C/C++ 종합 평가 조정환

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

답변 : 정신 없이 시작해서 C 와 C++ 그리 Python 맛보기 까지 달려왔습니다.

처음 접해보는 컴퓨터 언어이기 때문에 개인적으로 지식의 부족함이 많았지만,

강사님의 역치 때려박기에 정신 못 차리다 어느 순간 보니 그래도 조금은 이해하고 있는

수준으로 성장하였습니다. 앞으로도 더욱 정신 없이 때려 박히겠지만, 힘들 때 더욱

노력할 수 있는 학생이 되겠습니다. 감사합니다.

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

답변

1) 협업을 진행하는데 있어서 가독성을 높일 수 있다.

2) 각 Domain 간의 의존성을 최대한 배제하여 유지보수에 유리하다.

3) Domain 개념에 집중하여 코드의 복잡성을 감소 시킬 수 있다.

4) Entity를 관리 할 수 있다.

3. C/C++ 에서 Test 목적으로 GoogleTest를 사용하였습니다.

TDD(Test Driven Development)를 통해 얻을 수 있는 이점들에 대해 기술하세요.

답변

1) test를 통해서 예상대로 작동하지 않는 것을 방지하여 신뢰성을 높일 수 있다.

2) 개발 단계에서 각 기능마다 test를 통해 안정성을 확보할 수 있다.

3) 어디서 부터 잘못 됐는지 바로 파악이 가능하기 때문에 유지 보수에 용이하다.

4) 디버깅 시간을 단축 시켜 줄 수 있다.

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

답변

1) DDD로 구현한 Domain 중점 모델에서 각 Domain 마다 test를 수행할 때 더욱 더 신뢰성을 향상 시킬 수 있다.

2) test를 하면서 코드 작성에 시간이 소요 될 수 있지만, 결론적으로 test를 수행하고 넘어 갔을 때 최종 시간을 단축 시킬 수 있다.

3) 상호 보완적으로 DDD로 개발된 Domain 모델의 정확성을 빠르게 확인 가능하다.

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

답변

1) 가독성 : 협업 시 사용하는 언어의 획일화로 가독성을 증가시켜 각 팀원이 무슨 업무를 진행하고 있는지 빨리 파악이 가능하도록 해야함

2) 테스트 : 테스트(디버그)가 되지 않는 작업물은 공유하지 않아야 함

3) 이슈 관리 : 각자의 업무에서 이슈가 발생 됐을 때 그것을 공유하여 이슈 관리를 통해 중복되는 업무를 방지하고, 상대적으로 레벨이 낮은 팀원이 있더라도 같이 성장할 수 있음

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

답변 : 어떤 프로그램을 개발하는데 있어서 기능들이 서로 의존성이 가지게 될 때 향후 유지보수나 기능 추가 등의 변경 문제에 대응하기 어려운 상황이 발생할 수 있음. 이것을 방지하는 것이 IoC 인데 C 언어에서 함수 포인터 테이블을 사용하여 DDD를 통해 Domain 상호간의 의존성을 최소화하여 유지보수 및 변경에 용이하게 함

7. virtual method를 사용하는 이유에 대해 기술하시오.

답변 : virtual은 C에서 함수 포인터의 역할로 C에서 복잡하게 함수 포인터로 구현이 됐다면, C++에서는 virtual을 사용하여 문법상 복잡성을 최소화 시켜 주는 기능으로 마찬가지로 IoC를 달성하기 위함

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.

답변 : 우선 혼자 대응하기 어려운 문제는 즉시 이슈화 시켜 팀 전체에 이를 알리고 함께 해결할 수 있는 방안을 모색하는 것이 중요합니다.

그 이후에 팀에서 집단 지성을 발휘해서(?) 문제를 빠르게 해결 할 수 있고, 그렇지 않더라도 우선 내가 혼자 가지고 있는 것 보다 이미 이슈화를 시킨 상태이기 때문에 문제가 바로 해결이 되지 않더라도 협업에 지장 되지 않게 차근차근 해결해 나갈 수 있다.

9. Backlog를 작성함으로서 얻을 수 있는 이점들에 대해 기술하세요.

답변

1) Backlog를 작성함으로 팀 협업에 있어 상대방이 무슨 일을 하는지 바로 파악할 수 있음

2) 중복된 작업을 방지함으로 생산성을 증대할 수 있음

3) 폭포수 설계가 아닌 그때 그때 필요한 사항을 점검할 수 있어 에자일 하게 대응할 수 있음

4) 프로젝트가 터졌을 때 Backlog 단위로 분석하여 디버그 시간을 단축할 수 있음

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.

답변

1) Backlog의 잘못된 사용 : 기능적으로 Backlog를 작성하며 작업을 하다 보니 기능에 매몰되어 확장성을 보지 못함

2) 이슈 관리 : 발생되는 이슈를 개인적으로 해결하려고 노력하다 보니 다른 팀원들은 그 상황을 알지 못하고 또 그 상황이 반복되다 보니 자기가 만들고 싶은대로 만들어버림

3) DDD : 기본적으로 DDD를 잘 하지 못하여 의존성을 가져버리는 상황이 연출되어 결합을 어렵게 함

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

답변

1) 작업이 커짐에 따라 이전에 진행했던 작업의 추적이 어려움

2) 기능에 너무 매몰되다 보니 해당 기능에서 파생되는 곁가지를 놓치기 쉽고, 협업에 있어 결합성을 해침

3) 어쨌든 날긴 나는데 왜(?) 나는지 잘 모름...

[ 복합 문제 12 ~ 20 ]

<https://github.com/Jungpop/Comprehensive-Evaluation/commit/74edf29746cb6115154ca16d98b88df9c57aa5b6> (12~16번)

12. 속도 Entity를 작성합니다. (복잡도를 낮추기 위해 2D 로 제한)​

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

CREATE TABLE velocity (

number bigint NOT NULL PRIMARY KEY AUTO\_INCREMENT,

speed varchar(128) NOT NULL,

direction varchar(128) NOT NULL,

reg\_date datetime(6) DEFAULT CURRENT\_TIMESTAMP(6)

);

14. DB에 접속 할 수 있는 싱글톤을 생성합니다.​

15. 소수점 형태의 랜덤 sleep 기능을 구현합니다.​

16. 10번 정도 루프를 돌면서 0.3 ~ 1.0초 사이의 시간 이후 랜덤 속도 정보를 DB에 저장합니다.​

======================================================================

17. 가속도(Acceleration) Entity를 작성합니다 (마찬가지로 2D)​

18. 속도 정보를 토대로 가속도 정보를 산출합니다.​

19. 속도 정보를 토대로 전체 이동 거리를 계산합니다.​

20. 속도 정보를 토대로 현재의 변위 벡터를 계산합니다 (초기 위치 [0, 0] 으로 가정)

21. 현재 실시간으로 분석되는 영상 시스템을 사용하고 있습니다. 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 ?

답변

1) 먼저 DB에 어떤 데이터를 저장해야 할 지 고민해봐야 할 것 같습니다.

ex) 사람 수, 위치 등 어떤 정보를 저장해야 할지 결정

2) 저장 될 데이터 별로 Table을 따로 관리해야 할 것 같습니다. 이렇게 함으로 검색 및 sorting에 용이할 수 있습니다.

3) 저장하고 얼마간의 간격으로 불필요한 데이터를 삭제할지 고민해 봐야 할 것 같습니다.

ex) 30분만 저장한다 했을 때 자동차 블박처럼 30분 이후의 데이터는 삭제되고 현시간부터 30분 이전까지의 데이터만 관리하겠다 등으로 저장 공간을 관리

4) 사람이 한 명도 없다면 그 데이터를 저장해야 할 까 ? 싶습니다. 이런 경우 데이터를 저장하지 않고 바로 삭제(?) 함으로 저장 공간을 관리

​

22. 포인터가 필요한 이유가 무엇인지 기술하세요.

답변

1) 동적 메모리를 할당함으로 효율적인 메모리 관리를 할 수 있다.

2) 포인터로 함수 및 변수 호출 시 주소값만 전달하면 되기 때문에 오버로딩(? 자세한 단어를 잘 모르겠습니다.)을 방지할 수 있다.

​ 3) 함수 포인터를 사용 할 때 Domain 간의 호출이 가능함으로 DDD에 용이하다.

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

답변

1) Stack은 휘발성으로 한 번 사용되고 지워지기 때문에 Entity 관리를 잘 할 수 있다.

2) 만약 함수들이 Stack을 공유한다면, a라는 함수를 호출 했을 때 b 라는 함수에 영향을 줄 수 있기 때문에 시스템 오류(간섭)를 초래할 수 있다.

3) 함수는 호출된 Stack({})내에서만 유효하기 때문에 서로 Stack을 공유할 수 없다. 이는 호출 순서 등에 영향을 줄 수 있음

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 ?

답변 : 뭐 있습니까 DDD로 찢어 하나의 Domain으로 관리할 것 같습니다.