Regular paper

Journal of the KIECS. pp. 751-758, vol. 11, no. 8, Aug. 31 2016, t. 76, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2016.11.8.751

객체지향프로그래밍 언어 교육방법에 관한 연구

최세일*

A Study on Teaching the Object Oriented Programming Language

Se-Ill Choi*

요 약

자바를 비롯한 객체지향 언어를 쉽게 배우기 위해서는 그 언어가 제공하는 객체지향 프로그래밍 기능만을 습득하는 것만으로는 부족하고, 먼저 프로그래머 자신이 객체지향적 사고를 해야 한다. 그러나 객체지향 개념이 없는 한국어에 길들여진 한국 학생들이 객체지향적 사고를 한다는 것은 쉬운 일이 아니다. 따라서 한국인 학생들에게 객체지향 언어를 교육할 때는 객체지향적 사고에 기반 한 프로그래밍 교육을 실시해야 한다. 본 논문에서는 객체지향 언어를 교육할 때, 먼저 객체지향적 사고 전개 방법을 설명하고, 이를 기반으로 프로그래밍 언어를 사용하여 그렇게 전개한 사고를 표현하는 방법으로 프로그래밍 언어를 교육하는 방법을 소개한다.

ABSTRACT

Object Oriented Programming Languages including Java require Object-oriented thinking first for programming. However, for Korean students it is not easy since they are fully accustomed to Korean language which does not have the Object-oriented concept. This paper proposes a way of teaching the Object Oriented Languages to Korean students. It explains first how to organize our thoughts in Object-oriented way, and then how to express the thoughts with the languages.

키워드

Object-Oriented, Programming, Java, OOP, Language, Way of Thinking
객체 지향, 프로그래밍, 자바, OOP, 언어, 사고 방식

Ⅰ. 서 론

컴퓨터 관련학과에서 자바 언어는 기본적이며 필수적인 교과목이다. 그럴 수밖에 없는 것이 모든 정보기기에 자바 언어가 보편적으로 사용되고 있고, 이러한 현상은 갈수록 심화될 것으로 예상되기 때문이다. 그러나 국내 자바 교육현장에서 느껴지는 학생들의 자바 언어에 대한 태도는 자바 언어가 꼭 배워야 할 언어임에도 불구하고 배우기 어려운 언어로 인식하여 배우기를 기피하고 있다는 점이다. 이러한 문제는 한국인의 언어 습관과 매우 밀접한 관계가 있다. 즉 자바 언어는 영어와 유사한 사고방식을 기반으로 하기 때문에 학생들이 영어를 배울 때 느끼는 어려움을 자바 언어를 배울 때에도 똑같이 느낄 수밖에 없다.

영어와 한국어는 여러 가지 면에서 차이가 많지만 가장 큰 차이점은 객체를 개념화하고 표현하는데 필요한 기능의 차이에 있다. 영어에는 a, the와 같은 관사가 객체 개념을 다루기 위하여 제공되지만 한국어

* 교신저자 : 호남대학교 컴퓨터공학과
• 접 수 일 : 2016. 06. 28
• 수정완료일 : 2016. 08. 13
• 게재확정일 : 2016. 08. 24
• Received : Jun. 28, 2016, Revised : Aug. 13, 2016, Accepted : Aug. 24, 2016
• Corresponding Author : Se-Ill Choi
Dept. of Computer Engineering, Honam University
Email : sichoi@honam.ac.kr

에는 그러한 개념이 제공되지 않는다. 따라서 한국어에 습관화된 학생들은 객체를 다루는 능력이 부족할 수밖에 없고 결국 이러한 언어적 습관이 자바 언어를 배우는데 장애가 되고 있다[1].

이러한 언어적 차이를 이해하지 못한 상태에서 자바 언어를 공부하는 것은 단순히 자바 문법만을 공부하는 상태가 되고, 결국 영어를 배울 때 살아있는 영어를 배우지 못하고 영어 문법만을 배우게 되는 현상과 유사한 결과를 낳게 된다. 다시 말하면 객체지향 개념에 대한 이해가 부족한 상태에서 자바를 배우게 되면 자바 언어가 매우 어렵게 느껴 질 뿐만 아니라, 설령 자바 언어를 공부했다고 하더라도 자바 언어가 추구하는 객체지향 개념을 충분히 사용할 수 없다. 따라서 자바 언어를 쉽고 정확하게 공부하기 위해서는 먼저 객체지향적으로 사고하는 방법을 이해하고, 자바 언어를 사용하여 그러한 사고를 표현하는 방법을 동시에 습득해야 한다.

본 논문에서는 아무리 자바 언어를 쉽게 설명한다 하더라도 학생들이 사고방식을 개선하지 않는 한, 학생들의 자바에 대한 인식은 변하지 않을 것으로 보고, 근본적으로 학생들의 사고방식을 먼저 개선함으로서 자바를 쉽게 이해하도록 유도하는 교육방법을 제안하고자 한다.

2장에서는 국내 자바교육의 현황을 이해하기 위하여 국내 자바교육 도서를 분석하고, 3장에서는 객체지향 사고 방법과 객체지향 사고를 기반으로 한 자바프로그래밍 방법에 대하여 제안한다. 4장에서는 제안한 아이디어를 기반으로 자바를 교육하는 사례를 보이고, 5장에서 결론을 맺는다.

II. 관 련 연 구

현재까지 국내외에서 출간된 자바 언어 관련 교재를 살펴보면 자바 언어의 일반적 문법과 사용법을 설명하는데 그쳐있고 객체지향 사고방식에 대한 부분은 언급되어 있지 않다. 물론 자바 언어 교육 도중에 강사가 나름대로 객체지향 사고방식에 대하여 설명할 수도 있겠으나, 시중에 발간된 자바 교재나 국내에 발표된 자바 논문에서 객체지향 사고방식에 대한 설명을 찾아보기 힘든 것을 보면, 실제 자바 교육 현장에서 객체지향 사고방식에 대한 교육이 충분하지 않음을 추측할 수 있다[2].

객체지향의 개념에 대한 정의는 컴퓨터 프로그래밍 언어가 개발되면서 제기되었다. 컴퓨터 프로그래밍 언어는 자연언어와 달리 처음부터 특정한 목적을 가지고 개발되기 때문에 목적과 목표에 맞는 언어적 특징이 먼저 결정된다. 그러나 언어라는 의미에서 살펴본다면 프로그래밍 언어나 자연언어나 모두 상대에게 자기의 의사를 표현하는 도구이기 때문에 의사표시를 하는 사람이 편리하게 사용할 수 있는 구조를 갖추는 것이 바람직하다. 객체지향 언어라고 불리는 자바 언어가 널리 쓰이는 이유도 컴퓨터 구조에 적합한 언어라기보다는 프로그래머가 자신의 사고방식을 표현하는데 적합하기 때문일 것으로 추측된다[3][4].

우리가 일본어 교육이나 영어 교육에서 알 수 있듯이 같은 사고방식을 기반으로 사용되는 일본어가 전혀 다른 사고방식을 기반으로 하는 영어에 비하여 훨씬 쉽고 빠르게 배울 수 있는 것은 언어의 기반이 되는 사고방식이 언어 학습에 결정적 역할을 하기 때문이라 것을 증명해주고 있다. 따라서 언어 교육을 위해서는 언어의 문법 및 사용법 교육에 앞서 사고방식에 대한 교육이 선행 되어야 할 것으로 사료된다[5].

III. 객체지향 프로그래밍을 위한 객체지향 사고방식

본 논문은 자바 언어를 학습하기 위해서 먼저 객체지향 사고에 대한 개념을 확립하고, 이를 기반으로 객체지향적으로 프로그래밍하는 방법을 습득하는 순서적 학습 방법을 제안한다. 순서가 중요한 이유는 자바 언어가 아무리 객체지향 기능을 제공하고 있다고 하더라도 프로그래머가 먼저 객체지향적 사고를 하지 않으면 객체지향프로그래밍이 되지 않기 때문이다.

3.1 객체지향 사고

앞에서 언급한 것처럼 객체지향이란 개념 자체가 프로그래밍 언어에서 시작되었지만 이는 단지 프로그래밍 언어를 사용하기 위하여 요구되는 사고방식이 아니라 인간의 사고방식 유형중의 하나로 서구식 사고방식이 이를 대표한다.

객체지향 사고는 현실과 실체를 기반으로 사고를 시작하고 이를 확장해가는 사고의 전개방법을 말한다. 달리 이야기하면 사고의 시작은 현실과 실체를 기반으로 하고 사고의 전개는 귀납적인 방식으로 함을 의미한다[7]. 이러한 사고방식은 명분과 이념을 중심으로 사고를 시작하고 현실과 실체를 유도하는 우리의 연역적 사고 전개방식과 정반대의 사고방식이라고 말할 수 있다. 자바 언어의 학습이 어려운 이유가 바로 이러한 사고방식의 차이에서 기인되고 있다.

객체지향 사고방식에 대한 하나의 사례로 학생이라는 객체에 대하여 객체지향 방식으로 사고를 전개해보자.

1) 객체의 구분 - 한명의 인간이라는 실체가 존재한다. 객체지향 개념은 일단 실체로부터 사고를 시작한다.
2) 객체에 해당하는 개념의 정의 - 어떤 사람은 학생이라는 신분과 직장인 신분을 동시에 가지고 있고, 어떤 사람은 학생이라는 단일 신분만을 가지고 있으며, 어떤 사람은 휴학을 하고 일을 하고 있는 직장인이라는 단일 신분만을 가질 수도 있다. 객체지향적 사고는 해당 객체에 해당하는 개념을 나중에 전개한다.
3) 객체의 통합 및 통합객체의 구분 - 학생이라는 실체가 모여서 또 다른 실체를 만들 수 있는데 예를 들어 동아리 모임, 학과, 나아가서 단과대학 등과 같은 실체들은 작은 객체 단위가 모여서 만들어진 것들이다. 통합객체는 최소 단위 객체가 판별된 이후에 개념화된다.
4) 통합 객체에 대한 개념 정의 - 동아리, 학과, 단과대학 등과 같은 통합객체가 판별되면 이에 대한 개념이 정의되어야 한다. 예를 들어 테니스동아리, 기타동아리, 컴퓨터공학과, 법학과, 공과대학 등과 같은 개념 또한 객체가 판별된 다음에 정의된다. 이러한 개념의 전개 방식은 개념이나 명분이 먼저 설정되고 이에 대한 실체가 만들어 지는 우리의 사고방식과는 정반대의 논리 전개방식이다. 예를 들어 기타 동아리를 만들 때에도 기타를 좋아하는 사람이 모여서 기타동아리를 만드는 방법(귀납적 방법)[6]과 기타동아리가 정의된 후에 기타 동아리에 가입할 회원을 선발하는 반대의 방법(연역적 방법)이 있을 수 있다.

객체지향 사고는 최소의 객체 단위에서부터 객체를 판별하고 그 객체에 대한 개념을 정의한 다음, 다시 또 정의된 객체들이 모여서 이루는 통합객체를 판별하고 다시 이에 대한 개념을 정의한다. 이러한 사고과정은 통합객체가 만들어 지는 한 반복된다. 따라서 객체지향적 사고는 점차적으로 객체가 통합되어 가는 방향, 즉 상향식 사고와 객체를 기반으로 객체에 대한 개념을 정의하는 귀납적 사고방식을 통합적으로 이르는 개념이라고 정의할 수 있다.

3.2 객체지향 프로그래밍 절차

객체지향 사고에 대한 개념이 이해되면 이를 프로그래밍에 적용하는 방법에 대하여 알아야 한다. 프로그래밍에 적용하는 절차는 다음과 같다.

1) 실체에 해당하는 단위 프로그램 모듈을 판단한다. 예를 들어 회원가입모듈, 회원탈퇴모듈, 성적입력모듈, 성적출력모듈, 대출모듈, 반납모듈 등과 같은 기본 기능을 수행하는 단위 모듈 들을 말한다.
2) 모듈에 해당하는 개념을 정의한다. 판별된 모듈에 대하여 그 모듈이 사용하는 데이터 타입과 기능을 정의한다. 이는 자바의 클래스에 해당한다.
3) 프로그램 덩어리가 모여서 구성하는 복합모듈을 판별한다. 예를 들어 회원가입모듈과 회원탈퇴모듈이 통합된 회원관리모듈, 성적입력모듈과 성적출력모듈이 통합된 성적관리모듈 등과 같은 통합모듈을 의미한다. 소단위 기본기능 모듈들을 통합하다 보면 하나의 소단위 모듈이 다양한 통합단위 통합모듈에 중복되어 통합될 수도 있음으로 다양한 통합단위 모듈을 생각해 낼 수 있다.
4) 판별된 통합단위 모듈에 대한 개념을 정의한다. 개념의 정의는 소단위 모듈에서와 동일하게 자바 언어의 클래스에 해당한다.

객체지향 프로그래밍은 가장 적은 소단위 기능을 프로그래밍 할 수 있고, 그 프로그램을 통합하는 방법만 안다면 대단위 프로그램을 쉽게 작성할 수 있는 장점이 있다. 따라서 그러한 사고방식을 지원하는 언어가 있다면 쉽게 프로그래밍 할 수 있게 되는데 이러한 언어가 바로 자바 언어이다.

3.3 객체지향 프로그래밍 핵심사항 정리

기술된 객체지향 프로그래밍 방식을 자유롭게 활용하기 위해서는 다음과 같은 핵심사항을 이해할 필요가 있다.

1) 객체와 객체의 개념에 대한 확실한 구분 능력이 필요하다. 다시 말하면 객체와 그 객체의 개념을 정의하는 클래스에 대한 확실한 이해가 기본적으로 필요하다. 클래스를 객체에 대한 설계도라고 이해할 수도 있지만 클래스를 설계도라고 정의하게 되면 설계도를 기반으로 객체를 생성해야하기 때문에 객체지향이라기보다는 클래스 기반 사고가 된다. 따라서 객체지향 사고를 위해서는 객체를 먼저 생각하고 그 객체에 대한 개념을 클래스 형태로 정의하는 방향으로 사고를 전개해야 한다[8].
2) 객체가 다양한 개념으로 존재 가능하도록 허용하는 캐스팅 개념에 대하여 이해하여야 한다. 하나의 클래스에서 여러 가지 객체가 만들어 질 수도 있지만, 하나의 객체가 다양한 클래스로서 정의될 수 있음을 이해할 수 있어야 한다. 예를 들어 한명의 사람이 학생클래스에 속할 수도 있고, 직장인클래스에 속할 수도 있음으로 하나의 객체가 2가지 클래스로 사용됨을 이해하여야 한다[9].
3) 하위 객체가 통합되어 상위객체를 구성하는 의미와 방법에 대하여 이해하여야 한다. 세상의 모든 실체는 그 자체가 하나의 단위 객체가 되기도 하지만 여러 가지 소단위 객체가 모여서 하나의 대형 객체가 되기도 한다. 예를 들어 승용차는 엔진, 의자, 문짝 등과 같은 여러 가지 소형 객체들로 구성된 대형 객체가 되는데 승용차처럼 소형 객체를 통합하여 대형 객체를 만드는 방법에 대하여 정확하게 이해하여야 한다[10].

Ⅳ. 객체지향 사고를 기반으로 한 자바 프로그래밍 사례

사례로서 더하기와 빼기 연산을 수행하는 2칙 계산기를 자바를 기반으로 만들어 본다.

1) 소단위 모듈을 판별한다. 본 사례의 2칙 계산기에서는 더하기모듈과 빼기모듈이 필요하다. 먼저 그림 1과 같은 최소단위 모듈 중에 더하기 모듈을 작성한다.

```
int add(int x, int y){
    fstNum=x;
    sndNum=y;
    rsltNum=fstNum+sndNum;
    return rsltNum;
}
```

그림 1. 더하기모듈 프로그램
Fig. 1 Add module

더하기 모듈을 작성한 다음, 또 다른 최소단위 모듈인 빼기모듈을 그림 2와 같이 작성한다.

```
int sub(int x, int y){
    fstNum=x;
    sndNum=y;
    rsltNum=fstNum-sndNum;
    return rsltNum;
}
```

그림 2. 빼기 모듈
Fig. 2 Subtract module

2) 각 소단위 모듈에 대한 개념을 정의한다. 본 사례에서는 더하기모듈에 대한 클래스, 빼기모듈에 대한 클래스를 정의하는 작업을 의미한다. 첫 번째로 더하기모듈에 대한 개념화과정으로 그림 3과 같이 더하기모듈에 대한 클래스를 정의하고, 다음으로 빼기모듈에 대한 클래스를 그림 4와 같이 정의한다.

```
class AddClass{
    int fstNum, sndNum, rsltNum;
    int add(int x, int y){
        fstNum=x;
        sndNum=y;
        rsltNum=fstNum+sndNum;
        return rsltNum;
    }
}
```

그림 3. 더하기 모듈에 대한 클래스
Fig. 3 Class for the Add module

```
class SubClass{
    int fstNum, sndNum, rsltNum;
    int sub(int x, int y){
        fstNum=x;
        sndNum=y;
        rsltNum=fstNum-sndNum;
        return rsltNum;
    }
}
```

그림 4. 빼기 모듈에 대한 클래스

Fig. 4 Class for the Subtract module

3) 통합모듈을 판별한다. 본 사례에서는 2칙 계산기이다. 통합모듈은 더하기모듈에 대한 클래스와 빼기모듈에 대한 클래스를 부속품으로 하는 실체이다. 소단위 모듈에 대한 클래스들이 각각 정의되어 있음으로 이를 기반으로 그림 5와 같이 생성자를 사용하여 2개의 객체를 생성하고 이를 통합한다. 2개의 모듈을 통합하는 방법은 단지 생성자를 활용하여 객체를 통합하기만 하면 가능하고 굳이 별도의 또 다른 프로그래밍을 할 필요가 없다.

```
AddSubClass(){
    a=new AddClass();
    s=new SubClass();
}
```

그림 5. 통합모듈

Fig. 5 Integration module of the Add and Subtract modules

4) 통합모듈에 대한 개념을 그림 6과 같이 정의한다. 본 사례에서는 2칙 계산기에 대한 클래스이다.

```
class AddSubClass{
    AddClass a;
    SubClass s;
    AddSubClass() {
        a=new AddClass();
        s=new SubClass();
    }
}
```

그림 6. 통합 모듈에 대한 클래스

Fig. 6 Class for the Integration module

결론적으로 객체지향 사고를 기반으로 작성된 2칙 계산기는 다음 그림 7과 같다.

5) 작성된 2칙 계산기를 사용하여 실제 더하기와 빼기업무를 처리하는 프로그램을 작성한다. 객체지향 사고를 기반으로 작성된 2칙 계산기는 사용하고자 하는 목적, 즉 수행하고자 하는 더하기와 빼기를 추가로 프로그래밍하여 그림 8과 같이 추가함으로서 프로그래밍이 완료된다. 본 사례에서는 해당 업무를 처리하는 부분이 메인 프로그램에 삽입되어 있다.

```
class AddClass{
    int fstNum, sndNum, rsltNum;
    int add(int x, int y){
        fstNum=x;
        sndNum=y;
        rsltNum=fstNum+sndNum;
        return rsltNum;
    }
}
class SubClass{
    int fstNum, sndNum, rsltNum;
    int sub(int x, int y){
        fstNum=x;
        sndNum=y;
```

```
        rsltNum=fstNum-sndNum;
        return rsltNum;
    }
}
class AddSubClass{
    AddClass a;
    SubClass s;
    AddSubClass(){
        a=new AddClass();
        s=new SubClass();
    }
}
```

그림 7. 2칙 계산기 프로그램
Fig. 7 The Calculator of Add and Subtract functions

2칙 계산기는 그대로 사용될 수도 있지만 추가적인 소단위 프로그램을 작성하고 이들을 통합하여 더 큰 4칙 계산기, 복잡한 공학용 계산기 등으로 발전될 수도 있다.

사례에서 보듯이 객체지향 개념을 기반으로 작성된 프로그램은 조잡해 보이기도 하지만 크고 복잡한 프로그램을 쉽고 빠르게 작성할 수 있는 장점이 있다.

```
public class AddSubCal{
    public static void main(String[] args){
        int rslt;
        AddSubClass cal;
        cal = new AddSubClass();
        rslt = cal.a.add(6,2);
        System.out.println("Result="+rslt);
        rslt = cal.s.sub(6,2);
        System.out.println("Result="+rslt);
    }
}
```

그림 8. 2칙 계산기를 사용하는 프로그램
Fig. 8 Program using the Calculator

Ⅴ. 결 론

본 논문에서는 학생들에게 자바 언어 학습이 어려운 이유를 우리의 사고방식과 밀접한 관계가 있다고 보고, 자바 언어를 쉽고 정확하게 배우기 위해서는 객체지향 사고를 먼저 이해하고 이를 기반으로 프로그래밍을 진행하는 방식을 제안하였다.

1) 실체를 기반으로 개념을 유추하는 귀납적사고
2) 소형 실체들이 점점 통합되어 대형 실체가 되어가는 상향식사고

또한 이러한 사고방식을 기반으로 한 프로그래밍은 다음과 같은 핵심사항에 대한 이해를 요구한다.

1) 객체와 클래스의 구분을 명확히 할 수 있어야 한다.
2) 하나의 객체가 여러 가지 클래스를 전환할 수 있음으로 객체의 캐스팅에 대한 이해를 확실히 해야 한다.
3) 소형 객체가 통합되어 대형 객체가 되기 때문에 객체를 통합하는 방법에 대하여 확실히 이해하여야 한다.

객체지향 프로그래밍 방법이 정상적으로 정착되기 위해서는 이러한 연구를 기반으로 객체지향 프로그래밍 교재와 세부적인 커리큘럼이 제작되어야 할 것으로 생각된다.

References

[1] S. Choi, "A Study on Teaching the Java," *Proceeding of the Korea Institute of Electronic Communication Sciences*, vol. 9, no. 2, pp. 480-483, Ganhneung, Korea, Nov. 2015,
[2] E. Lee, W. Lee, J. Kim, "An Analysis of Informatics Curriculum for Object-Oriented Concept Learning", *P. The Korean Association of Computer Education*, vol. 18, no. 2, 2014, pp. 9-12.
[3] J. Choi, H. Kim, "Teaching Girls Object-Oriented Concepts using Storytelling", *Proceeding The Korean Association of Computer*

Education, vol. 15, no. 1, 2011, pp. 117-122.
[4] S. Choi, "A Curriculum to Improve the Lecture of Database SQL", *J. of The Korea Institute of Electronic Communication Sciences*, vol. 9, no. 9, 2014, pp. 1005-1010.
[5] S. Choi, "A Road-map for an e-Commerce Development", *J. of The Korea Institute of Electronic Communication Sciences*, vol. 7, no. 4, 2012, pp. 897-904.
[6] M. Ryu, "A Study on Effetive Teaching Method for Obect-Oriented Programming Language in Seconday Education", *Master's Thesis, Kyungsung University,* 2011, pp. 19-21.
[7] Y. Kim, Y. Jang, I. Yoon, W. Lee, J. Kim, "Design of Game based Educational Contents for Learning Object-Oriented Concepts", *P. The Korean Association of Computer Education*, vol. 17, no. 2, 2013, pp. 35-39.
[8] Stelios Xinogalos, "Object-Oriented Design and Programming: An Investigation of Novices' Conceptions on Objects and Classes", *ACM Transactions on Computing Education,* vol. 15, no. 3, Aritcle No. 13, Sep. 2015.
[9] S. Choi, "An Object-Oriented Modeling of Object-Oriented Software Development Methods : OMOS(Object-oriented software development Method for Object-oriented software System)", *J. of Korean Information Processing Society*, vol. 8, no. 4, 2001, pp. 401-408.
[10] K. Heo, Y. Kim, D. Yong, "Aviation Application : The Study of Framework Model for Software Productivity Enhancement in Object-Oriented Environment", *J. of The Korean Navigation Institute*, vol. 14, no. 45, 2010, pp. 900-908.

저자 소개

최세일(Se-Ill Choi)

1984년 한양대학교 전자공학과 졸업(공학사)
1989년 플로리다공과대학교 대학원 전산학과 졸업(공학석사)

2002년 모나쉬대학교 대학원 전산학과 졸업(공학박사)
1993년 ~ 현재 호남대학교 컴퓨터공학과 교수
※ 관심분야 : 소프트웨어공학, 데이터베이스, 전자상거래