|  |
| --- |
| Framework for Integrated Test |
| F I T |
| 소프트웨어공학 조사 보고서 |



|  |
| --- |
| 컴퓨터공학과 0491050 유준희  2008-10-04 |

Contents

[1. Fit란? 3](#_Toc210997190)

[2. Fit의 작업흐름 3](#_Toc210997191)

[Appendix A. Fit 설치하기 8](#_Toc210997197)

[Appendix B. Fitpro 설치 및 사용방법 13](#_Toc210997198)

[Appendix C. 소스코드 21](#_Toc210997199)

1. Fit란?

A

gile Manifesto가 2001년 선언되면서 강조한 4가지 조항 중에 고객과의 협력에 대한 내용이 있다. ‘계약에 의한 협상보다 고객과의 협조[[1]](#footnote-2)’를 우선시 한다는 조항이 바로 그것인데. 이는 시스템을 개발하는 것에 있어 문제가 생겼을 때의 조정을 얘기하는 것만이 아니라 고객의 요구사항을 잘 이해하는 것이 얼마나 중요한가도 알려준다. 요구사항에 대한 명확한 이해가 없으면 개발 도중이나 심지어 개발 후에 구현된 기능이 고객이 원하는 기능이 아닌 경우, 보수하는데 드는 비용은 커질 수 밖에 없기 때문이다. 정확한 요구사항 분석은 비단 개발자만의 문제가 아니다. 사용자가 원하는 기능이 제대로 동작하는지 테스트를 해야 하는 테스터에게도 요구사항이 원하는 입력과 출력이 어떤 것인지 명확히 알고 있어야 빈틈없는 테스트를 할 수 있다.

그렇다면, 이렇게 중요한 요구사항명세서를 누가 만드는 것이 가장 좋은가? FIT는 비전문가라도 한눈에 알 수 있는 문서를 통하여 고객-개발자-테스터 사이에 명확한 의사소통과 협동의 질을 높일 수 있으며 나아가 제품의 문서화에도 큰 도움을 줄 수 있는 통합테스트 환경을 제공한다.

1. Fit의 작업흐름

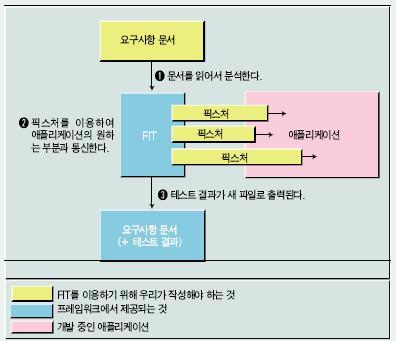


그림 – FIT의 구조[[2]](#footnote-3)

그

림 1은 FIT의 구조를 보여준다. 위 그림에서 입력으로 들어가는 요구사항 문서는 ‘실행 가능한 요구사항 명세서(executable specification)’이다. 이것을 만들기 위해 고객은 개발자, 테스터와 함께 수많은 토론과 학습을 통하여 요구사항을 명확히 해 나간다. 요구사항의 정리가 끝나고 나면 개발자, 테스터와 함께 일하는 고객은 완성된 요구사항 예제를 MS Word나 기타 WYSIWYG 도구를 이용하여 표에 정리하고 문서에 넣는다.

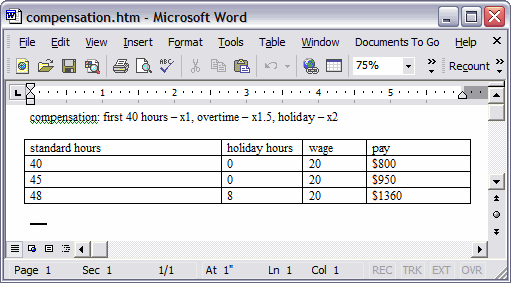


그림 – 고객이 초기화한 예제

이러한 예제는 비전문가인 고객이 만들었기 때문에 다른 비전문가라도 쉽게 알아볼 수 있다. 이 문서에 개발자와 테스터는 포함해야 할 새로운 사항들을 추가한다.

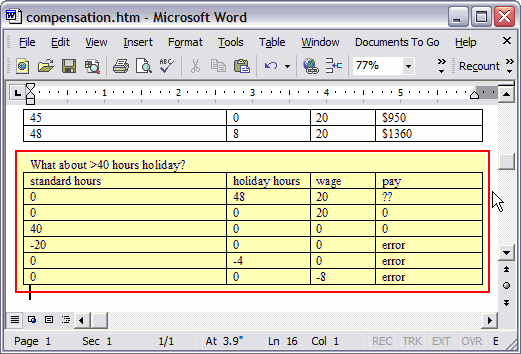


그림 – 테스터와 개발자의 추가사항

개발자는 FIT가 이해할 수 있도록 표를 수정한다.

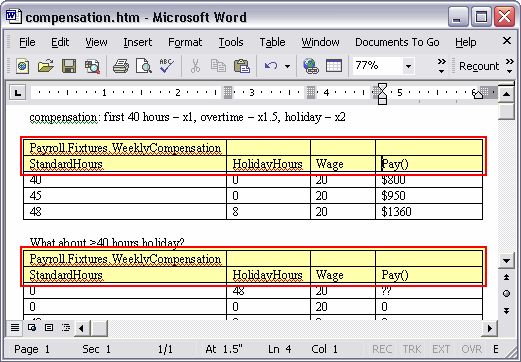


그림 – FIT가 이해할 수 있는 형식

개발자는 프로그램이 개발될 때 예제를 이해하고 실행할 수 있도록 “fixtures:”라는 작은 프로그램코드를 작성해야 한다. Fit 프레임워크는 이것을 쉽게 할 수 있는 기반을 제공한다. Fit확장툴인 Fitpro를 이용하면 이 Fixture를 자동으로 생성해 준다.

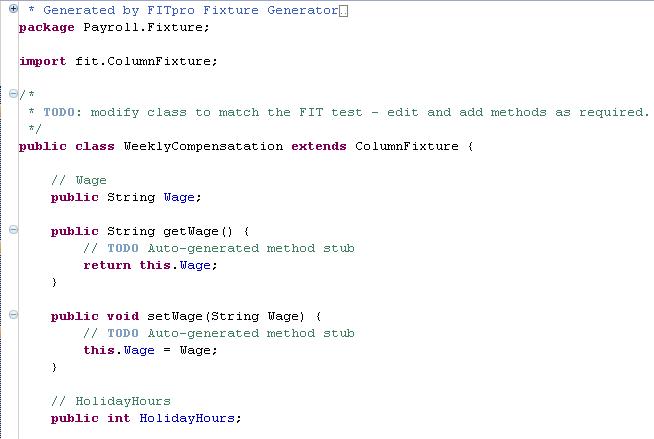


그림 - Fitpro를 이용해 생성한 Fixture의 일부

현재 생성된 템플릿에 코드를 넣지 않아 모든 기능이 구현되지 않았음을 나타내는 붉은색으로 색이 변해 있다. 코드가 구현되어 감에 따라 붉은색은 점차 푸른색으로 바뀌어 갈 것이다.

\*위에 나오는 값이 요구사항명세서에 따른 출력 값이고, 바로 밑의 값이 실제 코드가 출력한 값이다.

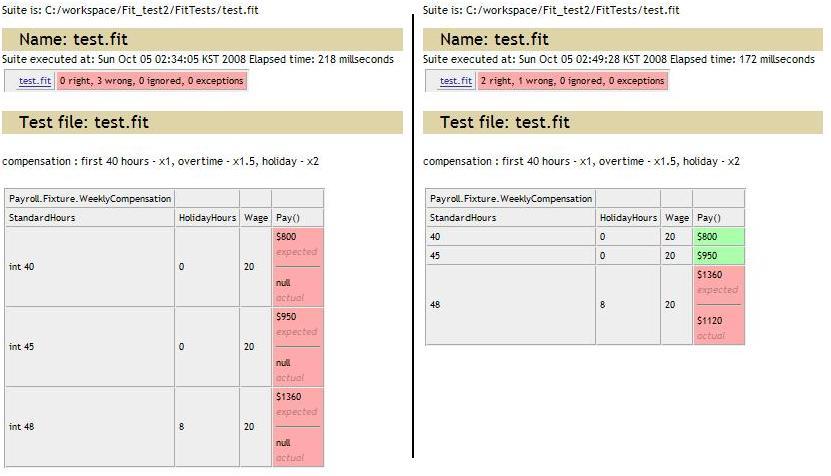


그림 – Fitpro를 통해 나온 문서들  
왼쪽 : 모든 기능이 구현되어 있지 않은 프로그램  
오른쪽 : 일부 기능이 구현되어 있는 프로그램

모든 구현을 완료한 후 Fit문서의 실행결과는 다음과 같다.

\*사용자가 입력한 내용에 대해 기대한 출력이 모두 나왔기 때문에 모두 녹색으로 표시되어 있다.

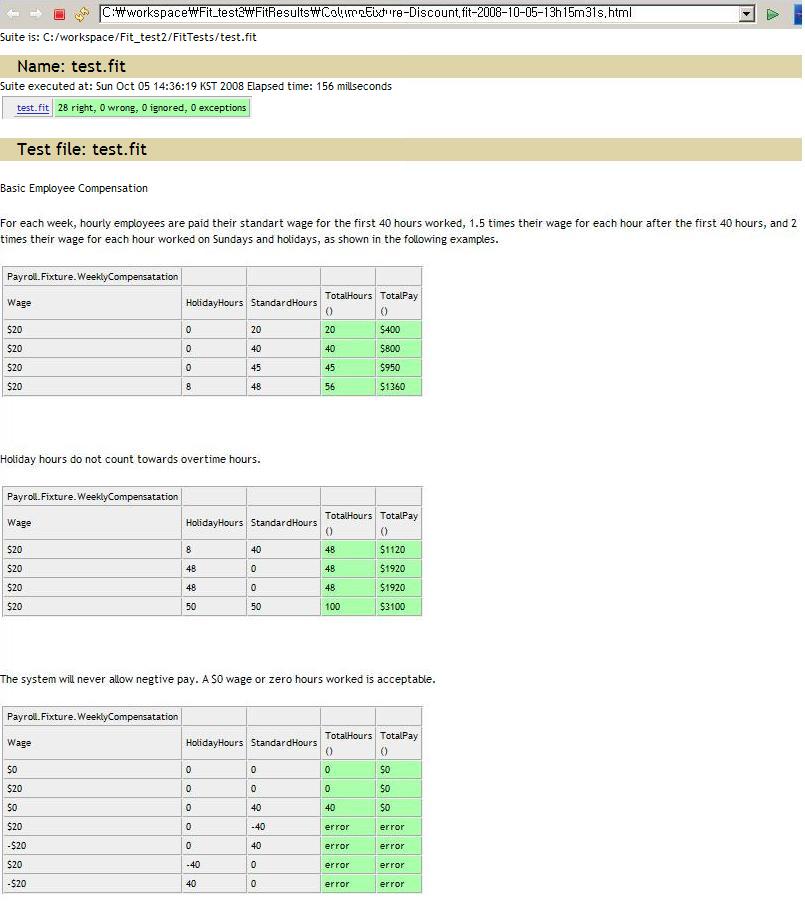


그림 - 완성된 Fit 문서

이렇게 Fit를 통해 생성된 문서는 시간이 지나고 나서 소프트웨어가 어떤 일을 하는지 의문이 생길 때 참고할 수 있는 훌륭한 자료가 될 수 있다. 만약 원하는 기능이 Fit 문서에 존재하지 않는다면 요구사항에 대한 내용을 추가하고 실행하면 된다.

Appendix A. Fit 설치하기

통합개발환경인 Eclipse – Ganymede와 Windows XP 환경에서 Fit를 사용하는 방법을 알아보자. 다음의 사이트에서 Fit를 다운받는다.

다운로드 링크

|  |
| --- |
| http://fit.c2.com/wiki.cgi?DownloadNow |

다운을 받아서 원하는 폴더에 압축을 풀면 다음과 같은 파일들이 나온다.

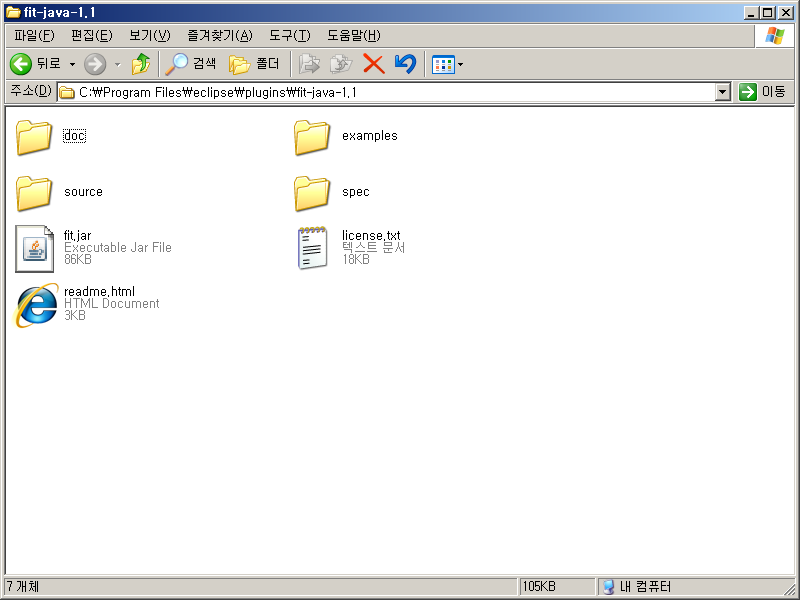


그림 - fit-java-1.1의 파일들

이 파일 중 fit.jar 파일이 가장 중요한 파일이다. 이 JAR파일을 컴파일 할 때 사용할 수 있게 설정 해 줘야 한다. Eclipse를 실행하고 프로젝트를 선택한 후 오른쪽 클릭을 하여 Properties를 선택하자. 메뉴바에 있는 Project → Properties도 같은 대화상자를 호출한다.

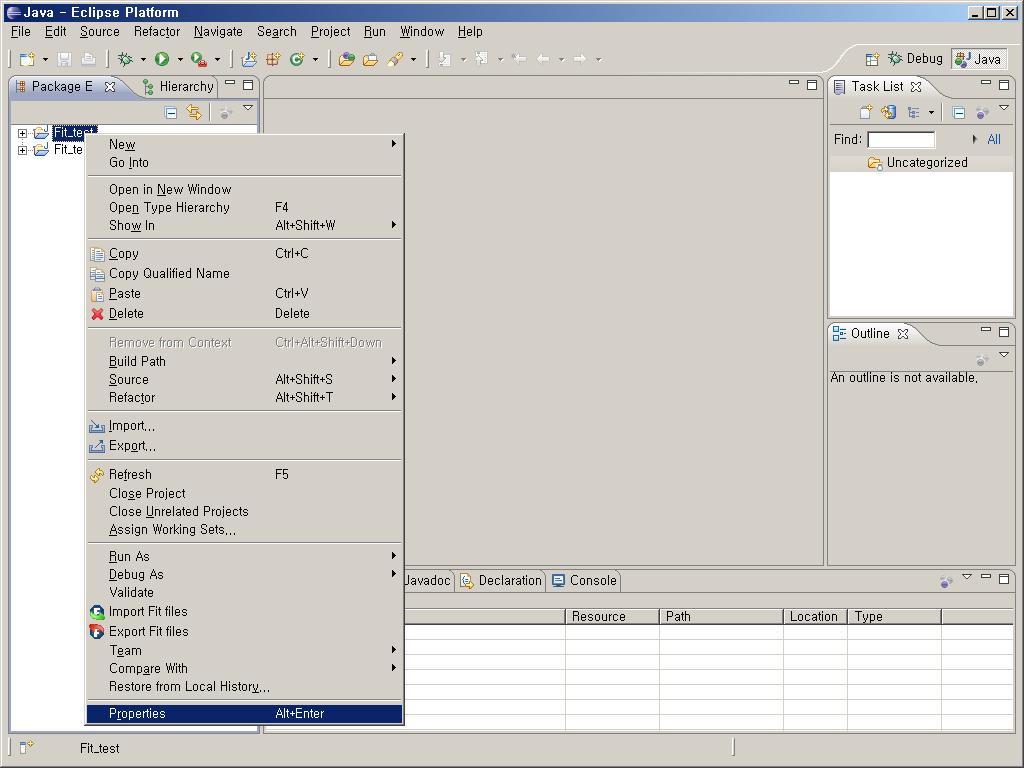


그림 - Project의 Properties 메뉴

JAR파일을 가져오는 방법은 여러가지가 있지만 Java Build Path → Libraries를 선택하여 외부JAR파일로 가져오는 방법을 소개하겠다.

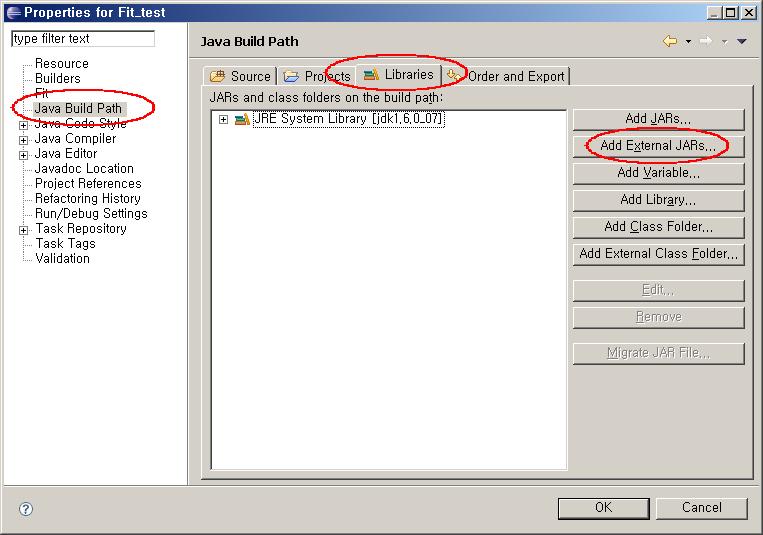


그림 - User Libraries

앞서 압축을 풀었던 폴더에 가서 fit.jar파일을 열도록 한다.



그림 1- fit.jar 파일 열기

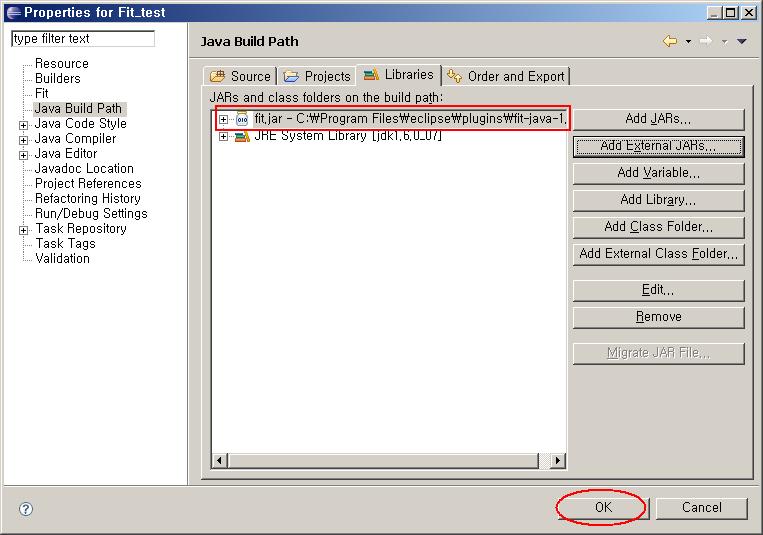


그림 - JAR 파일이 추가된 모습

이제 OK 버튼을 누르면 Fit를 사용하기 위한 설정은 끝난다. 설정이 잘 되었는지 확인하기 위해 소스코드에 다음과 같은 import문을 추가해 보자.

다음과 같이 Eclipse가 패키지의 내용을 출력해 주면 설정에 성공한 것이다.

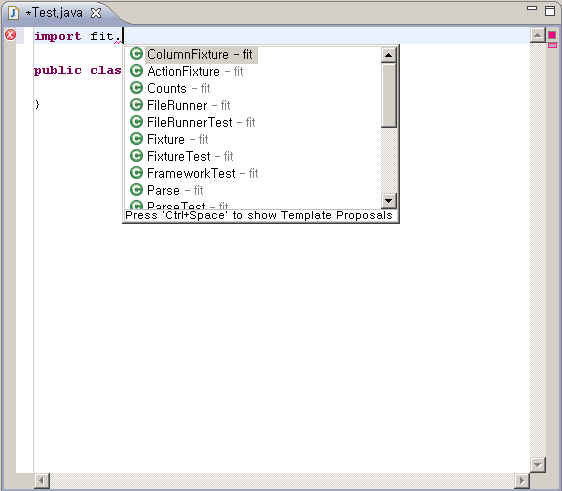


그림 - 설정 완료

Fixture 파일을 만드는 방법은 다음 사이트에 자세히 나와있다.

Fit Cook Tour

|  |
| --- |
| http://fit.c2.com/wiki.cgi?CooksTour |

Fit의 실행은 Command Prompt로 Fit가 있는 디렉터리로 이동하여 다음과 같이 입력한다.

|  |
| --- |
| Java -classpath fit.jar fit.FileRunner [input\_file] [output\_file] |

다음은 예제의 실행결과이다.

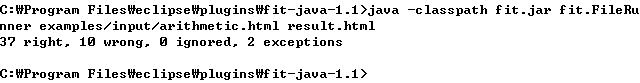


그림 - Fit의 명령행 실행

Appendix B. Fitpro 설치 및 사용방법

Fit는 충분히 유용한 도구임에는 틀림없다. 하지만 Fixture를 자동으로 생성하고 문서의 실행도 클릭 한번만으로 가능하다면 더욱 유용할 것이다. Fitpro는 html문서 에디터와 Fixture자동 생성도구 및 문서의 실행을 Eclipse의 Plug-in으로 만든 것이다. Eclipse - Ganymede에 Fitpro를 설치하여 보자.

다음URL은 Sourceforge.net의 Fitpro페이지이다.

|  |
| --- |
| http://sourceforge.net/projects/fitpro/ |

하지만 우리는 Eclipse – Ganymede의 Software Updates 기능을 이용하여 Fitpro를 설치할 예정이다. Eclipse.org페이지의 우측 상단 Search에서 Fitpro를 검색해 보자.

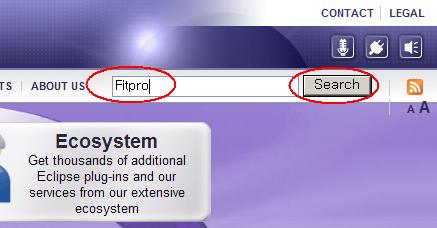


그림 - www.eclipse.org

관련 링크가 하나만 나올 것이고 클릭하면 다음과 같은 페이지가 나오는데 Update Site URL을 클립보드에 복사하자.



그림 - Fitpro의 Update Site URL

Eclipse – Ganymede를 실행하여 메뉴바의 Help → Software Updates를 선택하도록 하자.

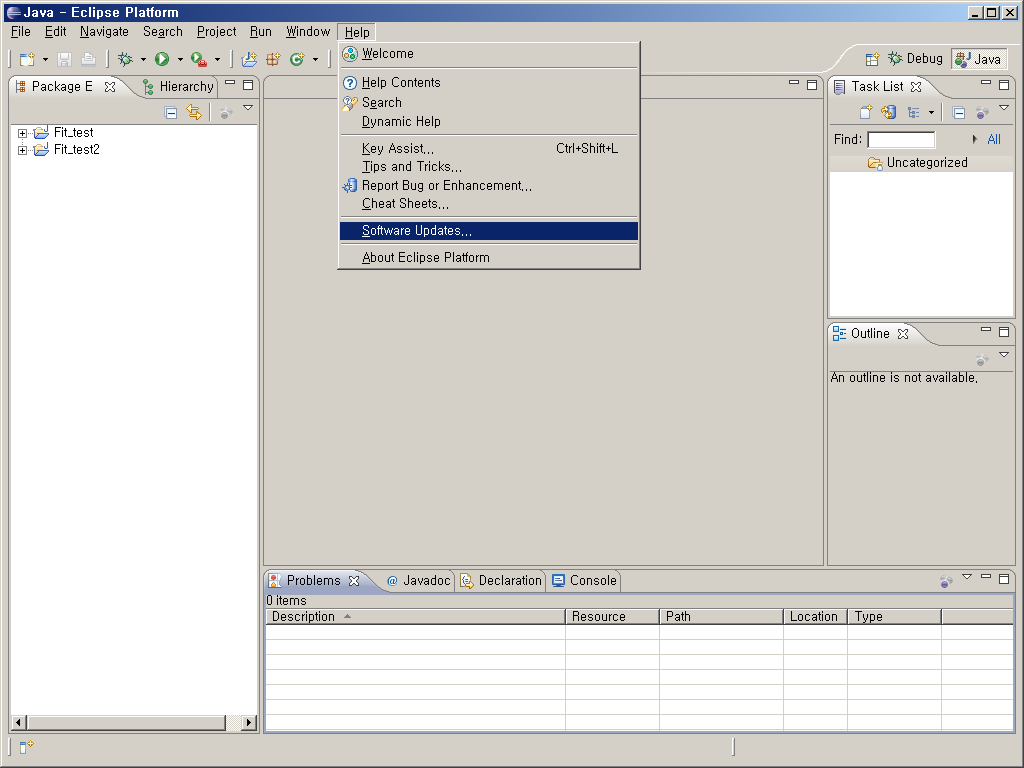


그림 - Help → Software Updates

선택하고 나면 다음과 같은 대화상자가 나타난다. 여기서는 Available Software탭을 선택한다.

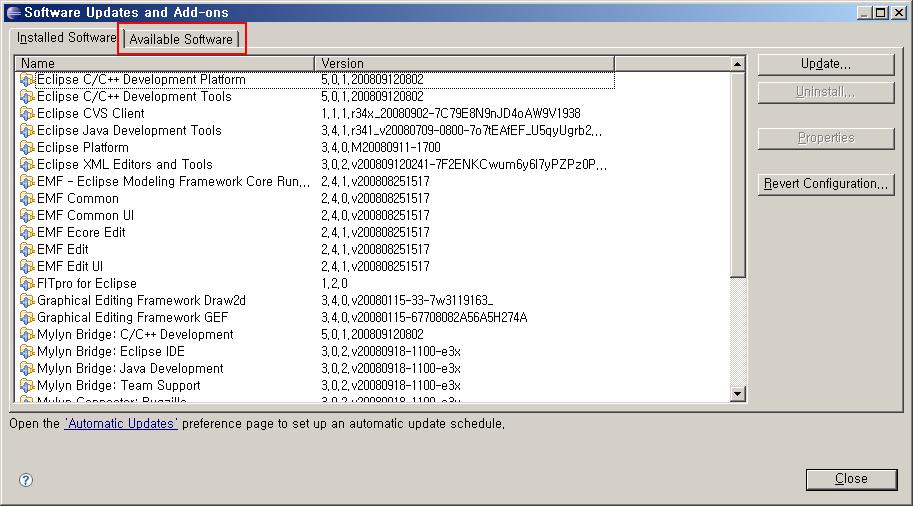


그림 - Software Updates 대화상자

Add Site를 눌러 앞서 클립보드에 복사했던 URL을 설정한다.

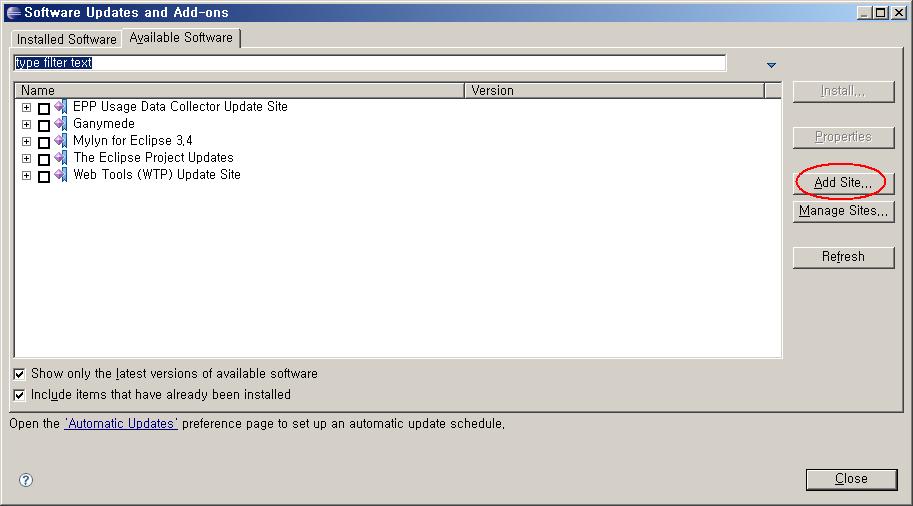


그림 - Available Software → Add Site

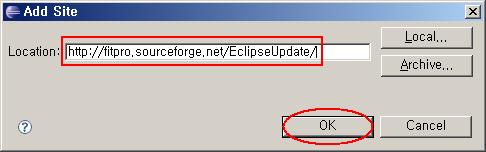


그림 - Fitpro의 Update Site 설정

추가된 사이트에서 FITpro for Eclipse를 선택한 후 Install을 누른다.

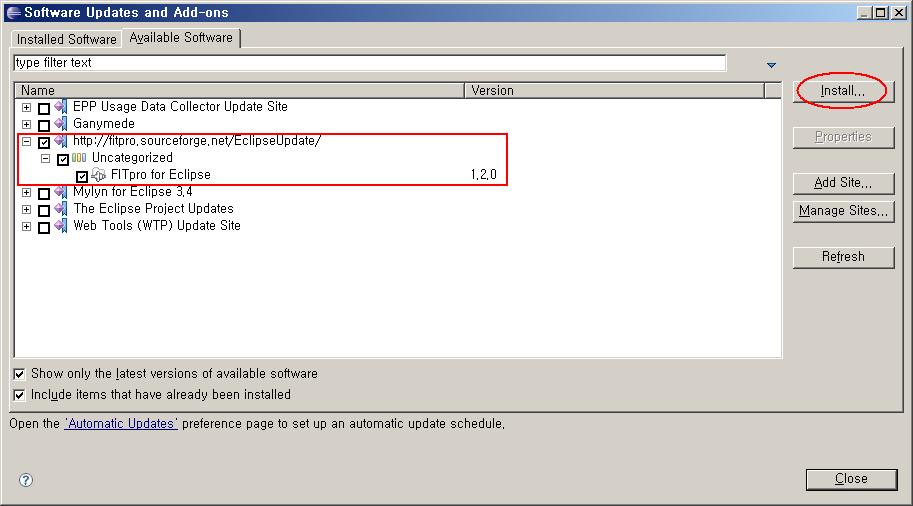


그림 - FITpro 설치 선택

나머지 대화상자에서는 잘 읽고 진행하면 무리가 없으리라 생각하여 Fitpro의 사용방법에 대해서 설명하고자 한다.

프로젝트 생성시 프로젝트 위자드에서 다음과 같이 선택하고 프로젝트 이름을 설정 후 Finish버튼을 누른다.

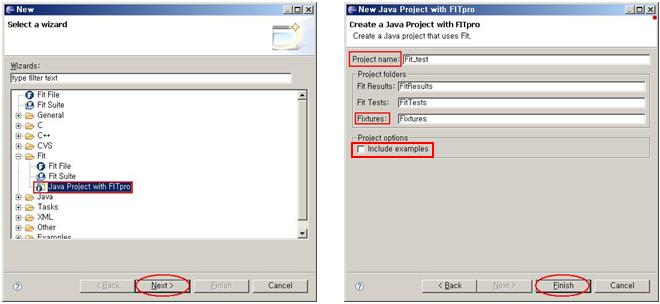


그림 - Java Project with FITpro

프로젝트 설정 후 Properties 대화상자를 연다.

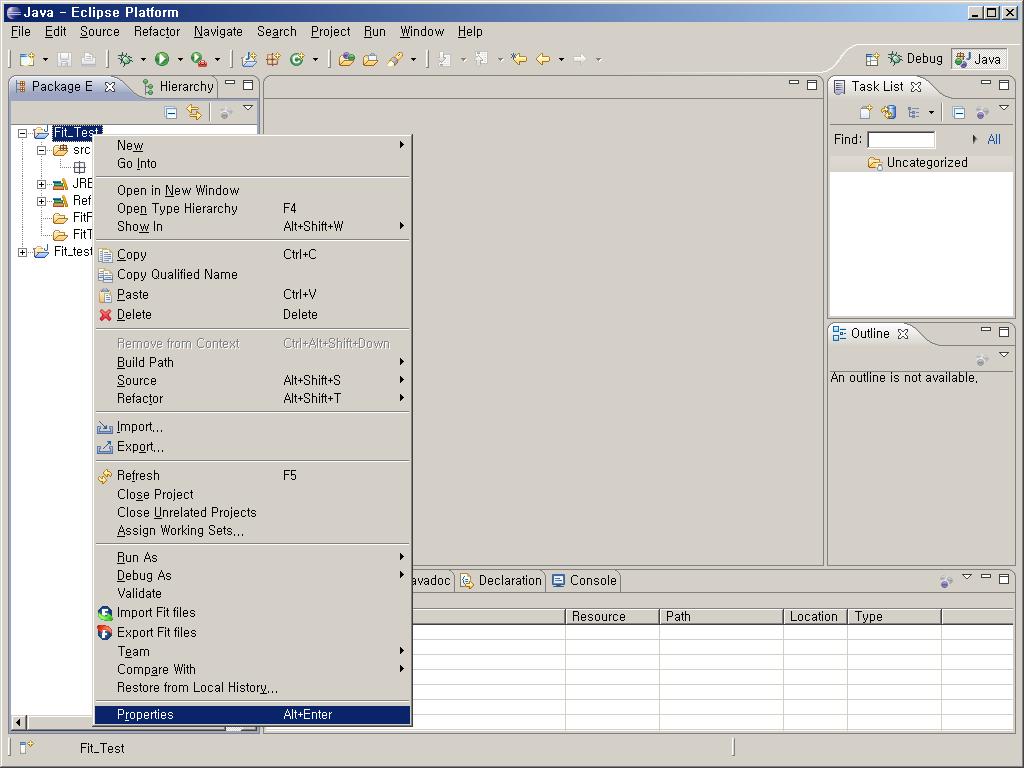


그림 - Project Properties



그림 - Properties 중 Fit 속성 변경

Fixture Source를 수정하는 까닭은 이 설정을 바꿔주지 않으면 Fixtures라는 패키지에 자동생성된 Fixture 파일이 생성되는데, Fixtueres패키지와 실행모듈이 링크가 안되어 있는지 문서 실행 버튼을 누르면 Class를 찾지 못하는 에러가 생기게 된다. OK를 누르고 나서 FitTests폴더에 Fit 문서를 생성해 보자.

FitTest폴더에서 오른쪽 클릭을 하여 New → Other 메뉴를 따라서 들어가면 다음과 같은 익숙한 대화상자가 뜬다.

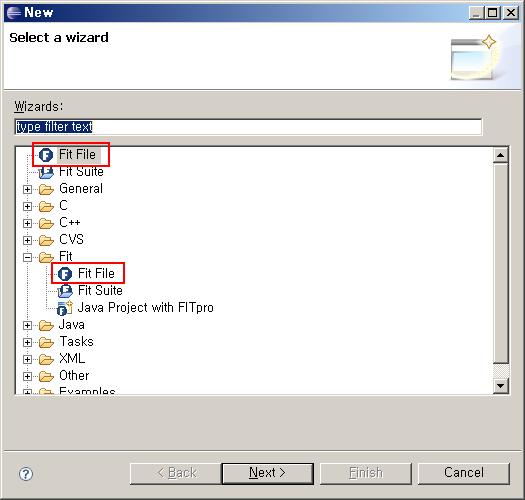


그림 - New 대화상자

둘 중 어느 것으로 하더라도 차이는 없다. Next를 눌러 파일의 이름을 정하고 Finish버튼을 누르면 파일이 생성된다.

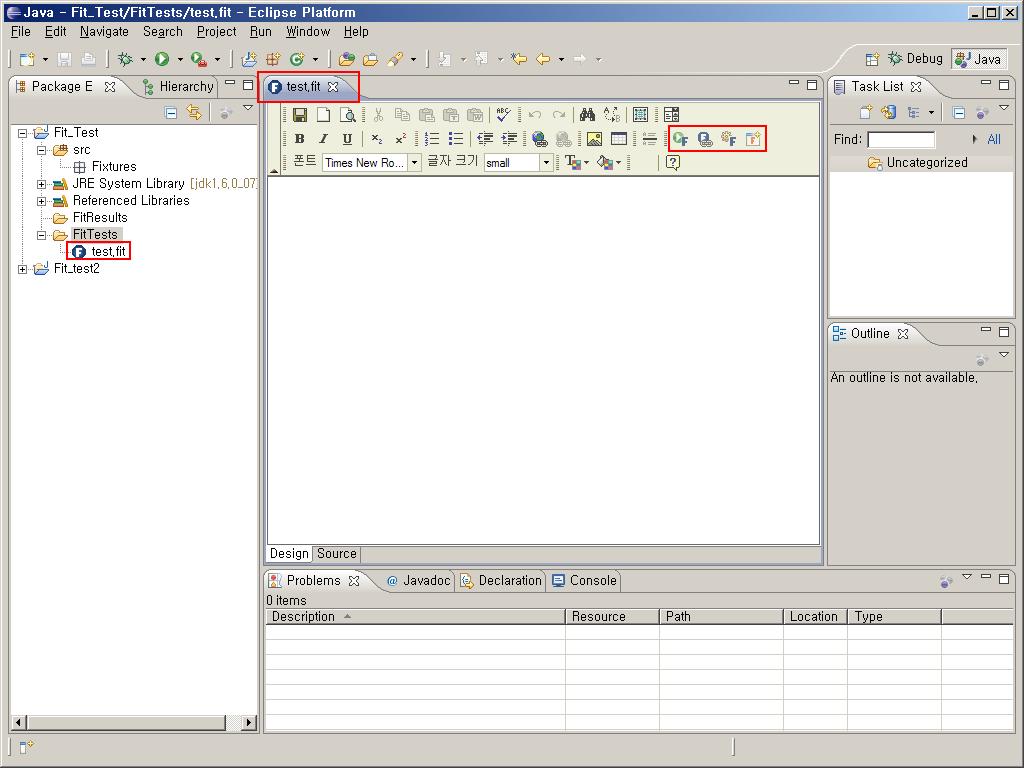


그림 - 생성된 Fit 문서와 편집기

Fitpro는 간단한 html문서 편집기를 제공한다. 오른쪽에 보이는 단추 묶음은 Fixture를 생성하거나 문서를 실행하는 단축버튼이다.

설정.JPG

그림 - Fit 단축버튼

왼쪽부터 차례로 Fit문서 실행, Fit문서 열기, Fixture생성, \*.java파일로부터 Fit문서 생성이다. 이때 \*.java파일은 Fixture구조를 가지고 있어야 한다. 간단히 Fit문서를 작성하여 Fixture파일을 생성하여 보자. Fit문서를 작성하는 방법은 다음의 URL을 참고하도록 하자.

Fit 문서 작성방법

|  |
| --- |
| http://fit.c2.com/wiki.cgi?CooksTour |

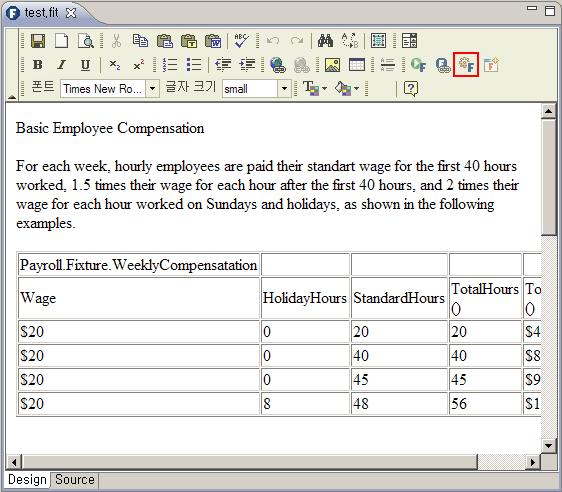


그림 - 이 문서에서 예제로 사용한 Fit문서

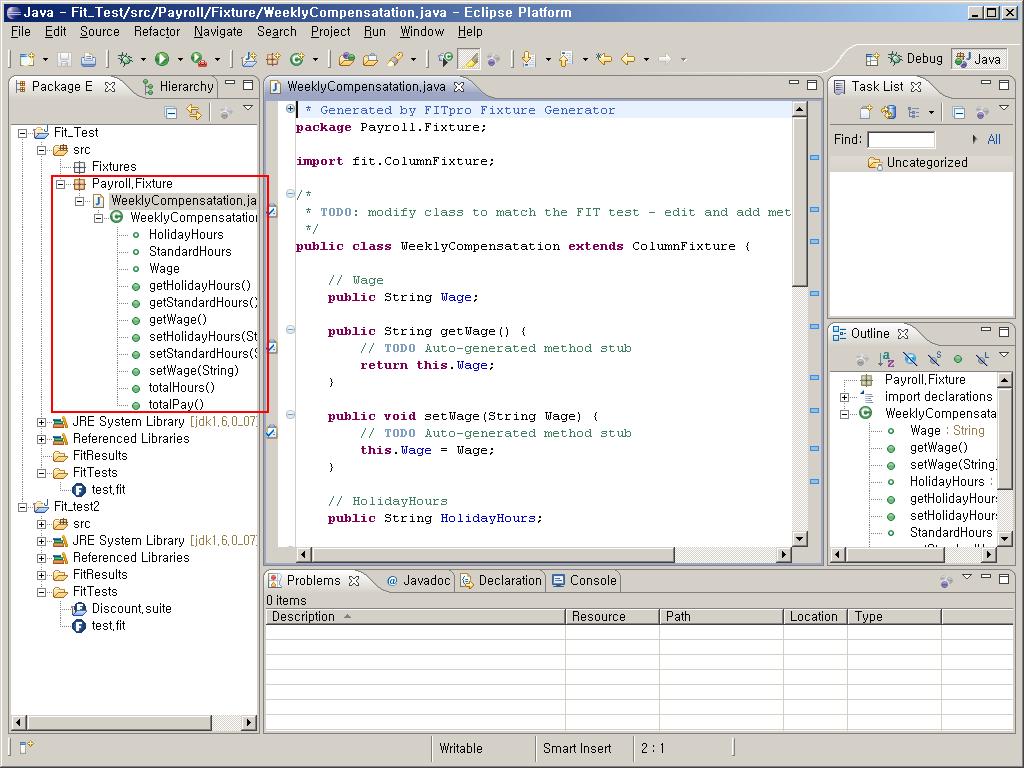


그림 - test.fit로 부터 생성된 Fixture

이 자동생성된 Fixture에서 재밌는 점은 멤버변수가 모두 public으로 되어있다는 점이다. 자동으로 생성된 set메소드들을 이용하여 Fit문서로부터 값을 읽어 멤버를 초기화 하는 것이 아니라 외부에서 바로 값을 넣어준다는 것을 잘 기억하고 코드를 짜야한다. 또한 Fit문서에서 error는 error라는 문자열을 출력하라는 의미가 아닌 Error객체를 Throw하라는 의미이다.

원하는 결과가 나올 수 있도록 코드를 작성한 후 test.fit에서 Run FIT Test버튼을 눌러보자.

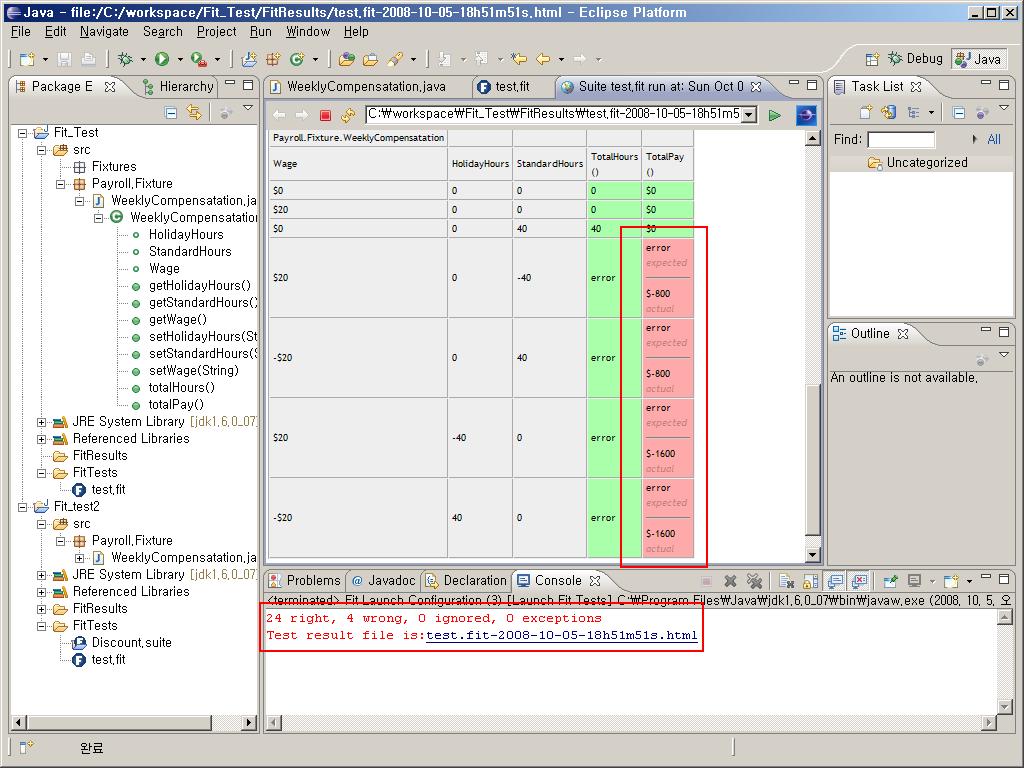


그림 - Fit문서의 실행

입력된 테스트케이스에 대하여 처리를 한 후에 콘솔출력을 해 준다. 링크를 눌러보면 결과 문서 창을 띄워주는데 문서를 보니 음수 값일 때의 처리가 잘 안된 것 같다. 코드를 수정한 후 다시 한번 실행해 보자.

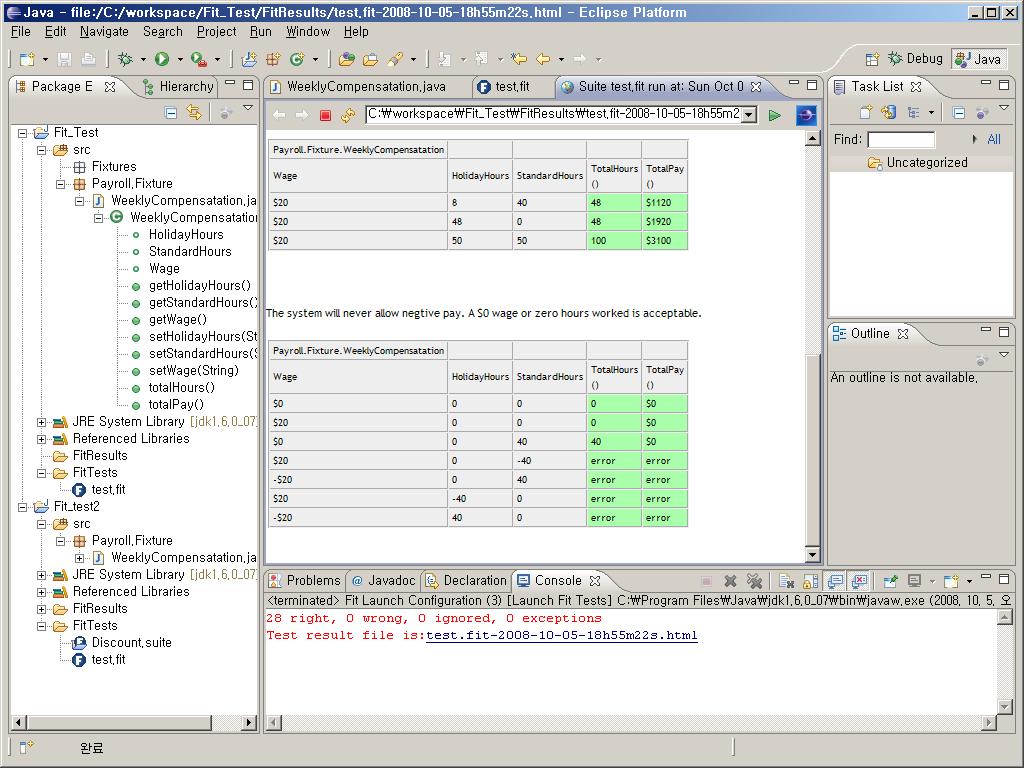


그림 - 모든 케이스를 통과한 Fit 결과문서

Fitpro는 Fit문서 에디터와 Fixture자동생성기능으로 훨씬 편리해 보인다. Fitpro에 대해 조사하면서 디버깅을 하는 것에 대해 지원하는지는 찾지 못했지만 통합테스트를 위한 다른 툴과 연계하여 사용하면 협동작업에 효율을 가져다 줄 것으로 보인다.

Appendix C. 소스코드

WeeklyCompensatation.java

|  |
| --- |
| /\*\*  \* Generated by FITpro Fixture Generator  \*/  **package** Payroll.Fixture;  **import** fit.ColumnFixture;  /\*  \* **TODO**: modify class to match the FIT test - edit and add methods as required.  \*/  **public** **class** WeeklyCompensatation **extends** ColumnFixture {  // Wage  **public** String Wage;    **public** String getWage() {  // **TODO** Auto-generated method stub  **return** **this**.Wage;  }    **public** **void** setWage(String Wage) {  // **TODO** Auto-generated method stub  **this**.Wage = Wage;  }    // HolidayHours  **public** **int** HolidayHours;    **public** String getHolidayHours() {  // **TODO** Auto-generated method stub  **return** String.*valueOf*(**this**.HolidayHours);  }    **public** **void** setHolidayHours(String HolidayHours) {  // **TODO** Auto-generated method stub  **this**.HolidayHours = Integer.*parseInt*(HolidayHours);  }    // StandardHours  **public** **int** StandardHours;    **public** String getStandardHours() {  // **TODO** Auto-generated method stub  **return** String.*valueOf*(**this**.StandardHours);  }    **public** **void** setStandardHours(String StandardHours) {  // **TODO** Auto-generated method stub  **this**.StandardHours = Integer.*parseInt*(StandardHours);  }    **public** String totalHours() {  // **TODO** Auto-generated method stub  **int** Wage2 = Integer.*parseInt*(**this**.Wage.replace("$", ""));    **if**(StandardHours < 0 || HolidayHours < 0 || Wage2 < 0){  **throw** **new** Error("지못미...");  }**else**{  **return** String.*valueOf*(StandardHours + HolidayHours);  }  }    **public** String totalPay() {  // **TODO** Auto-generated method stub    **int** result = 0;  **int** Wage2 = Integer.*parseInt*(**this**.Wage.replace("$", ""));      **if**(StandardHours < 0 || HolidayHours < 0 || Wage2 < 0){  **throw** **new** Error("^^ㅗ");  }**else** **if**(StandardHours <= 40){  result += StandardHours \* Wage2;  result += HolidayHours \* Wage2 \* 2;  **return** "$" + String.*valueOf*(result);  }**else**{  result += 40 \* Wage2;  result += (StandardHours - 40) \* Wage2 \* 1.5;  result += HolidayHours \* Wage2 \* 2;  **return** "$" + String.*valueOf*(result);  }  }  } |

1. Customer collaboration over contract negotiation – www.agilemanifesto.org [↑](#footnote-ref-2)
2. 월간 마이크로소프트웨어 2003년 6월호 ‘통합 테스트를 위한 프레임워크 FIT’, 강규영 [↑](#footnote-ref-3)