<Software Verification>

유아를 위한 주제별 영어 학습 놀이 프로그램 Verification

<Static Analysis Report #1>

Tema 4.
200911393 박현규
201010768 최정한
201111339 김민우
201211389 함진아

1. Checkstyle

- CheckStyle은 코딩 표준에 맞게 소스코드를 작성하도록 도와주는 도구이다.
- 개발자들이 잘 지키지는 않지만 중요한 것들을 지적해준다 (ex : private 설정)
- 협업시에 코딩스타일을 맞출 수 있고, 잠재적인 결함 발견도 가능하다. 팀의 코딩규칙을 문서로 정리해서 지키기로 하면, 규칙이 많아질 수록 어기기 쉽다.

- Checkstyle에서 기본적으로 제공하는 ruleset을 사용할 경우
- 14개의 카테고리 및 149개의 rule
  → 1201개의 warning 발생

CheckStyle Result

Warnings Trend

All Warnings	New Warnings	Fixed Warnings
1201	0	0

Summary

Total	High Priority	Normal Priority	Low Priority
1201	0	1201	0

Details

Files | **Categories** | Types | Warnings | Details

Category	Total	Distribution
Metrics	11	
Naming	11	
Whitespace	1179	
Total	1201	

1) Metrics

Checkstyle Warnings - Category Metrics

Summary

Total	High Priority	Normal Priority	Low Priority
11	0	11	0

Details

Files | Types | Warnings | Details

File	Total	Distribution
AlphaPanel.java	2	
Controller.java	1	
Game.java	1	
GamePanel.java	4	
TitlePanel.java	1	
WordPanel.java	2	
Total	11	

Type	Total	Distribution
ClassDataAbstractionCouplingCheck	4	
ClassFanOutComplexityCheck	3	
CyclomaticComplexityCheck	1	
JavaNCSSCheck	3	
Total	11	

* ClassDataAbstractionCouplingCheck
  - 클래스에서 사용된 레퍼런스 타입의 수를 계산한다.
  - 최대 7개까지가 적정선.

* ClassFanOutComplexityCheck
  - 참조하여 사용하는 class의 개수
  - 최대 20개까지가 적정선.

ex)

AlphaPanel.java:28, ClassDataAbstractionCouplingCheck, Priority: Normal

Class Data Abstraction Coupling is 10 (max allowed is 7) classes [ActionListener, File, FindWord, ImageIcon, JButton, JLabel, JTextField, KeyAdapter, LineBorder, MouseAdapter].

This metric measures the number of instantiations of other classes within the given class. This type of coupling is not caused by inheritance or the object oriented paradigm. Generally speaking, any abstract data type with other abstract data types as members has data abstraction coupling; therefore, if a class has a local variable that is an instantiation (object) of another class, there is data abstraction coupling. The higher the DAC, the more complex the data structure (classes) of the system.

AlphaPanel.java:28, ClassFanOutComplexityCheck, Priority: Normal

Class Fan-Out Complexity is 21 (max allowed is 20).

The number of other classes a given class relies on. Also the square of this has been shown to indicate the amount of maintenance required in functional programs (on a file basis) at least.

Game.java:46, JavaNCSSCheck, Priority: Normal

NCSS for this method is 54 (max allowed is 50).

Determines complexity of methods, classes and files by counting the Non Commenting Source Statements (NCSS). This check adheres to the specification for the JavaNCSS-Tool written by **Chr. Clemens Lee**.
Roughly said the NCSS metric is calculated by counting the source lines which are not comments, (nearly) equivalent to counting the semicolons and opening curly braces.
The NCSS for a class is summarized from the NCSS of all its methods, the NCSS of its nested classes and the number of member variable declarations.
The NCSS for a file is summarized from the ncss of all its top level classes, the number of imports and the package declaration.

Rationale: Too large methods and classes are hard to read and costly to maintain. A large NCSS number often means that a method or class has too many responsibilities and/or functionalities which should be decomposed into smaller units.

result

File	Line	Priority	Type	Category
AlphaPanel.java:28	28	Normal	ClassDataAbstractionCouplingCheck	Metrics
AlphaPanel.java:28	28	Normal	ClassFanOutComplexityCheck	Metrics
Controller.java:30	30	Normal	ClassDataAbstractionCouplingCheck	Metrics
Game.java:46	46	Normal	JavaNCSSCheck	Metrics
GamePanel.java:27	27	Normal	ClassDataAbstractionCouplingCheck	Metrics
GamePanel.java:27	27	Normal	ClassFanOutComplexityCheck	Metrics
GamePanel.java:53	53	Normal	JavaNCSSCheck	Metrics
GamePanel.java:83	83	Normal	CyclomaticComplexityCheck	Metrics
TitlePanel.java:21	21	Normal	JavaNCSSCheck	Metrics
WordPanel.java:28	28	Normal	ClassDataAbstractionCouplingCheck	Metrics
WordPanel.java:28	28	Normal	ClassFanOutComplexityCheck	Metrics

2) Naming

Checkstyle Warnings - Category Naming

Summary

Total	High Priority	Normal Priority	Low Priority
11	0	11	0

Details

Files | **Types** | Warnings | Details

Type	Total	Distribution
AbbreviationAsWordInNameCheck	1	
LocalVariableNameCheck	2	
MemberNameCheck	7	
StaticVariableNameCheck	1	
Total	11	

* LocalVariableNameCheck
  - Local 변수의 이름이 제시하는 format에 맞는가 ?
  - Naming의 전반적인 내용이 첫 글자의 대문자 사용에 대한 지적

AlphaPanel.java:34, MemberNameCheck, Priority: Normal

Name 'F' must match pattern '^[a-z][a-zA-Z0-9]*$'.

No description available. Please upgrade to latest checkstyle version.

AlphaPanel.java:124, LocalVariableNameCheck, Priority: Normal

Name 'Icon' must match pattern '^[a-z][a-zA-Z0-9]*$'.

No description available. Please upgrade to latest checkstyle version.

Game.java:20, StaticVariableNameCheck, Priority: Normal

Name 'ANSWERNUM' must match pattern '^[a-z][a-zA-Z0-9]*$'.

No description available. Please upgrade to latest checkstyle version.

GamePanel.java:48, MemberNameCheck, Priority: Normal

Name 'F' must match pattern '^[a-z][a-zA-Z0-9]*$'.

No description available. Please upgrade to latest checkstyle version.

result

File	Line	Priority	Type	Category
AlphaPanel.java:34	34	Normal	MemberNameCheck	Naming
AlphaPanel.java:124	124	Normal	LocalVariableNameCheck	Naming
Game.java:20	20	Normal	StaticVariableNameCheck	Naming
GamePanel.java:48	48	Normal	MemberNameCheck	Naming
MainPanel.java:20	20	Normal	MemberNameCheck	Naming
RankingPanel.java:16	16	Normal	MemberNameCheck	Naming
TitlePanel.java:19	19	Normal	MemberNameCheck	Naming
WordDic.java:15	15	Normal	AbbreviationAsWordInNameCheck	Naming
WordDic.java:15	15	Normal	MemberNameCheck	Naming
WordPanel.java:34	34	Normal	MemberNameCheck	Naming
WordPanel.java:123	123	Normal	LocalVariableNameCheck	Naming

3) Whitespace

Type	Total	Distribution
EmptyLineSeparatorCheck	89	
FileTabCharacterCheck	17	
GenericWhitespaceCheck	1	
NoWhitespaceAfterCheck	13	
WhitespaceAfterCheck	303	
WhitespaceAroundCheck	756	
Total	1179	

- EmptyLineSeparatorCheck
  → 초기화 시에 공백이 필요, 혹은 잘못된 공백 사용
- FileTabCharacterCheck
  → 소스코드 내 tab 사용
- WhitespaceAfterCheck
  → 토큰 뒤에 공백이 필요
- WhitespaceAroundCheck
  → 토큰 앞뒤에 공백이 필요

File	Line	Priority	Type	Category
AlphaPanel.java:28	28	Normal	WhitespaceAroundCheck	Whitespace
AlphaPanel.java:29	29	Normal	FileTabCharacterCheck	Whitespace
AlphaPanel.java:30	30	Normal	EmptyLineSeparatorCheck	Whitespace
AlphaPanel.java:31	31	Normal	EmptyLineSeparatorCheck	Whitespace
AlphaPanel.java:32	32	Normal	EmptyLineSeparatorCheck	Whitespace
AlphaPanel.java:33	33	Normal	EmptyLineSeparatorCheck	Whitespace
AlphaPanel.java:34	34	Normal	EmptyLineSeparatorCheck	Whitespace
AlphaPanel.java:35	35	Normal	EmptyLineSeparatorCheck	Whitespace
AlphaPanel.java:40	40	Normal	WhitespaceAroundCheck	Whitespace
AlphaPanel.java:40	40	Normal	WhitespaceAroundCheck	Whitespace
AlphaPanel.java:43	43	Normal	WhitespaceAfterCheck	Whitespace
AlphaPanel.java:43	43	Normal	WhitespaceAfterCheck	Whitespace
AlphaPanel.java:43	43	Normal	WhitespaceAfterCheck	Whitespace
AlphaPanel.java:44	44	Normal	WhitespaceAfterCheck	Whitespace
AlphaPanel.java:48	48	Normal	WhitespaceAroundCheck	Whitespace
AlphaPanel.java:49	49	Normal	WhitespaceAfterCheck	Whitespace
AlphaPanel.java:49	49	Normal	WhitespaceAfterCheck	Whitespace
AlphaPanel.java:49	49	Normal	WhitespaceAfterCheck	Whitespace
AlphaPanel.java:53	53	Normal	WhitespaceAroundCheck	Whitespace
AlphaPanel.java:53	53	Normal	WhitespaceAroundCheck	Whitespace
AlphaPanel.java:54	54	Normal	WhitespaceAfterCheck	Whitespace
AlphaPanel.java:54	54	Normal	WhitespaceAfterCheck	Whitespace
AlphaPanel.java:54	54	Normal	WhitespaceAfterCheck	Whitespace
AlphaPanel.java:65	65	Normal	WhitespaceAroundCheck	Whitespace
AlphaPanel.java:65	65	Normal	WhitespaceAroundCheck	Whitespace
AlphaPanel.java:65	65	Normal	WhitespaceAroundCheck	Whitespace
AlphaPanel.java:66	66	Normal	WhitespaceAroundCheck	Whitespace

2. PMD

- Program May Dependable의 약자
- 정해진 규칙에 따라 소스코드를 검사해서 위반사항을 찾아낼 수 있다.
- Eclipse –plugin으로 사용할 수 있으며, 별도로 커맨드라인에서 사용 가능하다.
- 분석 대상은 java, javaScript, XML 등이 있다.
- PMD에서 제공하는 카테고리로부터, 분석 대상 프로그램 언어인 Java에 맞는 카테고리 선정

● 카테고리의 큰 틀 16가지

Basic (rulesets/basic.xml)	**Code size (rulesets/codesize.xml)**
Naming (rulesets/naming.xml)	Javabeans (rulesets/javabeans.xml)
Unused code (rulesets/unusedcode.xml)	Finalizers
Design (rulesets/design.xml)	Clone (rulesets/clone.xml)
Import statements (rulesets/imports.xml)	Coupling (rulesets/coupling.xml)
JUnit tests (rulesets/junit.xml)	Strict exceptions (rulesets/strictexception.xml)
Strings (rulesets/string.xml)	Controversial (rulesets/controversial.xml)
Braces (rulesets/braces.xml)	Logging (rulesets/logging-java.xml)

Summary

Total	High Priority	Normal Priority	Low Priority
43	0	43	0

Details

Files | **Categories** | Types | Warnings | Details | New | Fixed

Category	Total	Distribution
Braces	13	
Code Size	23	
String and StringBuffer	1	
Unused Code	6	
Total	43	

1) Braces

- for, if, while, else 문장이 괄호를 사용하는지 여부 검사.

Type	Total	Distribution
ForLoopsMustUseBraces	1	
IfElseStmtsMustUseBraces	5	
IfStmtsMustUseBraces	7	
Total	13	

File	Line	Priority	Type	Category
AlphaPanel.java:65	65	Normal	IfStmtsMustUseBraces	Braces
Game.java:133	133	Normal	IfStmtsMustUseBraces	Braces
GamePanel.java:212	212	Normal	IfElseStmtsMustUseBraces	Braces
GamePanel.java:213	213	Normal	IfElseStmtsMustUseBraces	Braces
GamePanel.java:321	321	Normal	IfStmtsMustUseBraces	Braces
Rank.java:49	49	Normal	IfStmtsMustUseBraces	Braces
Ranking.java:48	48	Normal	IfElseStmtsMustUseBraces	Braces
RankingPanel.java:54	54	Normal	IfStmtsMustUseBraces	Braces
TitlePanel.java:96	96	Normal	IfElseStmtsMustUseBraces	Braces
TitlePanel.java:140	140	Normal	IfElseStmtsMustUseBraces	Braces
TitlePanel.java:155	155	Normal	ForLoopsMustUseBraces	Braces
WordDic.java:146	146	Normal	IfStmtsMustUseBraces	Braces
WordPanel.java:65	65	Normal	IfStmtsMustUseBraces	Braces

2) Code Size

- 과도하게 긴 메소드, 너무 많은 메소드를 가진 클래스, 리팩토링에 대한 유사한 후보들을 위한 테스트.

File	Total	Distribution
FindWordTest.java	1	
GamePanel.java	12	
GameTest.java	1	
TitlePanel.java	9	
Total	23	

ex)

GamePanel.java:27, TooManyFields, Priority: Normal

Too many fields.

Classes that have too many fields can become unwieldy and could be redesigned to have fewer fields, possibly through grouping related fields in new objects. For example, a class with individual city/state/zip fields could park them within a single Address field.

```
public class Person {   // too many separate fields
   int birthYear;
   int birthMonth;
   int birthDate;
   float height;
   float weight;
}

public class Person {   // this is more manageable
   Date birthDate;
   BodyMeasurements measurements;
}
```

GamePanel.java:27, CyclomaticComplexity, Priority: Normal

The class 'GamePanel' has a Cyclomatic Complexity of 9 (Highest = 24).

Complexity directly affects maintenance costs is determined by the number of decision points in a method plus one for the method entry. The decision points include 'if', 'while', 'for', and 'case labels' calls. Generally, numbers ranging from 1-4 denote low complexity, 5-7 denote moderate complexity, 8-10 denote high complexity, and 11+ is very high complexity.

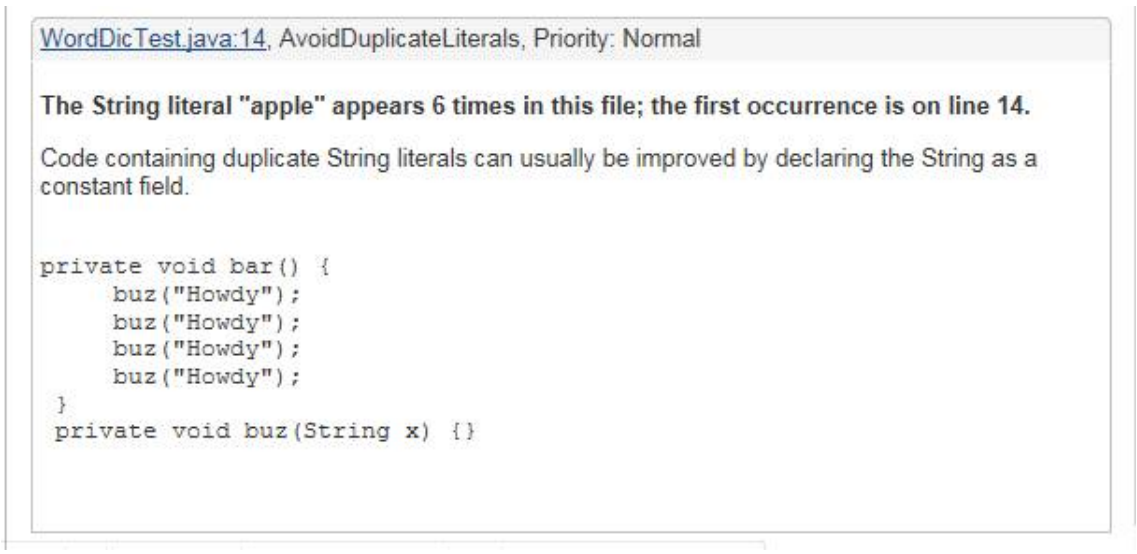
```
public class Foo {              // This has a Cyclomatic Complexity = 12
1   public void example()  {
2       if (a == b)  {
3           if (a1 == b1) {
                fiddle();
```

result

File	Line	Priority	Type	Category
FindWordTest.java:11	11	Normal	TooManyMethods	Code Size
GamePanel.java:27	27	Normal	TooManyFields	Code Size
GamePanel.java:27	27	Normal	CyclomaticComplexity	Code Size
GamePanel.java:27	27	Normal	ModifiedCyclomaticComplexity	Code Size
GamePanel.java:27	27	Normal	StdCyclomaticComplexity	Code Size
GamePanel.java:53	53	Normal	CyclomaticComplexity	Code Size
GamePanel.java:53	53	Normal	ModifiedCyclomaticComplexity	Code Size
GamePanel.java:53	53	Normal	StdCyclomaticComplexity	Code Size
GamePanel.java:83	83	Normal	StdCyclomaticComplexity	Code Size
GamePanel.java:83	83	Normal	CyclomaticComplexity	Code Size
GamePanel.java:204	204	Normal	StdCyclomaticComplexity	Code Size
GamePanel.java:204	204	Normal	CyclomaticComplexity	Code Size
GamePanel.java:204	204	Normal	ModifiedCyclomaticComplexity	Code Size
GameTest.java:7	7	Normal	TooManyMethods	Code Size
TitlePanel.java:14	14	Normal	CyclomaticComplexity	Code Size
TitlePanel.java:14	14	Normal	ModifiedCyclomaticComplexity	Code Size
TitlePanel.java:14	14	Normal	StdCyclomaticComplexity	Code Size
TitlePanel.java:14	14	Normal	TooManyMethods	Code Size
TitlePanel.java:21	21	Normal	StdCyclomaticComplexity	Code Size
TitlePanel.java:21	21	Normal	CyclomaticComplexity	Code Size
TitlePanel.java:21	21	Normal	ModifiedCyclomaticComplexity	Code Size
TitlePanel.java:124	124	Normal	StdCyclomaticComplexity	Code Size
TitlePanel.java:124	124	Normal	CyclomaticComplexity	Code Size

3) String

스트링 관련 작업을 할 때 발생하는 일반적인 문제들 규명. 스트링 리터럴 중복, String 구조체 호출, String 객체에 toString() 호출하기 등.



WordDicTest.java:14, AvoidDuplicateLiterals, Priority: Normal

The String literal "apple" appears 6 times in this file; the first occurrence is on line 14.

Code containing duplicate String literals can usually be improved by declaring the String as a constant field.

```
private void bar() {
     buz("Howdy");
     buz("Howdy");
     buz("Howdy");
     buz("Howdy");
 }
 private void buz(String x) {}
```

4) Unused Code

결코 읽히지 않은 프라이빗 필드와 로컬 변수, 접근할 수 없는 문장, 결코 호출되지 않는 프라이빗 메소드 등을 찾기.

Type	Total	Distribution
UnusedLocalVariable	1	
UnusedPrivateField	5	
Total	6	

Game.java:22, UnusedPrivateField, Priority: Normal

Avoid unused private fields such as 'name'.

Detects when a private field is declared and/or assigned a value, but not used.

File	Line	Priority	Type	Category
Game.java:22	22	Normal	UnusedPrivateField	Unused Code
Game.java:28	28	Normal	UnusedPrivateField	Unused Code
GamePanel.java:208	208	Normal	UnusedLocalVariable	Unused Code
Main.java:6	6	Normal	UnusedPrivateField	Unused Code
WordDicTest.java:8	8	Normal	UnusedPrivateField	Unused Code
WordDicTest.java:9	9	Normal	UnusedPrivateField	Unused Code

3. FindBugs

자바프로그램 분석도구로, 메릴랜드 대학에서 개발했다.
data flow, control flow 분석등을 이용한다.
버그 패턴을 자동으로 찾아서 알려준다 (ex : race condition, null Pointer exception 등)
source code(*.java 파일) 보다는 byte code(*.class 파일)를 기반으로 분석을 진행한다.●

FindBugs Result

Warnings Trend

All Warnings	New this build	Fixed Warnings
24	0	0

Summary

Total	High Priority	Normal Priority	Low Priority
24	4	20	0

Details

Files | **Categories** | Types | Warnings | Details | High | Normal

Category	Total	Distribution
BAD_PRACTICE	1	
CORRECTNESS	8	
EXPERIMENTAL	1	
I18N	3	
PERFORMANCE	8	
STYLE	3	
Total	24	

ex)

Child.java:23, DM_DEFAULT_ENCODING, Priority: High

Dm: Found reliance on default encoding in new Classes.Child(Superman): new java.util.Scanner(File)

Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

Controller.java:34, UWF_UNWRITTEN_FIELD, Priority: Normal

UwF: Unwritten field: Classes.Controller.child

This field is never written. All reads of it will return the default value. Check for errors (should it have been initialized?), or remove it if it is useless.

Controller.java:41, UWF_UNWRITTEN_FIELD, Priority: Normal

UwF: Unwritten field: Classes.Controller.parent

This field is never written. All reads of it will return the default value. Check for errors (should it have been initialized?), or remove it if it is useless.

Controller.java:68, NP_DEREFERENCE_OF_READLINE_VALUE, Priority: Normal

NP: Dereference of the result of readLine() without nullcheck in Classes.Controller.parentMode()

The result of invoking readLine() is dereferenced without checking to see if the result is null. If there are no more lines of text to read, readLine() will return null and dereferencing that will generate a null pointer exception.

Wordtrain.java:43, EI_EXPOSE_REP, Priority: Normal

EI: Classes.Wordtrain.getWordList() may expose internal representation by returning Wordtrain.wordlist

Returning a reference to a mutable object value stored in one of the object's fields exposes the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Returning a new copy of the object is better approach in many situations.

Wordtrain.java:109, RV_DONT_JUST_NULL_CHECK_READLINE, Priority: Normal

RV: Classes.Wordtrain.searchWordList(char) discards result of readLine after checking if it is nonnull

The value returned by readLine is discarded after checking to see if the return value is non-null. In almost all situations, if the result is non-null, you will want to use that non-null value. Calling readLine again will give you a different line.

Wordtrain.java:133, NP_DEREFERENCE_OF_READLINE_VALUE, Priority: Normal

NP: Dereference of the result of readLine() without nullcheck in Classes.Wordtrain.searchWordList(char)

The result of invoking readLine() is dereferenced without checking to see if the result is null. If there are no more lines of text to read, readLine() will return null and dereferencing that will generate a null pointer exception.

WordtrainTest.java:37, UC_USELESS_OBJECT, Priority: Normal

Useless object created

Our analysis shows that this object is useless. It's created and modified, but its value never go outside of the method or produce any side-effect. Either there is a mistake and object was intended to be used or it can be removed.

This analysis rarely produces false-positives. Common false-positive cases include:

- This object used to implicitly throw some obscure exception.

result

Files | Categories | Types | **Warnings** | Details | High | Normal

File	Line	Priority	Rank	Type	Category
FindWordTest.java:14	14	Normal	12	UWF_UNWRITTEN_FIELD	CORRECTNESS
FindWordTest.java:14	14	Normal	3	UR_UNINIT_READ	CORRECTNESS
FindWordTest.java:14	14	Normal	18	URF_UNREAD_FIELD	PERFORMANCE
Game.java:34	34	Normal	18	URF_UNREAD_FIELD	PERFORMANCE
Game.java:36	36	Normal	18	URF_UNREAD_FIELD	PERFORMANCE
GamePanel.java:84	84	Normal	19	SF_SWITCH_NO_DEFAULT	STYLE
GamePanel.java:299	299	Normal	18	SBSC_USE_STRINGBUFFER_CONCATENATION	PERFORMANCE
GameTest.java:9	9	Normal	18	URF_UNREAD_FIELD	PERFORMANCE
GameTest.java:10	10	Normal	3	UR_UNINIT_READ	CORRECTNESS
GameTest.java:10	10	Normal	12	UWF_UNWRITTEN_FIELD	CORRECTNESS
Rank.java:49	49	Normal	16	EQ_COMPARETO_USE_OBJECT_EQUALS	BAD_PRACTICE
RankTest.java:13	13	Normal	3	UR_UNINIT_READ	CORRECTNESS
RankTest.java:13	13	Normal	18	URF_UNREAD_FIELD	PERFORMANCE
RankTest.java:13	13	Normal	12	UWF_UNWRITTEN_FIELD	CORRECTNESS
Ranking.java:16	16	High	19	DM_DEFAULT_ENCODING	I18N
Ranking.java:26	26	Normal	17	REC_CATCH_EXCEPTION	STYLE
Ranking.java:37	37	Normal	20	OBL_UNSATISFIED_OBLIGATION_EXCEPTION_EDGE	EXPERIMENTAL
Ranking.java:37	37	High	19	DM_DEFAULT_ENCODING	I18N
TitlePanel.java:126	126	Normal	19	SF_SWITCH_NO_DEFAULT	STYLE
WordDic.java:37	37	High	19	DM_DEFAULT_ENCODING	I18N
WordDic.java:69	69	Normal	11	NP_NULL_ON_SOME_PATH_EXCEPTION	CORRECTNESS
WordDic.java:69	69	High	7	NP_ALWAYS_NULL_EXCEPTION	CORRECTNESS
WordDicTest.java:-1	-	Normal	18	UUF_UNUSED_FIELD	PERFORMANCE
WordDicTest.java:9	9	Normal	18	URF_UNREAD_FIELD	PERFORMANCE

4. JDepend

패키지 의존성과 관련된 설계 품질을 관리/분석 할 수 있다.
클래스 파일들을 읽어서 분석을 한다.
각 패키지별로 의존성 측정이 가능하며, 수치화, 그래프로 표현이 가능하다.
개발과정 중에도 아키텍쳐 품질을 평가할 수 있다.
XML 형식으로도 저장 가능하다.

Package	TC	CC	AC	Ca	Ce	A	I	D	V
project	47	47	0	0	1	0.0%	100.0%	0.0%	1

※ 결과가 가지는 의미

CC	Concrete Class
AC	Abstract Class 추상클래스나 인터페이스의 개수. 확정성의 척도
Ca	Afferent Couplings 현재 패키지에 종속성을 갖는 패키지 개수.
Ce	Efferent Couplings 현재 패키지가 종속하고 있는 패키지 개수.
A	추상화 정도. 0은 완전 구체적 패키지. 1은 추상적 패키지.
D	Main Sequence로 부터의 거리. Main Sequence란 추상적이면서 안정적이거나, 완전 구체적이면서 불완정한 패키지. 0일수록 가깝고 1일수록 멀다.

Term	Description
Number of Classes	The number of concrete and abstract classes (and interfaces) in the package is an indicator of the extensibility of the package.
Afferent Couplings	The number of other packages that depend upon classes within the package is an indicator of the package's responsibility.
Efferent Couplings	The number of other packages that the classes in the package depend upon is an indicator of the package's independence.
Abstractness	The ratio of the number of abstract classes (and interfaces) in the analyzed package to the total number of classes in the analyzed package. The range for this metric is 0 to 1, with A=0 indicating a completely concrete package and A=1 indicating a completely abstract package.
Instability	The ratio of efferent coupling (Ce) to total coupling (Ce / (Ce + Ca)). This metric is an indicator of the package's resilience to change. The range for this metric is 0 to 1, with I=0 indicating a completely stable package and I=1 indicating a completely instable package.
Distance	The perpendicular distance of a package from the idealized line A + I = 1. This metric is an indicator of the package's balance between abstractness and stability. A package squarely on the main sequence is optimally balanced with respect to its abstractness and stability. Ideal packages are either completely abstract and stable (x=0, y=1) or completely concrete and instable (x=1, y=0). The range for this metric is 0 to 1, with D=0 indicating a package that is coincident with the main sequence and D=1 indicating a package that is as far from the main sequence as possible.
Cycles	Packages participating in a package dependency cycle are in a deadly embrace with respect to reusability and their release cycle. Package dependency cycles can be easily identified by reviewing the textual reports of dependency cycles. Once these dependency cycles have been identified with JDepend, they can be broken by employing various object-oriented techniques.

Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
0	1	0.0%	100.0%	0.0%

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
None	project.AlphaPanel project.AlphaPanel$1 project.AlphaPanel$2 project.AlphaPanel$3 project.AlphaPanel$4 project.Controller project.FindWord project.FindWordTest project.Game project.GamePanel project.GamePanel$1 project.GamePanel$2 project.GamePanel$3 project.GamePanel$4 project.GamePanel$5 project.GamePanel$6 project.GameTest project.Main project.MainPanel project.MainPanel$1 project.MainPanel$2 project.MainPanel$3 project.MainPanel$4 project.MainPanel$5	*None*	org.junit