게임소프트웨어공학 2차시험 정리

Lecture 3

**소프트웨어 개발의 위기 (원인)**

1. 소프트웨어의 대규모화
2. 유지보수의 어려움
3. 개발 일정 및 소요 예산 예측의 어려움

* 높은 수준의 소프트웨어가 요구될 경우
* 개발 일정 측정 어려움, 소요 예산 예측 불가능

1. 신기술 적용에 대한 방법론 부재

소프트웨어 개발의 위기 (결과) – 안나올 가능성 높음

1. 프로젝트 예산 초과
2. 프로젝트 일정 지연
3. 낮은 품질
4. 유지보수 어려움

소프트웨어 개발 프로세스

1. 폭포수 모델 (시험 안나옴)
2. 프로토타이핑 모델

* 전체 소프트웨어의 일부를 빠르게 제작하는 방식.
* 장점

: 요구사항을 정확하게 판단할 수 있음

실현 가능성을 빠르게 확인 가능

* 단점

: 완제품으로 오해

프로토타입 개발 내용이 완제품에 반영되어 품질 저하

1. 나선형 모델

* 개발 단계를 반복적으로 수행하여 점차적으로 안정적인

소프트웨어를 개발하는 방식

* 장점

: 큰 시스템을 안정적으로 개발할 때 효과적

순차적으로 검증이 이루어지기 때문에 안정적인 문제 해결 가능

유지보수와 개발의 일원화 가능

* 단점

: 많은 투자가 필요하여 작은 규모의 프로젝트엔 어울리지 않음

관리가 복잡함

Lecture 4

애자일 선언문 – 빨간부분 빈칸 채우기로 시험나옴

우리는 소프트웨어를 개발하고 또 다른 사람의 개발을 도와주면서 소프트웨어 개발의 더 나은 방법들을 찾아가고 있다. 이를 통해 우리는 다음을 가치 있게 여기게 되었다.

공정과 도구보다 개인과 상호작용을,

포괄적인 문서보다 작동하는 소프트웨어를,

계약 협상보다 고객과의 협력을,

계획을 따르기 보다 변화에 대응하기를

가치 있게 여긴다. 이 말은, 왼쪽에 있는 것들도 가치가 있지만, 우리는 오른쪽에 있는 것들에 더 높은 가치를 둔다는 것이다.

**Git 정리**

- Git 은 트리구조와 gitID를 가지고 관리한다

- 브랜치를 따는 순간은 그 모든 데이터를 복사해간다. 거기서부터 바꾸기 시작한다는 의미. 데이터중복을 피하기 위해서.

# 깃 명령어 정리

-git init: 로컬저장소를 초기화

-git status: 현재 상태를 확인

-git add: 변화된 것 반영 준비

-git commit: 변화된 것을 로컬 저장소에 올림

-git log: commit 되었던 list 가 출력 됨.

-git branch: branch list 출력

-git branch branchname: branchname 으로 branch 생성

-git checkout branchname: branchname 으로 branch 이동

-git branch --[d/D] branchname: branchname branch 삭제

. Git reset --(hard/mixed/soft) CommitID

. Hard 라는 옵션은 워킹디렉토리에 있던내용물 뿐만 아니라 스테이징 에어리어에 있던 내용도 싹다 덮어 씌워버린다

. Mixed 라는 옵션을 하면 워킹디렉토리에 있던 내용물은 남겨두고 스테이징 에어리어에 있는건 없앤다.

. Soft 라는 옵션은 워킹디렉토리에 있던 내용과 스테이징 에어리어에 있는 내용도 남아있다.

+) 1차시험 이후 추가된 명령어

-git push: 로컬 저장소의 상태를 원격저장소에 올림

-git pull: 원격저장소의 상태를 로컬저장소로 받아옴

-git clone 저장소URL.git: 원격저장소를 로컬저장소로 최초로 가져올 때 사용

 # ;(세미콜론)을 사용하면 순차적으로 실행한다

Ex) git add a.txt; git commit -> add후 commit 한다.