 '/'

03. 등장 역할

- Prototype(원형) 역할
  - 인스턴스를 복사(복제)해서 새로운 인스턴스를 만들기 위한 메소드를 결정함
  - 예제에서는, Product 인터페이스에 해당됨
- ConcretePrototype(구체적인 원형) 역할
  - 인스턴스를 복사(복제)해서 새로운 인스턴스를 만드는 메소드를 실제로 구현함
  - 예제에서는, MessageBox나 UnderlinePen 클래스가 해당됨
- Client(이용자)의 역할
  - 인스턴스를 복사하는 메소드를 이용해서 새로운 인스턴스를 만듬
  - 예제에서는, Manager 클래스가 해당됨

05. 관련 패턴

- Flyweight 패턴
  - Prototype 패턴은 현재의 인스턴스와 같은 상태의 다른 인스턴스를 만들어 이용함
  - Flyweight 패턴은 하나의 인스턴스를 여러 장소에서 공유하여 이용함
- Memento 패턴
  - Prototype 패턴에서는 현재의 인스턴스와 같은 상태의 다른 인스턴스를 만듦
  - Memento 패턴에서는 스냅샷과 undo를 실행하기 위해 현재의 인스턴스의 상태를 보존함
- Composite 패턴 및 Decorator 패턴
  - Composite 패턴 및 Decorator 패턴을 많이 이용하면 복잡한 구조의 인스턴스가 동적으로 만들어지는 경우가 있음
  - 이런 경우에 Prototype 패턴을 사용하면 편리함
- Command 패턴
  - Command 패턴에 등장하는 명령을 복제하려고 할 때 Prototype 패턴이 사용되는 경우가 있음

06. 보강: clone( ) 메소드와 java.lang.Cloneable 인터페이스

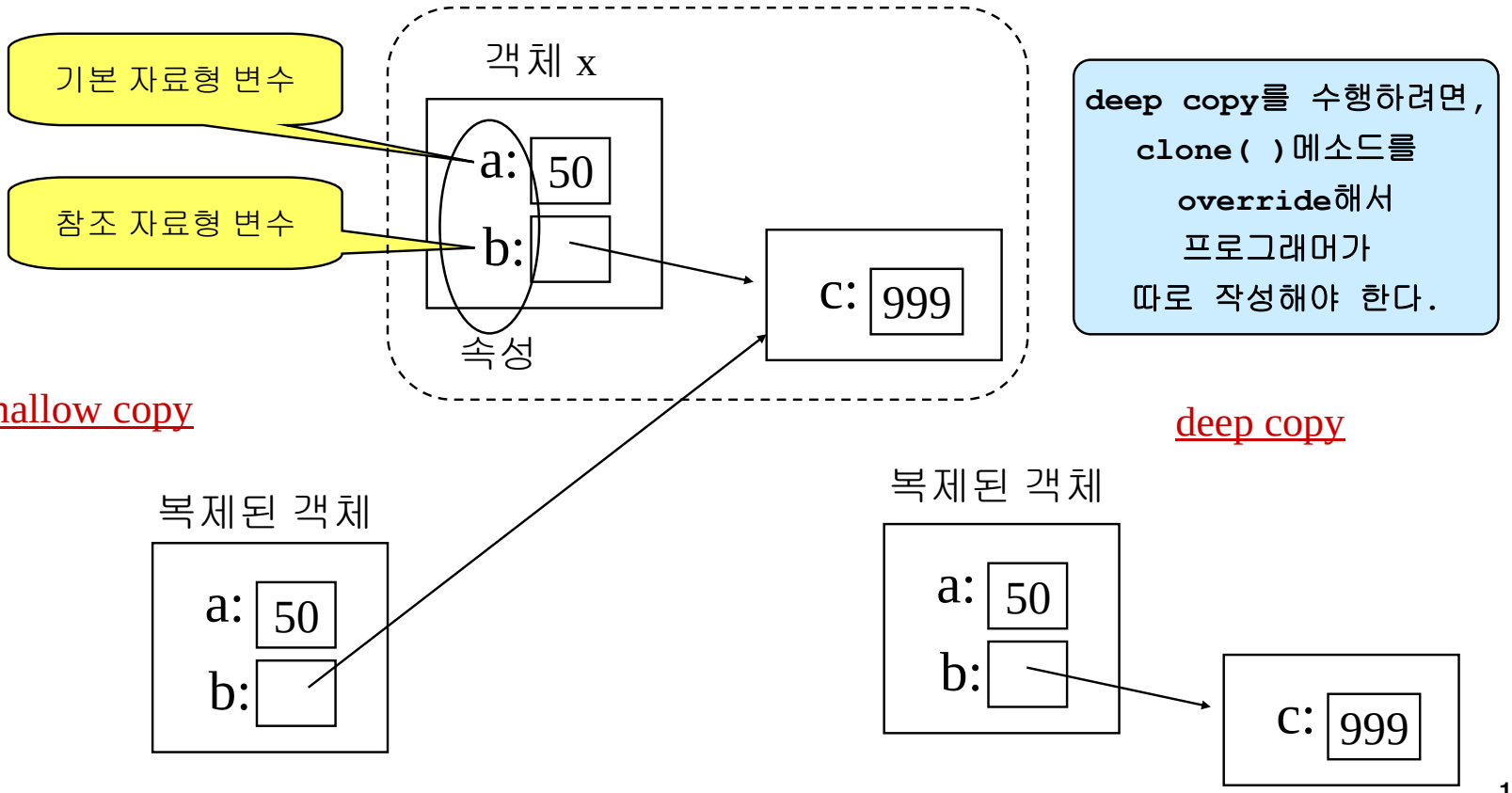
- 자바 언어의 clone
  - 인스턴스를 복사하는 장치로 clone( )메소드가 제공된다.
    - 이 메소드는 자기 자신만 호출할 수 있다.
  - 복사 대상이 되는 클래스는, 반드시 java.lang.Cloneable 인터페이스를 구현해야 한다.
  - Cloneable 인터페이스를 구현한 클래스의 인스턴스는, clone( ) 메소드를 호출하면 복사된다.
  - Cloneable 인터페이스를 구현하고 있지 않은 클래스의 인스턴스가 clone( )메소드를 호출하면, CloneNotSupportedException 예외가 발생한다.

06. 보강: clone( ) 메소드와 java.lang.Cloneable 인터페이스

- clone 메소드는 어디에서 정의되어 있는가?
  - 최상위 클래스인 java.lang.Object에 정의되어 있다.
  - 모든 클래스에서 clone 메소드를 상속함
- Cloneable 인터페이스에
  - clone( ) 메소드가 선언되어 있는 것은 아니다.
  - Cloneable 인터페이스는, clone( )에 의해 복사될 수 있다는 표시로만 사용된다.

06. 보강: clone( ) 메소드와 java.lang.Cloneable 인터페이스

- clone( ) 메소드는 'shallow copy'를 수행한다.
  - 즉, 속성의 값만을 복사한다.
  - 속성이 참조자료형인 경우, 속성이 가리키는 객체를 복사하지는 않는다.



07. 요약

- 인스턴스 복제를 용이하게 하기 위해서 Prototype 패턴을 이용하라.