

# 소프트웨어 V&V

CTIP

201511243 김동언

201511246 김상재

201511262 박우진

201711356 천세진

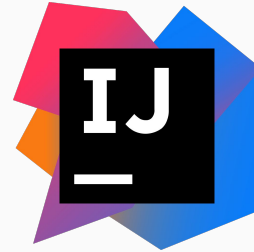
# Index

1. IDE (IntelliJ, Eclipse)
2. CI server (Jenkins, Github Actions)
3. Code Configuration Management (SVN, Git)
4. Requirements Management & Bug tracking (Github Project, Trello)
5. Unit Testing (JUnit 5)
6. Automatic Build (Maven, Gradle)
7. Team Communication (Discord, Slack)

# IDE (Eclipse VS IntelliJ)



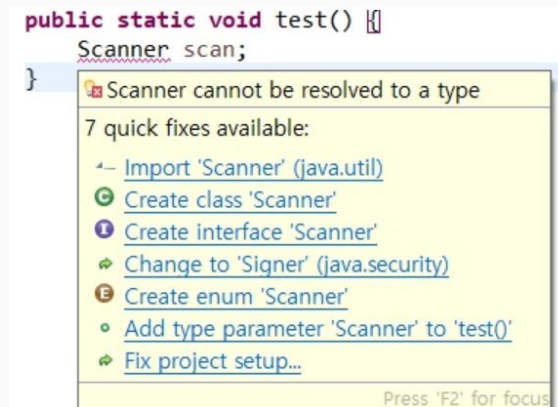
vs



# IDE - Eclipse

- Eclipse

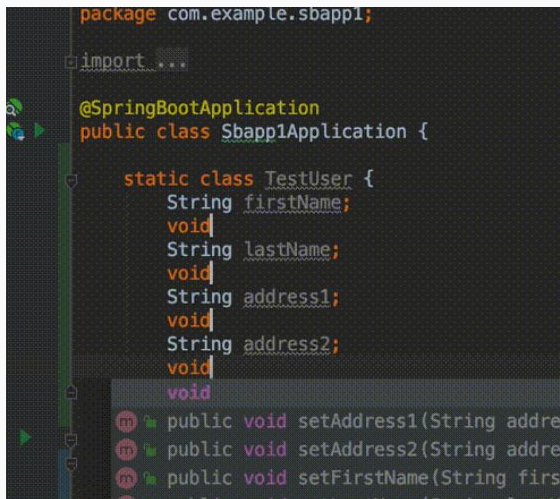
- 역사가 오래된(2004년 출시) IDE로 많은 개발자 커뮤니티, 잘 작성된 문서, 수 많은 플러그인들이 존재
- 웹, 모바일, 데스크탑, 엔터프라이즈, 임베디드 시스템 어플리케이션 개발에 사용
- 자바(JAVA)로 개발되어 있어 윈도우즈, 리눅스, 맥OS 등 대부분의 OS 지원
- 무료 라이선스
- 자바가상머신(JVM) 위에서 실행되므로 속도가 느림



# IDE - IntelliJ IDEA

- IntelliJ IDEA

- 자동완성(auto completion), 소스코드분석, 지능형리팩토링, 디버거, 테스트러너 기능이 비교된 IDE 중에서 가장 우수
- 편리한 기능 (스마트 코딩, 괄호 닫을 시 자동 정렬 등) 지원
- Git과의 연동이 편리함
- eclipse보다 상대적으로 프로그램이 가벼움
- 유료 라이선스지만 학생 라이선스 발급 => 무료로 사용 가능



```
package com.example.sbapp1;

import ...

@SpringBootApplication
public class Sbapp1Application {

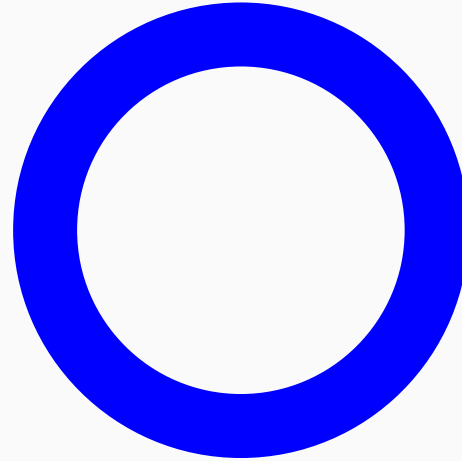
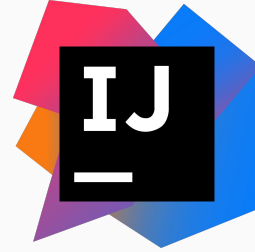
    static class TestUser {
        String firstName;
        void
        String lastName;
        void
        String address1;
        void
        String address2;
        void
        void
        public void setAddress1(String address)
        public void setAddress2(String address)
        public void setFirstName(String firstName)
```

intelliJ 스마트 코딩

# IDE (Eclipse VS IntelliJ)

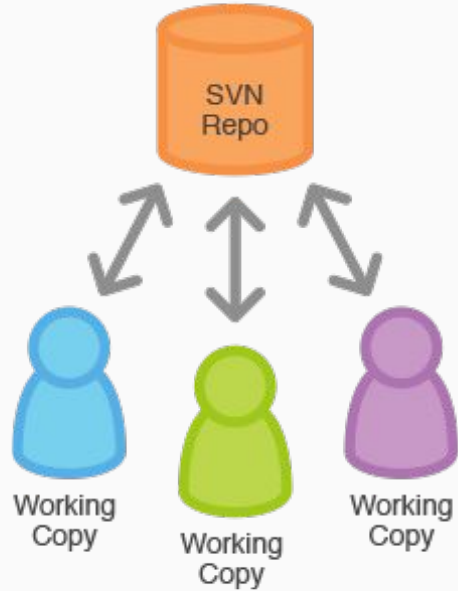


vs



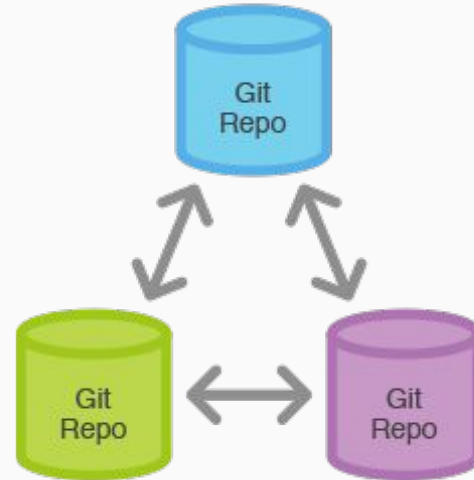
# Code Configuration Management

Central-Repo-to-Working-Copy  
Collaboration



SVN

Repo-to-Repo  
Collaboration



Git

# CI Server



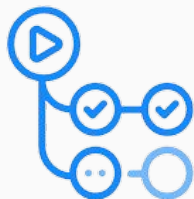
Travis CI

 **TeamCity**



**CODESHIP**

by CloudBees



GitHub Actions



GitLab

 **circleci**



**Jenkins**

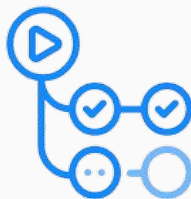




## Jenkins

- 2005년 2월 java.net에 출시한 CI 툴
- 수많은 플러그인들이 존재하며 다양한 플랫폼들과 연동 가능
  - Github, Sonarqube, Discord, Slack 등
- 로컬 서버 혹은 클라우드 서버 구축 후 사용 가능
- Pipeline을 통한 작업 자동화 가능

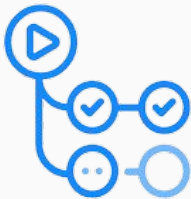
# CI Server - Github Actions



## GitHub Actions

- 2019년 11월에 출시한 workflows(작업흐름)들을 자동화해주는 기능
  - 각 프로젝트마다 workflows를 생성하여 소프트웨어 개발에 활용이 가능
- Continuous Integration, Continuous Deployment 이외의 추가 기능
  - 다양한 Testing 과 개성있는 workflow를 제작 가능
- 가상환경(컨테이너)에서 실행
  - 사용자가 따로 서버 세팅을 하지 않고 사용 가능
- github에서 제공하는 서버인 만큼 사용자가 많기 때문에 일정 제한이 존재

# CI Server - Jenkins vs Github Actions



GitHub Actions

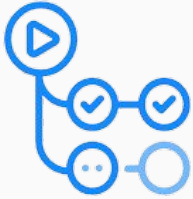
VS



Jenkins

서버 구축 <b>불필요</b>	서버 구축 <b>필요</b>
workflow로 작업 <b>자동화 가능</b>	pipeline로 작업 <b>자동화 가능</b>
가상머신(컨테이너) 환경에서 작동하므로 작업 완료 후 로그 접근 <b>불가</b>	로컬 혹은 클라우드 서버에서 작동하므로 작업 완료 후 로그 접근 <b>가능</b>

# CI Server - Jenkins vs Github Actions



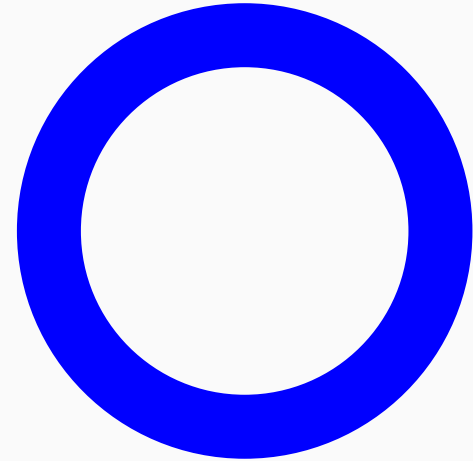
GitHub Actions



**VS**



**Jenkins**



# Automatic Build

**Maven™** vs **Gradle**



# Automatic Build



pom.xml을 이용한 정형화된 빌드 시스템

Maven을 사용할 수 있는 **변환 가능** 컨벤션 프레임 워크

뛰어난 프로젝트 정보 제공

**멀티 프로젝트**에 사용하기 좋음

2004년에 Ant를 개선하여 출시

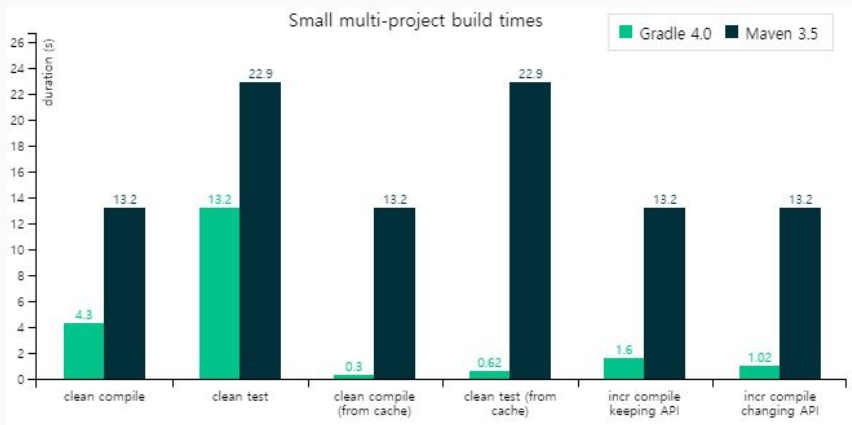
Apache Ivy에 기반한 강력한 의존성 관리

개발 가이드 라인 제공

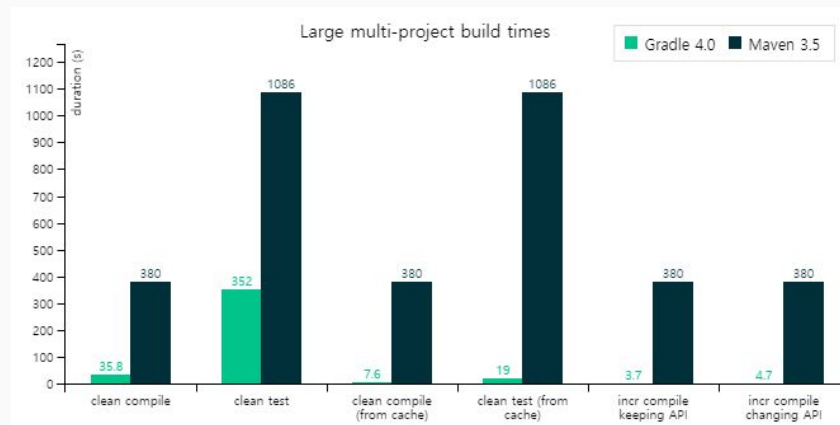
원격 저장소나 pom, ivy **파일 없이 연결되는** 의존성 관리 지원

# Automatic Build

- 실행 시간 비교



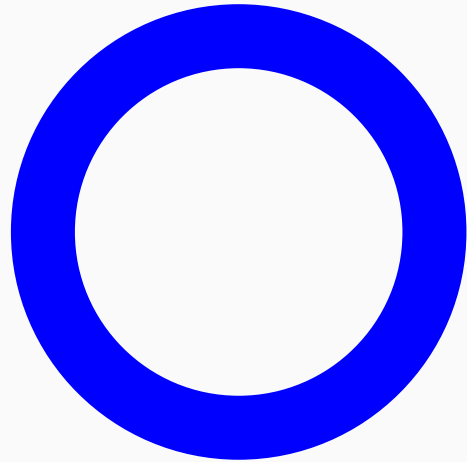
Small multi project build time



Large multi project build time

# Automatic Build

**Maven™** vs **Gradle**





# Unit Testing - JUnit 5



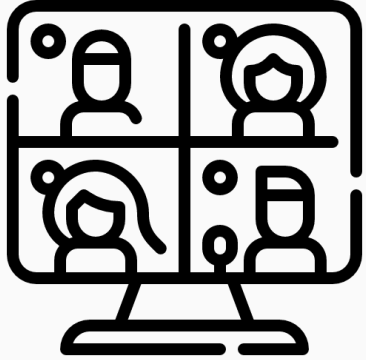
- Java 전용
- 자바용 Unit test Framework
- 컴파일 타임에 JAR로써 존재
- 가장 많이 사용

```
@Test
Run Test | Debug Test
public void requestStartStopwatch() {
    Stopwatch stp = new Stopwatch();
    stp.requestStartStopwatch(); // [st
    assertEquals(1, stp.getStatus()); //
}

@Test
Run Test | Debug Test
public void requestStopStopwatch() {
    Stopwatch stp = new Stopwatch();
    stp.requestStartStopwatch(); // [st
    stp.requestStopStopwatch(); // [sta
    assertEquals(0, stp.getStatus()); //
}
```

JUnit 5 Annotation Example

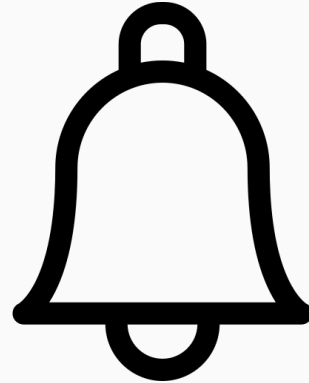
# Team Communication



소통



화면공유



알림



비용

# Team Communication



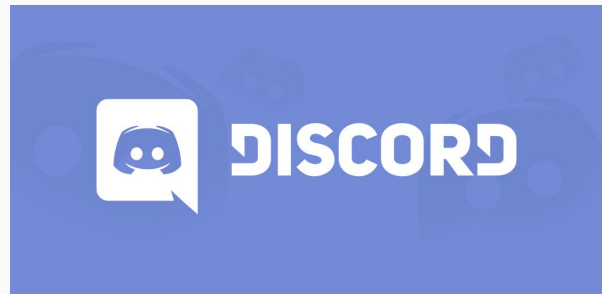
**slack** vs



**DISCORD**



# Team Communication



채팅, 음성/영상 통화	유료 플랜 가능	가능
화면 공유	유료 플랜 가능	무료 가능
Jenkins, Github 등의 알림	가능	가능
비용 여부	특정 기능은 유료가 필수	무료로 기능 사용 가능

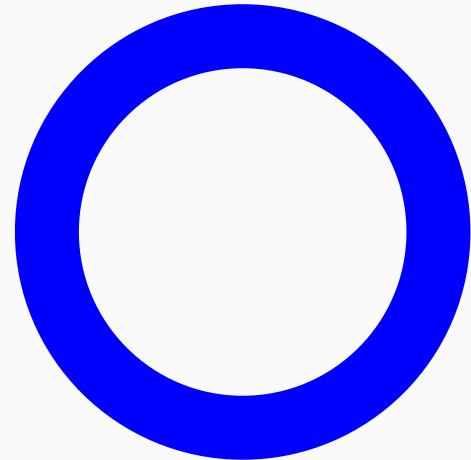
# Team Communication



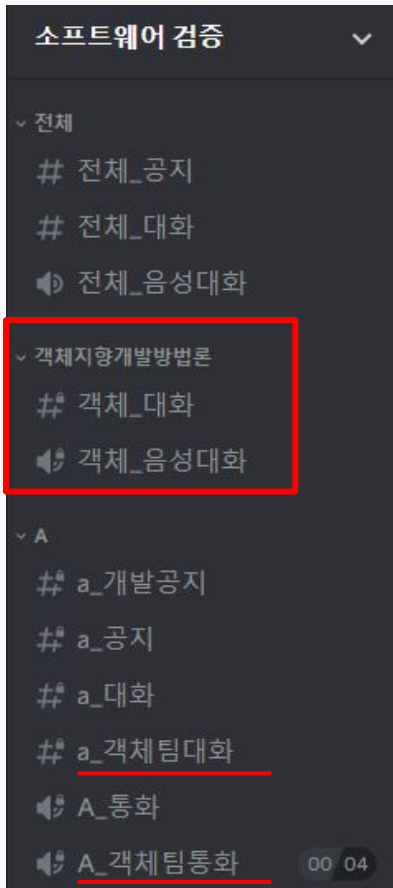
**slack** vs



**DISCORD**



# Team Communication

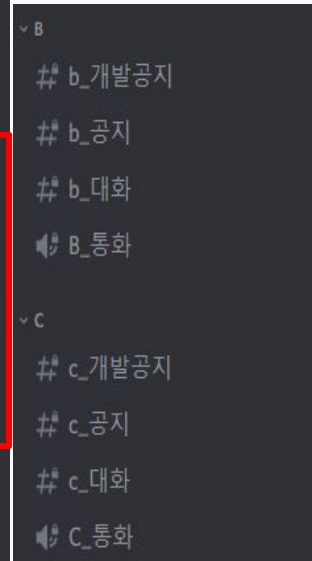


## <- 객체지향개발방법론

1. 검증팀 <-> 객체팀 간의 소통을 위한 전체 방
2. 객체팀들끼리의 대화방(검증팀 접근 **불가**)
3. 각 팀별로 대화
4. 객체팀 내부를 위한 방(검증팀 접근 **불가**)

## 소프트웨어 V&V ->

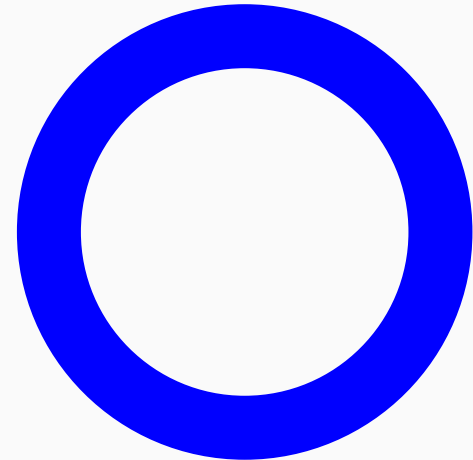
1. 검증팀 <-> 객체팀 간의 소통을 위한 전체 방
2. 검증팀 대화방 (객체팀 접근 **불가**)
3. 각 팀별로 대화
  - 모든 객체팀을 평가하기 위해 각 객체팀의 대화방에 접근은 가능해야 하지만, 객체팀의 독립적인 소통을 위하여 권한 설정을 통해 특정 채널은 **접근 불가**하게 설정



# Requirements Management & Bug Tracking



VS



# Requirements Management & Bug Tracking

8 labels	
<b>Todo</b>	수정해야하는 이슈
<b>Refuse</b>	해결하지 않을 이슈
<b>Middle Priority</b>	보통 우선순위 처리
<b>Low Priority</b>	낮은 우선순위 처리
<b>High Priority</b>	높은 우선순위 처리
<b>Build Success</b>	Build Success
<b>Build Failure</b>	Build Failure
<b>Accept</b>	해결할 이슈

**Bug Tracking**  
Updated 27 seconds ago

- 1. Todo**
  - B1 : 모든 길이 이상한 길이로 바뀌고 원하는 시간 변경이 불가능해진다. #8 opened by wjp103. **Todo**
- 2. Refuse**
  - B11 : d-day에서는 잔여일이 잘 나타나지만 %에서는 'done'으로 표시된다. #7 opened by wjp103. **Refuse**
  - Bug 02 : Call the method Thread.start() to execute the content of the run() method in a dedicated thread. #9 opened by wjp103. **Refuse**
- 3. Accept**
  - Critical 48 : The Cyclomatic Complexity of this method "iconical" is 18 which is greater than 10 authorized. #11 opened by wjp103. **Accept Low Priority**
- 3. In progress**
  - B4 : 타이머가 0이되어 알람이 울리고, 알람을 멈추면 상당한 타이머 시간으로 다시 재워진다는 내용이 스펙에 나와있지 않음(스펙추가 필요). #14 opened by wjp103. **Accept High Priority**
  - Critical 50 : Define a constant instead of duplicating this literal "000000" 3 times. #10 opened by wjp103. **Accept Middle Priority**
  - Critical 42 : Define a constant instead of duplicating this literal "zzzzzzzz" 4 times. #13 opened by wjp103. **Accept Low Priority**
- 3. Done**
  - Critical 03 : Refactor this code to not nest more than 3 if/for/while/switch/try statements. #16 opened by wjp103. **Accept High Priority**
  - Bug 10 : Logic.WatchSystem.digitalIdeal(Object) concatenates strings using + in a loop. #15 opened by wjp103. **Accept High Priority**
  - Critical 05 : Refactor this method to reduce its Cognitive Complexity from 53 to the 15 allowed. #12 opened by wjp103. **Accept High Priority**

**Requirements**  
Updated 4 minutes ago

- 1. Todo**
  - Time Keeping @3. #13 opened by wjp103. **Todo**
- 1. Refuse**
  - Time Keeping @2. #4 opened by wjp103. **Refuse**
- 3. Accept**
  - Time Keeping @2. #9 opened by wjp103. **Accept Low Priority**
  - Stopwatch @2. #11 opened by wjp103. **Accept Low Priority**
  - Timer @3. #8 opened by wjp103. **Accept Low Priority**
- 3. In progress**
  - Time Keeping @1. #12 opened by wjp103. **Accept High Priority**
  - Timer @2. #7 opened by wjp103. **Accept High Priority**
  - Stopwatch @1. #10 opened by wjp103. **Accept Middle Priority**
- 2. Done**
  - Timer @1. #6 opened by wjp103. **Accept High Priority**
  - Time Keeping @1. #5 opened by wjp103. **Accept High Priority**



# CTIP 구축 Overview

