Journal of the Korea Convergence Society
Vol. 8. No. 10, pp.53-60, 2017 https://doi.org/10.15207/JKCS.2017.8.10.053

Java에서 한글 식별자 사용에 관한 연구

양단희*
평택대학교 융합소프트웨어학과

A Study on the Use of Hangeul Identifier in Java

Dan-Hee Yang*
Department of Convergence Software, Pyeongtaek University

요 약 유니코드가 나오기 전까지는 프로그램용 이두문자의 사용이 불가피한 면이 있었다. 그러나 유니코드가 국제표준으로 명실공이 자리 잡은 지금에도 한글 식별자의 사용을 피해야 한다고 주장하는 것은 세종 이후 지속적으로 한글 사용을 천시하고 반대한 것과 비슷한 측면이 있다.
본 연구에서는 컴퓨터학과 학생들을 대상으로 한 설문조사를 통해 영어 식별자 사용을 선호하는 이유와 한글 식별자를 사용한다고 할 때 영어 대문자의 기능을 대체할 표기법에 대한 선호도를 조사하였다. 그리고 영어 식별자 사용의 대세론이 허망함에 대해 논의하고, 한글 식별자 사용을 위한 두 가지 명명법을 제안하였다. 소프트웨어 생산성 향상을 위해, 프로그래머의 작업 만족도 향상을 위해, 유지보수의 용이성을 위해 한글 식별자 사용이 조속히 보편화 될 필요가 있다.

• 주제어 : 식별자, 유니코드, 이두문자, 소프트웨어공학, 한글 프로그래밍

Abstract The use of 'Idumunja' for programs is inevitable before Unicode came out. However, even now that Unicode has been established as an international standard in both name and reality, there has been an insistence that the use of Hangeul identifiers should be avoided. This study surveyed the students for the reasons why they prefer to use English identifiers and for the notations that can substitute the function of English capital letters in using Hangeul identifiers. Then, we discussed the vanity of argument that the use of English identifiers is a global trend, and proposed two notations for the use of Korean identifiers. In order to improve the productivity of software, programmer's job satisfaction, and the ease of maintenance, the use of Hangeul identifiers should become rapidly common.

• Key Words : identifier, Unicode, Idumunja, software engineering, Hangeul programming

1. 서론

1980년대는 컴퓨터에서 한글 사용이 매우 제한적이고 부작용이 많아 한글이 IT 발전의 장애 요소로까지 인식되기도 하였다. 그러나 1995년 유니코드 2.0의 출현으로 소프트웨어에서 문자사용의 대변혁이 일어났다. 그리고 2000년경을 전후로 Java, C++, Visual Basic, Delphi 등 대부분의 프로그래밍 언어들이 한글은 물론 다국어 식별자 사용을 지원하게 되었다. 그런데 유니코드 사용의 이런 대세 속에서도 프로그래밍 언어를 배우는 학생들은 프로그래밍에서 식별자를 명명할 때 무의미한 영어 알파

*Corresponding Author : 양단희(dhyang@ptu.ac.kr)
Received September 2, 2017 Revised October 10, 2017
Accepted October 20, 2017 Published October 28, 2017

벳을 나열하거나 콩글리시 영어를 사용하는 경향이 있다[1].

프로그래밍 언어나 자연언어나 사람의 의사를 표현하는 도구이기 때문에 의사 표현자가 쉽고 편리하게 사용할 수 있어야 한다. 오세만[2]은 좋은 프로그래밍 언어의 조건으로 '프로그래머의 생각을 자연스럽게 표현할 수 있어야 한다.'고 하였고, 정영식[3]은 교육용 프로그래밍 언어의 조건으로 가독성(readability), 가작성(writability), 교정성(correctability), 전이성(transferability)을 내세웠다. 여기서 가독성은 다른 사람이 만든 소스코드를 쉽게 이해할 수 있음을 뜻하며, 한글 지원을 특징으로 한다. 가독성을 높이기 위해서는 클래스명, 인스턴스명, 변수명 등의 각종 식별자를 한글로 표현할 수 있어야 한다. 가작성은 프로그래머가 생각한 것을 쉽게 코드로 작성할 수 있음을 의미한다. 그래서 가작성을 높이기 위해서는 우리말을 옮기듯이 자연스럽게 표현할 수 있어야 한다.

그런데 대부분의 유력한 프로그래밍 언어들은 영어를 기반으로 설계되어 있다. 그래서 영어를 모어로 하는 사람은 프로그래밍 언어를 상대적으로 쉽게 배울 수 있는 반면, 비영어권에서는 영어가 프로그래밍 언어를 배우는데 장벽이 되고 있다[4]. 이점을 해결하기 위해 비영어권에서는 자국어 프로그래밍 언어를 개발하려는 노력을 꾸준히 기울려 왔다. 그러나 지금까지 한국어를 바탕으로 한 프로그래밍 언어는 별 주목을 받지 못했으며 한글 프로그래밍 언어가 실제 교육에 사용된 사례도 거의 없다[5,6].

프로그래밍 언어도 일종의 제2언어이다. 따라서 프로그래밍 언어를 습득하는 것은 외국어를 습득하는 것과 매우 유사한 측면이 많다[7]. 박경자의 연구[8]는 제2언어 학습에서 모국어를 학습자로부터 완전히 분리시키는 것은 불가능하며, 모국어를 이용함으로써 학습능률을 높일 수 있다고 주장하였다[9,10]. 박시균의 연구[11]는 학습자의 모국어와 모국 문화를 수업에 활용한 집단에서 제2언어 학습효과가 증진되고 문화적응 스트레스는 감소되는 효과가 있다고 밝혔다.

프로그래밍 학습의 효과를 극대화시키는 것은 사고의 표현을 얼마나 쉽게 할 수 있느냐에 달려 있다. 즉 학습자가 생각한 논리적 사고를 프로그래밍에서 표현할 때 매우 자연스럽게 이뤄져야 한다는 것이다. 이를 위해서는 학습자가 일상생활에서 사용하는 어휘와 문장이 최소의 변형을 통해 컴퓨터가 이해할 수 있는 형태로 표현될 수 있는 프로그래밍 언어를 배우는 것이 가장 효과적이다[3]. 실제로 박철의 연구[12]에서는 프로그래밍이 초급자 수준이면서 외국어 능력이 하위 50% 이내인 전문계 컴퓨터과 학생들을 대상으로 실험하여 모국어 기반 프로그래밍 학습이 주의집중과 만족감 요소를 포함한 전반적인 학습동기 향상에 상당한 도움이 되었다고 밝혔다.

프로그래밍에서 알고리즘과는 무관하나 가독성을 위해 매우 중요한 것이 식별자의 작명이다. Java에서 영어 예약어는 50개에 불과하나 클래스명, 변수명과 같이 프로그래머가 작명해야 할 식별자의 범위는 자연언어에서와 매한가지이다. 따라서 프로그래밍의 본질과는 무관한 영어 식별자를 사용하기 위해 노력할 필요 없이 오직 프로그래밍의 본질에만 집중할 수 있도록 한글 식별자의 사용을 보편화 한다면 프로그래밍 작업이 한결 수월하고 자연스럽게 될 것이다[13].

그래서 본 연구에서는 컴퓨터학과 학생들을 대상으로 설문조사를 통해 영어 식별자 사용을 선호하는 이유와 한글 식별자를 사용한다고 할 때 영어 대문자의 기능을 대체할 표기법에 대한 선호도를 조사하였다. 그리고 영어 식별자 사용의 대세론이 허망함에 대해 논의하고, 한글 식별자 사용을 위한 두 가지 명명법을 제안하였다.

2. 이론적 배경

2.1 모어의 언어적 생산성

일반적으로 모어에 대해 다음과 같은 기준이 제시된다[14]. 모국어는 고국의 언어를 지칭하는 용어이나, 우리처럼 단일 언어를 사용하는 국가에서는 모어와 모국어가 의미 구별 없이 혼용되어 쓰이고 있다.

① 태어나서 처음 습득한 언어에 기반을 둘 것
② 화자의 내적 정체성에 기반을 둘 것
③ 화자의 외적 정체성에 기반을 둘 것
④ 화자가 가장 잘 아는 언어에 기반을 둘 것
⑤ 화자가 가장 자주 사용하는 언어에 기반을 둘 것

언어적 생산성 측면에서 볼 때, 위의 모어 기준 중에서 중요한 것은 화자가 가장 잘 알고, 가장 자주 사용하는 언어라는 것이다. 우리에게 한국어는 모어이기 때문에 이보다 더 생산성 높은 언어는 존재할 수 없다.

2.2 한글 프로그래밍 언어의 현황

우리가 소프트웨어 강국을 지향할 때 영어를 못하더라도 쉽게 배울 수 있는 한글 프로그래밍 언어는 프로그래밍 작성과 유지보수에서 많은 이점이 있을 것이다. 그래서 1991년에 최초의 한글화된 언어인 한글 베이직 프로그래밍 언어 '한베'를 기점으로 1994년에 개발된 '씨앗', 2000년에 '두리틀', 2013년에 '한플' 등 한글 프로그래밍 언어가 간헐적으로 개발되어 왔다. 그러나 이들 모두, 사용자가 적고 지속적인 유지보수가 되지 않아 결국 사장되어 버렸다.

우리나라에 전산학이 학문 분야로 정립된 지 40여 년이 지났지만, 한글 기반 프로그래밍 언어 중에서 범용성을 지니며 사용 중인 언어는 하나도 존재하지 않는다는 것은 매우 안타까운 일이다. 일본은 십 수 가지의 일본어 프로그래밍 언어가 존재하여 초중등 교육에 사용되고 있다[6]. 그런데 몇 년 전부터 컴퓨팅적 사고(computational thinking) 교육의 열풍이 불면서 '와글', '약속', '아희' 등과 같은 한글 프로그래밍 언어가 잇달아 개발되고 있다[15].

2.3 Java의 식별자 명명법

<Table 1> Naming convention in Java

① An identifier is an unlimited-length sequence of Unicode letters and digits. ② An identifier begins with a letter, '$', '_'. The convention is to avoid beginning with '$' or '_'. ③ Subsequent characters may be letters, digits, '$', '_', including Hangeul and Chinese characters. ④ Use full words instead of cryptic abbreviations.

위의 Java 식별자 명명법을 보면 프로그래밍에서 모어 사용의 중요성을 인식하여 프로그래머가 모어를 사용하여 식별자를 만들 수 있게 해 놓았다[16]. 그리고 식별자는 암호 같은 약어 사용을 지양하고 완전한 단어를 사용할 것을 권장하고, 그 길이에 제한을 두고 있지 않다. 이러한 규정들은 프로그램의 가독성과 가작성을 높여 소프트웨어 개발의 생산성을 높이려는 데 그 목적이 있다.

3. 설문 조사 및 분석

3.1 설문조사 개요

<Table 2> Surveyee distribution

Grade	2	3	4	Total
# of Students	31	31	26	88

프로그래밍에서 한글 식별자 사용과 표기법에 대한 인식을 조사하기 위해 본 설문조사는 <Table 2>와 같이 수도권 내의 특정 대학교 컴퓨터학과 2, 3, 4학년의 특정 수업 수강생들을 대상으로 실시되었다. 그리고 설문조사는 성실한 답변을 유도하기 위해 실명제로 실시되었으며 설문에는 결석자를 제외하고 총 88명이 응답하였다.

3.2 설문조사 내용 및 분석

<Table 3> I know that Hangeul can be used as an identifier in programming languages that support Unicode such as JAVA, C, C ++, and Visual Basic.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
33.0%	37.5%	23.9%	3.4%	2.3%

<Table 3>은 식별자에 한글 사용이 가능함을 알고 있는지를 조사하기 위한 것으로 '긍정(매우 그렇다, 그렇다)' 비율이 70.5%며, '부정(아니다, 매우 아니다)' 비율을 5.7%에 불과했다. '보통이다'는 23.9%인데 이것은 뭐라고 단정하여 말할 수는 없으나 사용 사례를 본 적이 있는 정도일 것으로 추측된다.

한글 식별자 사용이 가능함에도 영어 식별자를 사용하고 있는 이유에 대해서는 <Table 4>에서와 같이 '한글 사용이 호환성과 같은 기술적인 문제를 야기 시킬 수 있기 때문' 34.1%, '프로그래밍 교재에서 영어 식별자를 사용하고 있기 때문' 28.4%, '국제화 시대에 부응하기 위해' 25.0%로 답하였다.

<Table 4> If you are using English as an identifier, why?

Reason	%
1) Because Hangeul can cause technical problems such as compatibility	34.1
2) Because English identifiers are usually used in programming textbooks	28.4
3) To meet the Internationalization era	25.0
4) the others	10.2
5) Because an professor taught me to use English	1.1

이것은 Java 프로그래밍 교재에서 '식별자로 한글을 사용할 수 있으나 실무 프로그램에서 한글 사용은 권장되지 않는다.'고 기술되어 있거나, 거의 모든 Java 교재에서 그러하듯이 한글 식별자가 가능하다고 한 줄 기술하고는 정작 예제 프로그램에서는 영어 식별자만을 사용하여 영어 식별자를 사용하는 것이 당연하고 바람직한 것으로 학생들을 묵시적으로 유도시킨 결과라고 생각된다.

<Table 5> The following English identifier program is naturally understood.

```
public class CompoundInterestMain {
 public static void main(String[] args) {
 int principal, period;
 double interestRate, amount;

 principal = 100;
 period = 12;
 interestRate = 0.05; // 5%

 amount = principal * (Math.pow((1 + interestRate),
period));
 System.out.printf("Amount: %.1f", amount);
 }
}
```

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
33.0%	37.5%	23.9%	3.4%	2.3%

<Table 6> The following Hangeul identifier program is naturally understood.

```
public class 복리_Main {
 public static void main(String[] args) {
 int 원금, 기간;
 double 이율, 원리합계;

 원금 = 100;
 기간 = 12;
 이율 = 0.05; // 5%

 원리합계 = 원금 * (Math.pow((1 + 이율), 기간));
 System.out.printf("원리합계: %.1f", 원리합계);
 }
}
```

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
33.0%	37.5%	23.9%	3.4%	2.3%

<Table 5>와 <Table 6>은 원리합계를 구하는 프로그램을 영어 식별자본과 한글 식별자본으로 각각 제시하고 프로그램의 이해도를 조사한 것으로 한글 식별자본이 영어 식별자본보다 이해하기 쉽다는 응답이 2배 이상 많았다. 이 결과는 설문에서 이 둘을 비교 평가하여 어느 쪽이 더 이해하기 쉬운지를 질의한 것은 아니고, 단순히 각 프로그램이 자연스럽게 이해되는지에 대한 평가였다.

<Table 7> Hangeul identifiers are easier to write and read than English's.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
33.0%	37.5%	23.9%	3.4%	2.3%

<Table 8> Hangeul identifiers are desirable in many ways if it does not cause technical problems.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
33.0%	37.5%	23.9%	3.4%	2.3%

<Table 7>에서 프로그램 작성과 판독에 한글 식별자가 더 용이하다는 응답률은 65.9%이고, 단지 6.8%만이 부정하였다. 그리고 <Table 8>에서 한글이 기술적인 문제를 야기 시키지 않는다면 한글 식별자를 사용하는 것이 바람직하다고 61.4%가 동의하였으며, 12.5%는 부정하였다. 그런데 한글 식별자의 장점에도 불구하고 영어 식별자 사용을 개인적으로 선호하는 이유는 다음과 같았다.

① 대부분의 SW개발업체에서 영어 식별자를 사용하고 있으며, 외국 프로그래머와 협업할 때를 대비해야 한다.

② 많은 소스코드들이 영어로 이미 생산되어 있어, 영어를 안 쓰다보면 기존 소스코드들을 대할 때 낯설고 어려울 것 같다.

③ Java의 키워드가 모두 영어이고, 한글 식별자를 사용하기 위해 타자할 때 한영 전환 과정이 번거롭다.

위 이유에서 한영 전환의 번거로움이라는 부수적인 것을 제외하면 영어 식별자 사용의 대세에 순응하고 준비해야 한다는 것으로 파악되어, 이것이 한글 식별자 사용의 보편화에 최대의 걸림돌임을 알 수 있다. 그리고 ② 번 응답은 한자를 평소에 쓰지 않더라도 한자를 읽는 데는 별 불편이 없다는 점을 간과하고 있는 듯하다.

Java에서 한글 식별자를 사용할 때 한글에는 대소문자 구분이 없기 때문에 이것을 대신할 수 있는 표기법이 필요하다. 이를 위해 <Table 9>와 <Table 10>의 설문을 실시하였다.

<Table 9> If words are consecutive but not a compound noun, it is more readable to connect them with '_' in Hangeul identifier. ex) 나의_폭, 최대_길이

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
33.0%	37.5%	23.9%	3.4%	2.3%

영어는 대소문자가 있어 식별자에 myWidth와 같이 카멜(camel) 표기법을 사용할 수 있으며, 이것은 가독성을 위해 Java에서 권장하는 표기법이다. 그래서 이를 대체할 표기법이 필요한데 <Table 9>의 설문에서는 58%가 식별자 내에서 '_'를 사용하여 단어를 연결하는 것을 선호하였고, 6.8%만이 타자할 때 번거롭다는 이유로 단어들을 그냥 붙여 쓰는 것을 선호하였다.

<Table 10> In English identifiers, the first letter of a class name is capitalized as follows: Student student = new Student(); What do you prefer for this in Hangeul identifiers?

Notations	Preference
1) 학교$ 학교 = new 학교$();	39.8%
2) $학교 학교 = new $학교();	36.4%
3) _학교 학교 = new _학교();	15.9%
4) 학교_ 학교 = new 학교_();	6.8%

Java에서 식별자에 특수문자로서는 '$'과 '_'만 예외적으로 포함시킬 수 있다. 그래서 <Table 10>은 이 두 특수문자 중 하나를 사용하여 클래스의 식별자와 인스턴스의 식별자를 구분 하는 표기법의 선호도에 대해 조사한 것이다. 학생들은 클래스의 식별자 뒤에 '$'를 붙이는 것에 39.8%가 선호하였고, 뒤에 붙이는 것에 대해 36.4%,가 선호하였다.

3.3 설문조사의 종합적 분석

설문 조사의 결과를 종합하면 프로그래밍 언어에서 유니코드를 지원하여 한글을 식별자로 사용할 수 있음을 70.5%가 이미 알고 있으며, 한글 식별자를 사용하는 것이 프로그램 작성과 판독에 용이하다는 것에 65.9%가 동의하였고, 기술적인 문제만 없다면 한글 식별자 사용을 61.4%가 지지하였다. 그러나 부정적인 12.5%는 한영 전환의 번거로움이라는 부수적인 것을 제외하면 영어 식별자 사용의 대세에 순응하고 준비해야 한다는 것으로 파악되어, 이것이 한글 식별자 사용의 보편화에 최대의 걸림돌임을 알 수 있었다.

그리고 58%가 한글 식별자 내에서 '_'를 사용하여 단어들을 연결하는 것을 선호하였고, 6.8%는 타자할 때 번거롭다는 이유로 부정하였다. 학생들에게 한영 전환의 번거로움이 문제인 것은 학생들은 간단한 프로그램만을 작성하기 때문에 프로그램 작성 당시의 편리함만을 추구할 뿐 가독성이나 미래의 유지보수에 대한 인식은 없기 때문인 것으로 보인다. 따라서 이에 대한 교육과 더불어 불가피한 경우가 아니면 한영 전환이 단지 귀찮다는 이유로 한글 사용이 무시되어서는 안된다는 것을 주지시킬 필요가 있다.

한글이 과학적으로 매우 우수한 문자임에도 대소문자 구분이 없다는 것은 문자의 기능적, 시각적인 측면에서 아쉬운 면이다. Java에서 클래스의 식별자는 대문자로 시작하고, 인스턴스의 식별자는 소문자로 시작함으로써 이 두 사이를 쉽게 구분 지을 수 있다. 그리고 '새싹' 한글 프로그래밍 언어에서는 '학교 학1 = 새 학교();' 식으로 표현하고 있는데, 인스턴스명을 '학1', '학2'와 같이 표현하는 것은 가독성 있는 명명과는 거리가 있어 보인다.

특수문자 '$'를 '클래스'를 뜻하는 기호로 간주한다면, '$'을 붙이는 위치는 언어적 사고방식과 관계되어 있어 보인다. Java에서 변수를 선언할 때 'int 합계'처럼 '데이터 타입'이 먼저 오고 '식별자'가 뒤에 온다. 그러므로 Java 언어 체계를 따르면 '$'가 식별자 앞에 붙는 것이 타당해 보이고, 한국어 체계를 따르면 '$'가 식별자 뒤에 붙는 것이 타당하다. 그런데 <Table 1>의 Java의 식별자 규약에서 식별자가 '$'나 '_'로 시작하지 않도록 권고하고 있는 점을 함께 고려하면 식별자 뒤에 '$'를 붙이는 것이 더 바람직한 것으로 결론지을 수 있다.

4. 영어 식별자 사용의 대세론에 대한 논의

본 설문조사의 종합적 분석 결과로 볼 때 프로그램의 가독성과 가작성을 향상시키기 위해 한글 식별자 사용이 활성화되고 보편화 되는 데는 무엇보다도 영어 식별자 사용의 대세론이 걸림돌이 되고 있다. 본 연구에서는 한글 프로그래밍 언어의 총체적인 효용성에 대한 논의는 논외로 하고, 영어 식별자 사용의 대세론, 좀 더 포괄적으로 보면 학문과 산업발전을 위한 영어 대세론은 한글 프

로그래밍 언어의 보급과 활성화를 위해서도 반드시 넘어야 할 산이다. 따라서 본 장에서는 영어 식별자 선호의 이유로 제시된 한글 식별자의 기술적인 호환성 문제, 국제화 역량 배양, 그리고 SW 산업체의 영어 식별자 선호에 대해 논의하겠다.

4.1 한글 식별자의 기술적인 호환성 문제

유니코드(Unicode)는 국제표준으로 제정된 2바이트계 만국 공통의 국제 문자 세트이다. 1995년도에 유니코드 2.0이 제정되면서 한글 1,1172자가 모두 수용되어, 운영체제, 컴파일러, DBMS 등 각종 소프트웨어에서 한글이 국제적으로 일관되게 지원될 수 있게 되었고, 국내에서는 이를 KSC 5700이라는 칭하고 국가표준으로 정하였다.

기존의 문자코드들은 그 규모나 범위 면에서 한정되어 있고, 다국어 환경에서는 호환되지 않는 결정적인 문제점이 있었으나 유니코드는 이러한 문제들을 모두 해결시킨 명실상부한 국제표준이다. 그래서 모든 SW 개발은 세계적인 경쟁력을 높이기 위해 소프트웨어 현지화와 국제화 전략을 적극 구사하고 있으며, 당연히 유니코드 체계를 따라서 개발하고 있다. 유니코드가 국제표준으로 자리 잡은 지 '20여 년'이나 지난 지금도 한글의 기술적인 호환성 문제를 염려하는 것은 지나친 기우라고 아니할 수 없다.

4.2 국제화 역량 배양

세종 시절의 한글 사용 반대론이 얼마나 허망한 것이었는가? 세종 당시 한글이 이두보다도 더 비속하고, 그저 쉽기만 한 것이라 한글 사용을 활성화 하는 것은 어려운 한자로 된 중국의 높은 학문과 멀어지게 만들어 우리의 학문과 문화 수준을 떨어지게 할 것이라고 주장하였다. 그런데 그때로부터 600여 년이 지난 현대에 이르러서도 논문이나 인터넷을 통한 국제적인 교류가 모두 영어로 이루어지기 때문에, 프로그래밍에서 한글 식별자를 사용하면 영어를 등한시하게 되어 국제적 기술 수준과 멀어지게 될 것이라는 주장이 있다[1]. 그런데 요즘은 정말로 가치 있는 기술들은 순식간에 한글로 번역되어 소개되고 있으며, AI 번역이 급격하게 진보하여 2024년에는 인간의 번역 실력을 앞지를 것이라는 예측도 나왔다.

2016년 세계 영어 능력 평가 지수(EF EPI: Education First English Proficiency Index)에서 독일은 61.58로 9위, 인도는 57.30으로 22위, 한국은 54.87로 27위, 프랑스 54.33으로 29위, 일본은 51.69로 35위이다[17]. 그런데 과학기술정책연구원에 따르면 2012년 기준으로 세계 패키지 SW 시장에서 미국이 점유율 65.35%로 1위, 2위 독일(6.48%), 3위 일본(2.56%)이다[18]. IT서비스 시장에서는 미국이 37.8% 점유율로 1위, 2위 일본(11.7%), 3위 인도(4.9%), 한국 7위(1.3%)이다.

이러한 세계 IT 산업 통계로 볼 때, 영어 구사력이 국제화 역량을 결정하는 중요한 지표라면 EF EPI가 우리보다 낮은 일본이 우리보다 IT 산업이 발전된 이유를 설명할 수 없고, 특히 미국과 마찬가지로 영어를 모국어로 사용하는 영국과 캐나다가 미국과는 현격한 차이를 보이는 이유를 설명할 수 없다. 언어는 생각을 표현하는 수단에 불과하다. 영어 구사력을 국제화 능력과 동일시하는 풍토는 우리나라 4차 산업혁명의 미래에 지속적인 걸림돌이 될 것이다.

4.3 SW 산업체의 영어 식별자 선호

SW 산업체에서 영어 식별자 사용을 선호하는 이유는 앞서 언급한 '한글 식별자의 기술적인 호환성 문제'와 '국제화 능력 배양'에 주로 기인할 것으로 생각된다. 한글 식별자의 기술적인 호환성 문제는 1995년도에 유니코드 2.0이 제정되어 SW 산업계에 명실상부한 국제표준으로 자리 잡기 전의 과도기 상태에서 발생된 문제일 뿐이다. 이제 그 후로 20여 년의 세월이 흘렀으므로 '구더기 무서워 장 못 담글까'라는 생각까지 든다.

영어 구사력을 통한 국제화 능력 배양의 필요성은 프로그래밍 능력은 나름 뛰어난데 이에 걸맞은 영어 능력이 뒷받침 되지 못할 때의 푸념일 수 있다. 모든 투자에는 기회비용이라는 것이 있다. 영어를 한마디도 못하지만 2008년 노벨물리학상 공동수상한 마스카와 도시히데 교수는 다음과 같이 영어의 가치를 평가하였다. "영어로 된 물리용어는 안다. 그러나 영어로 말할 수는 없다. 그러나 물리는 할 수 있다." 일본이 영어를 못하는 선진국이 된 배경에는 자국의 언어로 학문을 하는 풍토가 한 몫하고 있다는 분석도 있다[19]. 일본의 가와바타 야스나리가 노벨문학상 수상자로 발표된 직후 가진 기자회견에서 "이 상의 절반은 번역자인 에드워드 사이덴스티커 씨의 것이다." 라고 말했던 일화도 시사하는 바가 크다[20]. 일본에서는 영어 번역 전문가의 도움을 받아 세계 유수의 학술지에 게재한다고 한다.

5. 결론 및 제안

본 연구에서는 컴퓨터학과 학생들을 대상으로 설문조사를 통해 프로그램의 가독성과 가작성을 향상시키기 위해 한글 식별자 사용이 활성화되고 보편화 되는 데는 무엇보다도 영어 식별자 사용의 대세론이 걸림돌이 되고 있다고 분석하고 그 대세론의 근거에 대해 논의하였다.

유니코드가 나오기 전까지는 프로그램용 이두문자의 사용이 불가피한 면이 있었다. 그러나 유니코드가 국제표준으로 명실공히 자리 잡은 지금에도 한글 식별자의 사용을 피해야 한다고 주장하는 것은 세종 이후 지속적으로 한글 사용을 천시하고 반대한 것과 비슷한 측면이 있다. 영어 식별자 사용은 프로그래머가 프로그램의 본질인 창의성과 알고리즘에 집중할 수 있는 시간을 빼앗아 본말이 전도되는 현상이 생길 수 있고,심한 경우는 프로그래밍 학습이 특이한 영작문 학습의 일종으로까지 인식될 수 있다. 실제로 영어 식별자를 극단적으로 선호하는 프로그래머는 주석까지도 영어로 달기 위해 과도한 시간을 할애하기도 한다.

여러 측면에서 한글 식별자 사용이 바람직함에도 불구하고 국제적인 기술 교류와 협력을 위해 프로그래머 각자가 우수한 영어 실력을 배양해야 하고, 이를 위해 평소 프로그램 작성 때 영어를 사용하도록 노력해야 한다는 주장은 우리나라 문학도들이 국제적으로 인정받기 위해 혹은 국제적인 문학 교류를 위해 영어로 작품을 쓸 수 있는 능력을 배양해야 하고, 이를 위해 평소에 영어로 작품을 써야 한다는 주장과 다를 바 없다. 일상적인 생활에서는 모어가 우리의 사고를 자유롭게 한다. 그런데 영어 식별자 사용은 프로그래밍 언어를 처음 배우는 학생들에게 프로그래밍은 매우 난해한 것이라는 인식을 심어줄 수 있다. 우리가 무엇인가를 창의하려고 하더라도 모어로 된 지식을 바탕으로 해야 한다.

소프트웨어공학은 소프트웨어 제품의 품질을 향상시키고, 소프트웨어 생산성과 작업 만족도를 증대시키는 것이 목적이다. 코드 품질의 기준으로서 가독성과 가해성(comprehensibility)은 아무리 강조해도 지나치지 않는다. 모든 사람은 모어로 소통할 때 가장 편안한 마음이 들고, 맥락을 이해하는 능력이 최고치로 발휘된다. 모어는 번역과 같은 사고과정을 거치지 않고 자연스럽게 자신의 의사를 표현하고 이해할 수 있는 논리체계이다[21]. SW 산업계는 소프트웨어공학의 목표에 충실해야 한다. 우리나라의 각 분야에서 발굴의 성취를 이룬 사람들은 자신의 전공 영역에 충실한 사람들이다. 우리나라 SW 산업의 발전 요인을 4차 산업혁명에 부응하는 협력적 집단지성, 비판적·융합적·창의적 사고에 기반한 교육, 그리고 정부 정책과 투자 등에서 찾아야지, 엉뚱한 영어에서 찾고 있는 것은 연목구어가 될 수 있다.

결과적으로 소프트웨어 생산성 향상을 위해, 프로그래머의 작업 만족도 향상을 위해, 유지보수의 용이성을 위해 한글 식별자 사용이 조속히 대세로 자리 잡아야 한다. 한글 식별자 사용이 프로그래밍 언어 습득성과 소프트웨어공학 측면에서 적극 권장된다면 한글 프로그래밍 언어의 활성화에도 매우 긍정적인 영향을 미칠 것이다. 이를 위해 프로그래밍 교재의 저자와 교육자, 그리고 SW개발업체의 개발팀장들의 적극적인 역할이 필요하며, 아울러 '전자정부 표준프레임워크'에서 한글 식별자 사용을 권고하도록 ' 코딩 규약'을 제정할 필요가 있다. 이를 위해 본 연구에서는 한글 식별자 사용을 위한 두 가지 명명법을 다음과 같이 제안한다.

① 한글 식별자에서 여러 단어가 연속될 때 복합명사가 아닌 경우는 '_'을 사용하여 연결한다. 예) 나의_폭

② 한글 클래스명 끝에 '$'를 붙여 인스턴스명과 구분시킨다. 예) 학교$ 학교 = new 학교$();

ACKNOWLEDGMENTS

이 논문은 2015학년도 평택대학교 학술연구비의 지원에 의하여 연구되었음.

REFERENCES

[1] D. H. Yang, "A Software Developer Living in the Second 'Idumunja' Period", Review of Korean Society for Internet Information, Vol. 15, No. 2, pp. 17-23, 2014.

[2] S. M. Oh, Introduction to Compiler, Jeongiksa, 1994.

[3] Y. S. Jeong, "The Development and Evaluation of Educational Hangeul Programming Language 'HanScript'", J. of Korean Association of Computer Education, Vol. 7, No. 3, 2004.

[4] D. H. Yang, A Critical Review on the Inherent Problems of MOOC, J. of the Korea Convergence

Society, Vol. 6, No. 6, pp. 293-299, 2015.
[5] wikipedia. "Non-English-based Programming Languages", http://en.wikipedia.org/wiki/Non-English-based_programming_languages.
[6] J. S. Cheon, D. Kang, G. W. Kim, G. Woo, "A Concise Korean Programming Language 'Sprout'", J. of KIISE, Vol. 42, No. 4, pp. 496-503, 2015.
[7] G. M. Kim, H. S. Kim, A Case Study on Necessity of Computer Programming for Interdisciplinary Education, J. of Digital Convergence, Vol. 12, No. 11, pp. 339-348, 2014.
[8] G. J. Park, Language Teaching Studies, Seoul: Parkyeongsa, 2011.
[9] J. K. Kim, "The Effect Of Using L1 On English Prewriting", Bilingual Research, Vol. 44, 2010.
[10] B. S, Sin, "Direct Writing Versus Translation Writing = Experimental Study On The Effects of Korean on English Writing", J. of Studies in Language, Vol. 14, No. 1, pp. 127-142, 1998.
[11] S. G. Park, S. J. Lee, A Study on the Effectiveness of the Korean Teaching as L2 Using Learners' Languages and Cultures - Focused on the Novice Level of Married Women Immigrant Learners, Studies of Korean & Chinese Humanities, Vol. 43, pp. 145-173, 2014.
[12] C. Park, The Effects of the Native Language Based Programming Learning on Academic Motivation in Secondary Education, Konkuk University, Master thesis, 2010.
[13] J. Y. Seo, SW Education for Non-SW Major in Order to Nurture SW Convergence Talent, J. of Digital Convergence, Vol. 15, No. 7, pp. 123-132, 2017.
[14] Mother Tongue, https://ko.wikipedia.org/wiki/모어
[15] J. S. Cheon, G. Woo, "Saesark: A Korean Object-Oriented Programming Language for Beginners", J. of the Korea Contents Society, Vol. 16, No. 3, pp. 288-295, 2016.
[16] Java Convention, http://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html.
[17] The World's Largest English Language Proficiency Examination Index 6th Edition, http://www.ef.co.kr/epi/
[18] G. I. Yun, "Software Powerhouse USA, Growth Background", http://www.etnews.com/20150914000090, ETNEWS, Sep. 20, 2015.
[19] D. Y. Kim, "He could not speak English, but he received the Nobel Prize", http://www.hani.co.kr/arti/international/japan/472562.html, The Hankyoreh News, Apr. 12, 2011.
[20] Y. Heo, "Translator who brought 'Snow Country' to world literature: Saidenseutikeo", MK News, http://premium.mk.co.kr/view.php?no=18615, May 4, 2017.
[21] D. H. Yang, Educational Problems with MOOC, Suggestions, and Convergence of MOOC and Universities, J. of the Korea Convergence Society, Vol. 7, No. 3, pp. 121-129, 2016.

저자소개

양 단 희(Dan-Hee Yang) [정회원]

- 1989년 2월 : 연세대학교 전산과학과(이학사)
- 1991년 2월: 연세대학교 대학원 전산과학과(이학석사)
- 1999년 8월: 연세대학교 대학원 컴퓨터과학과(공학박사)
- 1991년 2월~1995년 2월: 현대전자 소프트웨어연구소
- 2001년 2월~현재 정보과학회/정보처리학회/인터넷정보학회 논문지 심사위원
- 2001년 3월~현재 평택대학교 융합소프트웨어학과 교수

<관심분야> : 인공지능, 기계학습, 정보보안, 소프트웨어공학, 컴퓨터교육