**C/C++ 종합평가 답안 작성**

작성자: 이재림

**문제 1. 대략적으로 1달 정도의 시간이 흘렀습니다. 1달의 시간 동안 느낀점과 간략한 소감을 작성해주세요.부가적으로 앞으로의 포부 또한 같이 작성해주시면 감사하겠습니다.**

솔직히 해당 과정에 대해 잘못된 정보를 전달받고 참여하게 되어, 처음에는 빨리 다른 길을 찾아봐야 하나 라는 생각이 들었습니다. 과정 특성상 협업도 많이 하게 될 것인데, 이 분야에 대해 아무 지식이 없는 제가 민폐를 끼치는 것이 아닌가 라는 걱정도 들었습니다.(성향상 다른 사람에게 피해를 주는 것을 정말 싫어합니다.) 그러나 수업을 듣고 프로젝트를 하면서 낯설었던 수업 내용들이 익숙해져 가며, 이 과정을 끝까지 수료를 해야 겠다는 다짐을 하게 되었습니다. 구체적으로 이 과정이 끝나면 하고 싶은 직무에 대해서는 더 찾아보고 고민해봐야 할 것 같지만, 우선 현재 상황에 충실하면 자연스럽게 정해지지 않을까 라는 생각이 들었습니다. 따라서 앞으로 남은 기간 동안 더 열심히 수업과 협업 프로젝트에 참여하고, 배운 내용을 혼자 정리 및 학습하는 시간을 가지는 등 제 역량을 향상시키도록 노력하겠습니다.

**문제 2. 언어를 가리지 않고 Domain Driven Design을 수행하여 얻을 수 있는 이점들에 대해 기술하세요.**

- 협업 프로젝트시 팀원 간의 의사소통이 원활해 집니다.  
- 새로운 개발자가 유입될 경우, 유입된 개발자의 업무 습득력이 빠를 것입니다.  
- 유지 및 보수가 향상됩니다.

**문제 3. C/C++ 에서 Test 목적으로 GoogleTest를 사용하였습니다. TDD(Test Driven Development)를 통해 얻을 수 있는 이점들에 대해 기술하세요.**

​- 전체적인 작업 흐름을 확인 할 수 있습니다.  
- Backlog 작성 및 관리가 용이합니다.  
- 코드의 신뢰성을 높이고 버그를 최소화 하는데 도움을 줍니다.

**문제 4. DDD(Domain Driven Design)과 TDD(Test Driven Development)의 관계에 대해 기술하세요.**

DDD는 비즈니스 도메인에 집중하여 모델을 개발하는 데 중점을 두고, TDD는 테스트를 통해 코드의 동작을 검증하고 유지보수성을 높이는 데 중점을 둡니다. 즉 TDD는 코드의 동작을 검증하기 때문에 DDD에서 개발된 도메인 모델을 안정적으로 유지할 수 있으며 기능 확장 및 변경시 용이합니다. 결론적으로 DDD와 TDD는 상호보완적인 관계라 할 수 있습니다.

​

**문제 5. 팀원들과 협업을 진행 할 때 중요하다 생각하는 요소들을 작성하고 각 요소들에 대해 조금 더 상세하게 기술하세요.**

팀원과의 협업 시 중요한 요소는 ‘의사소통’이라 생각합니다. 협업을 진행할 때 누가 어떤 작업을 수행하고 있는 지 알아야 하고, 소통을 함으로써 각 작업에 대한 빠른 판단 및 실행을 할 수 있습니다.

다음으로 중요하다 생각하는 요소는 ‘역할 분담’ 입니다. 효율적인 협업을 위해 구체적인 역할 분담이 필요하고, 맡은 역할에 따라 책임감을 갖고 이행하는 것이 필요하다 생각합니다.

마지막으로는 ‘피드백’ 입니다. 지속적인 성장과 개선을 위해서는 팀원 간의 피드백이 필요하며, 이러한 피드백을 적극적으로 수용하는 문화가 중요하다 생각합니다.

​

**문제 6. 함수 포인터 테이블을 사용하는 이유에 대해 IoC 관점에서 기술하세요.**

함수 포인터 테이블을 사용하면 객체가 직접적으로 특정 함수를 호출하는 대신, 해당 함수에 대한 포인터를 테이블을 통해 간접적으로 호출합니다. 즉 객체간의 의존성을 분리하여 결합을 낮추고 시스템의 유연성을 향상시킵니다. 또한, 함수 포인터 테이블을 사용함으로써 각각의 객체나 모듈을 단독으로 테스트하기 쉽습니다.

**문제 7. virtual method를 사용하는 이유에 대해 기술하시오.**

virtual method는 함수 포인터와 같은 역할을 수행하는데, C++에서 함수 포인터 문법의 난해함을 최소화시켜줍니다. 즉, virtual method는 객체지향 프로그래밍에서 다형성을 지원하여 코드의 유연성과 재사용성을 높여주며, 코드를 간결하게 만들어줍니다.

​

**문제 8. 협력형 과제를 진행하는 상황 혹은 회사에서 업무를 진행하는 상황에서 갑자기 대응하기 어려운 문제를 마주하게 되었습니다. 이런 경우 여러분들은 어떤 형식으로 문제에 대응 할 것인가요 ? 어떻게 대응 할 것인지 상세하게 기술하세요.**

발생한 문제가 혼자서 대응하기 어려운 것이라면 혼자서 계속 생각하고 고민하는 것은 시간 낭비일 것입니다. 따라서 팀원(혹은 상사)에게 도움을 요청하여 해당 문제에 대해 함께 논의하고, 해결 방안을 찾을 것입니다.

​

**문제 9. Backlog를 작성함으로서 얻을 수 있는 이점들에 대해 기술하세요.**

- 누가 어떤 목적으로 어떤 작업을 수행하는 지 알 수 있어 프로젝트 협업시 용이합니다.  
- 프로젝트의 요구 사항을 명확하게 이해할 수 있습니다.  
- 새로운 요구사항이나 변경 요청에 대한 처리가 용이합니다.

**문제 10. 여러분들은 이미 Board (게시판) 을 Console UI와 함께 구현해봤습니다. 이 때 여러분들이 각자 분업하여 만들었던 작업들이 존재합니다. 여러분들이 각자 분업하여 만든 내용물들이 다른 사람들의 결과물과 원활하게 결합하지 못하여 폐기처분하는 경우도 있었습니다. 혹은 끔찍한 혼종이 탄생하는 경우도 있었습니다. 이런 현상이 나타나게 된 근본적인 이유에 대해 기술해봅시다.**

우선 의사소통의 부재가 원인이 될 수 있습니다. backlog를 활용하지 않는 등 의사소통의 부재로 각자의 일을 독립적으로 진행하여, 다른 팀원들이 어떤 작업을 진행하고 있는지 알지 못하여 발생하는 것입니다. 좀 더 나아가 이러한 소통의 부재로 인해 각자 데이터를 주고받는 방식이 일치하지 않아, 호환성의 문제 등이 발생하여 끔찍한 혼종이 탄생게 됩니다. 또 다른 원인으로는 테스트 활용의 미흡함 입니다. 테스트를 진행하지 않고 작업을 하게 되면 전체적인 작업 흐름 파악이 어렵고, 버그 문제를 발견하기 어렵습니다. 즉 테스트를 진행하지 않고 결합하려고 할 때 여러 문제가 발생할 가능성이 높아지게 됩니다.

​

**문제 11. 기능 단위로 Backlog 를 작성하는 경우 어떤 문제들이 발생 할 수 있는지 기술해봅시다.**

기능 단위로 Backlog를 작성하게 되면 Backlog가 아닌 Todo 리스트 관리가 되어 버립니다. 즉, 세부 사항을 위주로 다루게 되어 복잡도가 증가하고 결과적으로 작업의 목적이 불분명해집니다. 결국 협업에 있어 의사소통이 어려워집니다.

**[ 복합 문제 12 ~ 20 ]**

깃허브 링크:

<https://github.com/Leejaelim/SDC-Comprehensive-Evaluation/tree/main/1%EA%B8%B0/first/jaelimlee/comp12>

**13. DB에 속도 정보와 등록 시간을 저장 할 수 있는 테이블을 작성합니다.**

CREATE TABLE velocity\_data (

id bigint NOT NULL PRIMARY KEY AUTO\_INCREMENT,

velocity\_speed double NOT NULL,

velocity\_direction double NOT NULL,

registration\_time datetime(6) DEFAULT CURRENT\_TIMESTAMP(6)

);

**문제 21. 현재 실시간으로 분석되는 영상 시스템을 사용하고 있습니다. 이 영상 시스템은 촬영하는 구간에 사람이 몇 명 있는지 확인 할 수 있습니다. 이러한 시스템을 특정 시간에 사람들이 붐비는 구간에서 사용하고자 합니다. 영상 시스템의 목적은 버스 정류장에 사람이 몇 명 있는지 판정하려고 합니다. 어떻게 DB에 데이터를 저장해야 스토리지를 효율적으로 사용 할 수 있을까요 ?**

우선 DB table을 버스 정류장 정보와 관련된 table, 버스 정류장에 머문 사람과 관련된 table을 설계합니다. 전자의 table에는 각 정류 장의 정보(고유 식별자, 위치-주소/경도/위도, 정류장의 이름 등)을 저장합니다. 후자의 table에는 측정된 사람의 수, 측정된 시간 정보를 저장합니다.또한 영상의 경우 용량이 크기 때문에 이미지 혹은 영상 파일을 파일 시스템에 저장하여 파일의 경로를 DB에 저장하는 것이 좋을 것 같습니다.

​

**문제 22. 포인터가 필요한 이유가 무엇인지 기술하세요.**

​포인터는 저수준의 메모리 조작이 가능하게 하며, 메모리의 효율적인 사용과 프로그램의 성능 향상을 도모하는 데에 도움을 주기 때문입니다.

**문제 23. 모든 함수들이 자신만의 개별적 공간인 Stack을 사용합니다. 함수들끼리 서로 Stack 을 공유하지 않는 이유에 대해 기술하세요.**

함수 호출의 독립성과 각 함수 호출 프레임이 자체적으로 스택 프레임을 가지기 때문입니다. 함수들 간에 스택을 공유하면 지역 변수나 매개 변수 등이 충돌할 우려가 있고, 각 함수 호출이 독립적으로 이루어지지 않아 예측할 수 없는 동작을 유발할 수 있습니다. 따라서 각 함수 호출은 자체적인 스택 프레임을 갖고 실행되어야 하며, 이를 통해 함수 간의 독립성과 안정성이 유지됩니다.

**[복합 문제 24~32]**

깃허브 링크:

<https://github.com/Leejaelim/SDC-Comprehensive-Evaluation/tree/main/1%EA%B8%B0/first/jaelimlee/comp24>

**문제 33. 처음에는 Domain 분리가 필요 없다 생각하였던 특정 Entity 내부의 어떤 멤버 변수 (필드) 가 있습니다. 시간이 지남에 따라 시스템이 커졌고 위 내부 변수가 수행하는 작업의 복잡도가 높아지고 있습니다. 여러분들이라면 이 녀석을 어떻게 처리하겠습니까 ?**

**[ 복합 문제 34 ~ 36 ]**

깃허브 링크: