객체지향 언어 교육을 위한 객체지향 개념 모델링에 관한 연구

이민나 †
† 한국외국어대학교

A Study on Object-Oriented Concepts modeling for Teaching Object-Oriented Language

Min-Na Lee†
† Hankuk University of Foreign Studies

요 약

객체지향 프로그래밍은 산업과 교육 분야에서 그 영향력이 점차 커지고 있다. 그러나 객체지향은 추상적이고, 복합적인 개념을 많이 포함하고 있어 처음 객체지향 언어를 배우는 초보학습자는 개념을 이해하는데 많은 어려움을 겪고 있다. 또한 객체지향 개념들은 상호 연관성이 매우 높아 기본 개념을 이해하지 못하면 다음 단계의 개념을 이해할 수 없다. 따라서 본 논문은 초보학습자의 객체지향 개념의 이해를 돕기 위해 클래스와 객체 모델, 클래스간의 상속 모델을 도식화하여 제안한다. 이 모델을 이용하여 객체지향의 핵심 개념인 클래스, 객체, 레퍼런스 변수, 상속, 오버라이딩, 다형성, 동적 바인딩의 이해도를 높일 수 있다.

1. 서 론

객체지향 프로그래밍은 최근 몇 년간 교육기관과 산업에서 가장 영향력 있는 프로그래밍 영역이 되었고[1][2][3]. 많은 대학에서 컴퓨터를 전공하는 전공자뿐만 아니라 비전공자에게도 객체지향 언어를 가르치고 있다. 또한 초·중·고등학교 소프트웨어 교육의 도입으로 정보 교과목에 객체지향 프로그래밍이 포함되었다.

비전공자와 초·중·고등학생들이 프로그래밍을 배우는 이유는 프로그래머의 기본 소양 교육뿐만 아니라 프로그래밍 교육을 통해 문제해결력, 창의적 사고력, 컴퓨터 사용 능력, 디버깅 작업을 통한 비판·반성적 사고 등의 종합적 사고 능력을 증진시키기 위함이다[4].

그러나 객체지향 개념은 추상적인 개념을 많이 포함하고 있고, 현실과 실체를 기반으로 하는 객체지향 사고를 요구한다[5]. 그래서 처음 객체지향 언어를 배우는 초보학습자는 객체지향 개념을 이해하는데 많은 어려움을 겪고 있다.

초보학습자의 객체지향 개념 이해도에 관하여 많은 연구가 진행되었는데 그 결과 초보학습자는 객체지향 개념의 기본인 클래스와 객체에 관하여 정확한 이해가 부족하며[6], 더욱이 상속, 캡슐화, 다형성, 동적바인딩 등과 같은 개념의 복잡성으로 인하여 초보학습자의 프로그래밍 수업 중도 포기율 또한 높다[7][8].

본 논문은 초보학습자가 이러한 객체지향 개념을 쉽게 이해할 수 있도록 객체지향 개념의 가장 기본이 되는 클래스와 객체의 개념에 관한 모델과 클래스간의 상속 개념에 관한 모델을 도식화하여 제안한다. 제안한 모델들을 이용하여 객체지향에 관한 핵심 개념들 즉, 클래스, 객체, 레퍼런스 변수, 상속, 오버라이딩, 다형성, 동적바인딩 등의 이해도를 향상시킨다.

현재 객체지향 언어로는 C++, Java, Python등 많은 언어들이 사용되고 있지만 본 논문에서는 Java를 이용하여 설명하였다.

2장은 객체지향 개념 학습에 관한 다양한 관련 연구를 분석하고, 3장은 클래스와 객체 개념의 모델, 클래스간의 상속 모델을 제안한다. 4장은 제안된 모델들을 기반으로 하여 객체지향의 중요한 개념들을 프로그램과 함께 설명하고, 5장에서는 결론을 맺는다.

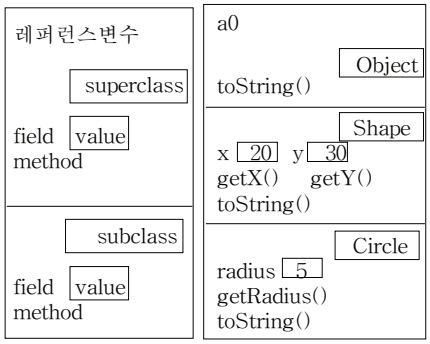
2. 관련연구

최근 객체지향 언어는 프로그래밍 언어 코스에 필수적인 과정이 되었으며, 객체지향 언어의 개념이해를 위해 절차적 언어 보다 먼저 학습되어야 한다는 주장도 널리 펴져 있다[2][3][9]. 그러나 객체지향 개념은 추상적이고 복합적인 개념을 많이 포함하고 있어 초보학습자에게는 여전히 어려운 프로그래밍 언어로 인식되고 있다.

그리하여 초보학습자의 객체지향 언어 학습을 돕기 위해 많은 연구가 진행되어 왔다. 최주희 외는 객체지

향 개념 학습을 위해 스토리텔링 방법과 언플러그드 학습 방법을 활용하였고[10], 김용천 외는 객체지향 개념 학습을 돕기 위해 학습자가 친숙하게 익힐 수 있는 게임에 대한 활동지를 개발하여 프로그래밍 학습을 진행한 게임 요소 기반 학습을 활용하였다[11].

객체지향을 활용하기 위해서는 객체지향의 핵심 개념의 이해가 필수적이므로, 이러한 핵심 개념의 정의와 관련되어 많은 연구가 진행되어 왔다. Gries는 클래스와 객체의 설명을 서류함과 그 안의 서류로 설명하였으며 객체와 상속 개념의 이해를 돕기 위해 객체와 상속을 [그림 1]과 같이 표현하였다[9]. 객체는 맨 위쪽에 레퍼런스 변수를 갖으며 그 아래 값을 갖는 속성과 메소드로 구성된다. 상속은 객체 안에 부모 객체를 위쪽에, 자식 객체는 아래에 표현한다.



[그림 1] Gries의 객체와 상속 모델

Armstrong은 객체지향 개념을 39개의 핵심 요소로 정의하였고 그중 상위 10개의 객체지향 핵심 개념은 <표 1>과 같다[11][12].

<표 1>객체지향의 핵심요소

개념	빈도수	비율
Inheritance	71	81%
Object	69	78%
Class	62	71%
Encapsulation	55	63%
Method	50	57%
Message Passing	49	56%
Polymorphism	47	53%
Abstraction	45	51%
Instance	31	35%
attribute	29	33%

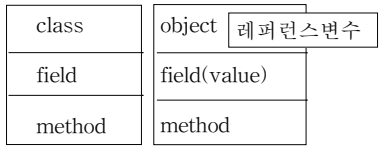
<표 1>에서와 같이 클래스와 객체는 객체지향의 중요 핵심 개념이며 가장 기본 개념이다. 그러나 일부 학생들은 클래스와 객체를 정확하게 이해하지 못하고 있으며[6], 학생의 반 이상은 클래스의 개념은 이해하고 있지만 객체에 관한 정확한 이해도가 낮으며, 프로그램 실행동안 정적인 클래스의 역할과 동적인 객체의 역할에 대한 이해도가 낮다[6].

Pedroni 외는 객체지향의 핵심개념 27가지를 추출하여 서로의 연관성을 비교한 논문에서 가장 기본에 되는 개념은 클래스, 객체, 레퍼런스변수, 메소드이고, 가장 복합 개념은 다형성, 동적바인딩, 정보은닉이며, 모든 객체지향의 핵심 개념들은 상호 연관성이 매우 높다고 발표하였다[13].

이렇게 객체지향 개념들은 서로 밀접한 관계가 있어 선수 단계의 개념을 이해하지 못하면 다음 단계의 개념을 이해할 수 없다. 이에 본 논문에서는 초보학습자의 객체지향 언어 학습을 돕기 위해 객체지향 개념을 도식화한 모델을 제안한다.

3. 객체지향 개념의 모델

3.1 클래스와 객체 모델

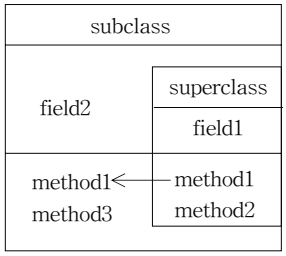


[그림 2] 클래스와 객체 모델

객체지향 설계의 모델링 언어인 UML은 클래스를 표현하는 좋은 모델이므로 본 논문에서는 클래스 모델로 UML 모델을 사용하며, 객체는 클래스로부터 생성되는 실체이므로 클래스 모델을 기반으로 한다.

클래스는 속성과 행위가 정의되며, 객체는 상태, 행위, 식별자를 갖는다. [그림 2]와 같이 객체의 속성을 선언하는 부분에 속성의 값(상태)를 같이 표시하고, 객체 생성시 필요한 레퍼런스 변수(식별자)는 객체를 참조하는 의미로서 객체 이름위에 표시한다. 레퍼런스 변수는 포스트잇 책갈피와 같이 객체 위에 붙일 수 있고, 뗄 수 도 있다.

3.2 상속모델



[그림 3] 클래스의 상속 모델

객체지향에서는 이미 존재하는 클래스에서 새로운 클래스의 파생을 상속이라 한다. 일반적으로 클래스간

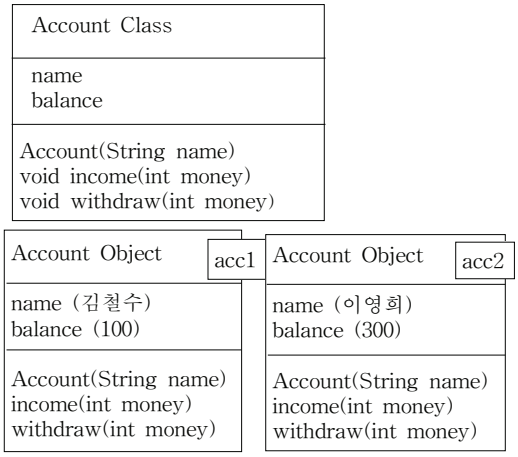
의 상속개념을 표현하기위해서는 부모 클래스와 자식 클래스를 UML 모델로 표현하고, 부모 클래스로 향하는 화살표를 그려 상속 관계를 나타낸다.

본 논문에서는 클래스간의 상속을 [그림 3]과 같이 포함관계로 표현한다. 부모 클래스는 자식 클래스에 포함되며, 자식 클래스가 부모 클래스의 메소드를 오버라이딩하는 경우 부모 메소드에서 자식 메소드로 향한 화살표로 표현한다.

기존 클래스간의 상속 표현은 거시적 관점으로 여러 클래스들 사이의 상속 관계를 전체적으로 파악할 수 있으며, 본 논문에서 제안한 상속 표현은 미시적 관점으로 부모 자식 클래스간의 관계를 상세히 표현한다.

4. 모델을 이용한 객체지향 개념 설명

4.1 클래스와 객체의 모델 설명



[그림 4] 클래스와 객체 모델의 예

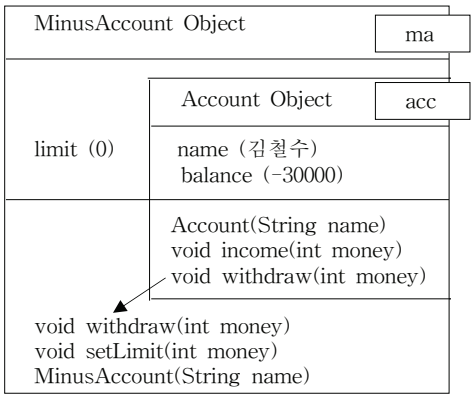
```
Account acc1 = new Account("김철수");  ①
Account acc2 = new Account("이영희");  ②
acc1.income(100);                      ③
acc2.income(300);                      ④
acc2 = acc1;                           ⑤
```

Account 클래스가 선언되어 있을 때, ①과 ②를 실행하면 [그림 4]와 같이 Account 객체 2개가 생성된다. 클래스는 객체 생성을 위한 추상적 개념으로 한번 선언되면 프로그램이 종료될 때까지 아무런 변화가 없는 정적인 역할을 하며, 객체는 프로그램 실행 중 생성, 속성 변경, 삭제 등 동적인 역할을 실행한다.

레퍼런스 변수 acc1과 acc2는 각 Account 객체를 참조하기 위해 각 객체에 붙어있다. ③과 ④와 같이 각 객체의 속성과 메소드는 해당 레퍼런스 변수를 통해서만 접근 및 실행 가능하다.

⑤는 레퍼런스 변수의 할당문으로 acc2 변수가 acc1 변수와 같은 객체를 참조하도록 하는 실행문이다. 이를 위해 acc2 레퍼런스 변수를 해당 객체에서 떼어 acc1 변수가 참조하는 객체에 붙인다. 이때 acc1이 참조하는 객체는 두 개의 레퍼런스 변수(acc1, acc2)를 가지며, 두 변수를 통해서 객체의 속성과 함수를 사용할 수 있다. 그러나 또 다른 객체는 이제 레퍼런스 변수가 존재하지 않으며, 레퍼런스 변수를 갖는 않는 객체는 garbage가 되어 삭제 처리된다.

4.2 상속 모델 설명



[그림 5] 객체의 상속 모델 예

```
MinusAccount ma=new MinusAccount("김철수");①
ma.income(50000);                          ②
ma.withdraw(50000);                        ③
Account acc = ma;                          ④
acc.withdraw(30000);                       ⑤
```

[그림 5]는 객체의 상속표현을 나타낸 것으로 ①에 의해 MinusAccount 객체가 생성되며, 그 안에 부모 객체인 Account객체를 포함한다. 레퍼런스 변수 ma를 통해 해당 객체안의 모든 속성과 메소드에 접근 및 실행이 가능하다.

②를 실행하기 위해 MinusAccount 객체가 선언한 메소드에서 income 메소드를 찾으나 존재하지 않는다. 그러나 부모 객체가 income 메소드를 가지고 있으므로 부모의 income 메소드가 실행되어 속성 balance 값이 50000으로 변경된다. ③을 실행하기 위해 MinusAccount 객체의 withdraw 메소드를 찾는다. 그런데 withdraw 메소드가 2개 존재하므로 화살표를 쫓아 오버라이딩 한 메소드를 실행하여, balance 값은 0으로 변경된다.

다형성은 객체지향의 매우 중요한 개념이다. ④는 다형성 개념을 포함한 실행문으로 레퍼런스 변수 acc와 ma가 같은 객체를 참조하도록 한다. 이때 acc가 Account를 참조하는 변수이므로 MinusAccount 객체 전체를 참조할 수 없으며, MinusAccount 안의 부모 객체인 Account 객체만을 참조할 수 있으므로 Account 객체에 acc를 붙이며, acc 변수는 해당

Account 객체안의 속성과 메소드만을 접근 및 실행할 수 있다.

동적 바인딩은 초보학습자가 가장 혼돈하는 개념으로 ⑤는 동적바인딩 개념을 포함하고 있다. acc변수는 Account 객체의 속성과 메소드만을 사용할 수 있으므로 Account 객체의 withldraw 메소드를 찾아 실행하여야 할 것 같지만 그렇지 않고 자식 객체가 오버라이딩한 메소드를 실행하여야 한다. 이를 설명하기위해 acc변수가 Account 객체의 withdraw 함수를 찾은 후 오버라이딩을 의미하는 화살표가 있으므로 화살표를 따라 MinusAccount의 withdraw 메소드를 실행한다. 그래서 속성 balance 값이 -30000으로 변경된다.

5. 결론

본 논문은 객체 지향의 추상적이고 복합적인 개념을 초보학습자가 쉽게 이해할 수 있도록 시각적으로 도식화된 클래스와 객체의 모델과 상속 모델을 제안한다. 이 모델을 통해 학습자는 다음과 같이 가장 기본이고 핵심인 객체지향 개념을 보다 쉽고 정확하게 이해할 수 있다.

첫째, 클래스의 정적인 개념과 객체는 생성, 변경, 소멸될 수 있는 동적인 개념의 이해를 돕는다. 둘째, 레퍼런스 변수의 정확한 이해를 돕는다. 객체는 레퍼런스 변수와 생성되며, 레퍼런스 변수를 통해 객체에 접근할 수 있으며, 레퍼런스 변수가 없는 객체는 삭제된다는 것을 이해할 수 있다. 셋째, 상속 모델을 통해 초보학습자가 가장 어려워하는 다형성과 동적바인딩의 이해를 돕는다. 부모 타입의 레퍼런스 변수는 자식객체가 포함한 부모 객체에 붙일 수 있으며, 해당 레퍼런스 변수는 부모 객체만을 접근할 수 있다. 이때 부모 객체의 메소드가 자식 객체의 메소드로 화살표가 되어 있는 경우는 화살표를 따라 자식 메소드를 실행한다.

향후 연구 과제로는 객체지향의 다른 개념들도 초보학습자가 보다 쉽게 이해할 수 있도록 본 논문에서 제안한 모델을 더욱 확대시켜 개발할 필요가 있다.

참 고 문 헌

[1] 이은정·이원규·김자미 (2014). 객체지향 개념 학습을 위한 정보 교육과정 분석. **컴퓨터교육학회학술대회논문지**, **18**(2), 9-12.

[2] Kolling, M. (1999). The problem of teaching object-oriented programming. *Journal of Object-Oriented Programming*, *11*(8), 8-15.

[3] Kolling, M. (1999). *The design of an object-oriented environment and language for teaching*. PhD. dissertation, Computer Science, University of Sydney.

[4] 유진아 (2008). **공개 소프트웨어 Python을 이용한 프로그래밍 교육에 관한 연구**. 석사학위 논문, 단국대학교 교육대학원.

[5] 최세일 (2016). 객체지향프로그래밍 언어 교육방법에 관한 연구. **전자통신학회논문지**, **11**(8), 751-757.

[6] Xinogalos, S. (2015). Object-Oriented Design and Programming : An Investigation of Novices' Conceptions on Objects and Classes. *ACM Transactions on Computing Education*, *15*(3), 13:1-13:20.

[7] Casperson, M. E. (2007). *Educating Novices in the Skills of Programming*. PhD. dissertation, Computer Science, University of Aarhus, Denmark.

[8] 최현종 (2011). 대학 프로그래밍 강좌를 위한 프로그래밍 교육 프레임워크. **컴퓨터교육학회논문지**, **14**(1), 69-79.

[9] Gries, D. (2008). A Principled Approach to Teaching OO first. P*roceedings of the 39th SIGCSE Technical Symposium on Computer Science Education,* 31-35.

[10] 최주희·김현철 (2011). 객체지향 개념학습에서의 스토리텔링 활용 방법 연구. **컴퓨터교육학회학술발표논문지**, **15**(1), 117-122.

[11] 김용천·장윤재·윤일규·이원규·김자미 (2014). 게임 요소 기반의 객체지향 개념 학습에 대한 수업사례 연구. **컴퓨터교육학회논문지, 17**(5), 1-13.

[12] Armstrong, D. J. (2006). The quarks of object-oriented development. *Communications of the ACM*, *49*(2), 123-128.

[13] Pedroni, M., &Meyer, B. (2010). Object-Oriented modeling of Object-Oriented Concepts : A Case Study in Structuring an Educational Domain. *Teaching Fundamentals Concepts of Informatics, 5941*, 155-169.