에이블러를 위한 PCM

Programmer Competency Matrix

안녕하세요 에이블러님들!!

이 페이지에서는 여러분께 PCM을 소개 해드릴거에요. PCM은 ‘Programmer Competency Matrix’ 로 직역하면 ‘프로그래머 능력 매트릭스’ 인데, 간단히 ‘개발자 역량 진단 표’ 라고 할 수 있겠습니다.

자, 먼저 여러분에게 역량진단표를 보여드리는 이유에 대해서 말씀 드리고 싶어요! PCM을 알려드리는 이유는 에이블러님들이 각자 본인의 위치를 측정해보고 어떤 공부를 더 해야 할 지, 어디로 어떻게 발전을 할 지 판단하는 기준이 되었으면 해요. PCM을 (그나마 객관적인) 기준으로 보시면 될 것 같습니다.

먼저, Aivler 자체 역량 진단을 소개 합니다!

Aivler 자체 역량 진단은 아래에서 소개 해드릴 ‘경력’ 개발자 진단표에서 신입 개발자를 위한 항목으로 추리고 변경하였답니다~ 총 20항목으로 커스터 마이징 했어요. 한번 보실까요?

[에이블스쿨] Aivler 자체 역량 평가

- Level 0 : 1점 / Level 1 : 2점 / Level 2 : 3점 / Level 3 : 4점
- 점수 합계 : ~30점 초급 // 31점~50점 중급 // 51점~65점 고급 // 65점 이상 특급

전산학

	Level 0	Level 1	Level 2	Level 3
data structures (자료 구조)	배열(array)과 linked list의 차이를 모른다	실제 프로그래밍 환경에서 배열과 linked list, dictionary의 차이를 설명할 수 있다	배열과 linked list의 속도/메모리타협관계를 설명할 수 있다. 해시의 충돌처리를 할 수 있고 우선순위 큐를 만들 수 있다	B트리, 이진트리, 피보나치힙, AVL트리, RED,BLACK 트리, Splay트리, Skip 리스트 등 고급 자료구조에 대해 이해하고 있다

	Level 0	Level 1	Level 2	Level 3
algorithms (알고리즘)	배열에 들어있는 숫자들의 평균을 낼 줄 모른다	sorting, searching, traversing 알고리즘을 이해한다	트리, 그래프, 단순한 greedy, divede, conquer 알고리즘을을 이해한다	프로그래밍 솔루션을 이해한다 그래프 알고리즘과 수치연산 알고리즘을 이해하고 NP 문제를 식별할 수 있다
systems programming (시스템 프로그래밍)	컴파일러, 링커, 인터프리터를 구분하지 못한다	컴파일러, 링커, 인터프리터를 이해한다. 하드웨어 레빌의 어셈블리 언어 동작을 이해한다. 가상 메모리와 페이징에 대한 이해가 있다	커널모드/유저모드의 차이를 알고 멀티스레딩, 동기화를 이해하고 그것들이 어떻게 구현되었는지 안다. 어셈블리 코드를 읽을 수 있다. 네트워크가 어떻게 동작하는지 안다., 프로토콜을 알고 소켓 수준의 프로그램을 읽을 수 있다	전체 프로그램 스택을 이해한다. 하드웨어 (CPU + Memory + Cache + Interrupts + microcode), 바이너리코드, 어셈블리, 정적/동적 링킹, 컴파일, 인터프리테이션, JIT컴파일, 가비지 컬렉션, 힙, 스택, 메모리 어드레싱을 구분해서 이해한다

운영체제

	Level 0	Level 1	Level 2	Level 3
리눅스	윈도우 밖에 사용해 보지 않았다	한두가지의 리눅스를 사용해 보았으며, 쉘 명령어를 좀 알고 있다	리눅스의 특정 벤더사의 서버를 설치, 운영해본 경험이 있다	리눅스로 서비스를 운영해 보았으며 각종 트러블 슈팅에 대한 경험이 있다
윈도우	특별한 경험이 없다	백업과 복원을 할 수 있고, 윈도우를 새로 설치하는데 문제가 없다	고급 테크닉을 숙지하고 있으며 최적화나 튜닝에 대해서도 관심이 많다	오류가 발생하면 덤프를 뜨고 로그를 읽어 어떤 문제가 발생했는지 알고 해결 할 수 있다

퍼블리싱

	Level 0	Level 1	Level 2	Level 3
웹페이지 코딩	간단한 HTML 문법과 CSS 문법을 알고 사용할 수 있다	div로 레이아웃을 잡고 코딩을 할 수 있다 (Box model을 이해한다)	웹 표준을 어느 정도 이해하고 있다 (DTD를 알고 있다)	웹 접근성과 시맨틱한 마크업에 대해 숙지하고 있으며 관심이 깊다
크로스 브라우징	인터넷 익스플로러밖에 사용해 보지 않았다	특정 모바일 디바이스에 최적화된 페이지를 만들어 본 경험이 있다	각 브라우저의 렌더링 이슈를 알고 있으며 해결해 본 경험이 있다	브라우저핵의 존재를 알고 있으며, 우아한 퇴보 기법의 일부를 숙지하여 코딩할 수 있다
JavaScript	아주 기초적인 JavaScript 문법을 알고 있다	Context의 역할과 this의 용법을 안다	closure를 이해하고 익명함수를 다룰 수 있다	각 브라우저별 JavaScript 엔진의 차이를 이해하고 크로스 브라우징이 가능한 코드를 작성할 수 있다
jQuery	간단한 셀렉터와 실행 문법을 알고 있다	다양한 플러그인을 사용해본 경험이 있다	제법 복잡한 셀렉터를 다룰 수 있고, 메소드 체이닝을 알고 있으며, 거의 모든 문법을 자유 자재로 다룰 수 있다	플러그인 제작이 가능하고 jQuery가 어떻게 구현되었는지 상당 수준을 이해한다

설계

	Level 0	Level 1	Level 2	Level 3

	Level 0	Level 1	Level 2	Level 3
객체 지향	인터페이스를 왜 써야 하는 지 모른다	캡슐화와 추상화를 이해하고 어떤 상황에서 써야하는지 알고 있다	디자인 패턴과 안티 디자인 패턴을 상당수 알고 있다	이를 통해 프레임워크를 구축해본 경험이 있다
DI	DI나 IoC가 무엇인지 모른다	왜 빈을 주입해야 하고 왜 이것이 낮은 결합도를 만드는지 안다	스프링에서 DI가 어떻게 구현되어 있는지 알고 있다	DI와 IoC에 대해 하루 종일 설교할 수 있다
모델링	논리적 모델과 물리적 모델이 무엇을 말하는지 모른다	논리적 모델과 물리적 모델을 구분하여 모델링할 수 있다	L자형 테이블이 왜 위험한지 알고 있다 (필드가 많이 늘어나는 것이 어떤 영향을 미치는지 이해한다)	정규화를 알고 이를 적용하여 모델링을 할 수 있다

Python

	Level 0	Level 1	Level 2	Level 3
문법	간단한 문법을 이해하고 프로그램을 작성할 수 있다	클래스 기반 프로그래밍을 이해하고 잘 사용할 수 있다	표준정규식을 어느정도 사용할 수 있다	앞선 개념들을 자유롭게 사용할 수 있다
빌드	빌드에 관심이 없다	커맨드 라인으로 빌드를 만들 수있다	빌드 스크립트를 직접 작성할 수 있다	문서, 설치스크립트, 릴리스 노트를 포함한 빌드 스크립트를 작성한다
테스트	테스트를 하지 않는다	문자열을 찍어가며 개발한다	유닛 테스트를 진행하며 개발한다	TDD를 이해한다

데이터베이스

	Level 0	Level 1	Level 2	Level 3
쿼리 작성	기본적인 CRUD를 작성할 수 있다	inline view를 사용하여 복잡한 질의문을 작성할 수 있다	각종 join을 다룰 수 있으며, 집계 쿼리, group by 등을 사용할 수 있다	100만 건 이상의 데이터가 누적되어 있는 상황에서 쿼리를 작성해본 경험이 있다
대용량 DB	실행 계획이 뭔지 모른다	Clustered Index와 Non-clustered Index의 차이를 안다	실행 계획을 변경하기 위해 hint를 사용할 수 있다	파티셔닝과 병렬 처리를 제어할 수 있고, 각종 튜닝 방법에 대해 알고 있다
DB 개념	제대로 데이터베이스에 대한 개념을 학습하지 못했다	ACID, 정규화, 트랜잭션을 이해한다	스키마를 정규화해서 정의할 수 있고, 뷰, 스토어드 프로시저, 트리거, 사용자 정의 타입을 다룰 수 있다	쿼리 트랜스포머를 어느 정도 이해하고 있으며, 내부적으로 인덱스가 어떻게 저장되어 있고 작동하는지 이해하고 있다

기타

	Level 0	Level 1	Level 2	Level 3
책	간단한 문법을 다룬 책을 위주로 몇 권 읽어 봤다	코드 컴플리트를 비롯한 스티브 맥코넬의 저서 또는 켄트벡이나 마틴 파울러의 저서를 읽어 봤다 (방법론에 대한 책)	이펙티브 C++, 이펙티브 Java, 그 외 각종 실용적이면서 중급 사용자에게 적절히 도움이 되는 책을 읽어 봤다	아트 오브 컴퓨터 프로그래밍, 자바스크립트 닌자의 비밀, 대용량데이터베이스 솔루션 등의 고급 서적을 읽어 봤다

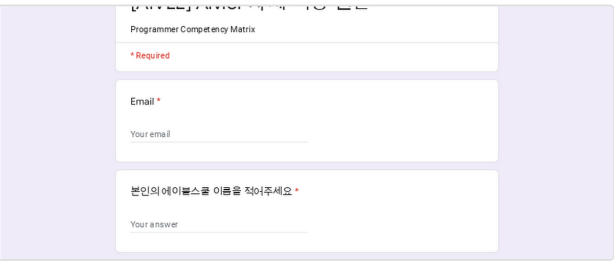
	Level 0	Level 1	Level 2	Level 3
외국어	한국어 아닌 자료로는 기술 습득을 못한다	영어로 된 기술 교재를 읽고 기술을 파악한다	영어권 외국인으로부터 개발업무지시를 받고 그에 따라 업무 수행 후 결과를 보고 할 수 있다	영어로 외국인에게서 S/W개발을 발주 받아서, 다른 외국인에게 개발업무를 아웃소싱 시키고 국내 팀과 연계하여 국제적인 프로젝트를 진행할 수 있다

위 에이블러를 위한 역량 진단은 아래 구글 폼에서 진행 하실 수 있습니다~!

[AIVLE] Aivler 자체 역량 진단

Programmer Competency Matrix

https://forms.gle/tRkP3oXs4oLSbA2W6



에이블러를 위한 역량 진단표는 아래의 경력 개발자 진단표에서 신입 개발자를 위한 항목으로 추리고 변경하였습니다!

다음은 경력 개발자를 위한 진단표 입니다

[에이블스쿨] 경력 개발자 역량 진단

- 총 40개 항목 Level 0 : 1점 / Level 1 : 2점 / Level 2 : 3점 / Level 3 : 4점 입니다
- 점수 합계 ~60점 초급 // 70점~90점 중급 // 100점~120점 고급 // 120이상 특급

Computer Science

	Level 0	Level 1	Level 2	Level 3
data structures (자료 구조)	배열(array)과 linked list의 차이를 모른다	실제 프로그래밍 환경에서 배열과 linked list, dictionary의 차이를 설명할 수 있다	배열과 linked list의 속도/메모리타협관계를 설명할 수 있다. 해시의 충돌처리를 할 수 있고 우선순위 큐를 만들 수 있다	B트리, 이진트리, 피보나치힙, AVL트리, RED,BLACK 트리, Splay트리, Skip 리스트 등 고급 자료구조에 대해 이해하고 있다
algorithms (알고리즘)	배열에 들어있는 숫자들의 평균을 낼 줄 모른다	sorting, searching, traversing 알고리즘을 이해한다	트리, 그래프, 단순한 greedy, divede, conquer 알고리즘을 을 이해한다	프로그래밍 솔루션을 이해한다 그래프 알고리즘과 수치연산 알고리즘을 이해하고 NP 문제를 식별할 수 있다
systems programming (시스템 프로그래밍)	컴파일러, 링커, 인터프리터를 구분하지 못한다	컴파일러, 링커, 인터프리터를 이해한다. 하드웨어 레빌의 어셈블리 언어 동작을 이해한다. 가상 메모리와 페이징에 대한 이해가 있다	커널모드/유저모드의 차이를 알고 멀티스레딩, 동기화를 이해하고 그것들이 어떻게 구현되었는지 안다. 어셈블리 코드를 읽을 수 있다. 네트워크가 어떻게 동작하는지 안다., 프로토콜을 알고 소켓 수준의 프로그램을 읽을 수 있다	전체 프로그램 스택을 이해한다. 하드웨어 (CPU + Memory + Cache + Interrupts + microcode), 바이너리코드, 어셈블리, 정적/동적 링킹, 컴파일, 인터프리테이션, JIT컴파일, 가비지 컬렉션, 힙, 스택, 메모리 어드레싱을 구분해서 이해한다

Software Engineering

	Level 0	Level 1	Level 2	Level 3
source code version control (소스코드 버전관리)	날짜 단위의 폴더를 백업	CVS, SVN, VSS 등의 사용을 시작했다	CVS/SVN에 대한 능숙한 사용, 브랜치/머지를 할 수있고, 패치를 만들 수 있고, 저장소 속성에 맞게 만들 수 있다	분산 VCS 시스템을 이해한다. Bzr/MBzr/Mercurial/Darcs/Git 등을 써봤다
build automation (빌드 자동화)	IDE에서만 빌드할 수 있다	커맨드 라인으로 빌드를 만들 수 있다	빌드 스크립트를 직접 작성한다	문서, 설치인스톨러, 릴리스 노트를 포함한 빌드 스크립트를 만든다
automated testing (테스트 자동화)	테스트는 테스터의 일이라고 생각한다	유닛 테스트를 짜고 새로 짜는 코드에 유닛 테스트를 작성하고 있다	TDD 방식으로 코드를 작성한다	기능적, 로드/성능적, GUI 측면의 테스트 자동화를 이해하고 실현한다

Programming

	Level 0	Level 1	Level 2	Level 3
problem decomposition (프로그램 분해)	라인단위의 복사/붙여넣기로 코드를 작성한다	문제를 여러 함수로 나누어 코딩한다	재사용 가능한 객체로 코드를 작성한다	적절한 알고리즘으로 제네릭/OOP방법을 써서 변경이 있을 법한 부분은 적절히 캡슐화하면서 코딩한다
systems decomposition (시스템 분해)	파일1개 클래스1개 이상의 범위를 생각하지 못한다	같은 플랫폼, 같은 기술 범위 내에서는 문제를 쪼개서 해결책을 설계해 낼 수 있다	다양한 기술과 시스템에 걸쳐 있는 문제에 대한 솔루션을 만들어 낸다	복잡한 제품들을 가시화 하여 설계하고 외부 시스템과 연동을 이끌어 낸다. 모니터링, 리포팅, 장애복구 등의 운영작업도 설계할 수 있다
communication (의사소통)	아이디어와 생각을 잘 표현하지 못하고, 맞춤법과 문법이 엉망이다	동료에게 무슨 의도로 말하는지 이야기할 수 있다. 맞춤법과 문법은 좋다	효과적으로 의사소통 할 수 있다	모호한 상황에서 생각/설계/아이디어/스펙을 이해하고 소통할 수 있으며, 상황에 맞게 소통할 수 있다
code organization within a file (파일 내 코드 구성)	코드 파일 내에서 구조화가 안되어 있다	메소드들이 논리적으로든 접근성으로든 어떻게든 구조화되어 있다	영역별로 그루핑 되어 있고, 주석도 잘 되어 있고, 서로 다른 파일간의 참조도 잘 설명되어 있다	파일은 라이선스 헤더도 있고, 요약 설명도 있고, 주석도 잘 되어 있고, 공백 사용은 일관성이 있고, 파일 자체가 보기 좋게 정렬되어 있다
code organization across files (파일 간의 코드 구성)	파일 간 구성에 대한 어떤 구조도 없다	파일들이 폴더로 나뉘어 있다	파일별로 고유한 목적이 있다. 예를들어 클래스 하나 정의, 기능하나 구현등	코드 구성이 설계와 잘 매치되어 코드 파일명만 보더라도 설계에 대한 이해를 할 수 있도록 만든다
source tree organization (소스트리 구성)	모든게 폴더 하나에 다 있다	논리적인 폴더로 나뉘어 있다	circular의존성이 없고 바이너리, 라이브러리, 문서, 빌드, 서드파티 코드등이 폴더로 구분되어 있다	물리적 코드 구성이 논리적인 계층을 잘 반영한다. 디렉토리명으로 시스템 설계에 대한 이해가 가능하도록 만든다
code readability (코드 가독성)	코드 가독성이 낮다	파일, 변수, 클래스, 메소드에 이름을 잘 부여한다	통상적이지 않은 코드나 버그수정, 전제조건 등에 대해서 주석을 잘 달아둔다	전제 조건은 asssert로 검증한다. nesting 단계가 깊지 않고, 자연스럽게 코드가 흐른다

	Level 0	Level 1	Level 2	Level 3
defensive coding (방어적 코딩)	방어적 코딩이 뭔지 모른다	인수를 다 체크하고, 크리티컬한 전제 조건에 대해서 assert를 사용한다	리턴 값도 체크하고, 예외를 항상 체크한다	방어적 코딩을 하기 위한 자신만의 라이브러리가 있다. 실패 케이스를 시험하는 유닛 테스트 코드를 작성한다
error handling (에러 핸들링)	정상적으로 작동될 경우만을 염두에 두고 코드를 작성한다	예외/에러가 생성되는 주변에 기본적인 에러 핸들링 코드가 있다	예외/에러로 가도 프로그램이 안정적인 상태에 있도록 유지한다. 리소스, 커넥션, 메모리 등이 깨끗하게 해제됨을 보장한다	가능한 예외 상황을 미리 감지해 내도록 코딩한다. 코드 전체에 대해 일정한 예외 핸들링 정책을 사용한다. 전체 시스템에 대한 예외 핸들링 가이드라인을 만든다
IDE	대부분 텍스트 에디팅에 IDE를 사용한다	인터페이스 이면에 숨어있는 IDE 기능을 메뉴에서 효과적으로 불러내서 쓴다	거의 모든 IDE 사용을 단축키로 한다	IDE에 매크로를 정의해서 사용한다
API	자주 문서를 봐야 한다	기억 속에 자주 쓰는 API는 들어있다	API에 대해서 폭넓고 깊이 있는 이해를 하고 있다	자주 호출하는 것에 대해서 단순화시키기 위해 API 위에 라이브러리를 추가로 개발하고 결과적으로 API의 부족한 점은 직접 채운다
frameworks (프레임워크)	코어 플랫폼 밖의 어떤 프레임워크도 쓰지 않는다	유명 프레임워크를 들어는 봤으나 써보지는 못했다	프레임워크를 여러 개 능숙하게 쓰고 해당 프레임워크를 효과적으로 잘 쓰는 전형적인 방법을 알고 있다	프레임워크를 직접 개발한다
requirments (요구사항)	주어진 요구사항을 코드 스펙으로 바꾼다	스펙에서 비어있는 케이스에 대한 질문을 할 수 있다	전체 그림을 이해하고 스펙으로 정의할 영역 전체를 도출해 낸다	경험에 기반해서 주어진 요구사항에 대해 더 좋은 대안과 플로우를 제시할 수 있다
scripting (스크립팅)	스크립팅 툴을 모른다	배치파일, 쉘 스크립팅을 한다	Perl/Python/Ruby/VBScript/Powershell류의 스크립팅을 한다	재사용 가능한 코드를 짜고 공개한다
database (데이터베이스)	제대로 데이터베이스에 대한 개념을 학습하지 못했다	데이터베이스의 기본 개념과 ACID, 정규화, 트랜잭션을 이해한다	실행될 쿼리를 염두해서 스키마를 정규화 정의할 수 있고, 뷰, 스토어드 프로시저, 트리거, 사용자 정의 타입을 다룰 수 있다	기본 DB관리, 성능 최적화, 인덱스 최적화, 고급 SELECT 쿼리를 작성할 수 있고, 커서를 이해하며, 데이터가 내부적으로 어떻게 저장되는지 알며, 인덱스가 어떻게 저장되는 지 알고, 데이터베이스 미러링, 복제를 이해하고, 2-페이스 커밋을 이해한다

Experience

	Level 0	Level 1	Level 2	Level 3
languages with professional experience (전문 경험을 가진 언어)	절차형 언어, 객체지향 언어	절차형 언어, 객체지향 언어, 선언적(SQL)언어를 알고, 동적/정적 타입과 약한/강한 타이핑을 이해하며 정적으로 유도된 타입을 이해하면 bonus	함수언어를 알고, lazy evaluation, currying, continuations를 이해하면 보너스 추가	병렬언어와 논리적 언어를 이해한다

	Level 0	Level 1	Level 2	Level 3
platforms with professional experience (전문 경험을 가진 플랫폼 수)	1	2-3	4-5	6+
years of professional experience (전문 경험 기간)	1	2-5	6-9	10+
domain knowledge (도메인 지식)	도메인 지식이 없다	같은 도메인의 프로덕트에 관련하여 적어도 1개에서 일해봤다	같은 도메인의 여러 프로덕트에 관련하여 일해봤다	도메인 전문가. 해당 도메인의 여러 제품을 설계하고 구현해봤다. 표준 도메인 용어 및 프로토콜에 대해서 잘 구사할 수 있다

Knowledge

	Level 0	Level 1	Level 2	Level 3
tool knowledge (툴 관련 지식)	자주 쓰는 IDE만 알고 있다	유명하거나 표준적인 다른 도구들을 쓸 줄 안다	에디터, 디버거, IDE에 대해 잘 알고 있고 오픈 소스 대체물도 잘 알고 있다 Scott Hanselman의 툴들은 대부분 알고 있다. ORM툴을 사용한다	도구를 직접 만들 수 있다
languages exposed to (노출된 언어 - Experience의 languages with professional experience 와 동일)	절차형 언어, 객체지향 언어	절차형 언어, 객체지향 언어, 선언적(SQL)언어를 알고, 동적/정적 타입과 약한/강한 타이핑을 이해하며 정적으로 유도된 타입을 이해하면 bonus	함수언어를 알고, lazy evaluation, currying, continuations를 이해하면 보너스 추가	병렬언어와 논리적 언어를 이해한다
codebase knowledge (코드베이스 지식)	코드베이스를 본 적 없다	코드 레이아웃에 대한 이해가 있고, 시스템 빌드에 대해서 이해한다	코드 베이스를 잘 알고 버그픽스를 코딩했고, 몇몇 기능도 추가했다	코드 베이스에여러개의 주요 기능을 넣었다. 대부분의 기능과 버그 수정에 소요되는 변경내역을 가시화 시킬 수 있다
knowledge of upcoming technologies (최신 기술 이해)	최신 기술을 모른다	해당 분야의 최신 기술을 들어는 봤다	알파 프리뷰, CTP, 베타를 다운로드해 봤다. 온라인 매뉴얼 등을 읽어 봤다	프리뷰를 시험해보고, 뭔가 만들어 봤다
platform internals (플랫폼 내부)	플랫폼 내부에 대해 전혀 모른다	플랫폼 내부가 어떻게 동작 하는지 기본 이해를 가지고 있다.	플랫폼 내부가 어떻게 동작하는지 잘 알고 있고, 가시화 시켜서 플랫폼이 어떻게 코드를 실행시키는지 설명할 수 있다	플랫폼 내부에 대한 정보를 제공하기 위해서 디컴파일, 디버깅, 디스어셈블등을 하거나 툴을 만든다
books (책)	간단한 문법을 다룬 책을 위주로 몇 권 읽어 봤다	코드 컴플리트를 비롯한 스티브 맥코넬의 저서 또는 켄트벡이나 마틴 파울러의 저서를 읽어 봤다 (방법론에 대한 책)	이펙티브 C++, 이펙티브 Java, 그 외 각종 실용적이면서 중급 사용자에게 적절히 도움이 되는 책을 읽어 봤다	아트 오브 컴퓨터 프로그래밍, 자바스크립트 닌자의 비밀, 대용량데이터베이스 솔루션 등의 고급 서적을 읽어 봤다
blogs (블로그)	들어봤지만, 별로 친하지 않다	주기적으로 기술, 프로그래밍, 소프트웨어 엔지니어링 블로그를 읽고 팟캐스트를 듣는다	펌글 블로그를 운영하고 유용한 팁과 기사를 모아서 관리하고 있다	프로그램에 대한 개인적인 통찰이나 생각을 적는 블로그를 개설해서 다른 사람과 공유한다

외국어

	Level 0	Level 1	Level 2	Level 3
Foreign Language (외국어)	한국어 아닌 자료로는 기술 습득을 못한다	영어로 된 기술 교재를 읽고 기술을 파악한다	유럽/미국인으로부터 개발업무지시를 받고 그에 따라 업무 수행 후 결과를 보고 할 수 있다	영어로 외국인에게서 S/W개발을 발주 받아서, 다른 외국인에게 개발업무를 아웃소싱 시키고 국내 팀과 연계하여 국제적인 프로젝트를 진행할 수 있다

문서화

	Level 0	Level 1	Level 2	Level 3
Format (문서 포맷)	문서 작성이 서툴고 무엇을 작성해야 하는지 자발적으로 판단하지 못한다	프로젝트에 필요한 문서 포맷이 어떤 것인지 알고있고 적어도 1개 이상의 프로젝트에서 단계별 문서를 작성해본적이 있다	여러 프로젝트에서 사용된 다양한 문서 포맷을 여러 세트 작성해본적이 있고, 여러 세트를 확보하고 있다	문서 세트를 종류 및 포맷을 직접 정의할 수 있고, 베스트 프랙티스를 알고 있다
Office Tools (오피스 툴 사용)	오피스 문서도구 사용법을 완전히 숙지하지 않은 상태이다.	오피스를 이용해서 각종 문서를 작성하지만, 문서 포맷이 주어지지 않으면 작성에 애를 먹는다.	오피스 사용법을 완전히 숙지하고 있으며, 단축키도 사용하고 효율적인 파일 처리 를 할 수 있다.	상당한 작업을 단축키로 수행하며, 매크로도 직접 정의해서 사용하고, 오피스 툴 외의 공개판 도구들도 적극적으로 사용해서 효과적인 문서를 생산한다.

[에이블스쿨] 경력개발자 역량진단표는 아래 링크의 pcm을 번역하여 제공해 드립니다

- https://sijinjoseph.netlify.app/programmer-competency-matrix/

자, 어떠셨나요? 저는 번역 작업 하면서 자아를 성찰하는 시간을 가졌답니다 :) 어느 정도 아는 부분이라고 생각했던 파트에서도 의외의 모르는 부분들이 나오더라고요..... 이게 뭐지? 하면서 아 앞으로 발전하려면 이런걸 공부 하면 되겠구나! 하는 가이드가 생겨서 제겐 의미가 있었답니다

Aivler 여러분에게 조금이라도 도움이 되었으면 좋겠습니다~ 파이팅!!

- [에이블스쿨] 경력개발자 역량진단표는 아래 링크의 pcm을 번역하여 제공해 드립니다
  - https://sijinjoseph.netlify.app/programmer-competency-matrix/
- [에이블스쿨] Aivler 자체 역량 진단표는 경력개발자 역량 진단표에서 신입 개발자를 위한 항목으로 추려서 제공해 드립니다