# 객체 지향 프로그램 자동 평가 시스템

서주은 ○ 최 김 남

한동대학교 전산전자공학부•

{22000350, , }@handong.ac.kr

# Automatic Evaluation System for Object-Oriented Programs

Jueun Seo○ Bokwon Choi Yeongbin Kim Jaechang Nam

School of Computer Science and Electrical Engineering, Handong Global University

**요 약**

현재 대부분의 자동 채점 시스템은 작성된 테스트 케이스만을 기준으로 코드 평가를 진행한다. 이러한 채점 방식은 각 프로그래밍 언어가 가지는 고유의 특성을 전혀 반영하지 않는다. 본 연구에서는 새로운 객체 지향 프로그램 자동 평가 시스템을 제안함으로써 사용자의 프로그램 설계 능력과 객체 지향 개념에 대한 이해도를 향상시키고자 한다. 이를 위해 대표적인 객체 지향 프로그래밍 언어인 Java 를 이용한 코딩 테스트 및 프로그래밍 과제를 채점할 수 있는 프로그램 분석 엔진과 웹 어플리케이션인 jChecker 를 설계 구현 하였다. 그 결과 채점자의 물리적인 채점 시간이 크게 단축되었고 코드 제출자는 피드백을 반영하여 객체 지향 개념을 적절히 코드에 적용할 수 있게 되었다.

## 1. 서 론

코딩 테스트 및 과제를 자동으로 채점하는 프로그램들은 대부분 테스트 케이스를 기반으로 작동된다. 하지만 제한된 자원으로 정확도와 완성도가 높은 테스트 케이스를 생성하는 데에는 많은 어려움이 있다.

이를 해결하기 위해서 기존의 논문들은 자동 채점 기술에 Formal Semantics 와 Hyper-safety verification 을 적용하였다. Liu et al.은 의미적으로 다른 실행 경로를 찾음으로써 Formal Semantics 을 통해 자동 채점을 시도하였다 [1]. Anil et al.은 Hyper-safety verification 을 통해 코드 자체의 정확성을 평가하여 테스트 케이스 기반 자동 채점 프로그램이 가지는 한계를 극복하고자 하였다 [2]. 두 논문은 코드의 출력값만 비교하는 것이 아닌, 소스 코드 자체를 분석하는 채점 방식을 제안했기에 의의를 지닌다. 하지만 위의 방법만으로는 각 프로그래밍 언어가 가지는 고유의 특성을 반영하는 데 한계가 있다.

이러한 한계점을 극복하기 위해 객체 지향 프로그램 자동 평가 서비스인 jChecker 를 개발하였다. jChecker 는 객체 지향 개념 검사를 지원하는 자동 채점 엔진 기반의 웹 서비스이다. 채점자가 테스트 케이스와 encapsulation, inheritance 등의 객체 지향 개념을 채점 기준으로 웹 어플리케이션에 입력하면 이를 기반으로 제출한 프로그램을 자동 평가한다. 사용자가 제출한 소스 코드는 엔진 단계에서 Abstract Syntax Tree(AST)를 통해 구문 분석 과정을 거치며 채점 시 field 접근자나 상위 class 이름 AST node 정보를 통해 객체 지향 개념의 구현 여부를 확인한다. 작성된 채점 기준표를 바탕으로 채점이 끝나면 결과와 그에 대한 피드백을 시각화하여 사용자에게 제공한다. 채점 결과를 통해 사용자는 부족한 부분을 인지하고 객체 지향에 대해 스스로 학습할 수 있다.

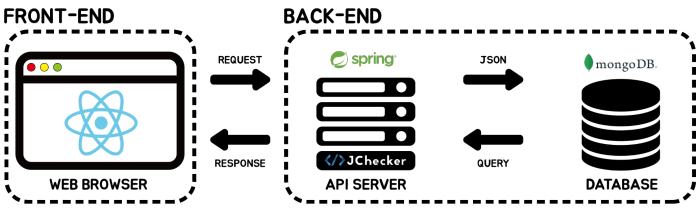


**그림 1**. 시스템 구조

## 2. 프로젝트 설계 및 해결방법

### 2.1 시스템 구조

그림 1 은 jChecker 프로젝트의 시스템 구조이다. 시스템은 백엔드, 프론트엔드, 데이터베이스 3 가지로 나뉜다. 객체 지향 개념을 효과적으로 확인하기 위해 소스 코드를 Abstract Syntax Tree (AST)를 이용하여 분석하는 엔진을 설계하였고, 이를 웹 서비스 형태로 제공하기 프론트엔드를 구현하였다.

1) 백엔드의 서버는 API 서버로 개발하였고 서버에는 채점 API 엔진이 내장되어 있다. 프론트엔드와 데이터베이스의 중간 다리 역할로서 채점 요청이 올 경우, 채점을 진행하여 데이터베이스에 결과를 저장한다. 이를 사용자에게 점수가 포함된 피드백 형식으로 반환한다. 이때 프론트엔드와의 정보

• 이 논문은 과학기술정보통신부의 소프트웨어중심대학 지원사업 (2017-0-00130)의 지원을 받아 수행하였음.

교환은 JSON 형태로 이루어진다.

2) 프론트엔드는 여러 사용자로부터 오는 다양한 요청을 처리한다. 백엔드 서버와 연결되어 있기에 사용자가 서버의 리소스를 필요로 하는 경우와 사용자가 프론트엔드에서 정보를 입력하여 저장하는 경우에 대한 처리가 가능하다. 본 프로젝트는 프론트엔드 개발에 React 를 사용함으로써 렌더링을 최적화하고 사용자 친화적인 인터페이스를 제공하고자 하였다.

3) 데이터베이스는 API 서버, 즉 백엔드의 요청에 맞게 채점 관련 데이터를 저장하고 반환한다. 데이터베이스로 mongoDB 를 사용하였다.

### 2.2 채점 API 엔진 상세 설계

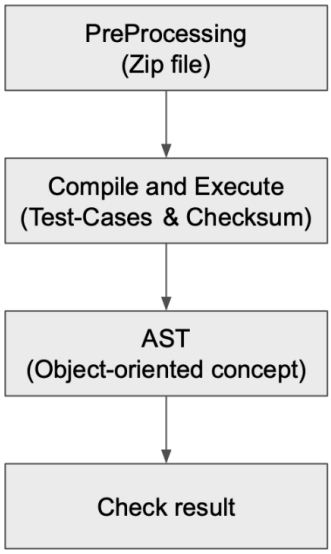


**그림 2**. 동작 및 흐름

본 절에서는 jChecker 기술의 핵심인 채점 엔진 설계를 상세하게 기술한다. 그림 2 는 자동 채점 API 엔진의 구체적인 동작 및 흐름을 나타낸다.

1) 전처리: 자동 채점을 진행하기 위해 사용자가 제출한 채점 폴더 경로 설정 등 채점 구동에 필요한 환경설정을 진행한다. 사용자는 Github 링크 또는 압축 폴더 형식으로 소스 코드를 제출한다. Github 링크의 경우, 링크를 참조하여 미리 설정한 경로에 git clone 명령어를 실행한다. 압축 폴더의 경우, Java 라이브러리를 사용하여 압축을 해제한다.

2) 컴파일 및 실행: 작성된 채점 기준에 따라 제출된 소스 코드를 컴파일하고 실행한다. 이를 통해 컴파일 가능 여부와 테스트 케이스 통과 여부를 확인할 수 있다. 채점 기준에 따라 프로그램 실행 결과가 파일로 생성될 경우, 정답 파일과 동일한 파일이 생성되었는지 체크섬 값을 비교하여 채점을 진행한다.

| AST를 이용해 채점할 수 있는 항목 |
|---|
| Encapsulation |
| Inheritance (SuperClass & Interface) |
| Custom Exception & Custom Data-Structure |
| Overriding method |
| Overloading method |
| Thread |
| Javadoc |
| Required Classes & Packages |
| Counting (Method, Field, Statements, etc…) |

**그림 3**. AST 기반 채점 가능 항목

3) 객체 지향 개념 평가: 그림 3 은 본 과정에서 평가하는 객체 지향 개념을 보여준다. 제출된 소스 코드에 필요한 객체 지향 개념 및 다양한 프로그래밍 요구 사항을 AST 정보로 확인한다. 예를 들어 encapsulation 구현 여부는 class 의 field 가 private 으로 선언되어 있는지, 해당 field 에 대한 getter method 및 setter method 가 정의되어 있는지를 통해 판단한다.

4) 결과 확인: 채점 결과를 JSON 형태로 반환한다. 마감 기한을 넘긴 지연 제출 여부를 추가적으로 확인한 후 채점 결과에 반영한다.

### 2.3 구현

1) 시스템 엔진: 자동 채점 API 엔진을 Java 로 구현하였다. Java 소스 코드의 AST 정보를 분석하기 위해 Eclipse JDT 를 사용하였다 [3]. AST 분석을 통해 class, package, Javadoc, method 관련 내용을 평가하였다. CSV 파일 처리와 엔진의 커맨드 라인 실행을 지원하기 위해 Apache commons CSV 와 CLI 라이브러리를 사용하였다. 채점 대상인 소스 코드의 의존성 및 설정에 관한 properties 를 확인하기 위해 commons configuration2 라이브러리도 활용하였다 [4]. 채점 기준과 결과, 데이터베이스 모두 JSON 형식의 데이터이므로 효율적인 데이터 처리를 위해 Gson 을 사용하였다 [5].

2) 백엔드: 서버 구현 시 최소한의 설정으로 간편하게 웹 어플리케이션을 개발하기 위해 Spring-boot 를 사용하였다 [6]. 외부 오픈 소스 라이브러리 사용과 배포 파일 생성을 위해


**그림 4**. 웹 사용자 접속 흐름

gradle 을 빌드 툴로 사용하였다. API 서버를 구축하고자 JSON 형식을 채택하였고 엔진에서 생성된 jar 파일을 백엔드 라이브러리에 포함하여 엔진의 여러 기능을 사용할 수 있게 설계하였다.

3) 프론트엔드: 완성도 높은 웹 서비스를 제공하기 위해 React 라이브러리를 사용하였다. Javascript 의 에러 가능성을 방지하기 위해 Typescript 의 사용 빈도를 높였으나 type 의 유연성을 보존해야 할 컴포넌트는 Javascript 로 구현하였다. 백엔드와 원활하게 데이터를 주고 받기 위해 Axios 를 사용하였다. 사용자에게 부드러운 웹 경험을 제공하고자 react-router-dom 라이브러리를 추가하였다. 원활한 렌더링은 immer 을 통해 보장하였다.

## 3. 설계평가

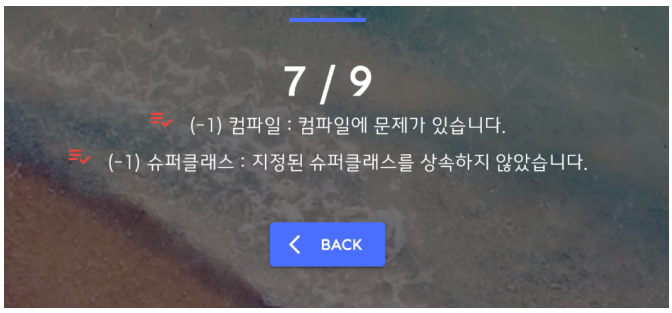


**그림 5**. 실제 피드백 예시

그림 4 는 jChecker 에 접속한 사용자의 흐름을 분석한 결과이다. 클래스 페이지(/classes)가 상호작용에 지속적으로 등장하는데 이는 사용자가 해당 페이지에서 여러 번의 채점을 시도하고 있음을 의미한다. 예를 들어 필요한 inheritance 가 제출된 코드에 구현되어 있지 않으면 이는 요구된 객체 지향 개념을 위반한 것이다. 그림 5 는 inheritance 개념을 구현하지 않은 코드에 대한 실제 피드백 화면이다. 사용자는 이에 대한 피드백을 반영하여 반복적으로 프로그램을 수정 및 확인할 수 있다. 채점자는 많은 수의 제출물을 수동으로 실행하지 않고 jChecker 의 결과를 점수에 반영함으로써 채점 시간을 단축할 수 있다. 이를 통해 jChecker 시스템의 도입으로 채점의 편리성이 향상되었음을 유추할 수 있다.

## 4. 결론 및 향후 방향

jChecker 는 채점 대상이 되는 소스 코드를 테스트 케이스로 채점함과 동시에 제출자가 객체 지향 개념을 정확히 이해하고 적용하였는지 평가함으로써 기존 채점 시스템이 가진 문제점을 극복하였다.

객체 지향 프로그램 자동 채점 기능을 웹 서비스로 구현하였고 사용자가 반복하여 자신의 코드를 평가 받을 수 있도록 설계하였다. jChecker 시스템을 통해 사용자는 자신의 코드를 직관적으로 확인할 수 있다. 또한 채점 결과를 반영한 피드백을 통해 개개인의 프로젝트에 객체 지향 개념을 효과적으로 적용하게 된다. 채점자 입장에서 채점을 진행하는 데 걸리는 시간을 크게 단축하였고 더 정확한 평가 결과를 제공할 수 있게 되었다.

지원하는 프로그래밍 언어를 확대하고 사용자 간의 코드 유사도를 분석하는 기능을 추가한다면 더 정밀하고 유연한 서비스를 제공할 수 있을 것으로 기대된다. jChecker 엔진 프로젝트를 오픈 소스로 공개하였다.[1]

## 5. 참고문헌

[1] X. Liu, S. Wang, P. Wang and D. Wu, "Automatic Grading of Programming Assignments: An Approach Based on Formal Semantics", 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), 126-137p, 2019

[2] Jude K Anil, Sumanth Prabhu S, Kumar Madhukar, and R Venkatesh, “Using hypersafety verification for proving correctness of programming assignments”, Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER '20), 81-84p, 2020

[3] https://www.eclipse.org/jdt/

[4] https://commons.apache.org/

[5] https://github.com/google/gson.git

[6] https://spring.io/projects/spring-boot

[1] https://github.com/HGUISEL/JChecker-Engine.git