다양한 게임 캐릭터 설계를 위한 효과적인 클래스 합성에 대한 연구

김종수*, 김태석
동의대학교 소프트웨어공학과
e-mail:seatree@deu.ac.kr

A Study on the Effective Class Composition for the Various Game Character Design

Jong-Soo Kim*, Tai-Suk Kim
Dept of Software Engineering, Dong-Eui University

요 약

현재 인터넷을 기반으로 일반인들에 많은 인기를 얻고 있는 게임의 대부분은 대규모 클라이언트를 수용하는 게임이다. 이러한 게임에서 개발자들은 클라이언트들에게 더 많은 흥미를 주기 위하여 다양한 형태의 캐릭터를 제공한다. 게임에서 사용되는 캐릭터는 인간을 닮은 것도 있지만, 의인화된 동물이나 사물도 많다. 그리고 게임에서 다양한 캐릭터들의 행동도 캐릭터에 부여되는 특성만큼 다양하다. 이러한 캐릭터들을 소프트웨어적으로 설계하기 위해서는 객체지향적인 언어의 사용이 많은 편리함을 제공해준다.

본 논문에서는 롤플레잉게임(Role-Playing Game)에서 보편적으로 등장하는 몇몇 캐릭터를 분석하여 클래스를 설계하고, 이들 클래스의 상속과 합성 기법을 통하여 게임 소프트웨어 설계에 효율적으로 사용할 수 있는 설계기법과 인터페이스를 이용해 캐릭터 클래스를 한 개의 형(Type)으로 묶을 수 있는 설계 기법을 제안한다.

1. 서론

네트워크 온라인 게임의 대부분은 사용자들에게 다양한 캐릭터를 제공하여 보다 흥미로운 게임을 할 수 있도록 해준다. 게임에서 사용되는 캐릭터는 인간에서부터 의인화된 개와 고양이와 같은 동물 그리고 버섯, 인조인간과 같은 다양한 몬스터들이 있다.

캐릭터들만이 표현할 수 있는 다양한 행동양식을 생각해 보면, 네트워크를 기반으로 하는 대규모 온라인 게임에서는 캐릭터들이 다양한 행동을 하도록 설계하는 것이 중요함을 알 수 있다. 그러나 인터넷을 이용하는 국내의 네트워크형 온라인 게임 개발 기술은 세계적으로 인정받고 있지만, 그러한 높은 기술력에도 불구하고 게임 소프트웨어의 특성상 설계와 관련된 자료의 공유가 힘들다[1].

게임 설계에서 캐릭터의 다양한 특성을 표현하기 위해 C++와 같은 객체지향언어를 사용하는 것은 여러모로 편리한 점을 제공해 준다. 이것은 코드의 재사용이라는 객체지향 언어가 가지는 장점이 때문이다[2].

본 논문에서는 인터넷을 이용하는 대규모 네트워크 롤플레잉게임(Role Playing Game)에서 보편적으로 등장하는 몇몇 캐릭터를 분석하여 클래스를 설계하고, 이들 클래스를 정련하는 과정에서 적용될 수 있는 추상화 기법을 통해서 상속 계층을 만들고, 클라이언트가 유연한 코드를 만들 수 있도록 하기 위한 합성을 통하여 게임 소프트웨어 설계에 효율적으로 사용할 수 있는 설계기법을 제안한다. 그리고 인터페이스의 사용을 통하여 게임 내에서 사용되는 캐릭터를 한 개의 형(Type)으로 묶을 수 있는 설계의 예를 보인다.

2. 온라인 게임의 캐릭터 요구 사항 분석

대규모 네트워크 게임에 등장하는 캐릭터는 여러

가지 종류가 있다. <표 1>은 게임 플레이어가 조작하는 대표적인 인간 캐릭터를 보여준다.

<표 1 > 인간 캐릭터

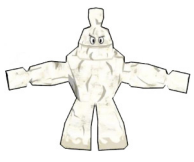
검사(Swordsman)	법사(Wizard)
	

RPG 게임에서 플레이어는 자신이 선택한 특정 캐릭터를 조작하여 역할을 수행함에 따라 경험치를 쌓고, 경험에 따른 레벨을 올리는 것이 주된 게임의 목표다. 캐릭터의 레벨이 높아짐에 따라 선택하게 되는 다양한 직업이 있다. 일반적으로 검사, 마법사, 궁수와 같은 직업이 있으며, 특정 캐릭터가 가지는 직업에 따라서 다른 직업을 가진 캐릭터와 차별화된 행동양식을 보인다. 이러한 직업은 레벨이 높아지면서 세분화되기도 한다. 게임의 사용자가 선택하여 조작하는 캐릭터에는 레벨에 맞게 다양한 무기와 방어구를 갖출 수 있다.

또한 캐릭터들 중에는 인간을 닮은 괴물도 흔히 등장하는데, 인조인간, 골렘(Golem), 마계인간, 신족과 같은 것이 있다. 그리고 특정한 캐릭터를 따라다니는 애완동물과 같은 개념의 캐릭터도 있는데, 개나 고양이, 그리고 독수리나 매와 같은 것이다.

게임 플레이어가 조작하는 인간 캐릭터는 자신이 가지고 있거나 게임을 통하여 획득한 다양한 무기와 기술을 사용하여 자신의 레벨을 올리기 위해서 <표 2>와 같은 여러 가지 몹(mob)이나 몬스터와 같은 캐릭터를 사냥할 수 있다.

<표 2 > 인간 캐릭터

버섯 몬	골렘
	

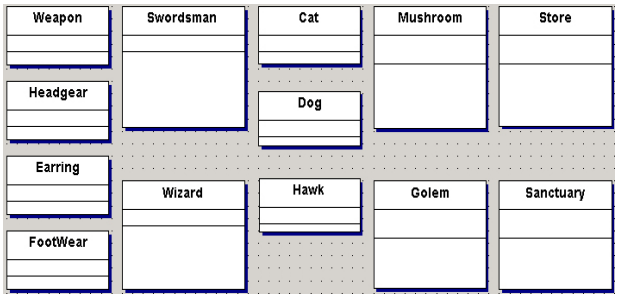
또 게임을 진행을 위해 전직을 하기 위한 건축물, 물건을 사고팔기 위한 상점과 같은 건축물과 같은 캐릭터가 필요하다.

3. 툴(Tool)을 사용한 클래스 설계

게임 캐릭터와 관련된 클래스를 설계하기 위해서 Together와 같은 객체지향 소프트웨어 설계 도구를 사용하는 것은 매우 효율적이다. Together에는 소프트웨어를 객체 지향적으로 설계하기 위한 많은 기능과 다이어그램을 제공하지만, 클래스다이어그램을 주로 사용하였다. 본 장에서는 2장의 요구 분석을 토대로 클래스를 추출하여 설계하고 합성해나가는 과정을 보여준다.

3.1 도메인에서 클래스 추출

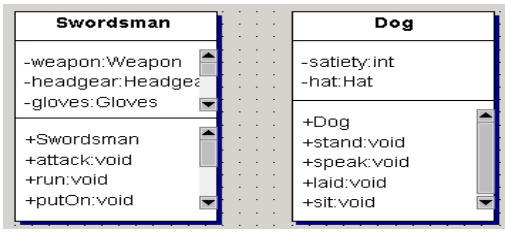
2장에서 정의된 요구분석을 통하여 검사, 마법사, 개, 고양이, 버섯등과 같이 클래스가 될 수 있는 객체를 추출하면 (그림 1)과 같다.



(그림 1) 문제영역에서 추출된 클래스

요구사항에서 선별된 명사형들 중에서 클래스로 선택된 것과 그렇지 않은 것 그리고 요구사항에는 없으나 효율적인 애플리케이션 설계를 위해 추가된 클래스가 있다.

(그림 2)는 요구사항에서 추출된 클래스 중 인간 캐릭터인 검사를 구현하는 Swordsman(검사) 클래스와 애완동물을 구현하는 Dog 클래스가 반복과 정련 단계를 거치면서 추가된 속성과 메소드를 보여준다.



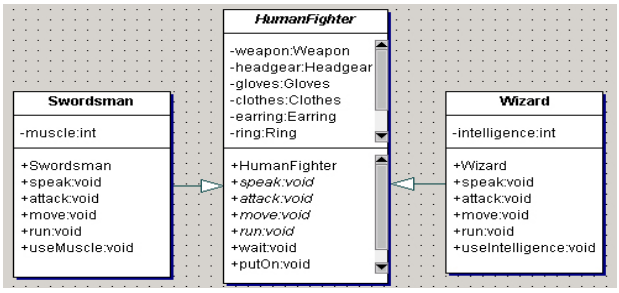
(그림 2) 클래스의 속성과 메소드 결정

다이어그램에서 Swordsman 클래스에 추가된 많

은 멤버들이 Wizard 클래스에도 나타나며, 마찬가지로 Dog 클래스에서 추가된 많은 멤버들이 Cat이나 Hawk 클래스에 나타난다. 이렇게 중복되어서 나타나는 클래스의 멤버들은 추상화 개념을 적용한 상속을 통하여 다시 정련할 필요가 있다.

3.1 추상화를 통한 클래스 상속

게임 플레이어가 조작하는 캐릭터는 축척한 경험치에 따른 직업 변환 시스템이 있다. 요구사항 분석에서 나타난 검사, 마법사, 궁수와 같은 직업이다. 이러한 직업을 가진 캐릭터는 (그림 2)의 Swordsman 클래스에서 나타난 많은 멤버들을 같이 공유한다. 기존에 설계된 Swordsman 클래스를 재사용하기 위해서 추상화 개념을 적용하여 클래스를 다시 설계할 수 있다. (그림 3)은 검사와 마법사 클래스의 추상화 작업 결과를 보여준다.



(그림 3) 검사와 마법사 캐릭터의 추상화

다이어그램에서 Swordsman과 Wizard 클래스는 추상 클래스 HumanFighter를 상속 받으며, 추상메소드인 speak(), attack(), move(), run()을 구현해야 한다. 그리고 Swordsman 클래스는 추가적인 속성 muscle을 Wizard 클래스는 intelligence를 가지고 있고 또한 각각의 클래스는 속성과 관련된 메소드인 useMuscle()과 useIntelligence()를 가진다.

위와 같이 추상클래스를 통한 상속은 상위 클래스의 속성과 메소드의 구현을 그대로 재사용한다는 면에서 장점이 있다. 그러나 HumanFighter 클래스의 추상메소드가 변경되면, 상속을 받은 Swordsman과 Wizard 클래스의 메소드도 같이 수정되어야 한다는 단점이 있다. HumanFighter 클래스가 가진 추상메소드인 speak()가 어떤 이유에서 Stirng()을 반환해야한다고 가정하자.

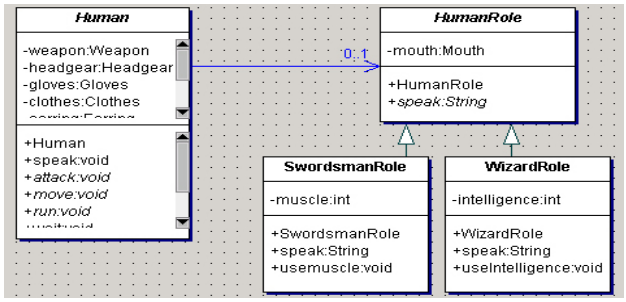
그러면 추상메소드를 구현해야 하는 Swordsman과 Wizard 클래스의 speak() 메소드들도 String을 반환하도록 코드를 변경해야 한다.

이렇게 자식클래스의 코드를 직접 변경해야 하는 것은 잦은 설계의 변경이 있는 경우나 많은 자식 클래스가 있는 경우 문제가 된다. 또 위의 경우 게임 플레이어가 얻은 경험치에 따라 검사는 마법사가 세분화된 다른 직업으로 전직하는 경우도 고려해야한다.

게임에 사용되는 캐릭터들 중에는 시간에 따라 객체의 상태가 변할 수 있는 경우가 있다. 한 개의 캐릭터가 검사나 마법사도 될 수 있고, 더 많은 기술을 사용하는 검사나 마법사로 전직을 할 수 있도록 설계하는 것이 좋다. 상속을 사용하는 경우 상속계층도의 수평에 있는 하위 클래스들은 호환될 수 없기 때문에 검사는 영구적으로 검사가 되고, 마법사나 궁수가 될 수 없게 된다. 왜냐하면 하위클래스는 서로 호환되지 않기 때문이다.

3.2 합성을 이용한 클래스 설계

한 개의 캐릭터가 전직을 가능하게 하기 위해서는 클래스가 합성되도록 설계하는 것이 효과적이다. (그림 4)는 클래스 합성의 예를 보여 준다.



(그림 4) 클래스의 합성을 통한 설계 예

다이어그램의 SwordsmanRole과 WizardRole 클래스를 이용하면, 검사는 검사의 고유한 역할을 할 수 있으며, 마법사도 고유한 역할을 할 수 있다.

또한 각각의 클래스가 공통적으로 수행할 수 있는 메소드 speak()가 있는데, SwordsmanRole 클래스와 WizardRole 클래스가 추상클래스인 HumanRole를 상속받아서 speak()를 구현하면, 캐릭터에 따른 관련 행위가 다르게 나타나게 된다.

만약 새로운 직업인 궁수가 추가된다고 하더라도 궁수역할 클래스를 추가하고 HumanRole 클래스를 상속 받으면 된다.

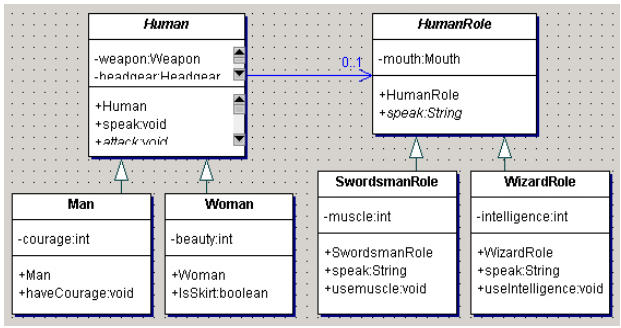
클래스의 합성을 통하여 (그림 3)에서 제시된

HumanFighter 클래스의 메소드 speak()의 구현을 (그림 4)의 HumanRole 클래스에 위임함으로 해서 Human 클래스의 speak() 메소드가 변경이 일어나더라도 검사와 마법사의 speak() 행동에는 변화가 생기지 않는다는 장점이 있다.

3.3 인터페이스를 이용한 설계

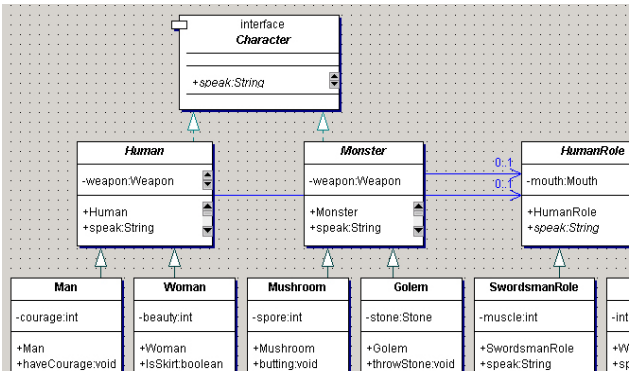
Human 캐릭터가 남자와 여자 캐릭터로 크게 나누어지고 각각이 다른 행동양식을 취하게 해야 할 경우, (그림 5)와 같은 설계가 가능하다.

남자와 여자 캐릭터는 "Man is a Human" 이라는 관계가 성립하므로 Human 클래스로부터 상속을 받을 수 있다.



(그림 5) 남자와 여자 캐릭터가 추가된 설계

다이어그램에서 Man과 Woman 캐릭터는 Human 클래스를 상속 받으므로 speak()를 구현하는데 아무런 문제가 없다. 캐릭터들 중에는 의인화된 Mushroom과 Golem과 같은 몬스터 캐릭터들도 있는데, 이러한 캐릭터들도 speak()와 같은 행위를 해야 한다고 가정하자. 이러한 경우 인터페이스를 사용한 설계가 효율적으로 사용될 수 있다. (그림 6은) Character 인터페이스를 이용한 설계를 보여준다.



(그림 6) 인터페이스를 이용한 몬스터 클래스 추가

추가된 Mushroom과 Golem 클래스는 추상 클래스 Monster를 상속 받는다. 또 Monster 클래스는 Character 인터페이스를 구현함으로써, Character 인터페이스를 이용하면, 게임에서 사용되어지는 인간, 동물, 몬스트와 같은 모든 캐릭터를 같은 형(Type)으로 묶을 수 있다. 그리고 각각의 캐릭터는 모두 speak()라는 인간이 가진 말하는 행위를 할 수 있게 된다.

4. 결론

본 논문에서 게임에서 사용되는 캐릭터의 종류와 각각이 가져야 하는 기능에 대한 요구분석을 기초로 클래스를 추출하였다. 그리고 추출된 클래스의 제련을 통하여 클래스의 속성과 행동양식을 결정하였다.

객체 지향 언어의 가장 큰 장점이라고 할 수 있는 코드의 재사용과 효율적인 구현을 위해서 상속을 통한 클래스의 추상화기법을 보였다. 클래스의 상속계층을 만드는 것은 객체지향언어가 가지는 큰 특징 중의 하나인 다형성(Polymorphism)을 구현할 수 있게 해 준다.

그리고 상속 계층에 있는 클래스들 사이에서 부모클래스의 메소드를 상속 받아 정의해야 하는 자식클래스가 부모클래스의 메소드가 변경됨으로써 코드를 수정해야 하는 잦은 변경 문제를 해결하기 위하여 클래스를 합성하는 효율적인 설계 기법을 보였다.

또한 게임에서 사용되는 다양한 캐릭터를 하나의 형(Type)으로 묶기 위해서 Character 라는 인터페이스를 이용한 설계기법을 보였다.

본 논문에서 네트워크 게임을 위한 효율적인 클래스의 상속과 합성에 대해서 제안 하였지만 게임과 관련된 여러 API의 구현에 효율적으로 적용될 수 있는 디자인 패턴에 대한 연구가 추가적으로 필요하다.

참고문헌

[1] 남재욱, "온라인 게임 서버 프로그래밍", 한빛미디어, pp. 18-30, 2004.

[2] 박지훈, "자바 디자인패턴과 리팩토링", 한빛미디어, pp. 144-177, 2003.