

# Code Slayer

Type Faster - Go Further

컴퓨터공학과 | 윤건호 팀장

컴퓨터공학과 | 안치원 팀원

TEAM

7

팀원 김상원 | 컴퓨터공학과

팀원 우은혁 | 컴퓨터공학과

- 과목명 : C++프로그래밍
- 담당교수 : 안용학
- 수업시간 : 13:30 ~ 15:00
- 제출일자 : 2020. 05. 13.

## 1. 문서 개요

### 가. 프로젝트 개요

- 프로그램명 : CodeSlayer
- 개발 배경 :
  - (1) 영어 코드 타자가 익숙하지 않은 사용자의 타자 실력 증진
  - (2) C++ 언어의 키워드를 활용하여 사용자의 코딩 생산성 증진
  - (3) 위 요소와 미니 게임을 결합하여 흥미 유발
- 주요 기능 :
  - (1) C++ 코드 타자 연습
  - (2) C++ 코드 빈칸 추론 게임

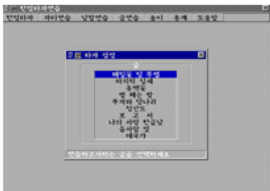
### 나. 목차

- 1) 문서 개요 | 프로젝트의 개요와 문서의 목차입니다.
- 2) 개발 내용 | 최종 개발 목표와 개발 내용입니다.
- 3) 클래스 설계 | 클래스를 정의하고 설계합니다.
- 4) 클래스 응용 | 클래스의 기능과 역할, 그리고 응용 명세서입니다.
- 5) 팀원 구성 | 팀과 팀원을 소개하고, 팀원의 역할을 서술합니다.
- 6) 개발 일정 | 프로그램 개발 일정입니다.
- 7) 기대 효과 | 프로그램 서비스 시 기대 효과입니다.


## 2. 개발 내용

### 가. 최종 개발 목표

사용자의 코딩 생산성 향상을 최종 개발 목표로 한다. 이를 달성하기 위해, 크게 두 가지의 개발 목표를 설정하였다. 첫째, “한컴타자연습”에서 착안한 타자 연습 기능이다. “한컴타자연습”에서는 단어연습, 짧은 글 연습, 긴 글 연습을 활용하여 사용자의 한글 타자 실력 향상을 돕는다. 이 프로그램에서는 “한글” 대신 “영어” 또는 “C++ 키워드 또는 코드”를 활용한다. 이를 통해 사용자는 영어, C++ 키워드, 그리고 코드에 익숙해질 수 있다. 이는 곧 코딩 생산성의 향상으로 이어진다. 둘째, “행맨”에서 착안한 미니 게임 기능이다. “행맨”은 플레이어가 숨겨진 철자를 추측해가며, 최종 단어를 맞히는 게임이다. 이 프로그램에서는 “행맨”의 기존 숨겨진 철자 대신, 특정 코드의 가려진 부분을 맞히는 형식으로 제작한다. 이를 통해 사용자는 코딩에 흥미를 느낄 수 있으며, 추측하는 과정을 통해 코딩 사고력을 증진할 수 있다.

타자연습 & C++																																						
	<table border="1"> <thead> <tr> <th colspan="4">Cand C++ Common Keywords</th> </tr> </thead> <tbody> <tr> <td>auto</td> <td>double</td> <td>int</td> <td>struct</td> </tr> <tr> <td>break</td> <td>else</td> <td>long</td> <td>switch</td> </tr> <tr> <td>case</td> <td>enum</td> <td>register</td> <td>typedef</td> </tr> <tr> <td>char</td> <td>extern</td> <td>return</td> <td>union</td> </tr> <tr> <td>const</td> <td>float</td> <td>short</td> <td>unsigned</td> </tr> <tr> <td>continue</td> <td>for</td> <td>signed</td> <td>void</td> </tr> <tr> <td>default</td> <td>goto</td> <td>sizeof</td> <td>volatile</td> </tr> <tr> <td>do</td> <td>if</td> <td>static</td> <td>while</td> </tr> </tbody> </table>	Cand C++ Common Keywords				auto	double	int	struct	break	else	long	switch	case	enum	register	typedef	char	extern	return	union	const	float	short	unsigned	continue	for	signed	void	default	goto	sizeof	volatile	do	if	static	while	<ul style="list-style-type: none"> <li>• “한컴타자연습”과 C++ 프로그래밍 언어를 결합</li> <li>• 사용자의 C++ 프로그래밍 언어 적응을 도움</li> </ul>
Cand C++ Common Keywords																																						
auto	double	int	struct																																			
break	else	long	switch																																			
case	enum	register	typedef																																			
char	extern	return	union																																			
const	float	short	unsigned																																			
continue	for	signed	void																																			
default	goto	sizeof	volatile																																			
do	if	static	while																																			

\* 타자 연습과 C++ 프로그래밍 언어를 결합한 코드 타자 연습 \*

행맨 & C++		
	<pre>/* 모든 자릿수의 합을 반환 함수 */ int sum(int number) {     if (number &lt; 10)         return number;     return (number % 10) + sum(number / 10); }</pre>	<ul style="list-style-type: none"> <li>• “행맨”과 C++ 프로그래밍 언어를 결합</li> <li>• 사용자의 코딩 사고력 증진</li> </ul>

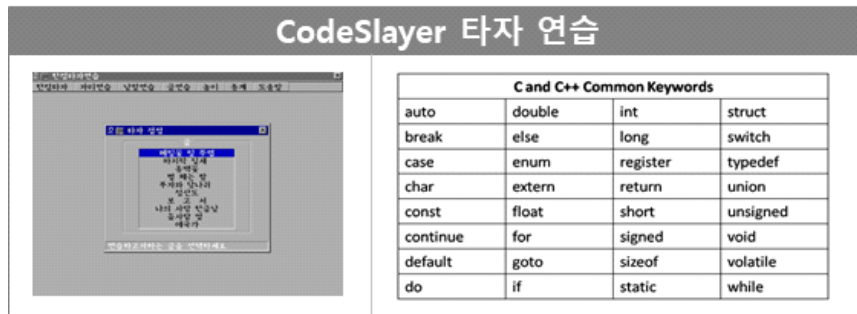
\* 행맨과 C++ 프로그래밍 언어를 결합한 미니 게임 \*

## 나. 개발 내용

### 1) 타자 연습 : 코드 타자 연습 기능 제공

사용자가 C++ 코드를 빠르고 정확하게 작성하도록 돕는다. 기본적인 틀은 다음과 같다. 첫째, 사용자에게 입력할 텍스트(C++ 키워드, 코드, 문장 등)를 출력한다. 둘째, 사용자는 주어진 텍스트를 최대한 빠르고 정확하게 입력한다. 셋째, 프로그램은 주어진 텍스트를 사용자가 얼마나 빠르고 정확하게 입력하였는지 측정한다. 마지막으로, 측정한 결과를 활용하여 사용자의 정확도와 타자 속도를 기록한다. 기록은 추후 열람할 수 있으며, 사용자는 자신이 얼마나 발전하였는지 확인할 수 있다. 텍스트의 길이 또는 사용자의 목적에 따른 다음 세 가지 기능을 제공한다.

- (1) 단어연습 : 텍스트 파일에서 무작위로 추출한 단어 하나가 주어진다. 단어에는 C++ 키워드, 프로그램에서 사용한 변수명 등이 포함된다.
- (2) 짧은 글 연습 : 텍스트 파일에서 무작위로 추출한 문장 하나가 주어진다. 문장에는 간단한 코드, C++에서 자주 사용하는 함수명 등이 포함된다.
- (3) 긴 글 연습 : 무작위 텍스트 파일에서 추출한 단락을 주어진다. 단락에는 일반적인 C++ 코드, 프로그램에서 사용한 함수 등이 포함된다.

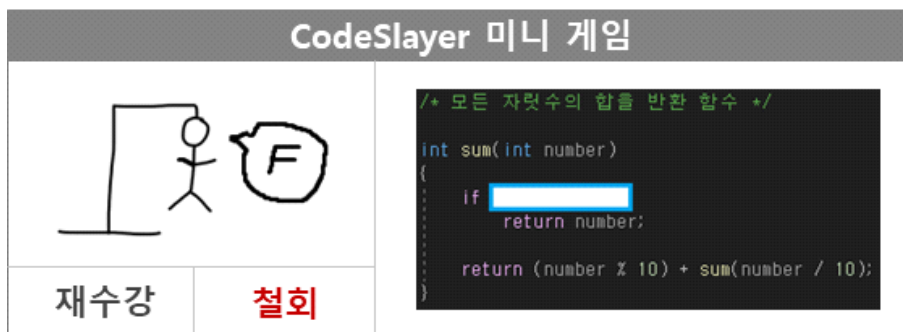


\* C++ 타자 연습 \*

### 2) 미니 게임 : “행맨”과 코딩을 결합한 미니 게임

CodeSlayer는 타자 연습뿐만 아니라, C++과 “행맨”을 결합한 새로운 게임을 제공한다. 이를 활용하여 사용자의 코딩 사고력 증진을 돕는다. 기본적인 틀은 다음과 같다. 첫째, 빈칸이 뚫려 있는 코드와 코드의 목적을 출력한다. 둘째, 사용자는 코드의 목적을 바탕으로 빈칸을 채워 넣는다. 셋째, 프로그램은 사용자가 입력한 코드의 답을 판별한다. 오답이 많을수록, 교수대 그림이 완성돼가고 사용자의 학점(목숨)은 떨어진다. 교수대 그림이 완성됨과 동시에 학점이 F가 되면, C++ 과목을 철회하면서 패배한다. “행맨”의 상세 규칙은 다음과 같다.

“행맨” : 행맨은 영어 단어 맞추기 게임 중 하나이다. 처음에는 글자 수만큼의 빈칸과 사람이 없는 교수대가 출력된다. 플레이어는 26개(영어 기준)의 철자 중 하나를 선택한다. 그 철자가 단어에 포함되어 있다면, 해당하는 빈칸에 철자가 채워진다. 만약 제시한 철자가 단어에 포함되어 있지 않다면, 교수대 아래에 밧줄, 머리, 팔, 손, 몸통, 다리, 발 순서로 사람이 그려진다. 그림이 완성되면 게임에서 패배하고, 단어를 완성하거나 알아내면(규칙에 따라 다름) 게임에서 승리한다.



\* C++ 미니 게임 \*

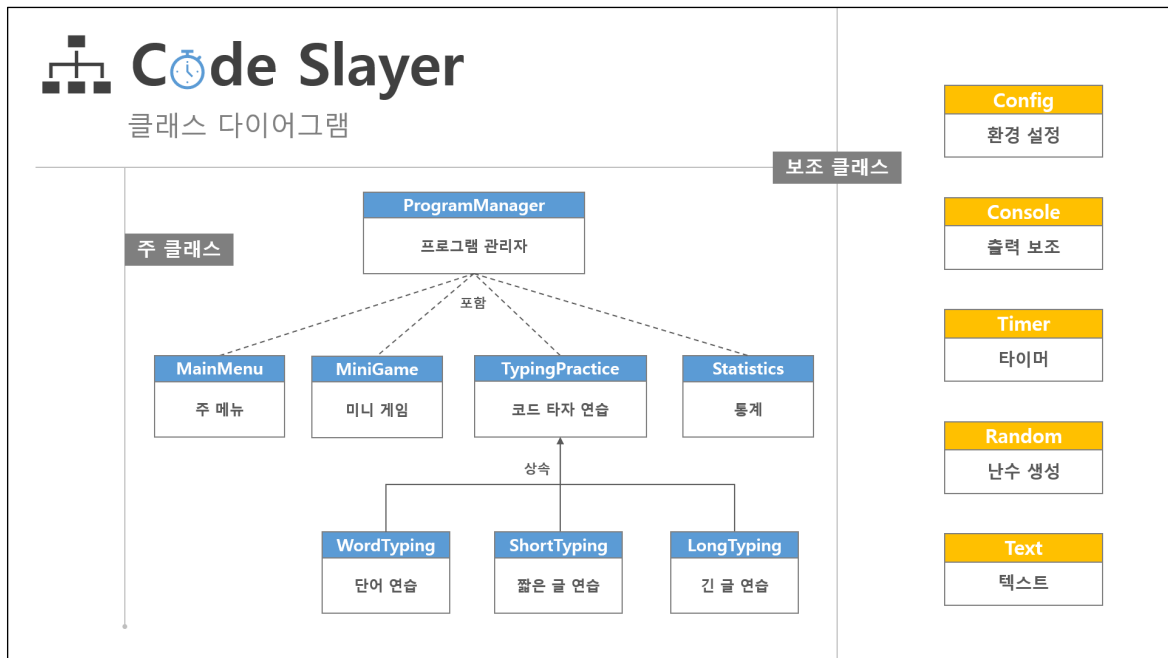
### 3. 클래스 설계

#### 가. 개요

프로그램에서 사용하는 **클래스의 설계 명세서**이다. 먼저, 전체적인 다이어그램을 제시하여 프로그램의 흐름 파악을 돕는다. 그다음으로, 프로그램의 주요한 기능들을 담당하는 주 클래스를 살펴본다. 마지막으로, 프로그램에서 사용하는 유용한 함수를 가지고 있는 보조 클래스를 살펴본다. 보조 클래스는 어느 클래스에 서나 포함하여 사용할 수 있다.

#### 나. 클래스 다이어그램

클래스 다이어그램을 활용하여 프로그램의 **전체 클래스 구조**를 설명한다. 프로그램의 전체적인 흐름은 최 상단에 있는 ProgramManager 클래스가 관리한다. 그리고 MainMenu, TypingPractice, MiniGame, Statistics 클래스는 각각 주메뉴, 코드 타자 연습, 미니 게임, 통계 기능을 담당한다. ProgramManager에 서는 네 가지 기능 중 사용자가 원하는 기능을 호출하여, 해당 객체의 Loop를 실행한다. TypingPratice는 세부적으로 단어연습, 짧은 글 연습, 긴 글 연습의 세 가지 기능으로 나누어진다.



\* 클래스 다이어그램 \*

#### 다. 주 클래스

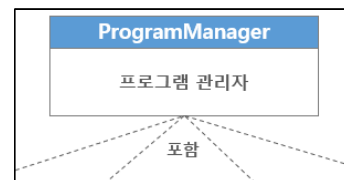
- 1) ProgramManager : 프로그램의 Main 기능을 담당하는 클래스이다. 프로그램이 구동하는 데에 필요한 객체들을 생성하고, 프로그램의 전체적인 흐름을 관리한다.

##### (1) SCREEN mCurrentScreen

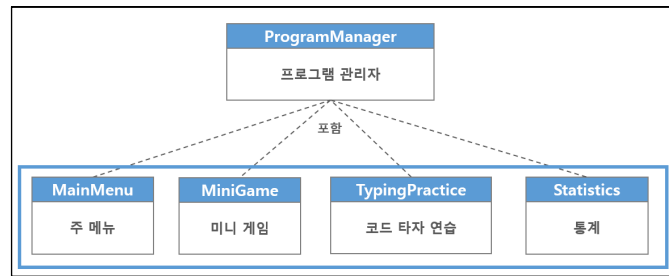
- enum class를 활용, 현재 선택한 기능을 저장한다.
- mCurrentScreen의 값을 switch문에 활용하여, 다음 기능으로 이동한다.

##### (2) void MainLoop()

- 프로그램의 메인 루프이다. 프로그램이 종료될 때까지 계속 반복한다.
- 반복문과 switch문을 활용하여 각 기능으로 이동한다.



\* 프로그램 관리 클래스 \*



\* 주요 기능 클래스 명세서 \*

2) **MainMenu** : 프로그램의 주메뉴를 제공하고, 다음 기능을 선택한다.

(1) **SCREEN mNextScreen**

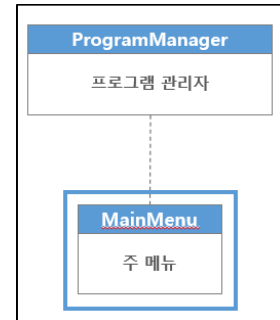
- 다음으로 이동할 스크린의 번호
- ProgramManager에 전달

(2) **void Main()**

- MainMenu 클래스를 실행한다.
- 주메뉴를 출력하고, 사용자의 다음 메뉴 선택을 입력으로 받는다.

(3) **SCREEN GetNextScreen()**

- ProgramManager에서 호출하여, mNextScreen 값을 반환받는다.
- ProgramManager는 반환받은 스크린으로 이동한다.



\* MainMenu \*

3) **MiniGame** : C++과 “행맨”을 결합한 게임을 제공한다.

(1) **void Main()**

- MiniGame 클래스를 실행한다.
- 게임의 전체적인 흐름을 관리한다.

(2) **Bool IsCorrect(string userAnswer, string correctAnswer)**

- 사용자가 입력한 문자열과 정답 문자열을 비교한다.
- 판별 후 값을 반환한다.



\* MiniGame \*

4) **TypingPractice** : 타자 연습에서 사용하는 기능을 제공한다.

(1) **Setter**

- 타자 연습에서 사용하는 변수의 값을 지정한다.
- 사용자의 타자 속도, 정확도, 정타, 오타 등이 포함된다.

(2) **Getter**

- 타자 연습에서 사용하는 변수의 값을 반환한다.
- 사용자의 타자 속도, 정확도, 정타, 오타 등이 포함된다.

(3) **void WriteFile()**

- 측정된 사용자의 타자 속도와 정확도를 파일에 기록한다.



\* TypingPractice \*

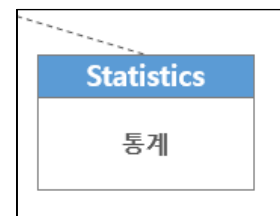
5) **Statistics** : 정확도, 타자 속도 등의 최근 기록을 확인할 수 있다.

(1) **void ReadFile(string path)**

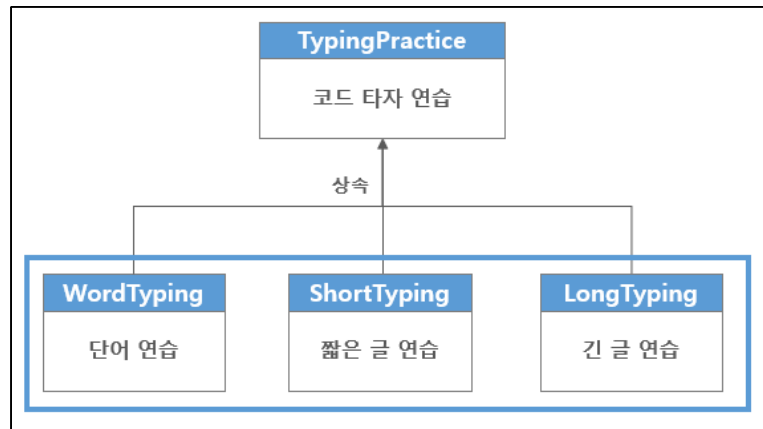
- 사용자의 기록을 텍스트 파일에서 불러온다.
- 불러온 기록을 객체에 저장한다.

(2) **void Main()**

- 통계 클래스를 실행한다.
- 사용자의 입력을 받아, 해당 기록을 출력한다.



\* Statistics \*

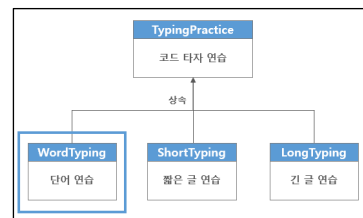


\* TypingPractice의 파생 클래스 명세서 \*

6) WordTyping : 단어연습 기능을 제공한다.

(1) void Main()

- 단어연습 클래스를 실행한다.
- 단어연습의 메인화면을 출력한다.
- 연습을 시작하거나, 주메뉴로 돌아갈 수 있다.

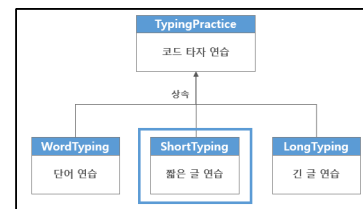


\* WordTyping \*

7) ShortTyping : 짧은 글 연습 기능을 제공한다.

(1) void Main()

- 짧은 글 연습 클래스를 실행한다.
- 짧은 글 연습의 메인화면을 출력한다.
- 연습을 시작하거나 주메뉴로 돌아갈 수 있다.

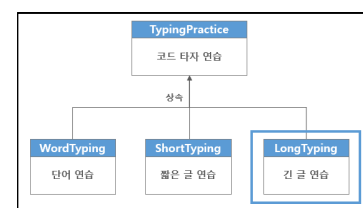


\* ShortTyping \*

8) LongTyping : 긴 글 연습 기능을 제공한다.

(1) void Main()

- 긴 글 연습 클래스를 실행한다.
- 긴 글 연습의 메인화면을 출력한다.
- 연습을 시작하거나 주메뉴로 돌아갈 수 있다.



\* LongTyping \*

## 라. 보조 클래스

프로그램에서 사용하는 **보조 클래스의 명세서**이다. 각 클래스에 상속하는 것이 아니라, 각 클래스에서 필요할 때 포함하여 사용한다. 프로그램의 환경 설정을 저장하는 Config, 콘솔 출력을 보조하는 Console, 시간을 관리하는 Timer, 난수를 생성하는 Random, 텍스트 관리를 담당하는 Text의 다섯 클래스가 있다.

Config	Console	Timer	Random	Text
환경 설정	출력 보조	타이머	난수 생성	텍스트

\* 보조 기능 클래스 명세서 \*

1) Config : 프로그램의 환경 설정을 관리한다.

(1) Getter

- 프로그램의 각 환경 설정값을 반환한다.
- 환경 설정값에는 텍스트 파일의 개수 등의 상수가 포함된다.
- 일반적으로 컴파일 타임에 초기화된 상수를 다룬다.

Config
환경 설정

2) Console : 콘솔 창의 출력을 보조한다.

(1) void SetCursorPosition(int x, int y)

- 콘솔 창의 커서를 (x, y)로 이동한다.

(2) void SetColor(string foreground, string background)

- 출력할 문자의 글자 색상과 배경 색상을 지정한다.

(3) void Clear()

- 콘솔 창을 비운다.

Console
출력 보조

3) Random : 범위 내의 무작위 난수를 반환한다.

(1) static int GetInteger(int rangeStart, int rangeEnd)

- rangeStart 이상 rangeEnd 이하의 정수를 반환한다.

(2) static double GetReal(int rangeStart, int rangeEnd)

- rangeStart 이상 rangeEnd 미만의 실수를 반환한다.

Random
난수 생성

4) text : 프로그램에서 사용하는 텍스트를 관리한다.

(1) Setter

- 텍스트의 문자열과 정보 변수를 저장한다.
- 정보 변수에는 텍스트의 길이, 텍스트 사용 여부 등이 포함된다.

(2) Getter

- 텍스트와 텍스트의 정보 값을 반환한다.
- 문자열과 텍스트의 길이, 텍스트가 사용되었는지의 여부 등이 포함된다.

(3) void ReadFile()

- 텍스트 파일에서 문자열을 불러온 다음, 멤버 변수에 저장한다.

Text
텍스트

5) timer : 타수 계산에 필요한 시간을 관리한다.

(1) Setter

- 시작 소요 시간, 걸린 시간을 저장한다.
- 걸린 시간은 (현재 시각 - 시작 시각)으로 저장된다.

(2) Getter

- 시작 시각, 걸린 시간을 반환한다.
- 타수 계산에 필요한 걸린 시간을 Getter를 이용하여 반환한다.

Timer
타이머

## 4. 응용 설계

### 가. 개요

설계한 클래스들이 프로그램에서 어떤 기능을 수행하는지, 즉 **클래스의 역할**을 서술한다. 보조 클래스는 어느 클래스에서나 포함하여 사용할 수 있으므로, 가장 먼저 보조 클래스의 기능을 살펴본다. 그리고 주 클래스의 기능과 역할, 그리고 보조 클래스 응용을 살펴본다.

### 나. 보조 클래스

<b>Config</b>	<b>Console</b>	<b>Timer</b>	<b>Random</b>	<b>Text</b>
환경 설정	출력 보조	타이머	난수 생성	텍스트

\* 보조 클래스 \*

#### 1) Config : 프로그램 환경 설정 관리

프로그램에서 사용하는 환경 변수들을 저장하고 관리한다. 각각의 클래스에서 상수를 (#define)으로 정의하는 대신, 하나의 Config 객체에서 관리하므로 유지 보수가 쉽다. 환경 변수들은 상수 멤버로 저장된다. 컴파일 타임에 초기화되고, 실행 중에는 수정할 수 없다. 프로그램 환경 변수에는 텍스트 프리셋 파일의 개수(MAX\_WORDTEXT, MAX\_LONGTEXT) 등이 있다. 프로그램의 시작할 때 하나의 객체를 생성하고, 각 클래스에서는 객체의 주소를 참조하여 접근한다.

<b>Config</b>
환경 설정

#### 2) Console : 콘솔 창 출력 보조

콘솔 창의 출력을 보조한다. 크게 세 가지의 기능을 멤버 함수로 제공한다. 첫째, 콘솔 창에서 문자의 출력 좌표를 지정한다. 레이아웃 또는 텍스트를 원하는 위치에 출력할 수 있다. 둘째, 출력 색깔을 지정한다. 문자의 글자색과 배경 색을 지정할 수 있다. 셋째, 화면을 비운다. 콘솔 출력 창의 모든 문자를 제거한다.

<b>Console</b>
출력 보조

#### 3) Timer : 시간 측정

시간 측정이 필요한 클래스에서 사용한다. 사용자가 첫 타를 치는 순간의 “시작 시각”과 (현재시간-시작 시각)으로 정의되는 “걸린 시간”을 멤버 변수로 가지고 있다. 타자 연습 클래스에서, 타자 속도를 계산할 때 필요한 “걸린 시간”을 측정한다. 걸린 시간을 반환받아, “순간 타자 속도”와 “평균 타자 속도”를 계산할 수 있다.

<b>Timer</b>
타이머

#### 4) Random : 무작위 난수 생성

사용자가 입력한 범위 내에서 무작위 난수를 생성하고, 정수 또는 실수로 반환한다. C++ <random> 라이브러리에서 제공하는 하드웨어 시드와 메르센-트위스터 19937 알고리즘을 활용한다. 따라서 품질이 높은 난수를 얻을 수 있다.

<b>Random</b>
난수 생성

#### 5) Text : 텍스트 관리

프로그램에서 사용하는 텍스트를 관리한다. Text 객체는 텍스트 파일에서 읽어온 문자열(string), 그리고 필요한 변수(bool isUsed, int TextLength 등)와 기능을 포함하고 있다. 프로그램에서는 텍스트 데이터를 Text 객체를 활용하여 관리한다.

<b>Text</b>
텍스트



## 다. 주 클래스

1) **ProgramManager** : 프로그램의 전체적인 흐름을 관리한다.

(1) 필요한 객체 생성

- 프로그램에서 사용하는 객체들을 생성한다.

(2) 소개 화면 출력

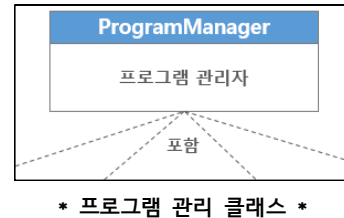
- 팀 로고와 게임 로고를 출력하여 프로그램을 소개한다.
- 몇 초 후, 주메뉴를 호출한다.

(3) 프로그램 흐름 관리

- 무한 루프와 switch()문을 활용하여, 각 기능을 순회한다.
- 사용자가 프로그램을 종료할 때까지 반복한다.

(4) 보조 클래스 응용

- Text : 팀 로고와 게임 로고를 프로그램에 저장한다.
- Console : 로고를 출력할 때 좌표와 색상을 지정하기 위해 사용한다.



\* 프로그램 관리 클래스 \*

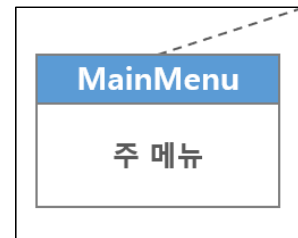
2) **MainMenu** : 주메뉴를 제공하고, 다음 메뉴를 반환한다.

(1) 주메뉴 기능

- 사용자에게 주메뉴를 출력한다.
- 사용자는 단어연습, 게임, 통계 등 다음 메뉴를 선택할 수 있다.
- 선택한 메뉴를 멤버 변수에 저장한다.

(2) 다음 메뉴 반환

- 함수를 이용, 다음 메뉴의 번호를 ProgramManager 클래스로 반환한다.
- 각 메뉴의 번호는 enum class SCREEN{...}에 정의되어 있다.



\* 프로그램 관리 클래스 \*

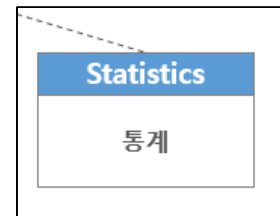
3) **Statistics** : 사용자의 기록을 통계 형태로 제공한다.

(1) 기본 구조

- 텍스트 파일을 다루는 기능 위주로 함수를 구현한다.
- 사용자가 원하는 통계 메뉴를 입력하면, 해당 메뉴의 통계를 제공한다.

(2) 보조 클래스 응용

- Console : 시각적 효과를 위해 색상을 지정한다.
- Text : 텍스트 객체를 활용하여 파일을 불러온다.



\* 통계 클래스 \*

4) **TypingPractice** 및 파생 클래스 : 타자 연습 기능을 제공한다.

(1) 기본 기능

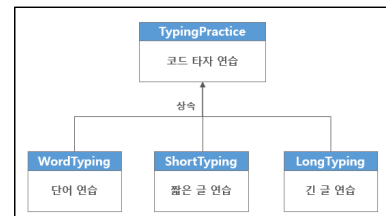
- 타자 속도 및 정확도 계산, 정답 판별 등의 기능을 포함한다.
- 각 메뉴에 해당하는 타자 연습을 실행한다.

(2) 파생 클래스들의 흐름

- 한 문항을 불러온다.
- 사용자의 입력을 받아, 타자 속도 등의 변수를 업데이트한다.
- 무작위의 다음 문항을 불러온다.
- 위 과정을 일정 횟수 반복 후, 기록을 출력하고 파일에 저장한다.

(3) 보조 클래스 응용

- Timer : 타자 속도를 계산할 때, 시간을 측정한다.
- Random : 새로운 텍스트 파일을 불러올 때, 무작위 난수 번호의 파일을 선택한다.



\* 타자 연습 클래스 \*

## 5) MiniGame : 미니 게임을 제공한다.

### (1) 기본 구조

- 게임의 규칙을 설명하고, 현재 학점(목숨)을 출력한다.
- 무작위 코드(적당한 길이의 함수) 한 개가 설명과 함께 출력된다.
- 사용자는 빈칸에 들어갈 적절한 코드를 작성한다.
- 정답을 맞히면 학점이 유지되고, 틀릴 때마다 학점이 한 단계씩 감소한다.
- 학점이 F가 되면, 과목을 철회하면서 패배하고 함수를 종료한다.
- 학점이 F가 아니면서 마지막 문제까지 끝냈다면, 사용자의 학점을 출력하고 함수를 종료한다.



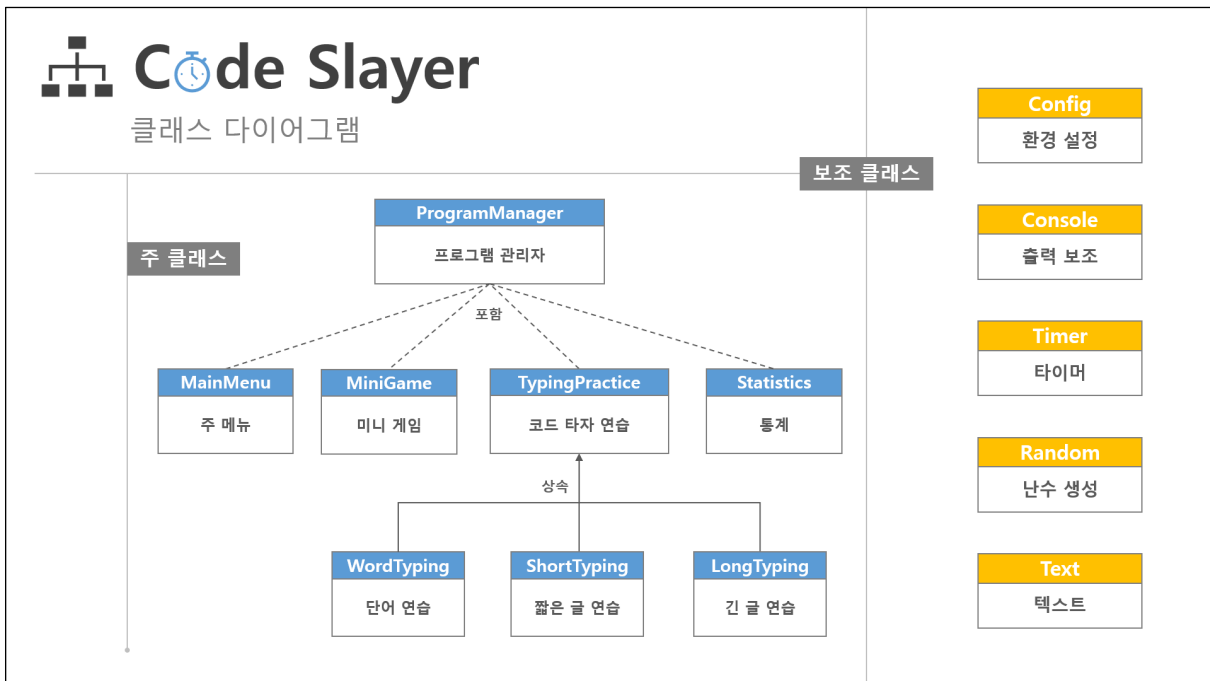
\* MiniGame \*

### (2) 정답 판별 함수

- 정답 문자열과 사용자의 입력 문자열을 비교한다.
- 일치하면 True, 틀리면 false를 반환한다.

### (3) 보조 클래스 응용

- Console : 좌표와 색상을 지정하여 출력한다.
- Text : 게임에서 사용할 텍스트를 Text 객체를 활용하여 사용한다.
- Random : 게임에서 사용할 프리셋 데이터를 무작위로 선택할 수 있다.

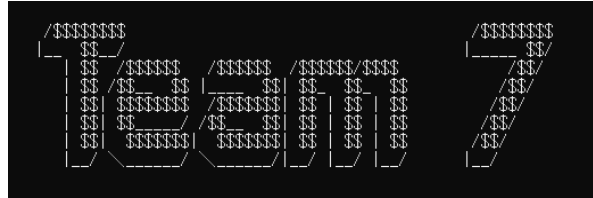


\* 전체 구조 \*

## 5. 팀원 구성

### 가. 팀 소개

- 1) 팀명 : Team 7
- 2) 팀장 : 윤건호
- 3) 팀원 : 김상원, 안치원, 우은혁



\* 팀 세븐 \*

### 나. 역할 분담

#### 1) 팀장 윤건호

- (1) 프로젝트 지휘
- (2) 프로젝트 관리
- (3) 프로젝트 문서화
- (4) 클래스 구현
  - 주 : ProgramManager, MainMenu, Statistics
  - 보조 : Config, Console, Random
- (5) 시나리오 테스트
  - 오류 수정
  - 성능 개선

윤건호	
번호	역할
1	프로젝트 지휘
2	프로젝트 관리
3	프로젝트 문서화
4	클래스 구현
5	시나리오 테스트

#### 2) 팀원 김상원

- (1) 클래스 구현
  - 주 : MiniGame
- (2) 데이터베이스 추가
  - 미니 게임 프리셋 데이터 추가
  - 타자 연습 프리셋 데이터 추가
- (3) 시나리오 테스트
  - 오류 수정
  - 성능 개선

김상원	
번호	역할
1	클래스 구현
2	데이터베이스 추가
3	시나리오 테스트

#### 3) 팀원 안치원

- (1) 클래스 구현
  - 주 : TypingManager, WordTyping, ShortTyping, LongTyping
  - 보조 : Timer, Text
- (2) 데이터베이스 추가
  - 타자 연습 프리셋 데이터 추가
  - 미니 게임 프리셋 데이터 추가
- (3) 시나리오 테스트
  - 오류 수정
  - 성능 개선

안치원	
번호	역할
1	클래스 구현
2	데이터베이스 추가
3	시나리오 테스트

#### 4) 팀원 우은혁

- (1) 클래스 구현
  - 주 : Statistics, WordTyping, ShortTyping, LongTyping
- (2) 데이터베이스 추가
  - 타자 연습 프리셋 데이터 추가
  - 미니 게임 프리셋 데이터 추가
- (3) 시나리오 테스트
  - 오류 수정
  - 성능 개선

우은혁	
번호	역할
1	클래스 구현
2	데이터베이스 추가
3	시나리오 테스트

## 6. 개발 일정

### 가. 개요

제안서 작성일로부터 6월 7일까지의 **CodeSlayer 개발 일정**을 나타낸다. 가장 먼저 전체적인 프로젝트 일정을 표의 형태로 제시한다. 그리고 개인 일정을 상술한다. 개인 일정은 각자 맡은 역할이 명확할 때는 자세하게 구분하였으나, 공통 작업의 경우에는 개인 일정 없이 전체 일정에만 작성하였다.

### 나. 전체 일정

구분	개발 세부 내용	5.7 ~ 5.13	5.14 ~ 5.21	5.21 ~ 5.28	5.29 ~ 6.4	6.5 ~ 6.7
주제 선정	주제 창출	■				
	주제 선정	■				
	아이디어 창출		■			
	아이디어 선정		■			
준비	유사 프로그램 분석		■			
	기초자료 수집		■			
	제안서 작성		■			
설계	클래스 구조 설계		■	■		
	기능 설계		■	■		
	디자인 설계		■	■		
개발	보조 클래스 개발		■	■		
	주 클래스 개발		■	■		
	클래스 간 연결			■	■	
	통합 개발			■	■	
개선	오류 수정			■	■	
	데이터 추가				■	
테스트	시나리오 테스트				■	■
	디버깅				■	■
최종 준비	최종 문서 작업					■
	프로젝트 공개					■
보수	데이터베이스 업데이트					■

\* 개발 일정표 \*

### 다. 개인 일정

다음 두 페이지는 **팀원들의 대략적인 개인 일정**을 나타낸다. 제안서 작성 시점부터 6월 7일까지의 일정을 크게 세 부분, 5월 넷째 주, 5월 다섯째 주, 6월 첫째 주로 나누었다. 각자 주요 담당 클래스들을 설정하였고, 각자 그 부분을 중점적으로 구현할 계획이다. 다른 팀원들의 진행 정도에 따라 자신이 담당하는 클래스를 지속해서 수정 및 개선한다.

프로그램 구현 과정에서 서로 긴밀하게 협력해야 해결할 수 있는 작업이 많을 것으로 예상하였다. 따라서 공통 사항으로 처리한 일정들이 있으며, 이러한 공통 일정은 개인 일정에 표기하지 않았다. 나타내지 않은 모든 일정은 협력하여 처리한다.

# 윤건호

## 상세

### 01 5월 4주차

클래스 구현

#### 클래스 구현

1. 보조 클래스 Config, Console, Random을 완성한다.
2. 타 클래스의 구현을 보조한다.

### 02 5월 5주차

클래스 구현, 데이터 추가

#### 클래스 구현, 데이터 추가

1. 주 클래스 ProgramManager, MainMenu를 구현한다.
2. 팀원과 논의 후, 필요한 클래스를 추가로 구현한다.
3. 프로그램에서 사용하는 프리셋 데이터를 추가한다.

### 03 6월 1주차

시나리오 테스트

#### 시나리오 테스트

1. 클래스의 다양한 시나리오를 테스트한다.
2. 오류와 수정하고 프로그램을 개선한다.

# 김상원

## 상세

### 01 5월 4주차

클래스 구현

#### 클래스 구현

1. MiniGame 클래스의 전체적인 틀을 구현한다.
2. 보조 클래스를 결합하고, 레이아웃을 완성한다.

### 02 5월 5주차

클래스 구현, 데이터 추가

#### 클래스 구현, 데이터 추가

1. MiniGame 클래스의 변수와 함수를 구체화한다.
2. Main(), Corrector(), DrawMan() 등이 포함된다.
3. 게임에서 사용하는 프리셋 데이터를 추가한다.

### 03 6월 1주차

시나리오 테스트

#### 시나리오 테스트

1. MiniGame 클래스의 다양한 시나리오를 테스트한다.
2. 오류와 수정하고 클래스를 개선한다.

## 안치원

상세

### 01 5월 4주차

클래스 구현

#### 클래스 구현

1. 보조 클래스 Timer, Text을 구현한다.
2. 해당 보조 클래스를 응용하는 주 클래스를 수정한다.

### 02 5월 5주차

클래스 구현, 데이터 추가

#### 클래스 구현, 데이터 추가

1. 주 클래스 TypingManager를 구현한다.
2. 중요한 클래스이므로 팀원과 긴밀하게 협력한다.
3. 프로그램에서 사용하는 프리셋 데이터를 추가한다.

### 03 6월 1주차

시나리오 테스트

#### 시나리오 테스트

1. 작성한 클래스의 다양한 시나리오를 테스트한다.
2. 오류와 수정하고 클래스를 개선한다.

## 우은혁

상세

### 01 5월 4주차

클래스 구현

#### 클래스 구현

1. Statistics 클래스를 구현한다.
2. 보조 클래스를 결합하고, 레이아웃을 완성한다.

### 02 5월 5주차

클래스 구현, 데이터 추가

#### 클래스 구현, 데이터 추가

1. WordTyping, ShortTyping, LongTyping 클래스를 구현한다.
2. 중요한 클래스이므로 팀원과 긴밀하게 협력한다.
3. 프로그램에서 사용하는 프리셋 데이터를 추가한다.

### 03 6월 1주차

시나리오 테스트

#### 시나리오 테스트

1. 클래스의 다양한 시나리오를 테스트한다.
2. 오류와 수정하고 클래스를 개선한다.

## 7. 기대 효과

### 가. 현주소

컴퓨터와 스마트폰이 널리 보급된 오늘날, 한글 타자가 느린 것을 불평하는 사람은 거의 없다. 하지만 영어 타자와 코딩은 그렇지 않다. 자주 사용할 일이 없기에, 한글 타자 속도와 수 배까지 차이가 나고 있는 한다. 따라서 우리에게 친숙하지 않은 영어와 코드로 이루어진 C++ 언어를 작성하기는 쉽지 않다.

```

// Change Color Name to Integer
// Console: ColorNameToNumber(const string& colorName) const
int Console::ColorNameToNumber(const string& colorName) const
{
    if (colorName == "default")
        return -1;

    if (colorName == "random")
    {
        int colorSet[5] = { 12,10,3,13,14 };
        return colorSet[rand() % 5];
    }

    else if (colorName == "red") return 12; // RED
    else if (colorName == "green") return 10; // GREEN
    else if (colorName == "blue") return 3; // BLUE
    else if (colorName == "purple") return 13; // PURPLE
    else if (colorName == "yellow") return 14; // YELLOW
    else if (colorName == "white") return 15; // WHITE
    else if (colorName == "black") return 1;
}

```

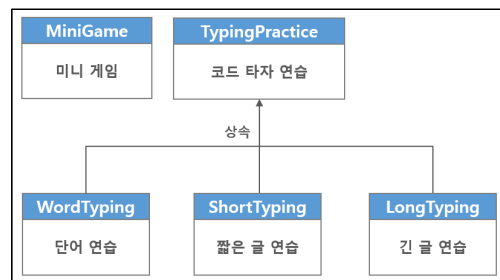
\* 잦은 오타 \*

Result	Score	Code
Wrong Answer	0 /100	C
Wrong Answer	0 /100	C
Output Limit	0 /100	C
Compile Error	0 /100	C
Run-Time Error	0 /100	C
Compile Error	0 /100	C
Run-Time Error	0 /100	C
Run-Time Error	0 /100	C

\* 부족한 시험 시간 \*

### 나. CodeSlayer 개발 후 기대 효과

CodeSlayer는 C++ 언어와 타자 연습, 그리고 게임을 결합한 프로그램이다. 크게 두 가지 주요 기능을 제공한다. 첫째, 코드 타자 연습 기능이다. 이를 활용하여 사용자의 타자 실력(정확도, 속도)을 증진한다. 둘째, 미니 게임 기능이다. 이를 활용하여 사용자가 코딩에 흥미를 느끼게 한다. CodeSlayer가 제공하는 두 가지 기능을 적절하게 활용한다면, 사용자는 C++ 언어에 더 익숙해질 것이다. 이는 결과적으로 사용자의 코딩 생산성 향상으로 이어진다.



\* CodeSlayer가 제공하는 주요 기능 활용 \*

**C++ 언어 습득**

한글	영문	한글	영문	한글	영문
auto	자동	double	이중	int	정수
break	중단	else	그렇지 않으면	long	긴
case	경우	enum	열거형	register	등록
char	문자	extern	외부	return	반환
const	상수	float	부동소수점	short	짧은
continue	계속	for	반복	signed	부호
default	기본	goto	이동	sizeof	크기
do	반복	if	만약	volatile	변동
				static	정적
				while	동안

\* C++ 언어 체화 \*

**코딩 생산성 향상**

Result	Score	Code
Wrong Answer	0 /100	C
Wrong Answer	0 /100	C
Output Limit	0 /100	C
Compile Error	0 /100	C
Run-Time Error	0 /100	C
Compile Error	0 /100	C
Run-Time Error	0 /100	C
Run-Time Error	0 /100	C

Submit	My Score
1	100 /100
1	100 /100
1	100 /100
1	100 /100
1	100 /100
1	100 /100
1	100 /100
1	100 /100

\* 코딩 생산성 향상 \*