정적 분석 실습

1. 소 개

정적 분석은 프로그램을 실제로 실행하지 않고 소스코드 또는 객체코드를 검사하는 프로그램 분석이다. 정적 분석을 하기 위한 많은 상용 및 오픈 소스 정적분석 도구가 있다. 이번 실습에서는 그 중 하나인 SpotBugs를 사용하여 제공하는 Java 프로젝트에서 버그를 찾아볼 것이다. 이번 실습의 목적은 SpotBugs와 같은 도구가존재한다는 것을 알고, 이를 사용하는 방법과 보고된 문제를 해석하는 방법에 대해 실제 경험을 얻고 코드 품질 향상에 도움이 될 수 있도록 하는 것이다.

2. 분석 도구

이번 실습에 사용할 툴은 SpotBugs 버전 3.1.5 이다. Eclipse 에서 Java 프로젝트와 함께 작업할 예정이므로 Eclipse 플러그인을 사용할 것이다. SpotBugs 는 Java 바이트 코드를 분석하고 버그 패턴의 개념을 바탕으로 광범위한 문제를 탐지하는 정적 코드 분석 툴이다. SpotBugs 는 실행 환경으로 최소 Java 7 이 필요하다.

3. 테스트 대상(SUT):

이번 실습에서 테스트할 프로젝트는 HospitalSystem 1.0 이다. 환자와 의사를 체크인/체크아웃 하고 환자를 적절한 의사에게 보내기 위한 간단한 Java 프로젝트이다. 아래의 "HospitalSystem Overview and Tool Setup"에서 자세한 명세를 확인할 것

과제 1: 수동 코드 검사

✓ "First Task"를 살펴보고 5 분 동안 가능한 많은 오류를 찾으세요

과제 2: 도구 Setup

✓ SpotBugs 플러그인 설치
✓ HospitalSystem 을 가져오고 SpotBugs 를 실행하시오.
✓ "HospitalSystem Overview and Tool Setup"에서 지침을 확인할 것.

과제 3: 이슈 분석

✓ SpotBugs 의 이슈의 의미와 분석, 방법을 이해하려면 "Analyzing an issue"참조.

First Task

```
import java.util.ArrayList;
public class User{
	private int id;
	private String firstName;
	private String lastName;
	private int age;
	private String profession;
	private ArrayList<User> children;

	public User(int id,String firstName,String lastName,int age){
		this.id = id;
		this.firstName = firstName;
		this.lastName = lastName;
		this.age = age;
		this.profession = profession;
	 }
	 public String getChildName(Userchild){
		if(!this.children.contains(child)){
		    new Exception("Invalidargument!");
	 }
	 else{
		String name = null;
		if(child.getFirstName()!=null){
		    name = child.getFirstName();
		}

		if(name =="Harry"){
		     name.replace('r','p');
		}

		if(name != null || name.length() > 0){
		      name.concat(child.getLastName());
		}
	    }
	    return this.getChildName(child);
	 }
	 public String getFirstName(){
		return firstName;
	 }
	 public String getLastName(){
		return lastName;
	 }
}
```

HospitalSystem Overview and Tool Setup

시스템 개요: Hospital System은 환자 수납과 퇴원, 의사의 출근과 퇴근을 관리하는 간단한 Java 프로그램이다. 이 프로젝트는 이번 실습을 위해 만들어졌으므로, 완전한 시스템으로 생각하면 안된다.

시스템 기능:

- ✓ 의사를 추가할 때 방이 배정되는데, 의사가 좋아하는 방이 있다면 그 방에 자동으로 지정되고 방이 가득 차면 첫 번째 빈방에 배정된다.
- ✓ 환자가 추가될 때는 원하는 전문분야의 담당의사에게 배정된다. 그 의사가 진료하지 않는다면 환자는 일반 의사에게 배정된다. 일반 의사가 진료하지 않는다면 가장 짧은 대기자 명단을 가진 의사에게 배정된다.
- ✓ 의사가 체크아웃(퇴근)할 때는 방을 비우고 모든 환자는 가장 짧은 대기열의 의사 대기자 명단에 추가된다.
- ✓ 환자 체크아웃
- ✓ 체크인 된 모든 환자와 의사는 Overview 탭에서 확인할 수 있다.
- ✓ 병원의 모든 객실 개요는 Running rooms 탭에서 확인할 수 있다.

Eclipse Workspace에 HospitalSystem 추가

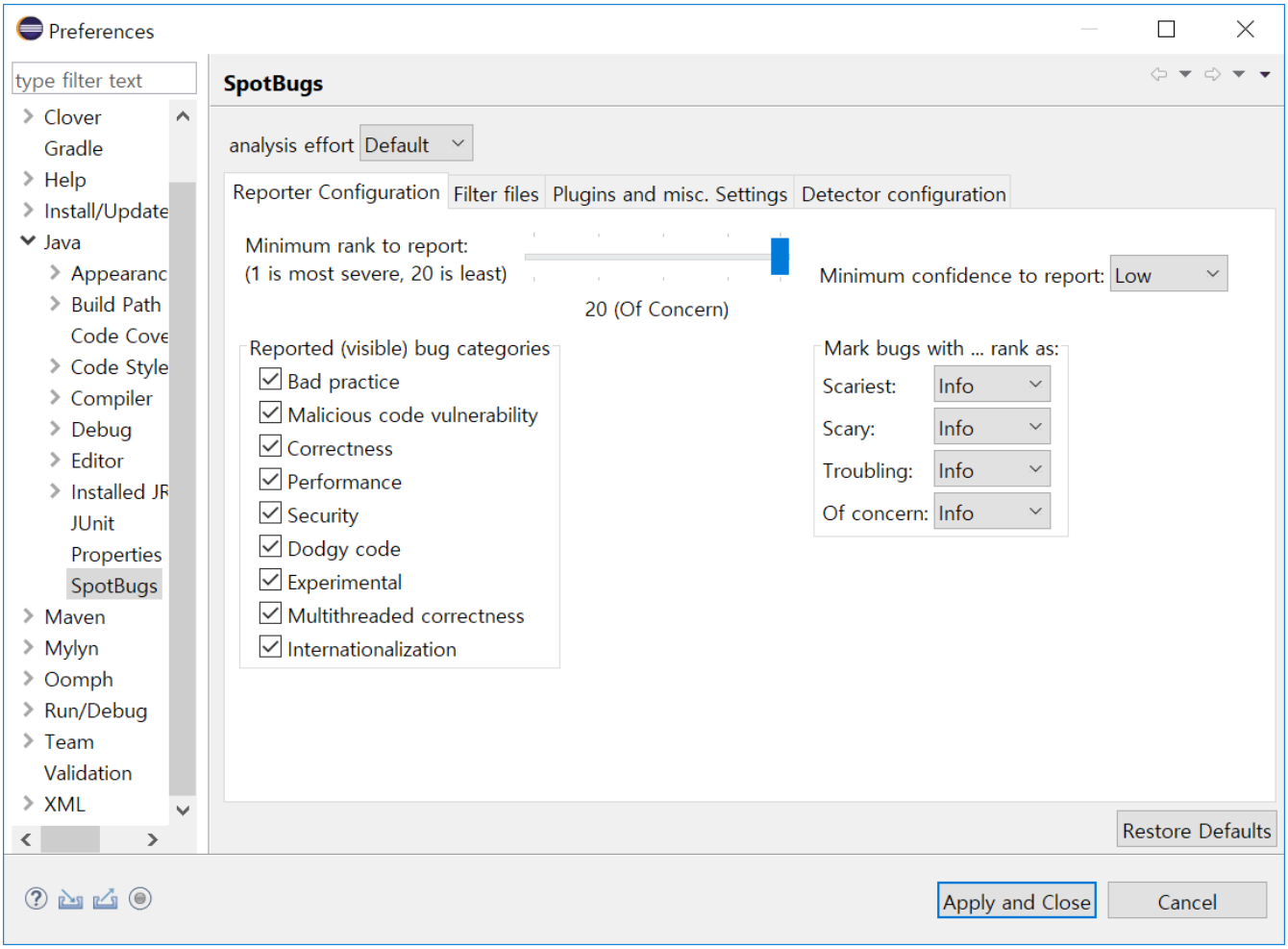
1. Eclipse 실행.
2. "HospitalSystem.zip" 압축 해제
3. Eclipse에서 "File > Import" 클릭
4. "Gradle > Existing Gradle Projects" 선택하고 "Next" 클릭
5. "Browse" 클릭 "HospitalSystem" 압축 해제한 위치 선택
6. "HospitalSystem"이 "Projects" 박스에 보여야한다, 선택한 후 "Finish" 클릭

Eclipse에 SpotBugs 추가.

1. Eclipse에서 "Help > Eclipse Marketplace" 클릭
2. "Find" 입력창에 "SpotBugs" 입력
3. 검색 결과에서 "SpotBugs Eclipse Plugin 3.1.5" 찾아보기
4. "Install > Confirm > I accept.. > Finish > OK" 선택
5. Eclipse 재시작
6. "Windows > Preferences > Java > SpotBugs" 클릭
7. "Minimum rank to report" 20으로 설정
8. "Minimum confidence to report"를 "Low"로 설정

9. 모든 카테고리에서 "Reported (visible)" bug categories"를 true로 변경
10. 모든 옵션을 "Mark bugs with... rank as"에서 "info"로 변경

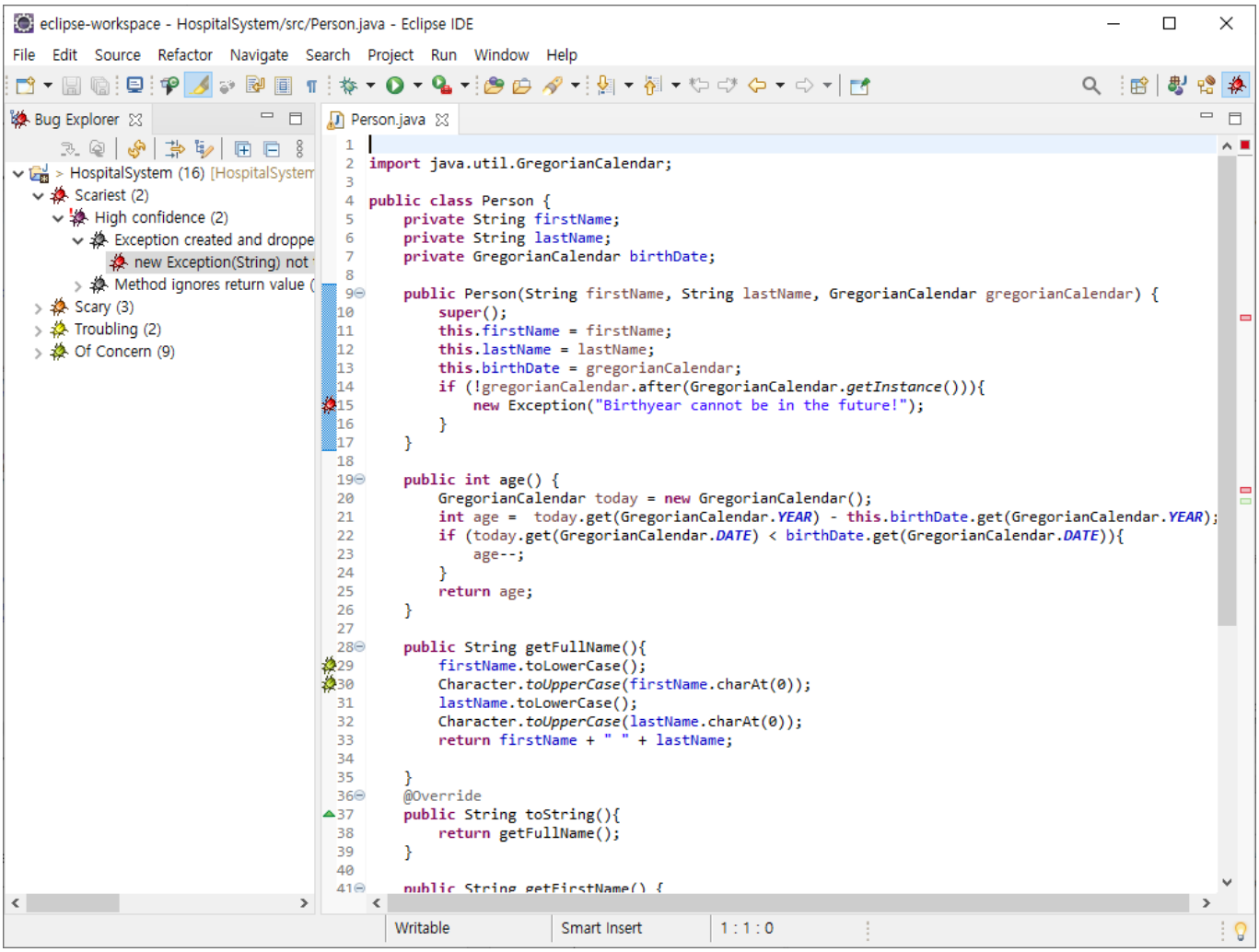
모든 단계 이후 설정 창은 캡쳐와 같아야 한다.



11. "Apply and Close" 또는 "OK" 클릭

SpotBugs 실행

1. Package explorer에서 "HospitalSystem" 오른쪽 클릭.
2. "SpotBugs > Find Bugs" 클릭
3. 버그 탐색이 끝날 때까지 대기.
4. 오른쪽 상단 모서리에 "Open Perspective > SpotBugs > OK" 를 클릭



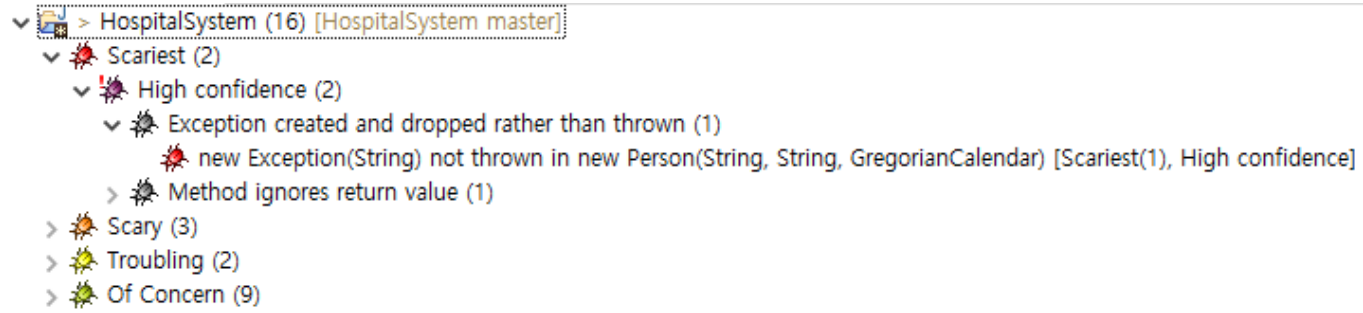
버그 탐색

SpotBugs Perspective에서 SpotBugs로 발견한 버그에 대한 정보를 볼 수 있다. 버그 목록을 보려면 "Bug Explorer" 창의 "Hospital System" 옆에 있는 작은 화살표를 클릭하세요.

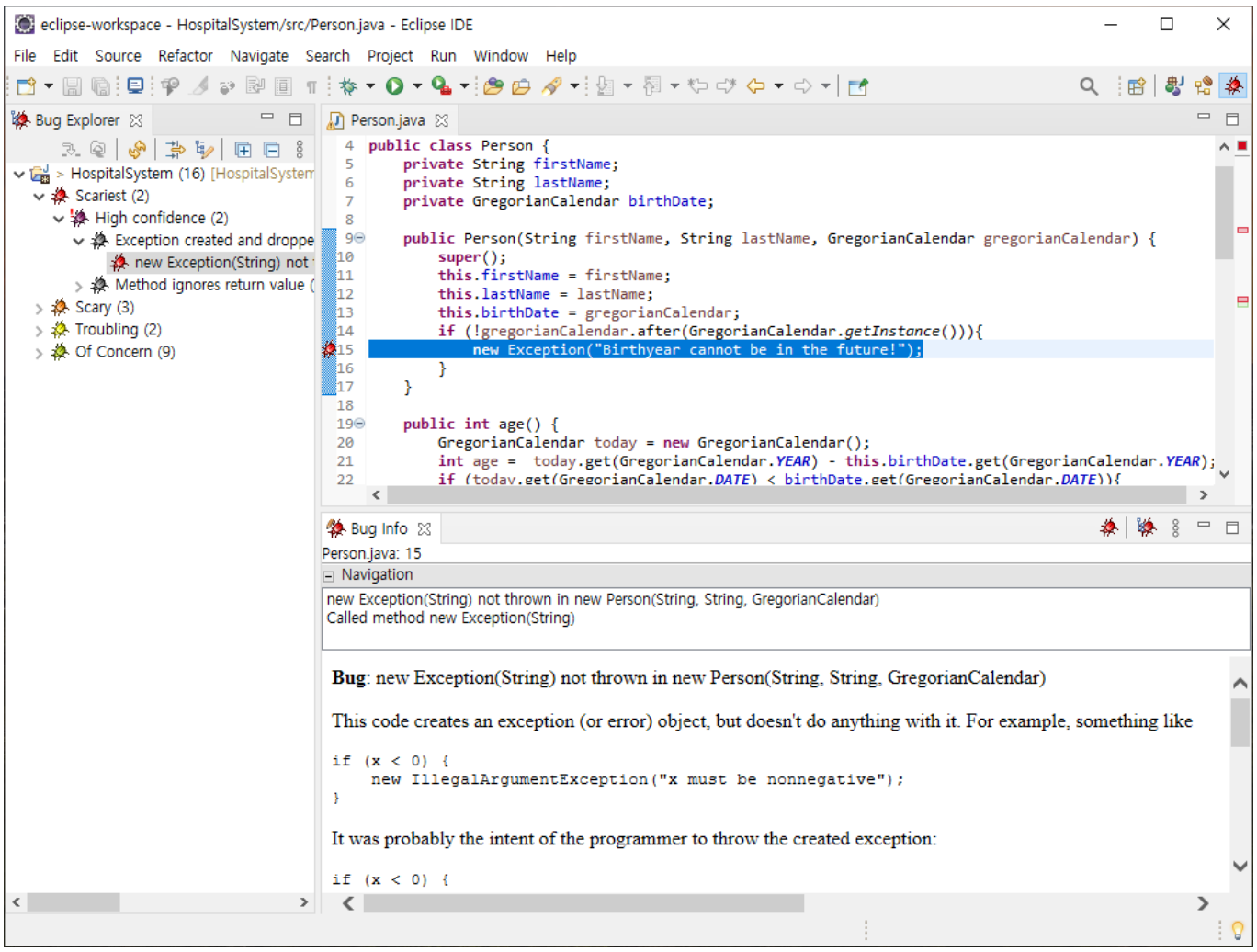
SpotBugs는 3가지 방법으로 버그를 분류한다.

1. Rank – 버그의 심각도 측정. 20(최저)에서 1(최고)까지 있다.
   심각도는 다음과 같이 4개로 분류된다: Concern, Troubling, Scary, Scariest
2. Confidence – 거짓 양성(오류가 아닌데 오류라고 보고)보다 진짜 오류일 가능성 측정.
3. Category – 버그의 유형 기록. 총 9개의 법주가 있는데, 몇 가지 예시로는 Correctness, Bad Practice, Performance 등이 있다. 추가로 모든 버그와 패턴을 일치시킨다.
   (관심이 있다면 밑 링크에서 패턴에 대한 정보 제공:
   Bug descriptions — spotbugs 4.2.3 documentation)

Eclipse에서는 발견된 버그의 계층을 탐색하여 개별 버그를 찾을 수 있다.



버그 설명을 한번 클릭하면 SpotBugs가 문제가 있는 코드 위치를 탐색하고, 자세한 정보는 "Bugs Info" 탭에서 확인할 수 있다.



같은 줄에 2개의 버그가 있는 경우, 버그 설명을 한 번 클릭하면 그 중 하나에 대한 정보를 볼 수 있고, 더블 클릭하면 다른 것에 대한 정보를 볼 수 있다.

Analyzing an issue

SpotBugs 가 보고한 예제 버그를 분석해보자. SpotBugs 는 항상 탐지한 버그에 대한 간략한 설명을 제공한다. 예시는 아래와 같다.

Bug: *Call to Person.equals(Integer) in Person.addFriend(Person)*

This method calls equals(Object) on two references of different class types and analysis suggests they will be to objects of different classes at runtime. Further, examination of the equals methods that would be invoked suggest that either this call will always return false, or else the equals method is not be symmetric (which is a property required by the contract for equals in class Object).

버그 설명을 더블 클릭하면 버그가 발견된 코드로 이동한다.

```
public void addFriend(Person person){

        if (person.equals(id)){
            System.out.println("One cannot be in his/her own friends list!");
        }
        else {
            friends.add(person);
        }
}
```

SpotBugs 가 보고한 이슈를 분석할 때, 가장 중요한 것은 실제 결함인지, 가짜 결함인지 구분하는 것이다. 예시의 경우는 클래스 Person 에서 객체와 정수 값을 비교하려고 하는 것이기 때문에 맞는 오류를 찾은 것 같다. 그럼에도 불구하고 *equals()* 함수가 재정의 되었을 가능성이 있으며, 따라서 비교가 의도된 구현일 수 있다.

Person 클래스의 두가지 구현 사례를 가정해보자:

Case 1

Code:

```
import java.util.ArrayList;
public class Person {
    private String name;
    private int id;
    private ArrayList<Person> friends;

    public Person(String name, int id, ArrayList<Person> friends) {
        super();
        this.name = name;
        this.id = id;
        this.friends = friends;
    }

    public void addFriend(Person person){
        if (person.equals(id)){
            System.out.println("One cannot be in his/her own friends list!");
        }
        else {
            friends.add(person);
        }
    }
}
```

결론: *equals()* 함수가 재정의되지 않으므로, *addFriend()* 함수는 잘못 사용되었다 따라서 SpotBugs 가 실제 결함을 발견하였다.

Case 2

Code:

```
import java.util.ArrayList;
public class Person {
    private String name;
    private int id;
    private ArrayList<Person> friends;

    public Person(String name, int id, ArrayList<Person> friends) {
        super();
        this.name = name;
        this.id = id;
        this.friends = friends;
    }

    @Override
    public boolean equals(Object object){
        try {
            return (int)object == id;
        }
        catch (Exception e) {
           return false;
        }
    }

    public void addFriend(Person person){
        if (person.equals(id)){
            System.out.println("One cannot be in his/her own friends list!");
        }
        else {
            friends.add(person);
        }
    }
}
```

결론: 이 코드가 일부 Java 코딩 규약을 어기고 있지만, 정수 값을 인수로 받아들이고 이를 해당 사람의 ID 와 비교하는 방식으로 *equals()* 함수를 재정의 한다. 이것은 보고된 버그가 의도한 구현이고, 따라서 거짓 양성(오류가 아닌데 오류라고 보고)임을 알 수 있다. 그럼에도 불구하고, 이해 가능성과 확장성 때문에 *equals()* 함수는 코딩 규약에 맞추도록 수정되어야 한다.

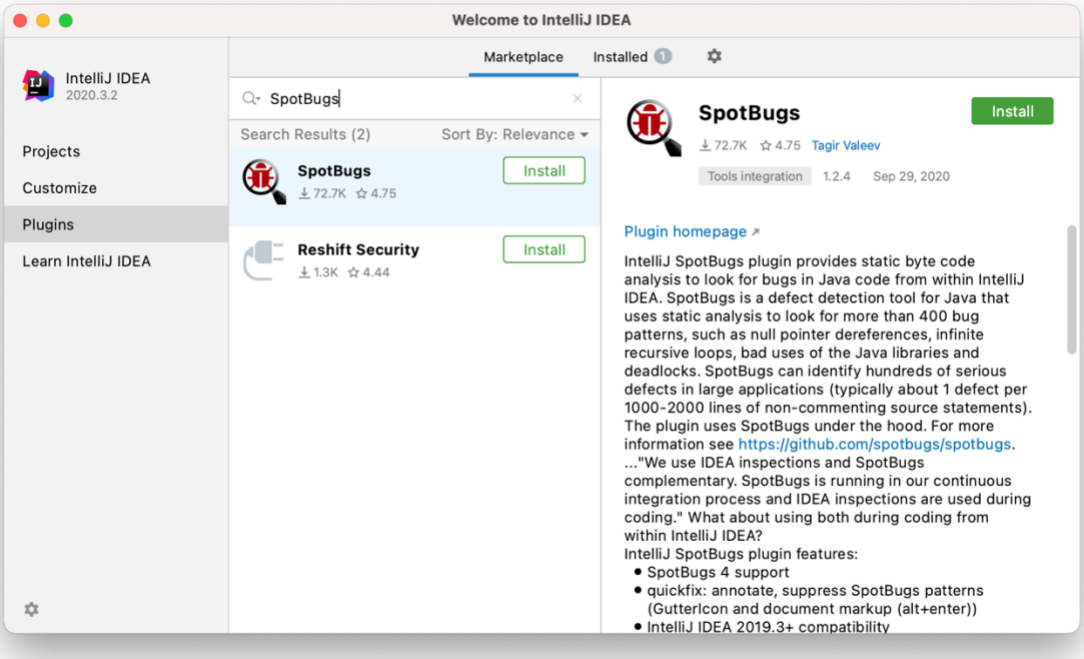
Others

IntelliJ IDEA에 HospitalSystem 추가

1. "HospitalSystem.zip"의 압축 해제
2. IntelliJ IDEA실행 후 Open project -> build.gradle 파일 선택
3. " Open as a Project " 선택

IntelliJ IDEA에 SpotBugs 추가

1. IntelliJ IDEA(2020.3)기준 시작 화면에서 "Plugins"를 클릭
2. 검색창에 "SpotBugs" 검색 후, SpotBugs 선택 및 설치
3. IntelliJ IDEA 다시 시작



IntelliJ SpotBugs 사용법

1. SpotBugs는 하단 툴바에 위치
2. "Analyze Project Files" 버튼 클릭
3. "Group by bug rank" 클릭

IntelliJ의 FindBugs의 ' Confidence' 확인 방법

1. 발견된 버그 클릭
2. 오른쪽 상단 창에 [버그 세부 정보와 신뢰도("Priority")] 확인 가능

IntelliJ의 FindBugs 추가 설정

1. File -> Settings -> Tools -> SpotBugs 클릭
2. "Report" 탭 선택
3. Minimum rank > 20 설정
4. Minimum confidence > Low 설정
5. Reported bug categories > 모두 선택
6. 확인(OK) 클릭