

SW DEVELOPMENT PRACTICES

TEAM TELETUBBIES

INDEX

1. 팀원 소개
2. Best Practice 조사
3. Best Practice 제안
4. Best Practice 검증
5. Q&A



TEAM TELETUBBIES

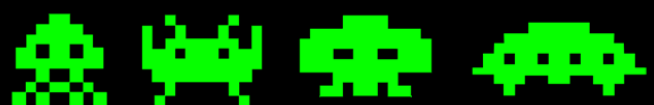
LV.1 강예원 소프트웨어학부 2019054693

LV.2 김예진 소프트웨어학부 2019069034

LV.3 전민지 소프트웨어학부 2019025823

LV.4 정연주 소프트웨어학부 2019014739

LV.5 한수빈 소프트웨어학부 2019071994



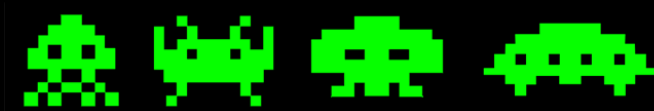
2. Best Practice 조사



소프트웨어 개발이란 ?

시장의 목표나 사용자의 요구를 소프트웨어 제품으로 만들기 위한 요구사항, 설계, 구현, 테스트, 유지 보수 등의 일련 과정을 말한다.

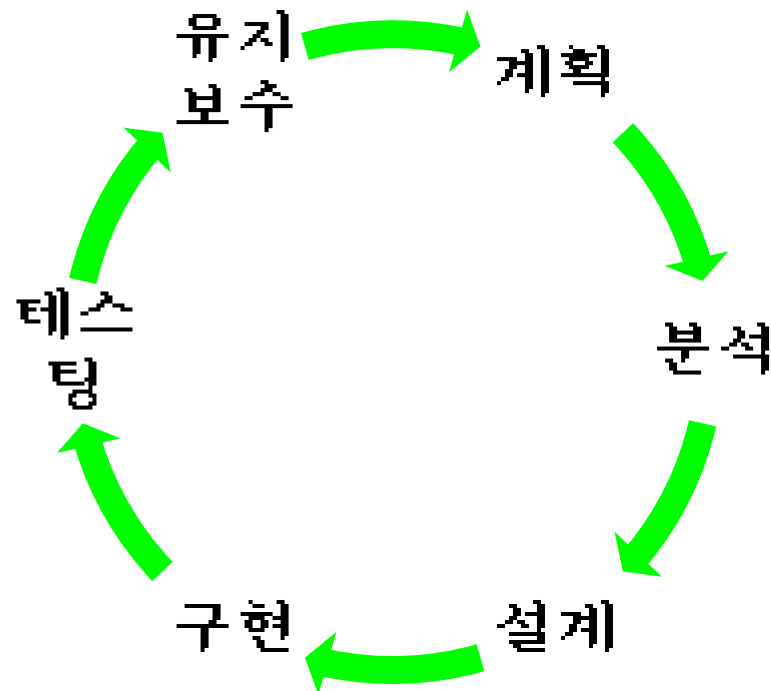
2. Best Practice 조사 - 소프트웨어 개발 과정



소프트웨어 개발 생명 주기(Software Development Life Cycle)

소프트웨어 개발부터 폐기까지 전 과정을 하나의 생명 주기로 정의하고 단계 별 공정을 체계화

소프트웨어 제품의 종류에 따라 다른 작업 내용과 작업 순서가 요구된다.



2. Best Practice 조사 - 소프트웨어 개발 과정 모델



프로세스 모델 적용



개발 진행 상황 명확히 파악 가능

각 단계의 산출물 통해 검토 용이

2. Best Practice 조사 - 소프트웨어 개발 과정 모델



폭포수 모델(Waterfall Model)

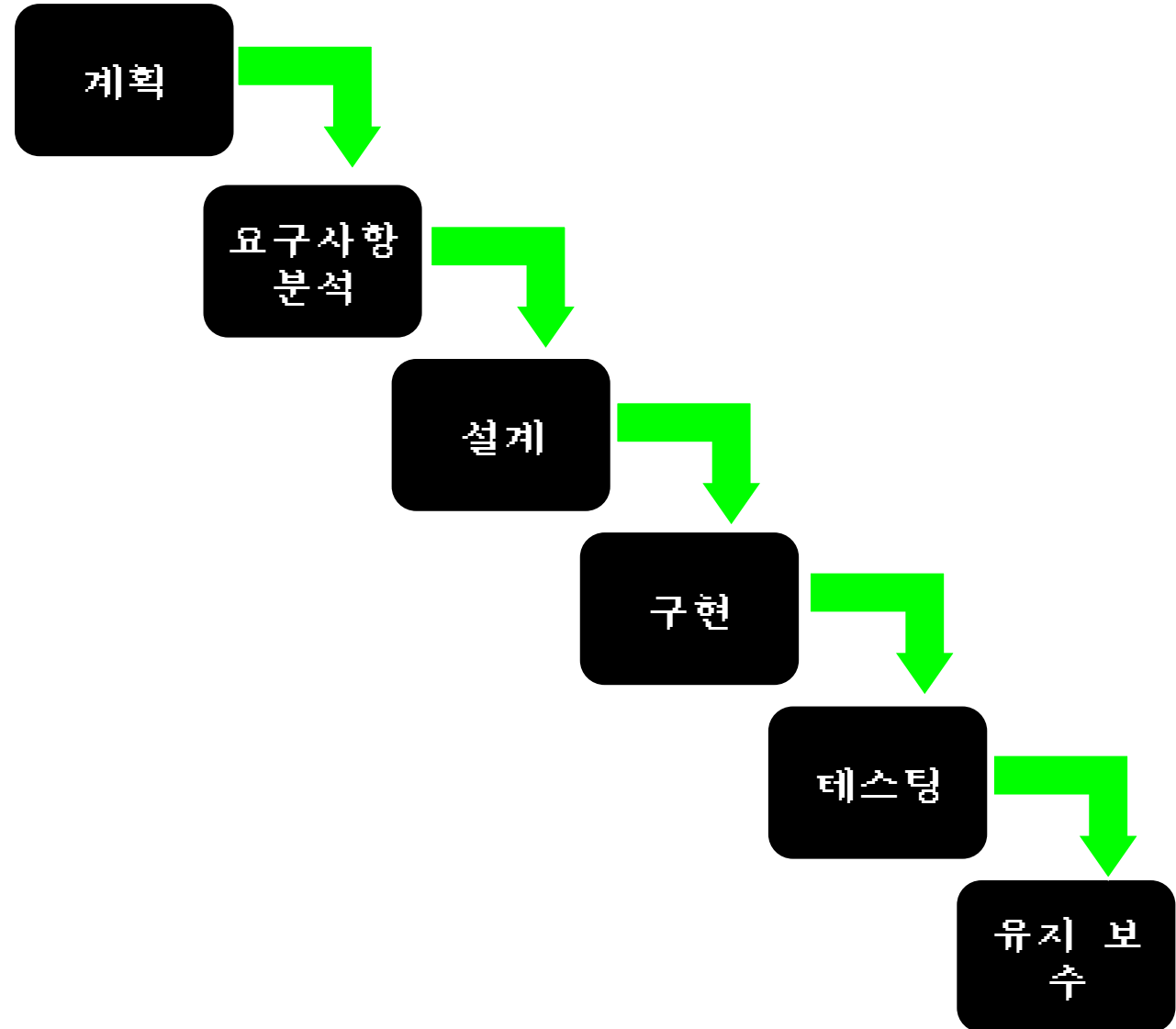
앞선 단계를 끝내면서 결과물을 다음 단계를 담당하는 그룹에게 넘기는 방식

+

- 프로젝트 진척도와 책임 소재 파악 용이

-

- 변경 사항 반영에 시간이 오래 소요되어 니즈를 충족하지 못하는 상황 발생



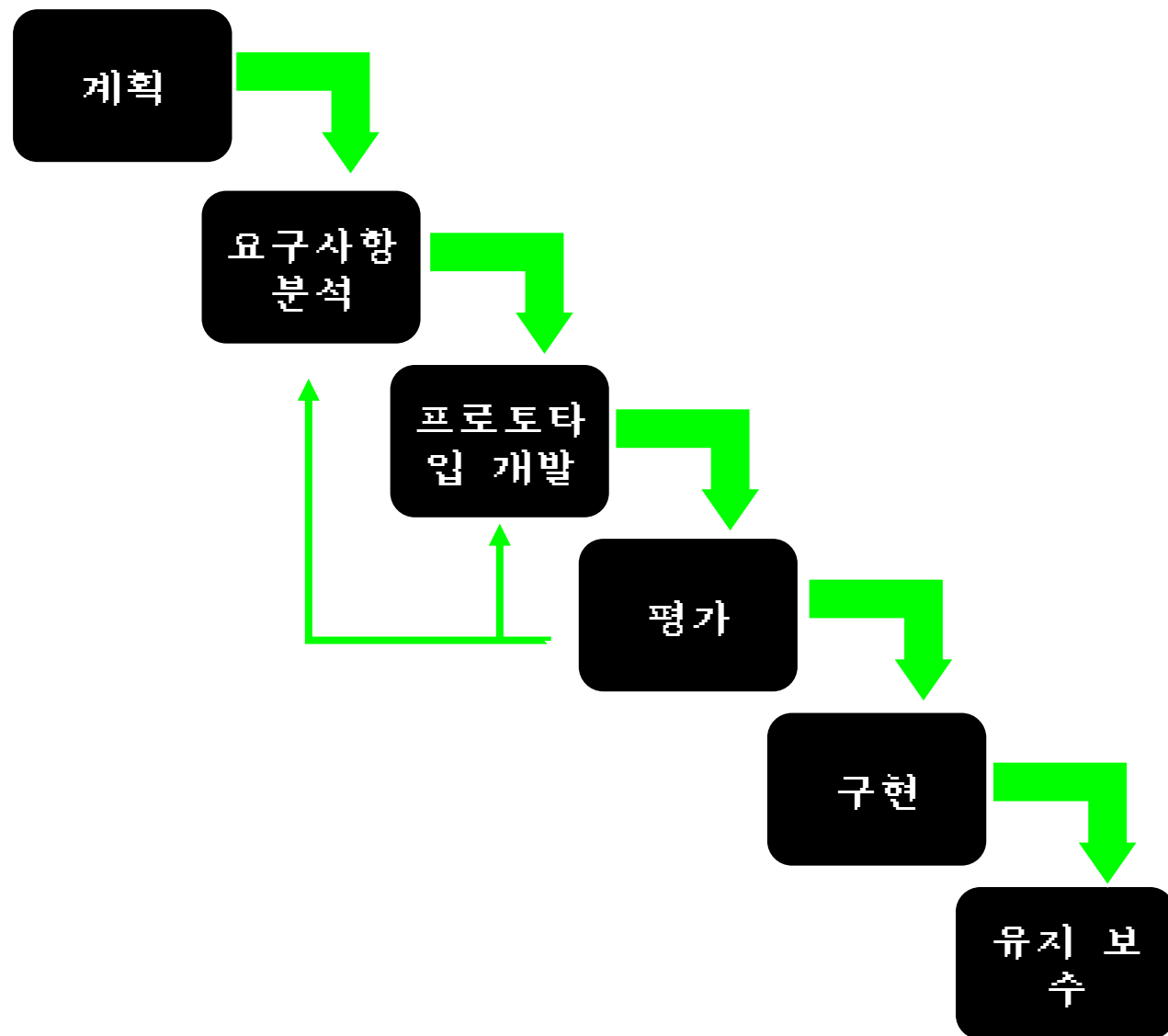
2. Best Practice 조사 - 소프트웨어 개발 과정 모델



프로토타입 모델(Prototype Model)

핵심 기능이나 위험성이 높은 기능을 시제품으로 먼저 만들어 평가한 후 구현

- +**
 - 요구사항 도출과 시스템 이해 용이
 - 의사소통 향상
- - 사용자가 시제품을 완제품으로 오해 가능
 - 폐기되는 시제품에 따른 개발자의 불만 발생 가능



2. Best Practice 조사 - 소프트웨어 개발 과정 모델



나선형 모델(Spiral Model)

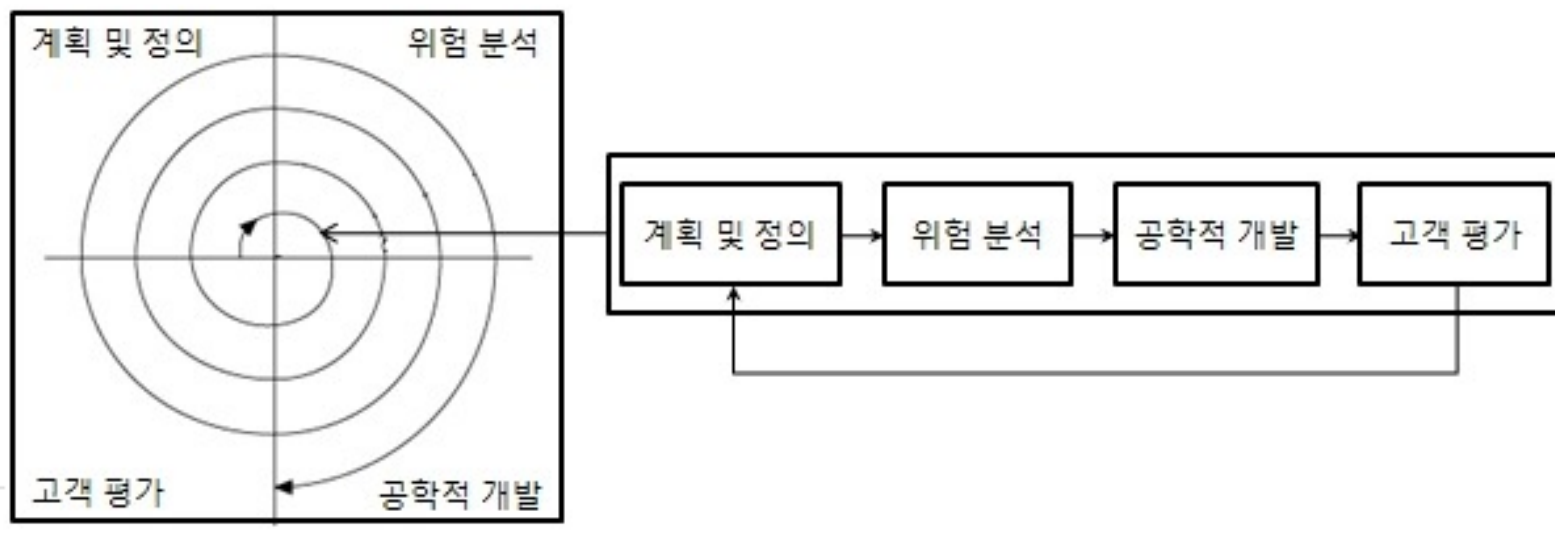
나선을 따라 돌 듯이 여러 번의 개발 과정을 거쳐서 점진적으로 완성하는 방식

+

- 실패 위험 감소
- 소프트웨어의 강인성을 높이며 정확성 획득 가능

-

- 프로젝트 기간이 오래 걸림
- 반복 단계가 길어질수록 프로젝트 관리가 어려움



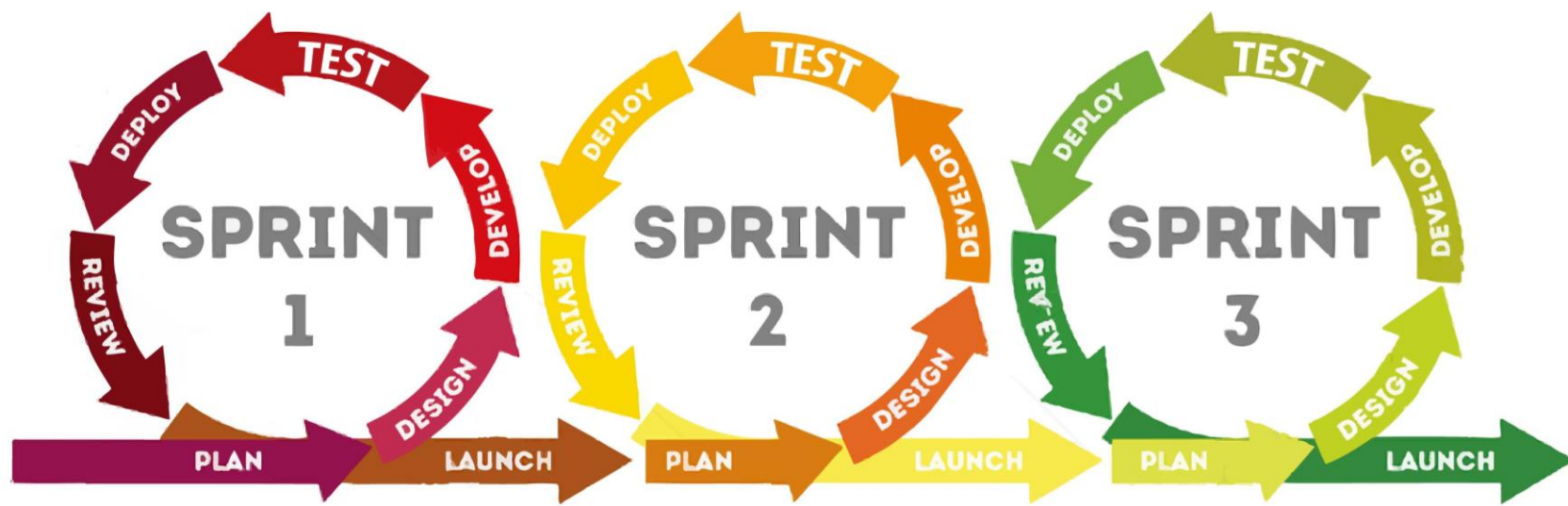
2. Best Practice 조사 - 소프트웨어 개발 과정 모델



애자일(Agile)

시장 변화에 기민하고 유연하게 대처하기 위해 우선순위 중심으로 반복 개발을 하는 방법

유동적이고 개방적인 소프트웨어와 요구사항의 변경에 대처하기 위해 만들어짐
제한된 시간과 비용 안에서 정보는 불완전하고 예측은 불가능하다는 전제를 가짐



2. Best Practice 조사 - 소프트웨어 개발 과정 모델



데브옵스(DevOps)

Development와 Operation의 합성어로 개발 담당자와 운영담당자가 연계하여 협력하는 개발 방법

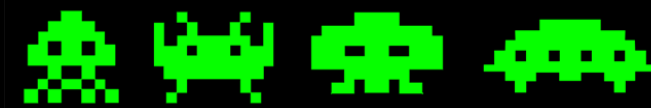
개발/테스트팀과 운영/IT 인프라팀 사이의 간격을 좁히는 것에 초점
개발 초기부터 실행이 가능한 상태로 유지하며 퀄리티 컨트롤을 적용





! 개발 목적, 개발 제품의 종류에 따른 적절한 환경 구축 중요 !

2. Best Practice 조사 - 구축 환경



모델링

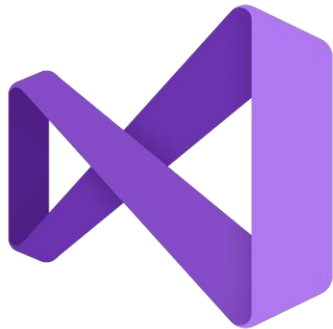


Visual Paradigm



StarUML

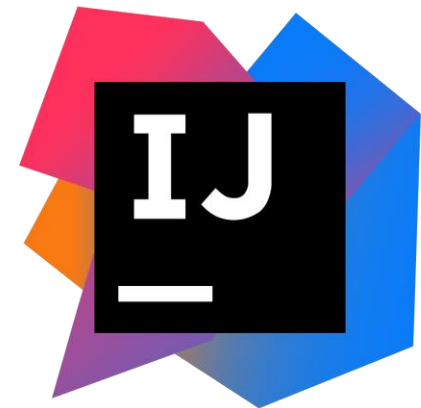
통합 개발 환경(IDE)



Visual Studio IDE



Eclipse



IntelliJ

2. Best Practice 조사 - 구축 환경



텍스트 편집기



Atom



Visual Studio Code



Sublime Text

DBMS



Oracle DB



MySQL

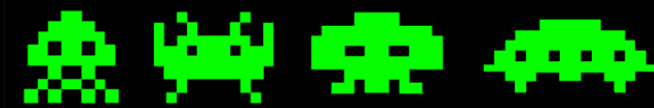


PostgreSQL



mongoDB

2. Best Practice 조사 - 구축 환경

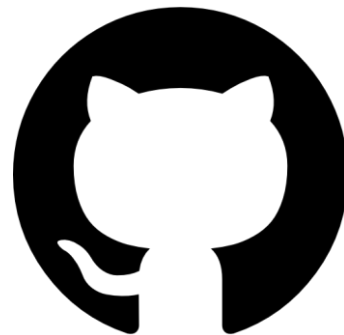


버전 관리 시스템(VCS)

- Local VCS : 서버 없이 로컬 컴퓨터 내에서 버전을 관리, 데이터베이스 만으로도 구현이 가능하므로 단순하고 개인 프로젝트에 적합
- 중앙집중식 VCS : CVS
- 분산 VCS : Git - > Github, bitbucket, gitlab
Team Foundation Server



git



2. Best Practice 조사 - 구축 환경



프로젝트 관리 도구



Jira



Asana



Slack

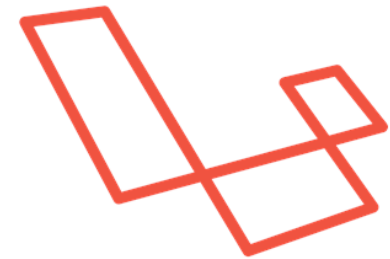
Web Framework



Spring



Django



laravel

Laravel

2. Best Practice 조사 - 구축 환경



Test Framework

JUnit

JUnit

*j*behave

JBehave

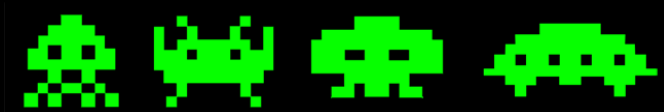


Cucumber

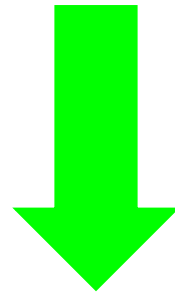
통합 및 배포



Jenkins
Jenkins

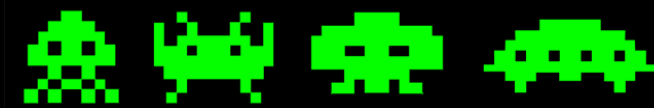


현업 개발자와의 인터뷰



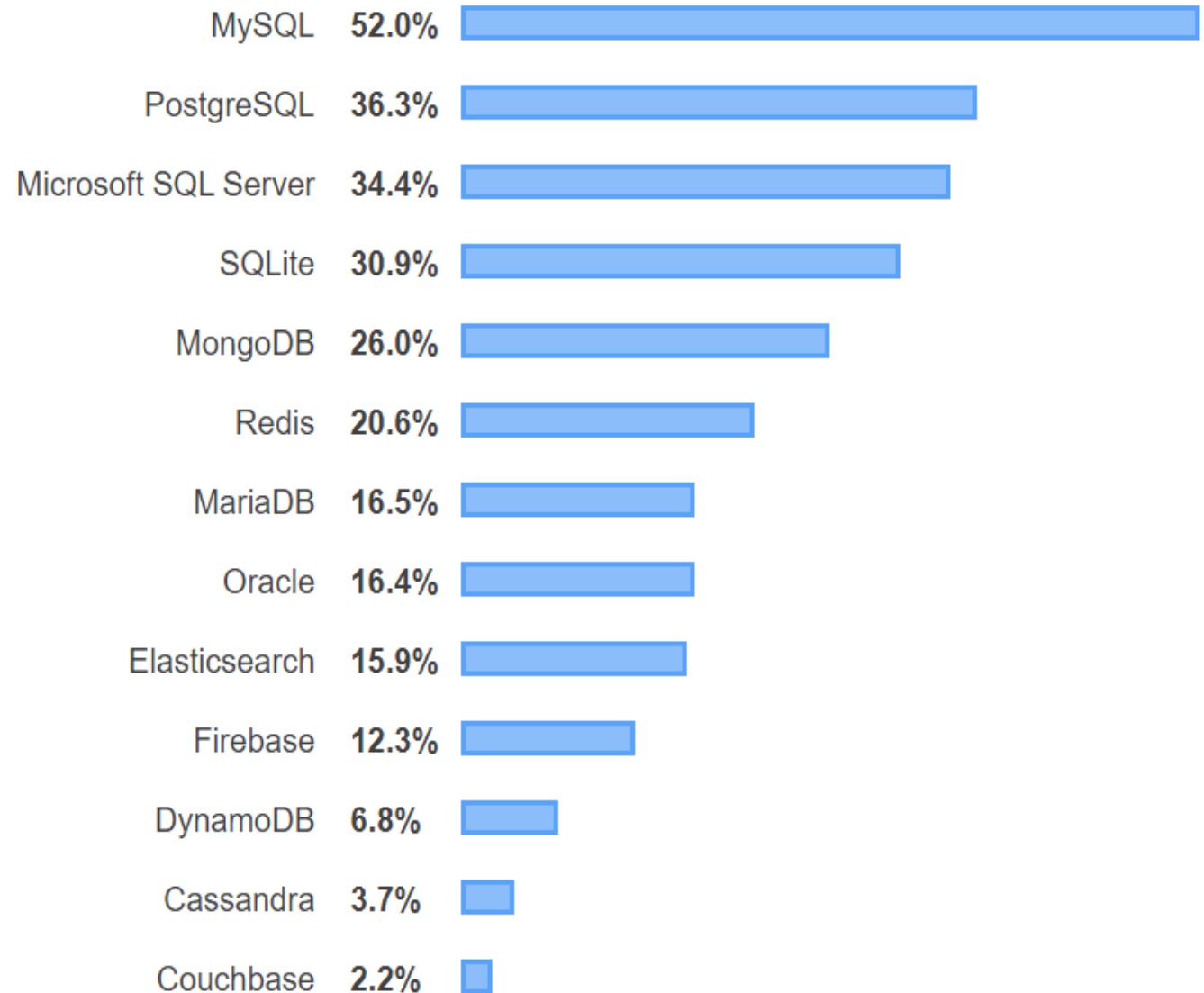
Github와 Jira를 많이 사용

2. Best Practice 조사 - 최신 트렌드



Stack Over Flow의 조사 결과

- Database

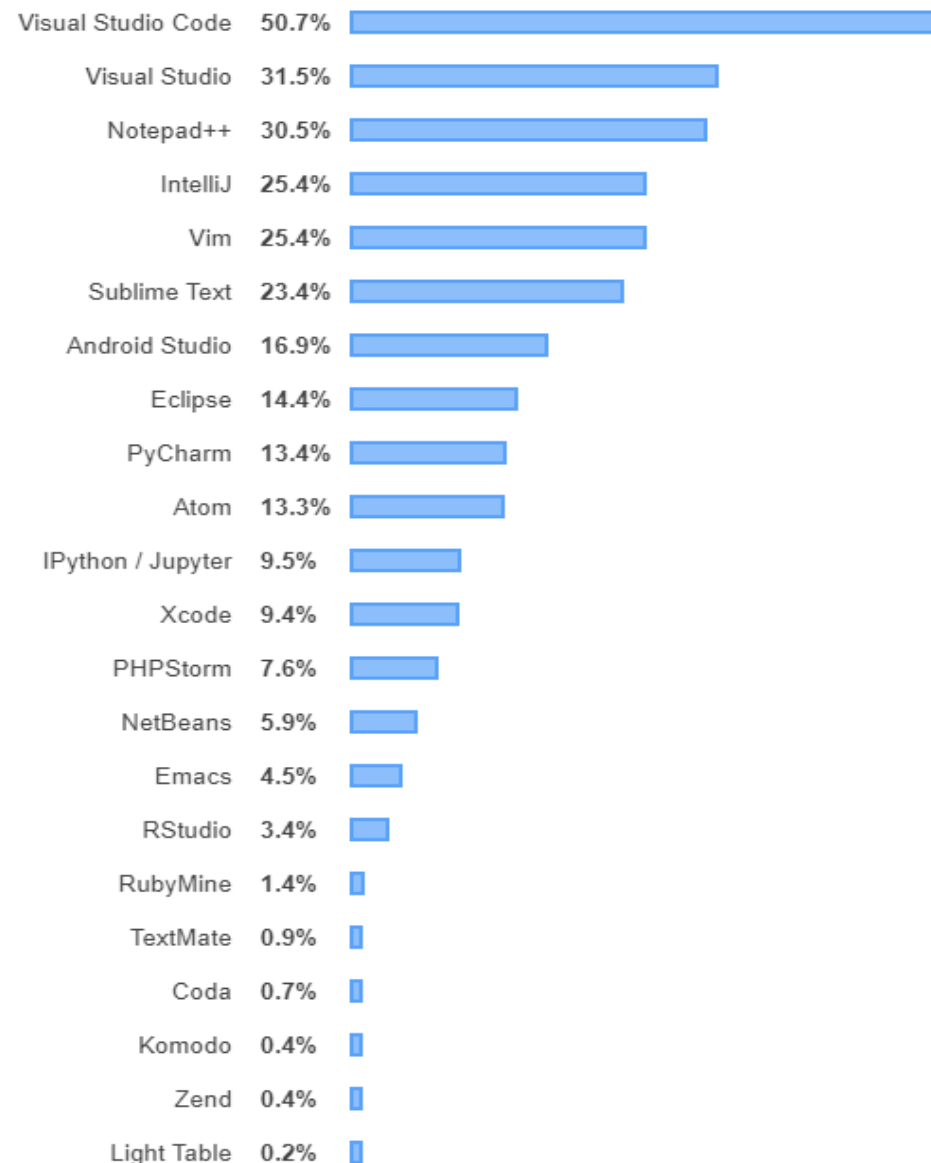


2. Best Practice 조사 - 최신 트렌드

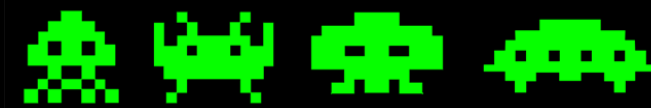


Stack Over Flow의 조사 결과

- Development Environment

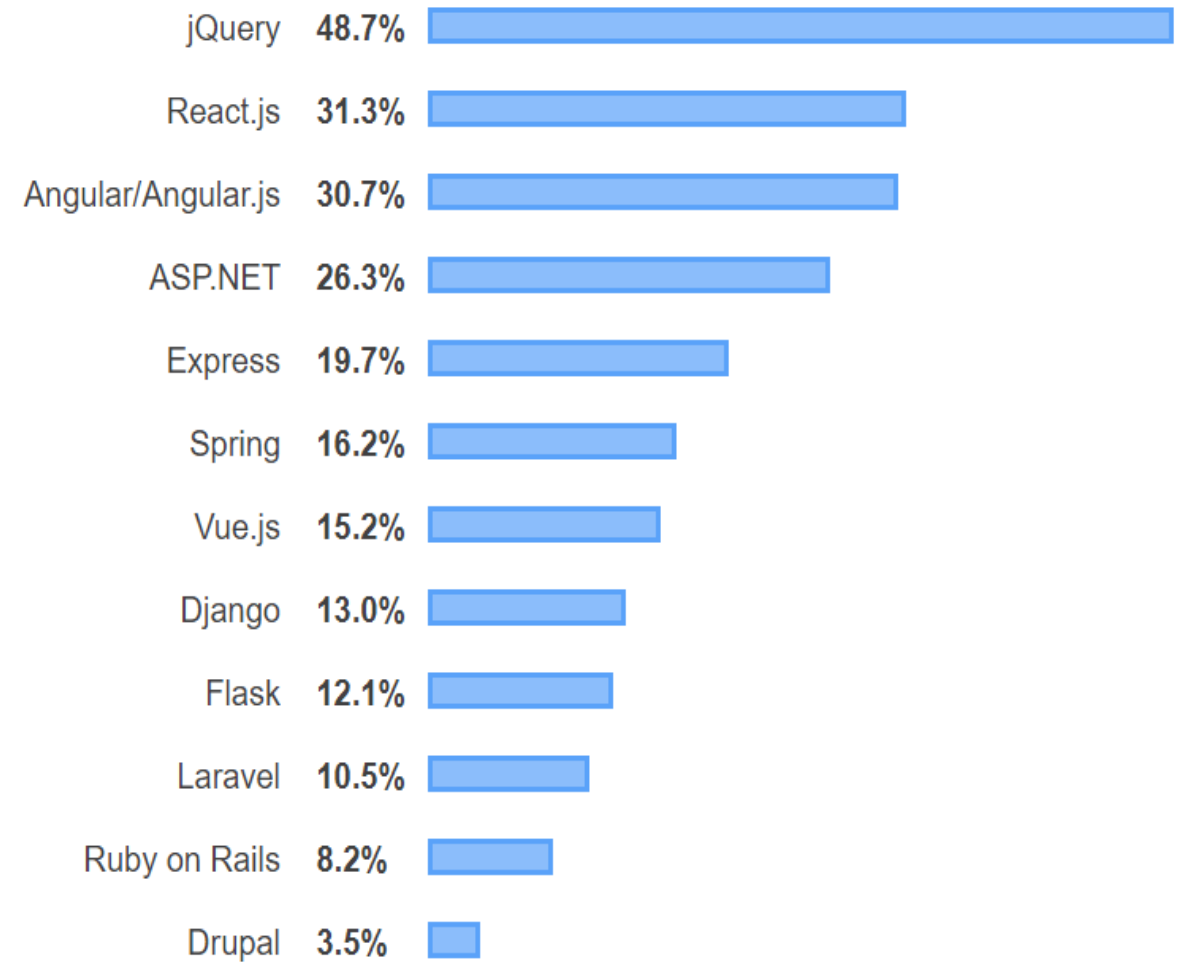


2. Best Practice 조사 – 최신 트렌드



Stack Over Flow의 조사 결과

– Web Framework



2. 개발 프로세스



- Week 1 (11/24 ~ 11/30)

Requirements analysis, additional requirements meetings.
Choosing software development tools to match our project.

- Week 2 (12/1 ~ 12/7)

Share tasks using project management tools.
Start Software Development.

- Week 3(12/8 ~ 12/12)

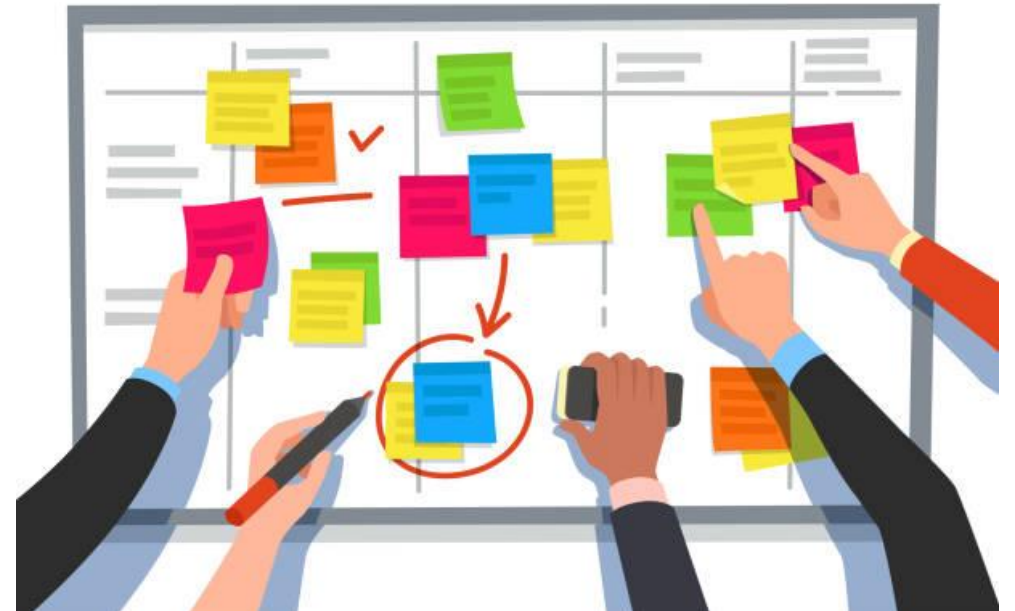
Testing, Maintenance
Software quality control
Release

3. Best Practice 제안 - 개발 프로세스 모델



Kanban

- Quickly add errors or additional requirements.
- Get a quick view of everything we need to do, what we're doing, and what we've done.



3. Best Practice 제안 - 프로젝트 관리

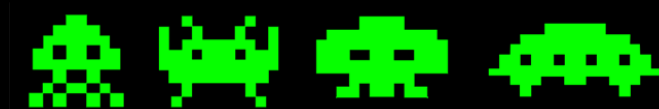


Your entire project, in a single glance.

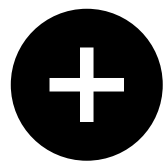


1. 이번 프로젝트는 빠르게 개발해야 한다.
2. 업무와 현황을 한눈에 파악할 수 있어야 한다.
3. 이슈를 실시간으로 생성할 수 있어야 한다.
4. Github와 연동이 되어야 한다.

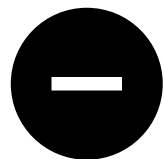
3. Best Practice 제안 - 프로젝트 관리



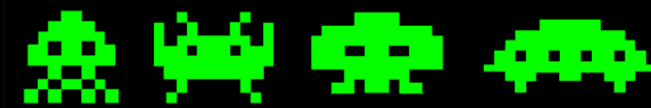
Redmine



일감 진행에 따른 이메일 통보가 가능하다.
일정 관리가 가능하다.



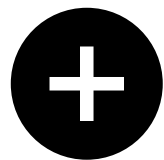
Jira가 Redmine보다 업무 현황을 한눈에 파악하기 쉽다.



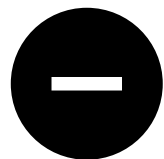
Trello



Your entire project, in a single glance.



직관적으로 이슈의 상태를 확인할 수 있다.

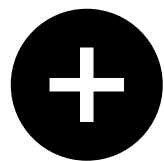


이슈 검색 기능이 정확하지 않다.

이슈의 히스토리 관리가 어렵다.

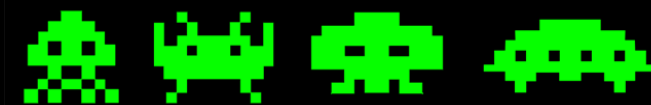


Jira



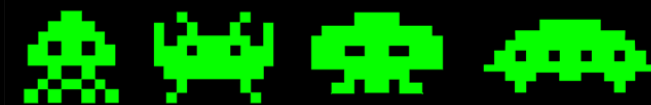
- 업무 현황을 한눈에 파악할 수 있다.
- Github와 연동이 가능하다.
- 업무 상태 변경이 쉽다.
- 팀원들에게 업무를 할당할 수 있다.
- 보고서를 작성할 수 있다.

3. Best Practice 제안 - 형상 관리

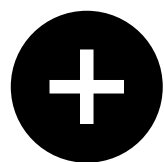


PERFORCE

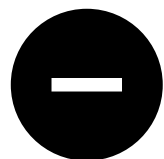
1. 언제 어디서나 협업이 가능해야 한다.
2. 히스토리 관리가 편해야 한다.
3. 체계적으로 관리하고 유지가 가능해야 한다.



Subversion



중앙 저장소에 반영이 된다.

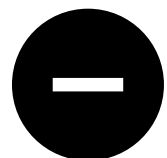


충돌 확률이 높다.

중앙 저장소에 에러가 생기면, 모든 작업이 마비가 된다.



CVS

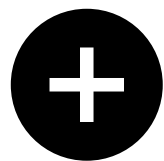


커밋 중 오류가 발생하면 롤백이 되지 않는다.

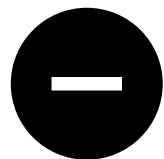


Perforce

PERFORCE



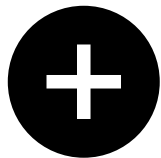
빠르고 히스토리 검색이 편하다.



파일명이 바뀌면, 히스토리 추적이 어렵다.



Git



- 처리 속도가 빠르다.
- 히스토리 관리가 쉽다.
- 어디서나 협업이 가능하다.

3. Best Practice 제안 - 소프트웨어 품질 관리



요구사항을 모두 충족하였는지 확인하기 위해 품질 관리 단계 진행

1. 개발 전, 필수 요구사항들과 추가 요구사항들을 요구사항 명세서로 정리.
2. 코드 리뷰와 평가
3. 오류 수정 위해 1과 2를 반복
4. 요구사항에 부합하는지 확인



통합 테스트(Integration Test)란?

모듈을 통합하는 과정에서 모듈 간 호환성의 문제를 찾아내기 위해 수행되는 테스트

통합 테스트(Integration test)가 왜 아닌가?

- 유닛 테스트는 다른 컴포넌트들과 독립적인 반면 통합 테스트는 그렇지 않음
- 따라서 유닛 테스트를 작성하는 것보다 복잡하고 오랜 시간이 소모됨
- 유닛 테스트만으로 충분하다고 느끼지 못할 때 사용됨

3. Best Practice 제안_테스팅



E2E 테스트(End-to-end Test)란?

현장에서 실제로 수행되는 것처럼 데이터베이스, 네트워크, 하드웨어, 시스템 등을 이용하여, 시스템의 기능을 처음부터 끝까지 테스트하는 것

E2E 테스트(End-to-end test)가 왜 아닌가?

- 데이터베이스에 자료가 정확하게 들어가 있는지, 서버는 일정시간안에 응답해야 하는 등 점검해야 할 항목이 많기 때문에 이번 프로젝트에 적합하지 않음

3. Best Practice 제안 - Testing



유닛 테스트(Unit Test)란?

컴퓨터 프로그래밍에서 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차

즉, 모든 함수와 메소드에 대한 테스트 케이스를 작성하는 절차

프로그램의 각 부분을 고립시켜서 각각의 부분이 정확하게 동작하는지 확인하는 것이 목적

유닛 테스트(Unit test)가 왜 필요한가?

- 단시간 내에 에러를 파악하고, 바로 잡을 수 있도록 해줌
- 지속적인 유닛 테스트 환경을 구축하면 어떠한 변화가 있더라도 코드와 그 실행이 의도대로인지를 확인하고 검증할 수 있게 됨

3. Best Practice 제안 - Testing



JUnit은 단위 테스트 framework 중 하나로 문자 혹은 gui 기반으로 실행된다.

Why you need JUnit testing

- 코드의 오류를 찾아, 최적화된 코드를 유추해준다.
- 테스트 기반 환경을 사용할 때, JUnit은 효율적이다.

3. Best Practice 제안 - 이슈관리



지라(Jira)란?

버그 추적, 이슈 추적, 프로젝트 관리 기능을 제공하는 소프트웨어

지라(Jira)가 왜 필요한가?

- 팀원들과 이슈를 바로 공유 가능
- 담당자 지정을 통해 효율적인 업무 분배 가능
- 이슈 해결에 대한 히스토리가 남기 때문에 후에 비슷한 이슈가 발생했을 때 처리 과정을 되짚어 볼 수 있음
- 요구사항을 스토리로 올리고, 메인 기능들을 에픽으로 묶어 기능 별 수정과 관리 용이
- 에픽으로 이슈를 묶게 되면 칸반보드에서도 어떤 에픽인지 한눈에 파악 가능
- 에픽 별 이슈의 진행상황을 %로 확인 가능

3. Best Practice 제안 - 통합 및 배포



젠킨스(Jenkins)란?

소프트웨어 개발 시 빌드, 테스트, 배포 등의 지속적인 통합을 자동화해주는 툴
다수의 개발자들이 하나의 프로그램을 개발할 때 버전 충돌을 방지하기 위해 각자 작업한 내용을 공유 영역에 있는 Git 등의 저장소에 빈번히 업로드함으로써
지속적 통합이 가능하도록 해 줌

젠킨스(Jenkins)가 왜 필요한가?

- 빌드와 테스트, 배포 자동화
- 프로젝트 표준 컴파일 환경에서의 컴파일 오류 검증
- 코딩 규약 준수여부 체크
- 각종 배치 작업의 간략화

3. Best Practice 제안 - 도구 및 환경



이클립스(Eclipse)와 인텔리제이(IntelliJ)란?

자바의 개발 도구로 IDE(통합 개발 환경)이라고 함
일반적인 IDE의 기능들처럼 소스 코드 편집기, 로컬 빌드 자동화, 디버거 등의 기능이 있음

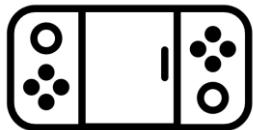
툴 팁을 통해 해결책을 알 수 있음

- 이클립스 : 전반적인 해결책 모두
- 인텔리제이 : 사람들이 많이 선호하는 기능 우선 설정 (툴 팁 처음 열었을 때)

이클립스(Eclipse)와 인텔리제이(IntelliJ)가 왜 필요한가?

- 평소 팀원들이 익숙한 개발 환경이 좋을 것이라 생각함
- 이클립스와 인텔리제이는 프로젝트 호환이 잘 됨

4. Best Practice 검증 - 요구사항 분석



PART ONE

Add 2nd Player

- 각 우주선은 다른 색깔로 표시
- 각 *player*의 점수는 분리하여 저장



PART TWO

Three Game Difficulty

- *Three Levels* : EASY, NORMAL, HARD
- 모든 레벨은 다른 게임 요소를 가짐
- 모든 점수는 난이도와 함께 저장

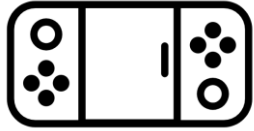
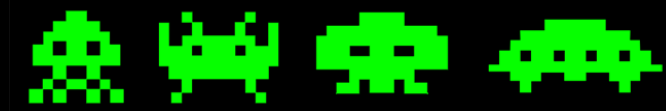


PART THREE

Better Interface

- *ESC key*를 통해 일시정지
- *Reset Highscore*

4. Best Practice 검증 - 요구사항 분석

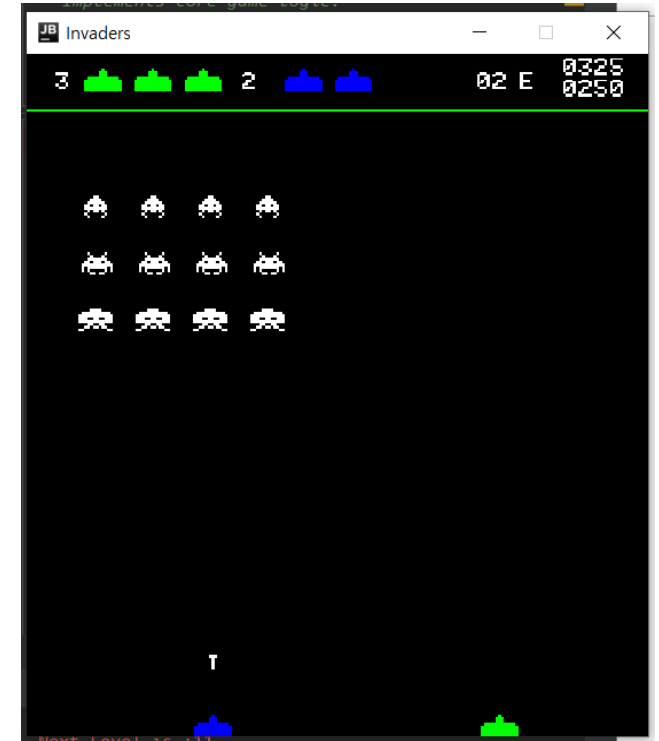


PART ONE

Add 2nd Player

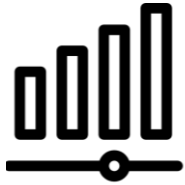
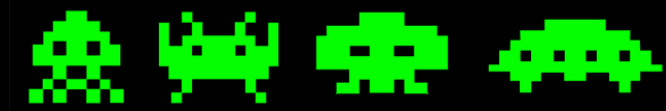


*Each Spaceship is
displayed
in different color*



*Each player's score
is recorded
separately*

4. Best Practice 검증 - 요구사항 분석



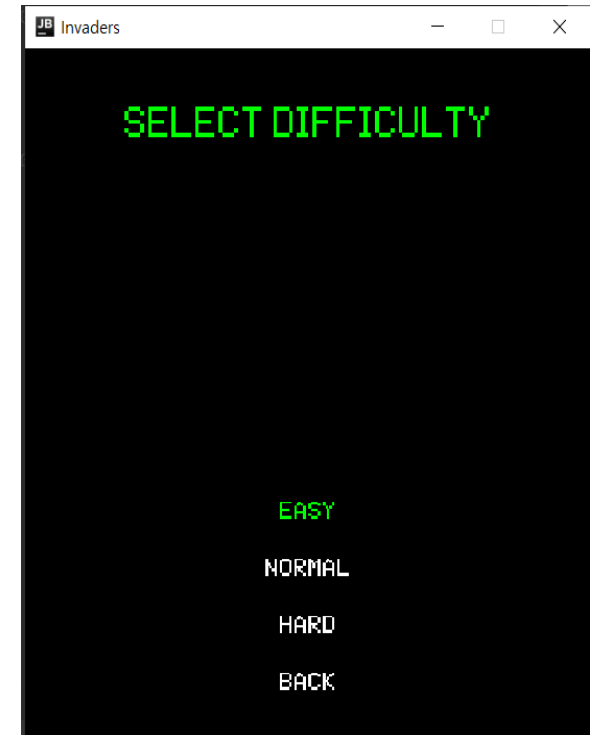
PART TWO

Three Game Difficulty



Main menu

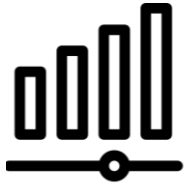
Select player



Three Levels

EASY, NORMAL, HARD

4. Best Practice 검증 - 요구사항 분석



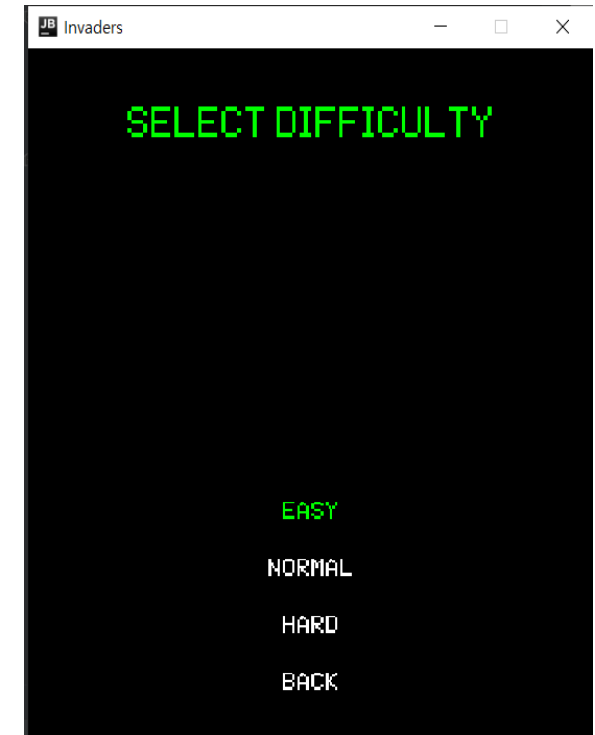
PART TWO

Three Game Difficulty



Main menu

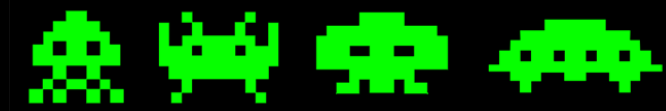
Select player



Three Levels

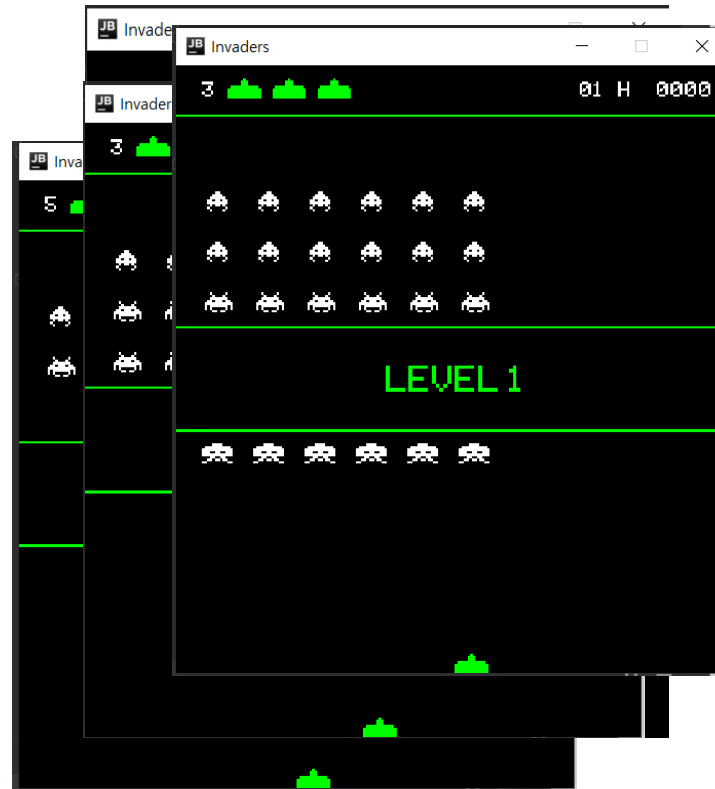
EASY, NORMAL, HARD

4. Best Practice 검증 - 요구사항 분석



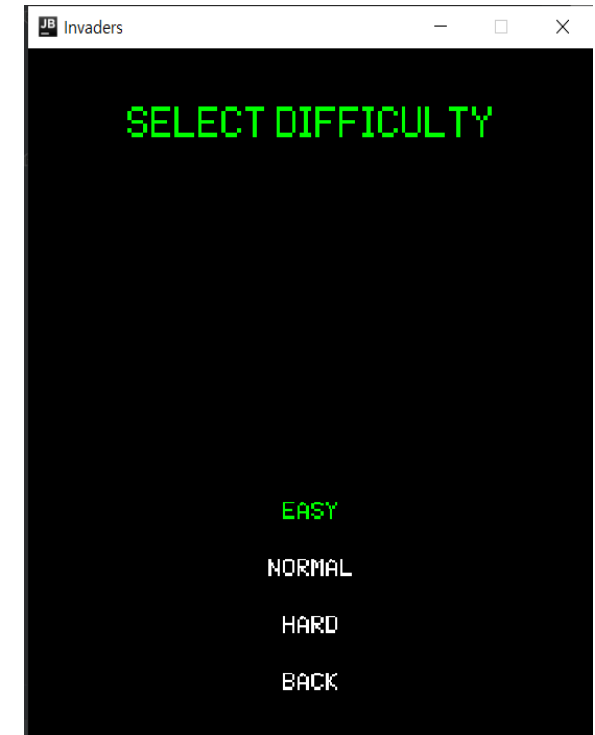
PART TWO

Three Game Difficulty



Main menu

Select player



Three Levels

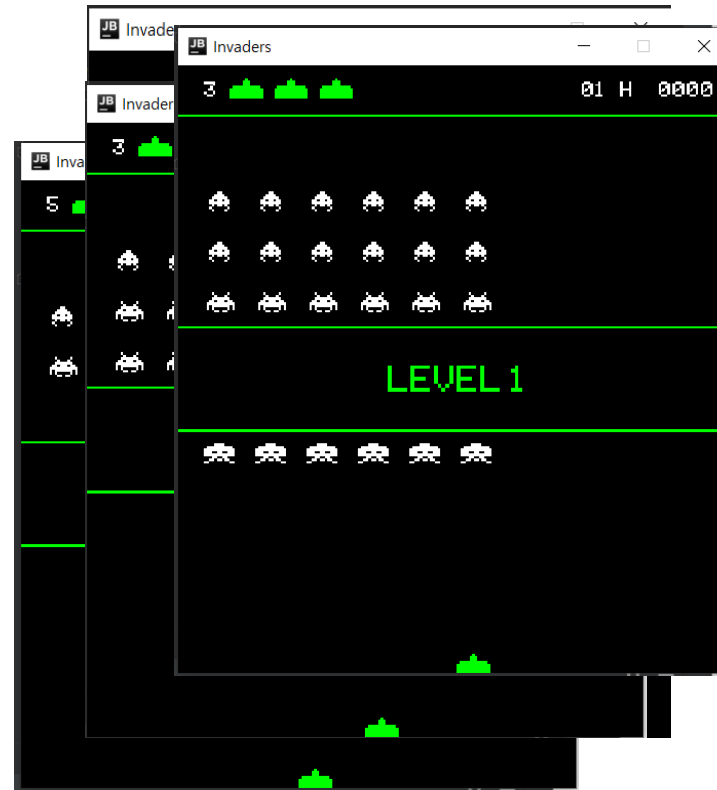
EASY, NORMAL, HARD

4. Best Practice 검증 - 요구사항 분석



PART TWO

Three Game Difficulty



Main menu

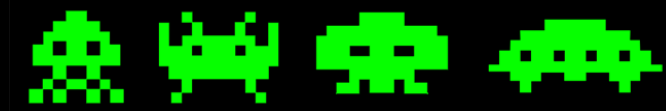
Select player



Three Levels

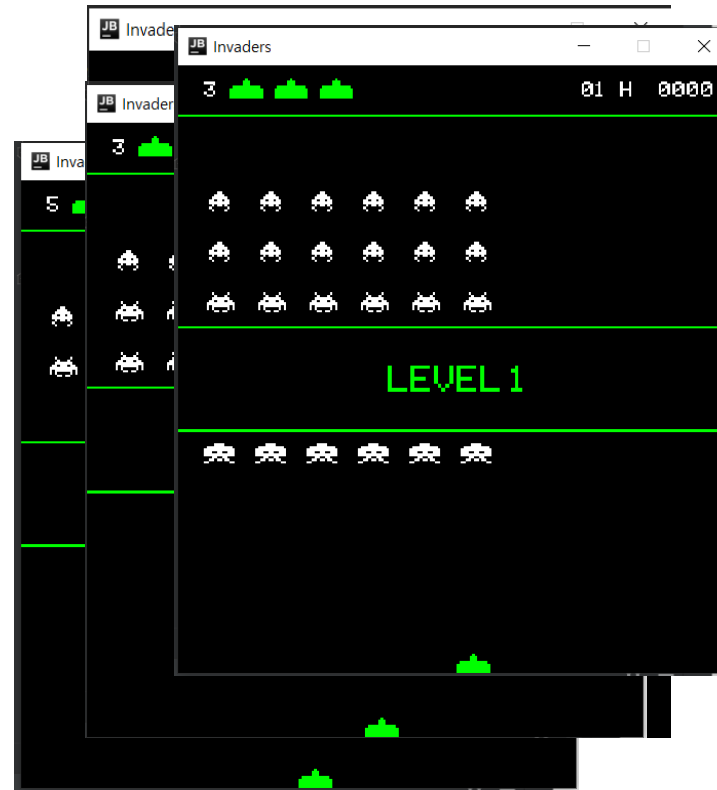
EASY, NORMAL, HARD

4. Best Practice 검증 - 요구사항 분석



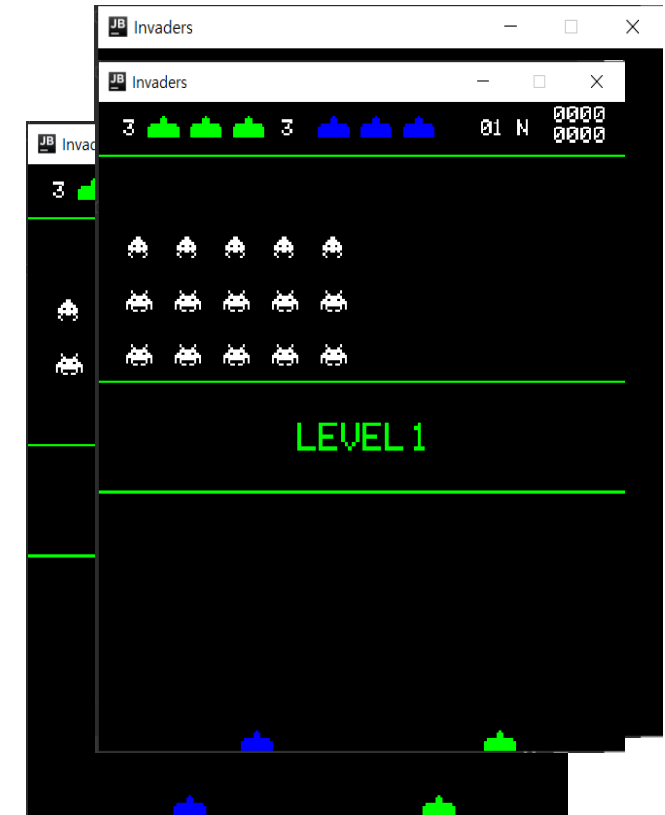
PART TWO

Three Game Difficulty



Main menu

Select player



Three Levels

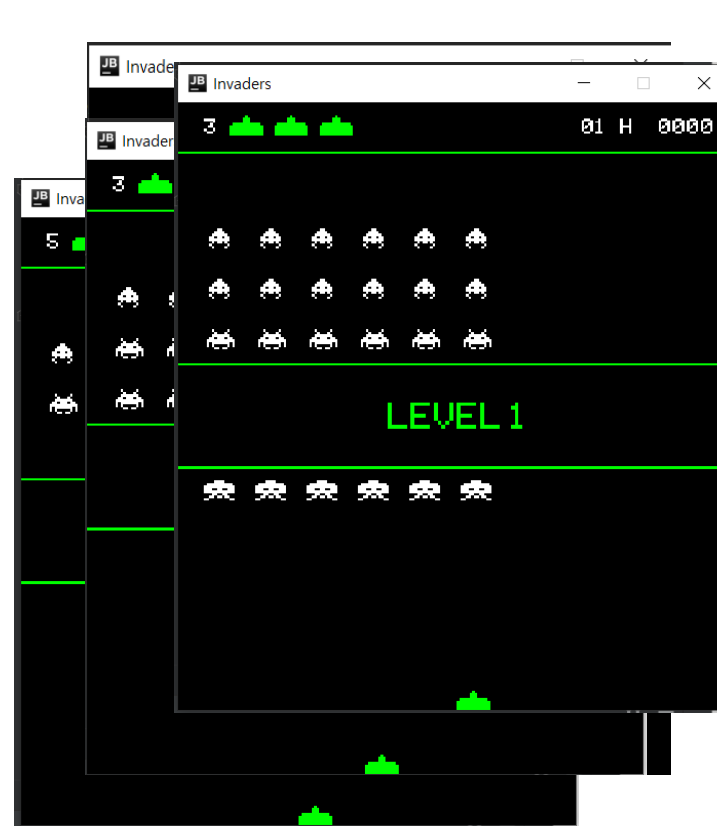
EASY, NORMAL, HARD

4. Best Practice 검증 - 요구사항 분석



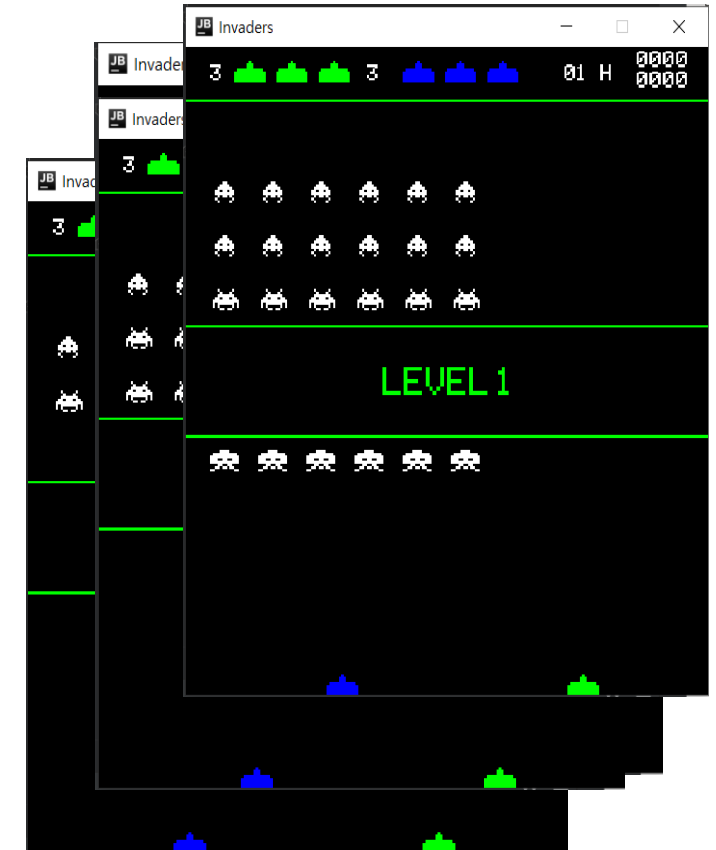
PART TWO

Three Game Difficulty



Main menu

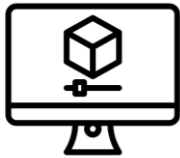
Select player



Three Levels

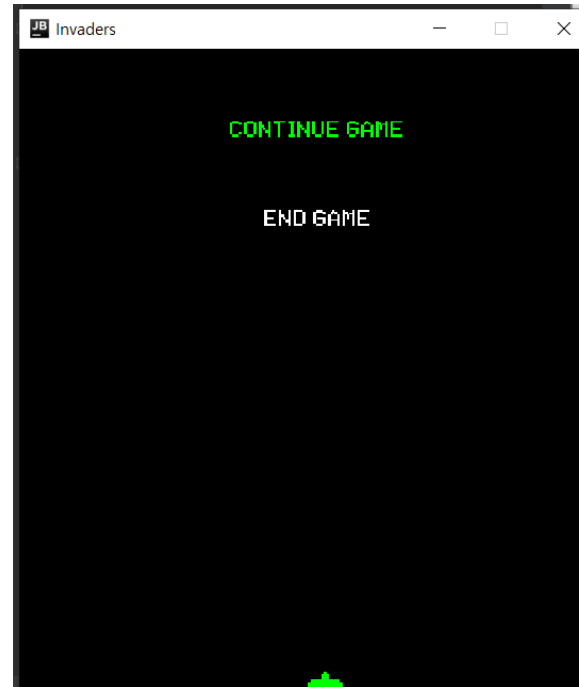
EASY, NORMAL, HARD

4. Best Practice 검증 - 요구사항 분석



PART THREE

Better Interface

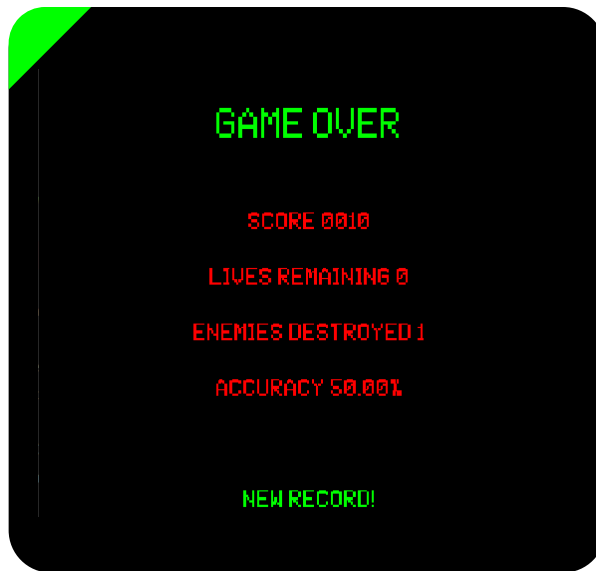
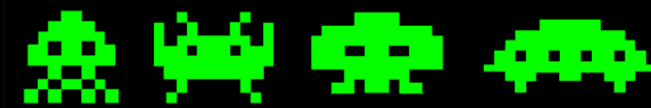


ESC key to pause game



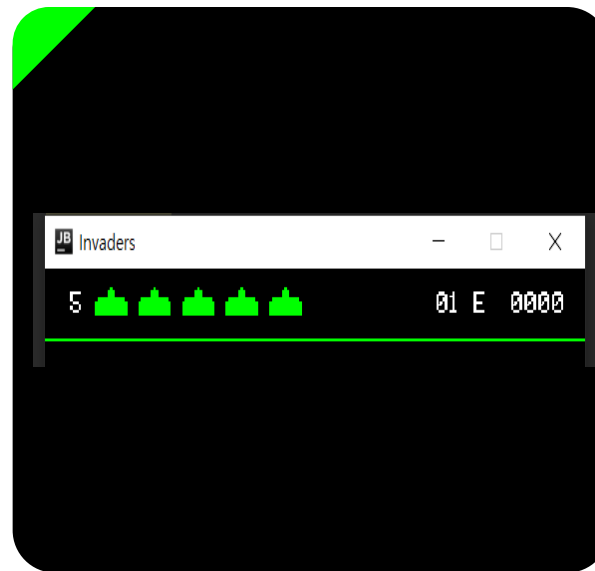
Reset Highscore

4. Best Practice 검증 - 요구사항 분석



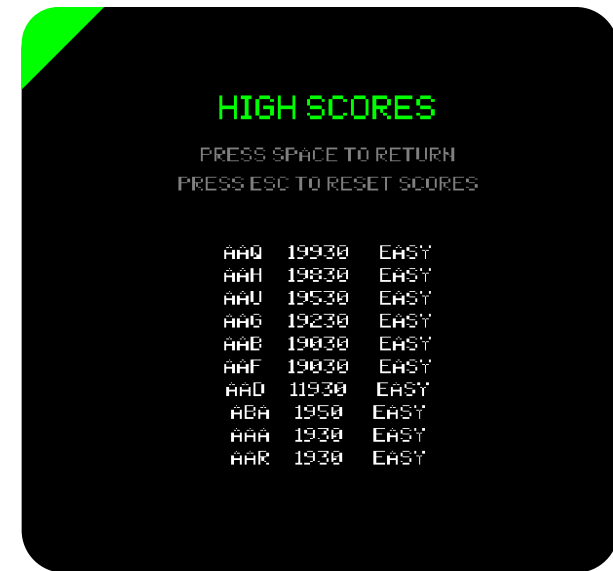
NEW HIGHSCORE == RED

최고 기록 갱신 시
빨간색으로 점수 표시



CURRENT LEVEL&DIFFICULTY

게임 화면에서
현재 레벨과 난이도 표시



MORE HIGHSCORE

최고기록 메뉴에서,
Top 10 점수 표시

4. Best Practice 검증 - 요구사항 분석



INSERT BGM

배경음악 추가로
더 즐거운 게임 가능



BACKWARD OPTION

난이도 선택에서,
뒤로가기 기능 추가

4. Best Practice 검증 – JIRA 활용



1) 칸반 보드

프로젝트 / teletubbies / TEL 보드

칸반 보드

🔗 ☆ 릴리즈 📄 ...

🔍 👤👤👤👤👤👤 내 이슈만 최근 업데이트됨

백로그 3

요구사항
📌 ↑ ●●● TEL-1

아이템 먹으면 중 레벨업
📌 ↑ ●●● TEL-13

상단에 하이스코어 표시
📌 ↑ ●●● TEL-14

개발하기로 선택됨 3

플레이어 1명 더 추가하기
2인용으로 만들기
📌 ↑ TEL-9 👤

esc키를 누르면 게임 pause 하는 기능 추가하기
인터페이스 개선하기
☑️ ↑ TEL-19 👤

점수저장 안되는 오류
오류사항
☑️ ↑ TEL-36

진행 중 3

난이도 easy, normal, hard로 분리 후 개별 설정
요구사항 2번
☑️ ↑ ●●● TEL-6 👤

플레이어 2명 색깔 다르게 설정하기
2인용으로 만들기
☑️ ↑ ●●● TEL-11 👤

게임 난이도에 따라서 적 우선선의 점수 다르게 하기
요구사항 2번
☑️ ↑ ●●● TEL-22 👤

완료 23

최고기록 깨지면 점수 빨갈게
☑️ ↑ ●● TEL-28

TEL-1 요구사항

상단에 현재 레벨 표시
📌 ↑ ●●● TEL-15

등수 10등까지 표시
📌 ↑ TEL-16

점수 창에 난이도 표시하기
📌 ↑ TEL-23

배경 음악 넣기
📌 ↑ ●● TEL-24 👤

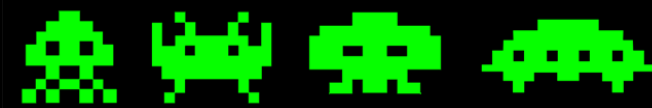
난이도 상 중 하
요구사항 2번
📌 ↑ ●● TEL-2 👤

2인용
📌 ↑ ●● TEL-3

게임 난이도 선택 창 추가하기
요구사항 2번

💡 Quickstart

4. Best Practice 검증 - JIRA 활용



2) 작업 업로드 및 담당자와 보고자 할당

TEL-9

피드백 보내기 1

플레이어 1명 더 추가하기

첨부 에픽에 이슈 생성 이슈 연결

설명

- 플레이어 1명 더 추가하기
- 플레이어 2명 색깔 다르게 설정하기
- 플레이어 각각 점수 매기기

이 에픽의 이슈

100% 완료

✓ TEL-10	플레이어 1명 더 추가하기	↑	강예원	완료됨
✓ TEL-11	플레이어 2명 색깔 다르게 설정하기	↑	강예원	완료됨
✓ TEL-12	플레이어 각각 점수 매기기	↑	한수빈	완료됨
✓ TEL-17	플레이어 인원 선택화면	↑	강예원	완료됨
✓ TEL-26	플레이어 한명 죽으면 죽은 상태 유지	↑	강예원	완료됨

완료됨 완료

담당자 강예원

보고자 한수빈

레이블 없음

우선 순위 ↑ High

Epic Name 2인용으로 만들기

필드 5개 더 보기
Story Points, 최초 추정, 시간 추적, 컴포넌트 및 수정 버전

생성됨 2020년 11월 24일 오후 3:38
업데이트됨 9시간 전

구성

4. Best Practice 검증 - JIRA 활용



3) Github 커밋과 연동

TEL-2 / ☒ TEL-6

난이도 easy, normal, hard로 분리 후 개별 설정

[첨부](#) [하위 작업 생성](#) [이슈 연결](#) [...](#)

설명

각각의 레벨의 적, 총알, 내 우주선 속도와 그리고 총알 개수 설정.

- 총 레벨 13단계로 세분화

easy 1레벨 ~ 7레벨

normal 4레벨 ~ 10레벨

hard 7레벨 ~ 13레벨

easy level - 기본 목숨 5개 + level 3씩 올라갈 때 마다 보너스

normal - 기본목숨 3개 + level 3씩 올라갈 때 마다 보너스

hard - 기본 목숨 3개 + level 2씩 올라갈 때 마다 보너스

Confluence 페이지

연급됨

[소프트웨어개발실무 프로젝트 요구사항](#)

활동

[댓글 추가...](#)

프로 팁: 눌러서 **M** 댓글 추가

피드백 보내기 2

완료됨 ☒ 완료

담당자 김예진

보고자 김예진

개발

- 1개 브랜치
- 2개 커밋 10일 전
- 1개 풀 리퀘스트 MERGED

레이블 없음

Epic Link [요구사항 2번](#)

우선 순위 ↑ Medium

☒ **필드 4개 더 보기**

최초 추정, 시간 추적, 컴포넌트 및 수정 버전

생성됨 2020년 11월 24일 오후 3:30

업데이트됨 2020년 12월 2일 오후 9:56

구성

4. Best Practice 검증 - JIRA 활용



4) 스토리에서 하위 작업 생성

TEL-1

4. 추가 요구 사항

- > 게임 시작하기 전에 마우스나 키보드 누르면 게임 시작하기
- > 아이템 떨어뜨려서 아이템 먹으면 총 레벨 업하기
- > 상단에 하이스코어와 레벨 표시하기
- > 레벨이 올라가거나 난이도 올라가면 적이 쓰는 총의 개수 늘리기
- > 등수 10등까지 표시하기
- > 점수 창에 난이도 표시하기
- > 배경 음악 넣기
- > 처음 시작 카운트 다운 때 뽀(5)뽀(4)뽀(3)뽀(2)뽀(1) 소리하게 하기

하위 작업

TEL-13	아이템 먹으면 총 레벨업	↑	BACKLOG
TEL-14	상단에 하이스코어 표시	↑	BACKLOG
TEL-15	상단에 현재 레벨 표시	↑	완료됨
TEL-16	등수 10등까지 표시	↑	완료됨
TEL-23	점수 창에 난이도 표시하기	↑	완료됨
TEL-24	배경 음악 넣기	↑	완료됨

66% 완료

피드백 보내기

2

👍

🔗

...

✕

Backlog

담당자

할당 해제됨

보고자

전민지 | ERICA 소프트웨어전공

개발

1개 커밋

17일 전

레이블

없음

우선 순위

↑ Medium

필드 6개 더 보기

Story Points, 최초 추정, 시간 추적, Epic Link, 컴포넌트 및 수정 ...

생성됨 2020년 11월 24일 오후 1:56

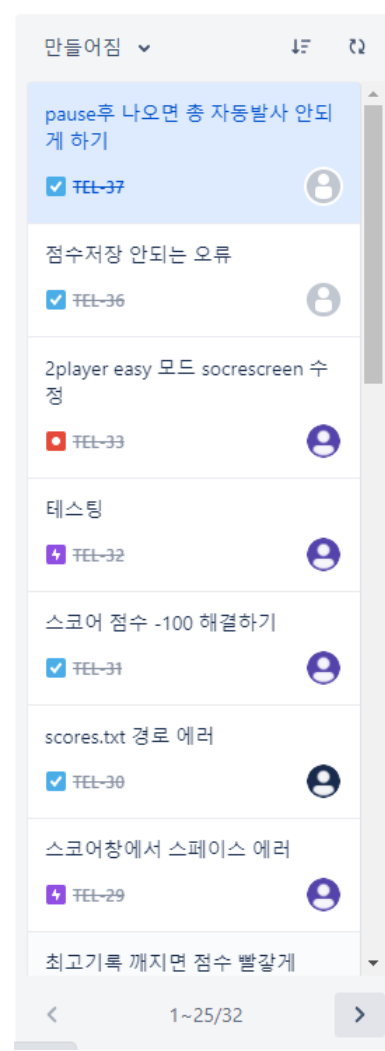
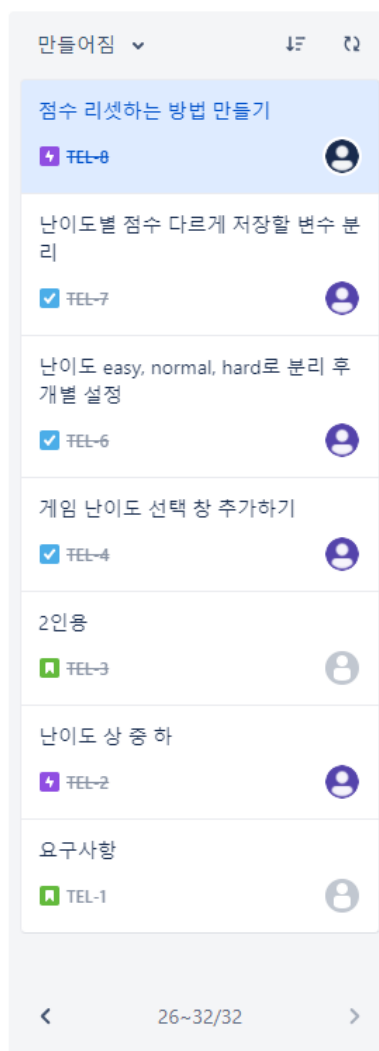
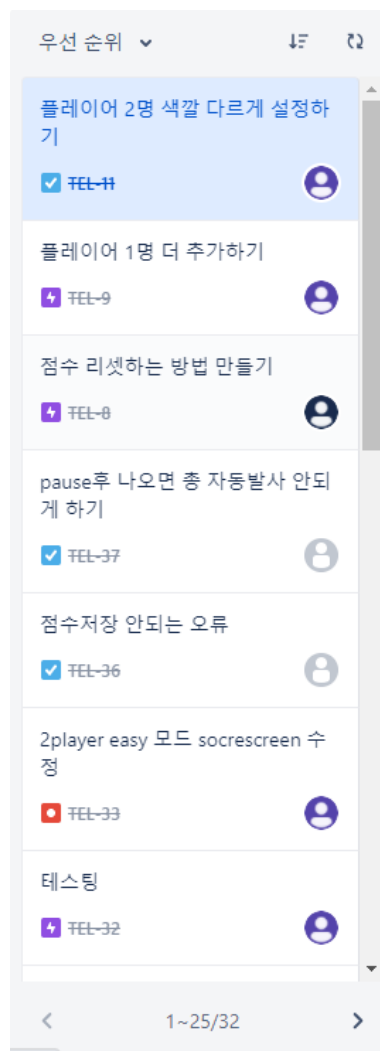
구성

업데이트됨 2020년 12월 1일 오후 11:37

4. Best Practice 검증 - JIRA 활용



5) 우선 순위



4. Best Practice 검증 – JIRA 활용



6) 보고서 작성




Requirements				
	Requirement	Details	Importance	Jira Issue
1	2인용으로 만들기	1. player1과 player2는 다른 색의 우주선을 갖는다. 2. player1과 player2의 점수를 각각 기록한다.	HIGH	<ul style="list-style-type: none"> TEL-9: 플레이어 1명 더 추가하기 완료됨 TEL-11: 플레이어 2명 색깔 다르게 설정하기 완료됨 TEL-12: 플레이어 각각 점수 매기기 완료됨 TEL-17: 플레이어 인원 선택화면 완료됨 TEL-26: 플레이어 한명 죽으면 죽은 상태 유지 완료됨 TEL-27: 난이도 선택 화면에서 뒤로가기 기능 추가 완료됨
2	게임 난이도 설정	1. Easy, Normal, Hard 3단계로 난이도를 구성한다. 2. 각각의 레벨은 적, 총알, 우주선의 속도가 다르게 한다. 3. 기존의 게임과 다른 난이도를 가지도록 한다. 4. 난이도 별로 게임 점수를 개별적으로 저장한다.	HIGH	<ul style="list-style-type: none"> TEL-2: 난이도 상 중 하 완료됨 TEL-7: 난이도별 점수 다르게 저장할 변수 분리 완료됨 TEL-6: 난이도 easy, normal, hard로 분리 후 개별 설정 완료됨 TEL-22: 게임 난이도에 따라서 적 우주선의 점수 다르게 하기 완료됨 TEL-4: 게임 난이도 선택 창 추가하기 완료됨
3	인터페이스 개선	1. ESC키를 이용하여 게임을 일시 중지하는 기능을 추가한다. 2. 점수를 리셋하는 기능을 추가한다.	HIGH	<ul style="list-style-type: none"> TEL-8: 점수 리셋하는 방법 만들기 완료됨 TEL-20: 점수 리셋창 만들기 완료됨 TEL-19: esc키를 누르면 게임 pause 하는 기능 추가하기 완료됨
4	추가 요구사항	1. 마우스나 키보드 자판을 누르면 게임을 시작하는 기능을 추가한다. 2. 게임 중 아이템을 떨어뜨려 아이템 획득 시에 총 레벨을 더해준다. 3. 게임 화면 상단에 하이스코어와 레벨을 표시한다. 4. 레벨이나 난이도가 올라갈 시에 적이 쓰는 총의 개수를 늘린다. 5. 동수를 10등까지 표시한다. 6. 배경 음악을 삽입한다. 7. 점수 창에 난이도를 표시한다.	NORMAL	<ul style="list-style-type: none"> TEL-15: 상단에 현재 레벨 표시 완료됨 TEL-24: 배경 음악 넣기 완료됨 TEL-23: 점수 창에 난이도 표시하기 완료됨 TEL-16: 동수 10등까지 표시 완료됨 TEL-14: 상단에 하이스코어 표시 완료됨

4. Best Practice 검증 - Github 활용



1) 코드 리뷰




hansususu commented 10 days ago

Collaborator

😊 ...

깔끔하게 잘 구현하셨습니다! 수고 많으셨습니다!




yeonjujeong reviewed 10 days ago

View changes

Invaders-master/src/screen/GameScreen.java

```
98 +         else if (this.difficulty.equals("H")) {
99 +             this.score += enemyShip.getPointValue() * POINT_HARD;
100 +         }
101 +         else this.score += enemyShip.getPointValue();
```




yeonjujeong 10 days ago

Collaborator

😊 ...

난이도에 따라 배를 파괴했을 때 점수까지 다 다르게 주었군요!



yeonjujeong 10 days ago

Collaborator

😊 ...

아주 세세하게 한 것 같네요!!!

4. Best Practice 검증 - Github 활용



2) 추가 사항 언급



yewonkang00 commented 10 days ago

Collaborator

😊 ...

가능하다면 'score을 reset하시겠습니까' 라고 확인창을 띄운 뒤 확인을 누르면 reset되도록 하는 것도 좋을 것 같습니다!!



yejin00 commented 9 days ago

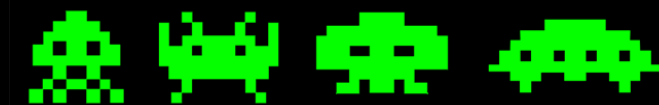
Collaborator

😊 ...

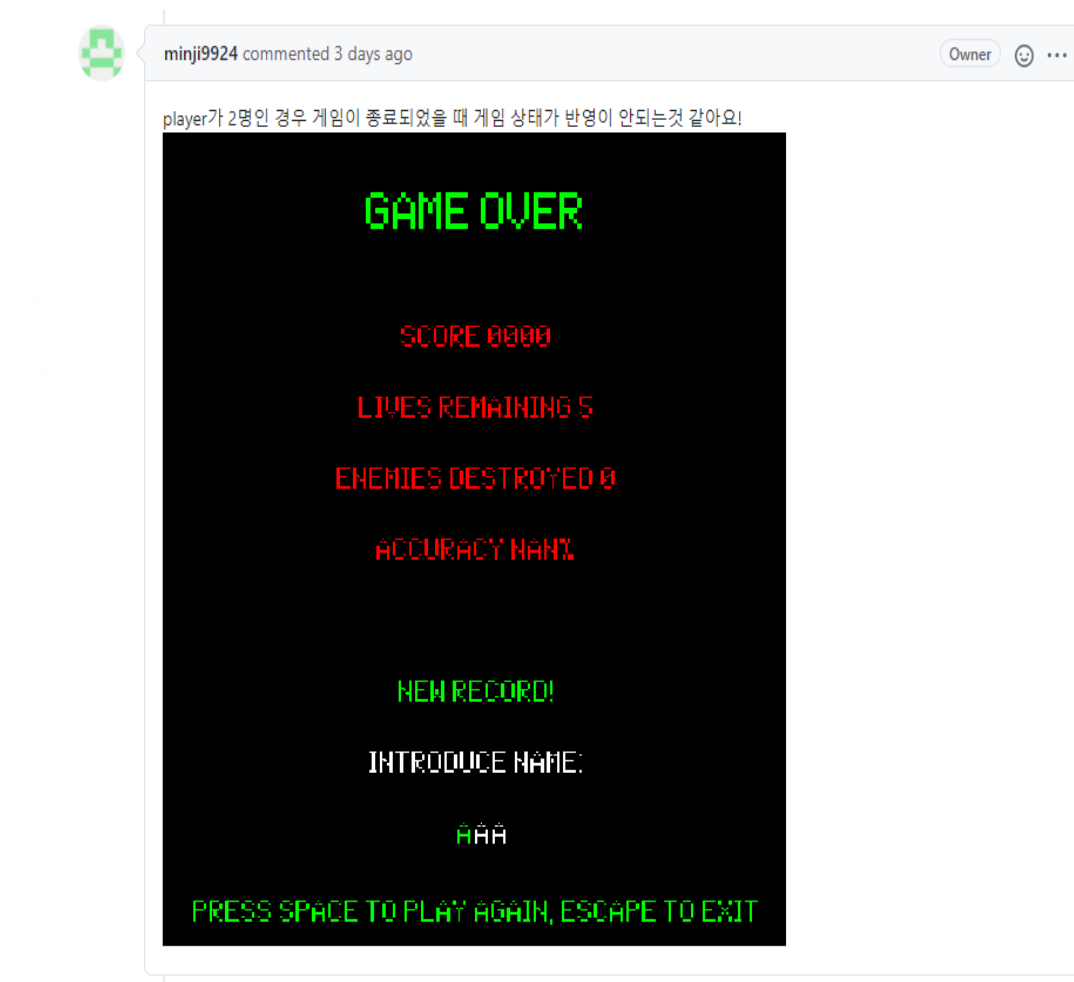
죽은 플레이어 죽은채로 할지 아니면 살릴지에 대해 다 같이 고민해보는게 좋을 것 같아요
뒤로가기 버튼 너무 좋아여 !!!

 1

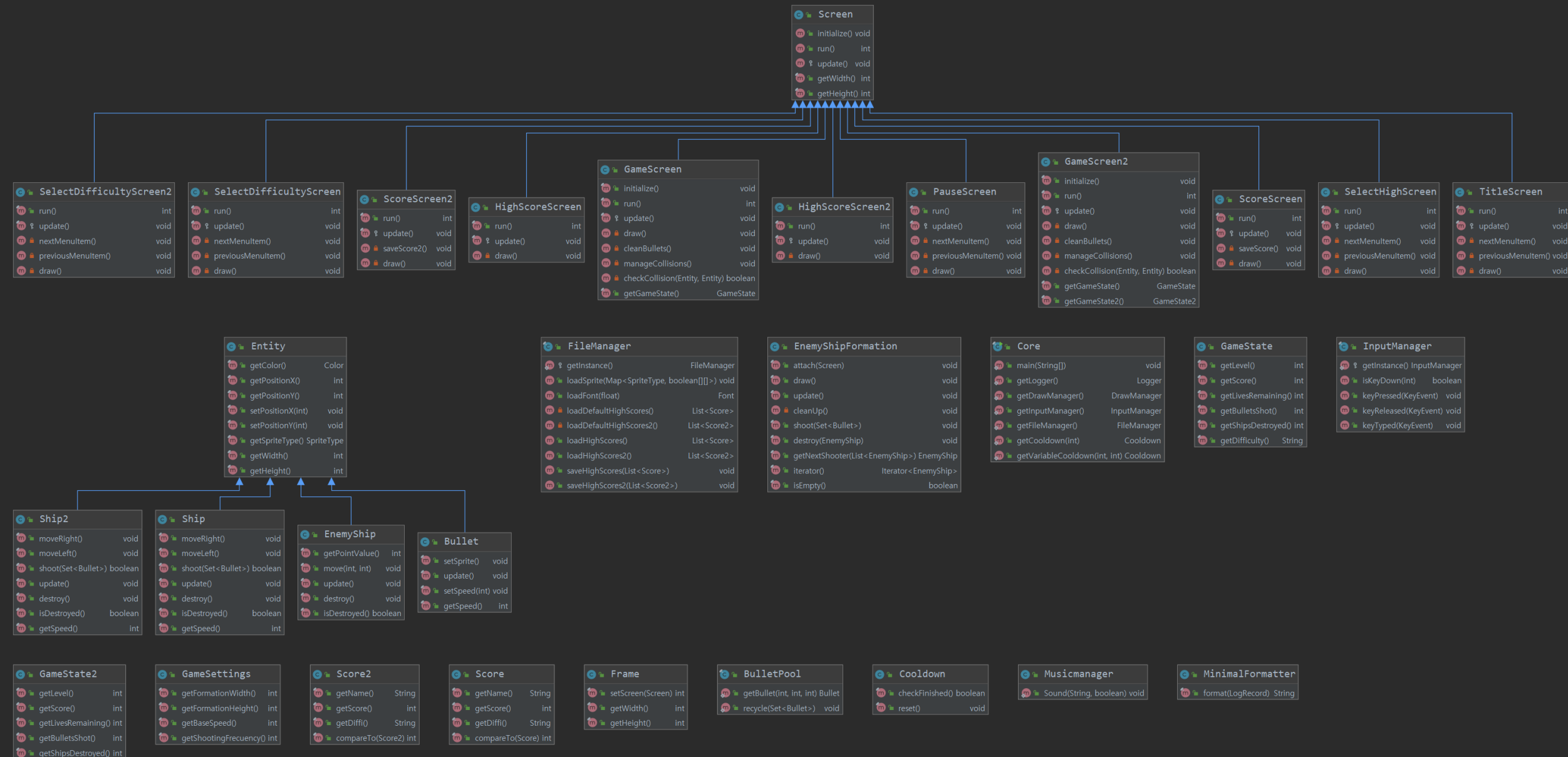
4. Best Practice 검증 - Github 활용



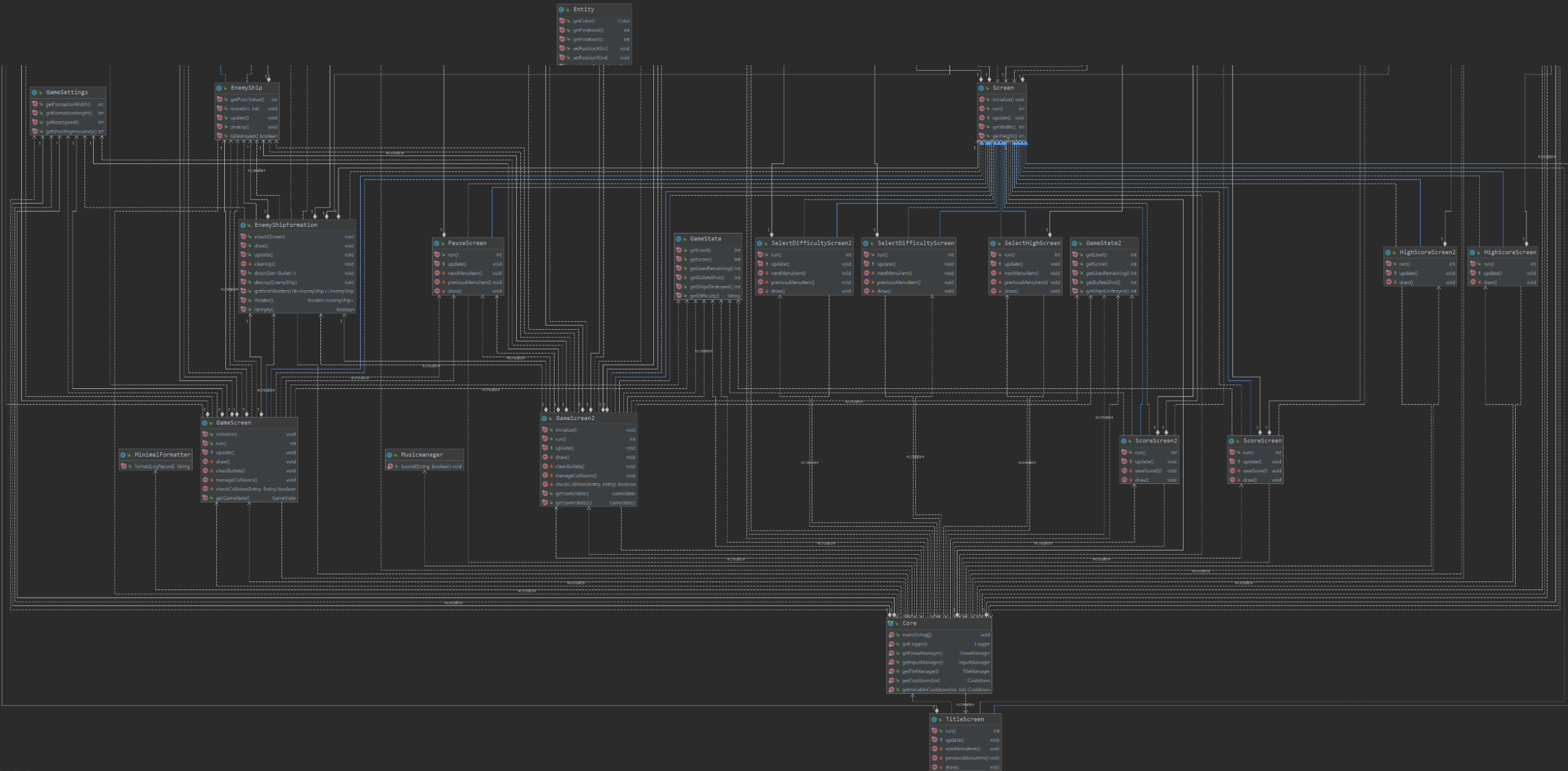
3) 오류 사항 언급



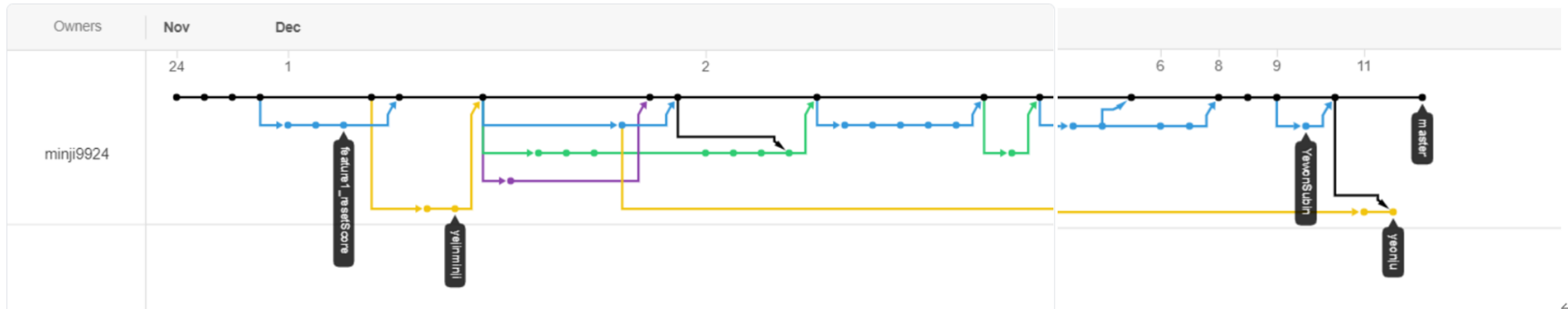
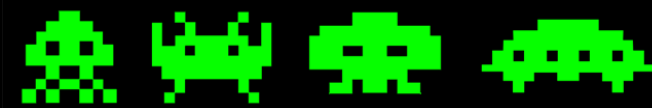
4. Best Practice 검증 - CLASS DIAGRAM



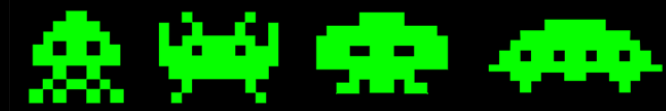
4. Best Practice 검증 - CLASS DIAGRAM



4. Best Practice 검증 - NETWORK BRANCH



4. Best Practice 검증 - Junit Test



Brief support with annotations.

```
@Test
public void testSum() {

}

@BeforeClass
public static void setUpBeforeClass() throws Exception {

}

@AfterClass
public static void tearDownAfterClass() throws Exception {

}

@Before
public void setUp() throws Exception {

}

@After
public void tearDown() throws Exception {

}
```

- assertEquals(a, b)
- assertEquals(a, b)
- assertEquals(a, b, c)
- assertSame(a, b)
- assertTrue(a)
- assertNotNull(a)

4. Best Practice 검증 - JUNIT TEST



```
GameState gameState_normal = new GameState( level: 1, difficulty: "N", score: 0, livesRemaining: 3, bulletsShot: 0, shipsDestroyed: 0);
GameState gameState_easy = new GameState( level: 1, difficulty: "E", score: 0, livesRemaining: 5, bulletsShot: 0, shipsDestroyed: 0);
GameState gameState_hard = new GameState( level: 1, difficulty: "H", score: 0, livesRemaining: 3, bulletsShot: 0, shipsDestroyed: 0);
```

@Test

```
void getDifficulty() {
    assertEquals(gameState_normal.getDifficulty(), actual: "N");
    assertEquals(gameState_easy.getDifficulty(), actual: "E");
    assertEquals(gameState_hard.getDifficulty(), actual: "H");
}
```

@Test

```
void getLevel() {
    assertEquals(gameState_normal.getLevel(), actual: 1);
    assertEquals(gameState_easy.getLevel(), actual: 1);
    assertEquals(gameState_hard.getLevel(), actual: 1);
}
```

@Test

```
void getScore() {
    assertEquals(gameState_normal.getScore(), actual: 0);
    assertEquals(gameState_easy.getScore(), actual: 0);
    assertEquals(gameState_hard.getScore(), actual: 0);
}
```



빌드 #4 (2020. 12. 15. 오전 10:27:04)



Changes

1. Update 요구사항.txt ([commit: dcf9a93](#)) ([details](#) / [githubweb](#))



Revision: dcf9a93facb9205df64e332176096592edaa4a93

- refs/remotes/origin/master

4. Best Practice 검증 - 결과



Agile - Kanban

개발 프로세스 모델 미적용과의 차이 -> 효율적 개발, 향상된 협업

칸반 채택 -> 요구사항 파악 용이, 짧은 기간에 적합

JUnit

게임 진행 중, 새롭게 갱신되는 것을 테스트하지 못함

모든 것을 테스트하지 못함

-> 다른 테스트 방식 채택

4. Best Practice 검증 - 결과



Jira

담당자와 보고자를 2명 이상 배정할 수 없다.

Github commit과 연동하기 어렵다.

-> Github commit tag에 issue code를 포함해야 한다.

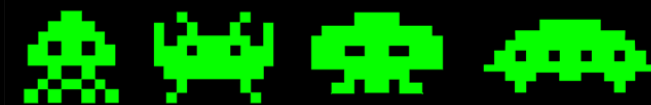
Github

사소한 문제로 충돌이 쉽게 일어난다.

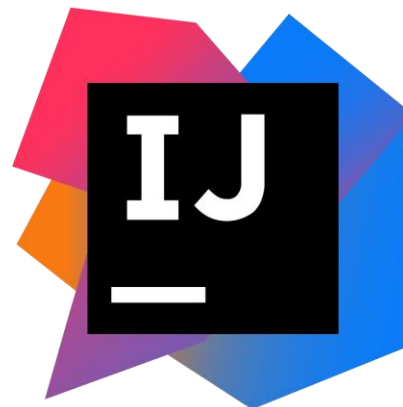
매우 많은 기능과 도구

-> 초기에 적응하기 어려움

4. Best Practice 검증 - 결과



VS



Project Structure 설정이 다름.
파일 경로 설정이 다름.
-> 같은 IDE를 사용

4. Best Practice 검증 - 결과

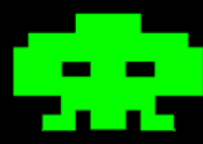


Jenkins

환경 설정(서버, 연동)이 어렵다.

CI / CD를 통해 버그를 더욱 빠르게 자주 수정할 수 있다는 것을 알게 되었다.

Q&A



THANK YOU!

