



BlueJ에서의 단위테스트 Unit Testing in BlueJ

Version 1.0
for BlueJ Version 1.3.0

한국어 버전 1.0
BlueJ Version 1.3.0用

Michael Kölling
Mærsk Insitute
University of Southern Denmark

번역: 황석형
선문대학교 컴퓨터정보학부



Copyright © M. Kölling

차 례

1. 소개	3
2. 단위테스트 기능설정	5
3. 테스트 클래스의 생성	6
4. 테스트 메소드의 생성	7
5. 테스트 시행	8
6. 테스트 결과 해석	12
7. 설비(Fixture)이란?	14
8. 테스트 설비의 생성과 사용	15
9. 수작업에 의한 테스트 메소드 작성	16
10. 테스트 먼저 작성하기	17
11. 다중 클래스 테스트	18
12. 요약	19

1. 소개

요약	BlueJ에서는 JUnit를 통합한 형태의 회귀테스트(regression testing)기능을 제공한다.
-----------	------------------------------------------------------------

1.1 튜토리얼 문서에 대하여(취급범위와 대상독자)

본 튜토리얼문서에서는 BlueJ환경에서 제공하고 있는 단위테스트기능을 소개한다. 이미 여러분들은 BlueJ의 일반적인 기능들에 대하여 익숙해 있을 것으로 가정한다. 그렇지 않을 경우에는 “BlueJ튜토리얼”문서를 먼저 읽어보기 바란다.

<http://www.bluej.org/doc/documentation.html>로부터 PDF문서를 입수할 수 있다.

또한, 독자 여러분들은 단위테스트에 대한(또는, 적어도 일반적인 테스트에 관한) 기본적인 사항들에 대해서 알고 있을 것으로 가정한다. 아래 절에서는 몇가지 기본사항에 대해서 설명하도록 한다.

1.2 단위 테스트란?

“단위테스트(unit testing)”라는 용어는 소프트웨어 시스템을 몇 개의 독립된 요소들로 분리하여 개발단위별로 테스트하는 것을 가리킨다. 객체지향 시스템에서는, 이와같은 개별 단위들(units)로서 클래스와 메소드가 이에 해당한다. 따라서, 본 문서에서 “단위테스트”라는 용어는 BlueJ환경하에서 메소드와 클래스에 대한 개별적인 테스트를 지칭한다.

본 튜토리얼 문서에서는 체계적인 단위테스트를 지원하는 BlueJ툴에 대해서 설명한다. 독자여러분들이 BlueJ의 상호작용적인 특성에 익숙하다면, 각 메소드들을 개별적으로 상호 작용하면서 테스트하기 수월하다는 사실을 발견하게 될 것이다. 이와같은 특징을 “임기응변식 특화형 테스트(ad-hoc testing)”라고 부르기로 한다.

특화형 테스트는 유용하지만 체계적인 테스트로서 충분하지 않다. BlueJ의 단위테스트 기능은 테스트를 기록하고 재시행하기 위한 도구를 제공한다. 이와같은 도구에 의해 (일반적으로 시스템을 변경한 후에) 단위테스트를 수월하게 반복할 수 있게 된다. 또한, 개발자는 최근의 변경에 의해 기존의 기능들을 훼손하지 않음을 보증할 수 있다. 이를 회귀테스트(regression testing)라고 부른다.

단위테스트와 회귀테스트는 고전적이지만, 최근의 XP(eXtreme Programming)방법론¹⁾ 뿐만아니라, Java를 위한 단위테스트 도구인 JUnit 등에서 최근까지도 널리 사용되고 있는 테스트방법론이다.

1) XP에 관한 상세한 사항들은, Kent Beck, "Extreme Programming Explained:Embrace Change" Addison Wesley, 1999.와 <http://www.xprogramming.com/xpmag/whatisxp.htm> 등을 참고하기 바란다.

JUnit는 Erich Gamma와 Kent Beck에 의해 제작된 회귀테스트용 프레임워크이다. <http://www.junit.org>로부터 관련 소프트웨어와 다양한 정보를 얻을 수 있다.

1.3 BlueJ에서의 단위테스트

BlueJ의 체계적인 테스트 도구는 JUnit을 기반으로 하고 있다. 따라서, JUnit 사용에 관한 몇가지 지식을 알고 있으면 BlueJ에서 단위테스트를 이해하는데 도움이 될 것이다. 이에 관한 관련 논문(현재로서는 제공되고 있지 않지만, 추후에 제공예정)을 읽어보기 바란다. 또한, JUnit의 웹사이트도 훌륭한 출발점이 될 것이다.

BlueJ의 단위테스트는, JUnit의 회귀테스트와 BlueJ의 상호작용적인 테스트기능을 혼합한 형태로 제공된다. 이 두가지 테스트기법들이 모두 제공되고 있다. 또한, 두가지 시스템을 조합한 새로운 기능도 제공되고 있다. 예를들면, 향후의 회귀테스트를 위한 JUnit테스트 메소드들 자동적으로 생성하기 위하여 상호작용적인 테스트 순서를 기록할 수 있다. 이에 대한 구체적인 예는 본 문서의 후반부에 설명하도록 한다.

BlueJ의 단위테스트기능은 Andrew Patterson에 의해 Monash University에서 박사과정 연구의 일환으로 설계/구현되었다.

2. 단위테스트 기능의 설정

요약	테스트 도구들은 설정메뉴(preferences)에서 설정변경을 하므로써 활성화된다.
-----------	------------------------------------------------

BlueJ에서는 테스트기능이 비활성화 상태로 초기화되어 있다. 테스트 도구를 사용하기 위해서는, "도구-옵션..."을 선택하고, "테스트도구 활성화(Show Unit Testing Tools)"레이블이 붙어있는 체크박스를 선택하도록 한다.

이와같이 하면, BlueJ인터페이스화면에 3개의 요소들(메인 창의 툴바에 몇가지 버튼들과 "recording"지시자가 활성화되고, "보기"메뉴에는 "테스트 결과보기(Show Test Results)"가 나타나며, 컴파일 완료된 클래스의 팝업 메뉴에는 "테스트클래스 생성(Create Test Class)"라는 항목이 활성화된다.

3. 테스트클래스 생성

요약	클래스의 팝업메뉴에서 “테스트클래스생성”항목을 선택하여 테스트클래스를 생성한다.
-----------	----------------------------------------------

BlueJ의 클래스 또는 메소드에 대한 테스트를 실시하기 위한 첫 번째 단계는 테스트클래스를 생성하는 것이다. 테스트클래스는 프로젝트 클래스(“참조클래스(reference class)”라고 부르기)와 연관되는 클래스로서, 테스트클래스에는 참조클래스의 메소드를 위한 테스트가 포함된다.

본 튜토리얼문서에서는 BlueJ배포판의 examples디렉토리에 포함되어있는 people프로젝트를 예제로 사용한다. 독자여러분들은 people프로젝트를 BlueJ를 오픈하여 본 튜토리얼문서를 읽어보면서 조작해 보기 바란다.

여러분들은 컴파일완료된 클래스에 대하여 마우스의 오른쪽버튼을 클릭하여 나타난 팝업 메뉴의 "테스트클래스 생성(Create Test Class)"항목을 선택하므로써 테스트클래스를 생성할 수 있다. 생성된 테스트클래스의 이름은 테스트대상 클래스의 이름뒤에 “Test”라는 접미어가 붙여져서 자동으로 지정된다. 예를들면, 테스트대상클래스의 이름이 "Student"인 경우, 생성되는 테스트클래스의 이름은 “StudentTest”가 된다.

테스트클래스는 그림1과 같이, <<unit test>>라는 태그가 붙여진 상태로 별도의 색깔이 부여되어, 테스트대상클래스에 첨부된 형태로 화면에 나타난다. 테스트대상클래스를 드래그 하더라도 해당 테스트클래스 또한 같이 움직임을 알 수 있다.

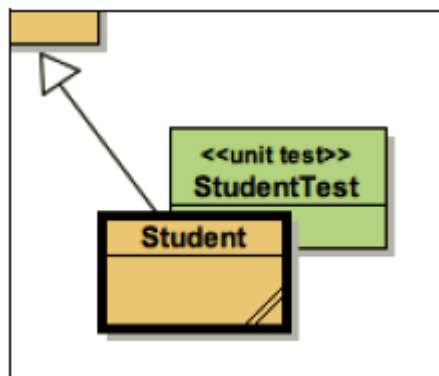


그림 1 테스트대상클래스와 테스트클래스

테스트클래스들은 BlueJ환경에서는 특별히 취급된다. 테스트클래스들은 기본적인 기능들(열기, 편집, 컴파일, 삭제)뿐만아니라, 테스트와 관련된 특수기능들(그림2)을 갖는다. 이와 같은 기능들이 활성화되기 위해서는 테스트클래스가 컴파일되어야 한다.

테스트클래스 자체를 생성하는 것만으로는 어떤 테스트도 만들어지지 않으며, 다만, 테스트

생성에 관한 선택사항들만을 사용자에게 제공할 뿐이다. 따라서, 테스트클래스는 앞으로 생성할 테스트를 보유하려는 목적으로 사용한다.

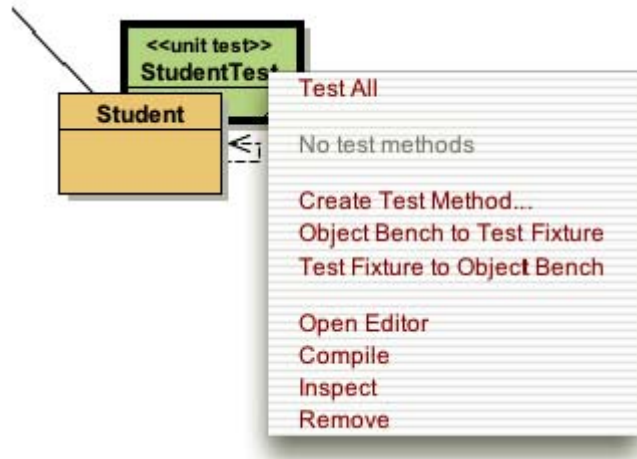


그림 2 테스트클래스의 팝업메뉴

4. 테스트메소드의 생성

요약	테스트클래스의 메뉴중에서 “테스트메소드생성(Create Test Method...)”를 선택하여 테스트메소드를 생성한다.
-----------	---------------------------------------------------------------------

Student클래스의 객체들은 2개의 메소드, 즉, setName과 getName(Person클래스로부터 계승)을 갖게 되며, 각각은 학생의 이름을 지정하거나 알아내기 위해 사용된다. 이러한 메소드들이 정상적으로 실행되는지를 점검하기 위하여 테스트를 생성해보기를 원한다고 가정해 보자.

StudentTest클래스에 대하여 Create Test Method...항목을 선택한다. 테스트메소드는 단일 테스트를 구현하게 된다(즉, 1개 단일 기능에 대한 테스트를 하는 것이다).

이와같은 기능을 선택한 후에, 해당 테스트에 대한 이름을 지정해야 한다. 테스트 이름은 항상 “test”라는 접두어를 붙여야하며, 이러한 규칙을 지키지 않았을 경우에는 자동적으로 “test”라는 접두어가 부여된다. 따라서, “testName” 또는 “name”을 입력하게 되면, “testName”이라는 테스트메소드가 생성된다.

테스트 이름을 지정한 후에 OK를 선택하면, 모든 상호작용들이 해당 테스트내용으로서 기록된다. “기록(recording)”지시등이 on상태로 되고, 해당 테스트 기록을 종료 또는 취소하기 위한 버튼이 활성화된다.(그림3)

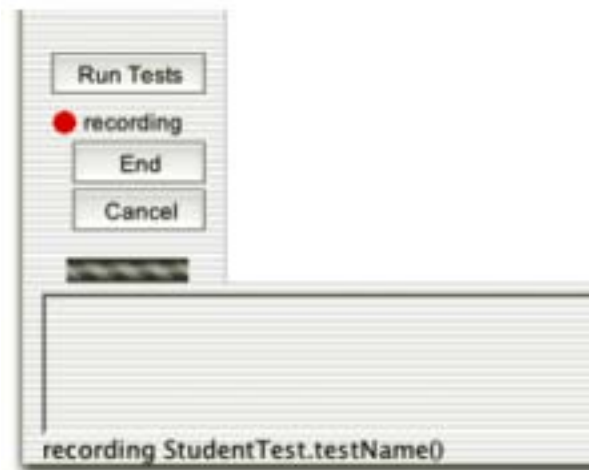


그림 3 테스트기록을 위한 테스트 버튼들

주어진 예제에서 테스트를 기록하기 위하여, 다음과 같이 수행한다:

- (1) 매개변수없이 생성자를 사용하여, Student객체를 생성한다.
- (2) Person클래스로부터 계승받은 setName(newName)메소드를 호출하여 "Fred"라는 이름을 부여한다.
- (3) getName()메소드를 호출한다.

getName메소드를 호출하게 되면 메소드 실행결과를 보여주는 화면이 나타난다. 테스트를 기록하는 동안, 결과화면에는 해당결과에 대한 검증사항들(assertions)을 지정할 수 있는 부분이 나타난다(그림4). 이와같은 검증항목을 이용하여 해당 테스트에 대한 기대결과를 지정할 수 있다. 해당예제에서는, "Fred"라는 문자열이 메소드호출에 대한 기대결과이므로, 이것을 검증사항 항목에 지정하였다(그림4).

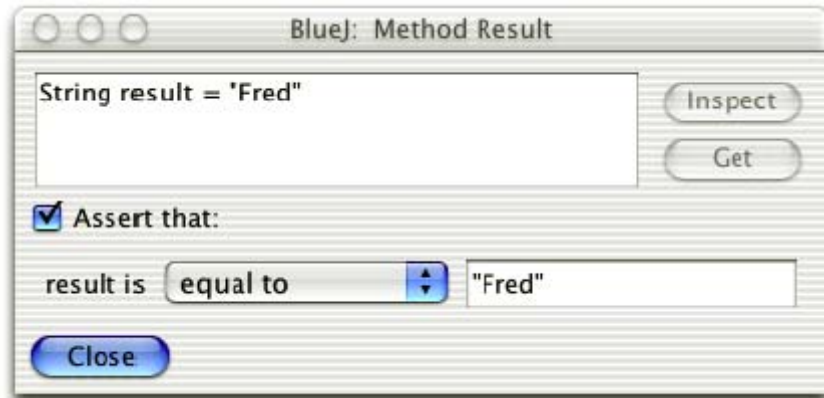


그림 4 검증사항(assertion)을 기입한 결과화면

동일성에 대한 테스트(tests for equality), null값인지 여부 등, 다양한 형태의 검증사항들을 지정할 수 있다.

위의 예에서 해당 테스트케이스가 완결되었으므로, 테스트 기록을 완료하기 위하여 테스트 기록 지시자 아래에 있는 "End"버튼을 클릭할 수 있다.

테스트를 종료함으로써, 테스트클래스에 해당 테스트메소드가 추가된다. 이와같은 테스트 메소드는 실행가능한 상태가 된다.

또한, 테스트기록을 종료하기 위하여 "Cancel"버튼을 누를 경우, 테스트기록은 폐기된다.

이상에서 살펴본 예제와 유사한 방법으로, 독자여러분들은 다양한 테스트를 기록할 수 있다. 프로젝트내의 각 클래스들은 자기자신을 위한 테스트클래스를 갖출 수 있으며, 각 테스트 클래스들은 다수의 테스트를 가질 수 있다.

각 테스트기록들은 여러개의 액션들로 구성될 수 있으며, 각 액션에는 여러개의 인스턴스 생성동작과 여러개의 검증사항(assertion)을 포함할 수 있다.

5. 테스트 실행

요약	“Run Tests”버튼을 클릭하여 모든 테스트를 실행한다. 개별적으로 테스트를 수행하고자 할 경우에는 테스트 클래스의 메뉴에서 선택하여 실행시킨다.
-----------	-------------------------------------------------------------------------------------

테스트들이 기록되면, 실행시킬 수 있다. 테스트클래스들은 참조클래스와 같은 형태의 Java클래스들이므로, 실행시키기 전에 컴파일되어야 한다.

BlueJ에서는 각각의 테스트들을 기록한 후에 테스트클래스들을 자동적으로 컴파일할 수 있다. 검증사항에 기입된 표현식에 에러가 포함되었을 경우, 또는 테스트클래스가 수작업에 의해 작성되었을 경우에는, 사전에 테스트클래스를 반드시 컴파일해야 한다.

이렇게 함으로써 이제 테스트클래스에 대하여 마우스 오른쪽 버튼 클릭을 하여 해당클래스의 팝업메뉴에 기록된 테스트를 살펴볼 수 있게 된다. 그림5에는 위에서 살펴보았던 테스트 메소드(testName)와 2번째 테스트(testStudentID)에 관한 예가 표시되어 있다.

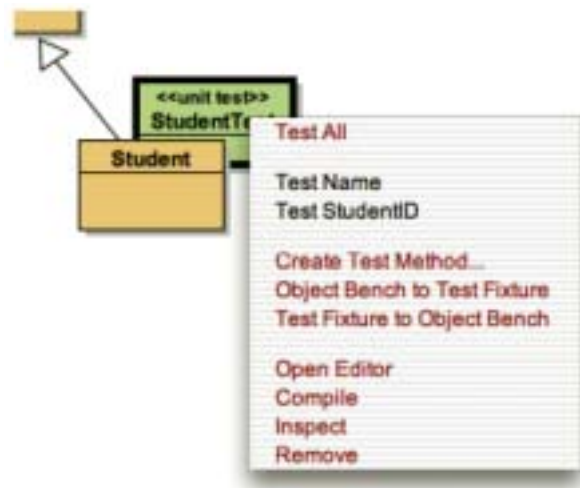


그림 5 2개의 테스트메소드가 표시된
테스트클래스의 메뉴

위에 표시된 메뉴에서 한개의 테스트를 선택함으로써 개별적으로 테스트를 수행할 수 있다. 또한, 테스트클래스의 메뉴에 있는 “Test All”항목을 선택함으로써 해당클래스에 정의된 모든 테스트를 실행시킬 수 있다.

개별적으로 테스트를 수행할 경우, 다음과 같이 두 가지 경우 중, 어느 한 가지 경우가 발생할 수 있다.

- (1)테스트가 성공(검증사항이 모두 만족)하는 경우, 성공을 알리는 설명문이 윈도우 하단에 있는 프로젝트 윈도우 상태바에 표시된다.
- (2)테스트가 실패(검증사항이 불만족되거나 기타 다른 문제점이 발생)하는 경우, 테스트결과 윈도우에는 해당테스트 실행결과에 대한 상세한 사항들이 표시된다(그림6).

만약 모든 테스트들을 실행시켰을 경우에는, 테스트결과 윈도우에는 테스트 출력결과가 표시된다.

메인 윈도우에 있는 테스트기록 지시자 상단에 있는 "Run Tests"버튼도 사용할 수 있다. "Run Tests"버튼을 활성화시키면 패키지내의 모든 테스트 클래스들에 있는 모든 테스트를 실행시킬 수 있다. 이와같은 방법은 해당 패키지에 대한 모든 테스트를 실행시키는 표준적인 방법이다.

6. 테스트결과에 대한 해석

요약	테스트결과 윈도우에는 테스트실행에 대한 요약과 더불어 실패한 경우의 세부사항까지 표시된다.
-----------	----------------------------------------------------

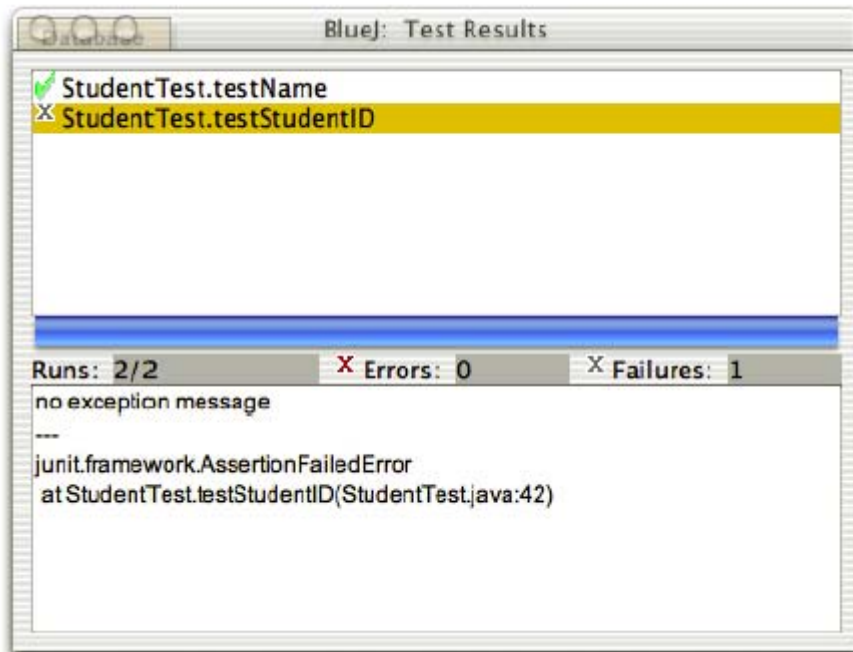


그림 6 테스트결과 윈도우

테스트가 실행되면, 테스트결과윈도우에는 테스트결과에 대한 요약내용이 출력된다(그림6). 테스트결과윈도우의 상단패널에는 실행된 모든 테스트의 목록이 성공 또는 실패를 알려주는 아이콘과 함께 표시된다. 녹색마크는 성공한 테스트를, 회색마크는 실패한 테스트, 그리고 빨간색마크는 에러를 각각 표시한다.

실행테스트와 에러 그리고 실패에 대한 각각의 건수들은 테스트결과윈도우의 중간부분에 요약되어 있다.

검증사항 중에서 만족되지 않는 것이 있을 경우, 해당 테스트는 실패(회색마크)한다. 예를 들면, 테스트검증사항이 어떤 특정한 메소드의 결과가 null이 되면 `없다`라고 규정할 수 있다.

테스트는 에러를 포함할 수도 있다. 테스트를 실행한 결과, 예상치 못한 예외처리가 발생하는 등의 에러가 발생하는 경우가 이에 해당한다.

성공하지 못한 테스트에 대하여, 테스트목록 중에서 해당테스트를 선택함으로써 테스트 실패에 관한 세부사항들을 화면에 표시할 수 있다. 테스트결과 윈도우의 최하단 패널에는 테스트 실패 또는 에러에 관한 상세한 정보가 표시된다.

테스트윈도우의 중간부분에 있는 바에는 테스트실행에 대한 주요사항이 요약되어 표시된다. 즉, 해당 바가 녹색이 되면, 모든 테스트가 성공했음을 표시하는 것이고, 빨간색인 경우에는 적어도 한개의 테스트가 실패하여 문제가 발생했음을 알려준다.

MacOS에서는 이와같은 바의 색깔이 변화하지 않음에 주의하기 바란다.

7. 설비(fixture)란?

요 약	테스트설비(test fixture)는, 테스트에 대한 시작 지점으로 사용되는 객체들의 집합이다.
------------	-------------------------------------------------------

때로는 테스트를 수행할 때, 실제 테스트를 시작하기에 앞서 몇가지 객체들을 준비해둘 필요가 있다. 예를들면, 몇가지 테스트를 위하여 'people'프로젝트내의 Database클래스에 대하여, Database객체와 Student객체 그리고 Staff객체 등을 준비하고, 또한, student 및 staff 멤버를 특정한 상태(name과 age값)로 설정해둘 필요가 있다.

독자여러분들은 이와같이 필요로하는 객체들을 생성하고 이들을 적절한 상태로 설정할 수 있다. 그러나, 보다 세련되게 테스트를 수행하려면 이와같은 방식은 지루할 수도 있으므로, 이와같은 방식보다 더 좋은 방안을 사용할 수도 있다.

특정한 테스트클래스에 대한 모든 테스트를 수행하기 위한 시작점으로서, 특정한 상태를 갖는 객체들을 Object bench에 만들어 둘 수 있다. 이와같이 테스트를 위해서 사전에 만들어지는 객체들의 집합을 테스트설비(fixture)라고 부른다.

테스트설비는 사전에 정의될 수 있으며, 동일한 테스트클래스의 모든 테스트 시작에 앞서서 자동적으로 설정될 수 있다. 따라서, 각각의 개별적인 테스트에 대한 부담을 경감시킬 수 있게 된다.

8. 테스트설비(test fixture)의 생성과 사용

요약	테스트클래스에 대한 테스트설비를 생성하기 위하여, 필요시되는 객체들을 object bench에 생성하고, 테스트클래스의 메뉴중에서 "ObjectBench To Test Fixture" 항목을 선택한다.
-----------	------------------------------------------------------------------------------------------------------------------

필요시되는 객체들을 생성하여 테스트설비를 만들고, 필요한 상태를 설정하기 위하여 해당 객체에 대하여 메소드를 호출한다.

예를들면, people프로젝트에 있는 database클래스를 테스트하기 위하여, Database객체와 "Fred"라는 이름을 갖는 Student객체가 필요하게 된다. "Fred"라는 이름을 갖는 Student객체는 database에 격납시키고, "Jane"이라는 이름을 갖는 Staff객체는 database에 격납시키지 않는다. 이를 위하여, 해당객체들을 생성시키고 "Fred"를 database에 격납시키기 위해 필요한 메소드 호출을 수행한다.

object bench의 상태가 현재 수행하려는 테스트를 시작하기에 적합하다면, DatabaseTest 클래스의 "ObjectBench To Test Fixture"기능을 선택한다.

이와같은 기능을 선택함으로써 Object Bench로부터 모든 객체들이 삭제되면서, 해당클래스에 대한 테스트설비가 생성된다.

임의의 클래스가 테스트설비를 갖게 되면, 모든 테스트의 시작시기에 해당설비가 재생성될 것이다. 예를들면, Database클래스에 대해 (해당 테스트클래스의 "Create Test Method"를 선택하여)새로운 테스트를 생성했을 경우, 테스트설비에 정의된 상태가 자동적으로 재저장된다. 이와같은 테스트설비내의 객체들은 테스트기록을 시작할 때 정의된 상태로 Object Bench에 나타난다.

또한, 테스트설비상태는 테스트클래스의 메뉴에 있는 "Test Fixture To Object Bench"를 선택하므로써 Object Bench에 다시 생성될 수도 있다. 이와같이 함으로써 테스트설비를 나중에 확장(새로운 테스트 메소드가 테스트설비 객체를 추가적으로 필요로)할 때 유용하다.

이와같은 경우, 테스트설비 상태를 다시 생성하기 위하여 "Test Fixture To Object Bench"를 사용할 수 있으며, 수작업에 의해 설비 상태에 필요시되는 추가사항을 기입한다. 이와같은 확장작업이 종료되면, 테스트설비를 저장하기 위하여 "Object Bench To Test Fixture"를 선택한다. 변경전의 테스트설비는 새로운 것으로 바뀌게 된다.

9.수작업에 의한 테스트 메소드 작성

요약	테스트 메소드는 테스트클래스의 소스코드에 직접 작성해 넣을 수 있다.
-----------	----------------------------------------

사용자와의 상호작용과 object bench상태를 기록하여 테스트 메소드들과 테스트설비들을 생성하는 것은 단위테스트를 작성하는 일개 수법에 불과하다. 또다른 방법으로서, 수작업에 의해 테스트 메소드들을 작성할 수도 있다.

테스트클래스는 일반 프로젝트에 있는 클래스들과 마찬가지로 Java클래스이므로, 같은 방식으로 취급할 수 있다. 즉, 소스코드를 보기 위하여 에디터를 오픈하여 편집할 수도 있고, 컴파일과 실행 등을 수행할 수도 있다.

JUnit의 전통적인 사용방법(BlueJ사용법이 아님)에 있어서는, 수작업에 의한 테스트메소드 작성법은 표준적인 방법이며, BlueJ에서도 이와같은 방법을 사용할 수 있다. 테스트를 상호작용방식으로 기록하는 것은 수작업에 의한 테스트 작성방식에 추가된 내용일 뿐, 대체방법이 아니다.

JUnit에 친숙하지 않은 독자들은, 상호작용방식으로 테스트 설비와 몇가지 테스트 메소드를 작성하여 테스트클래스의 소스코드를 검사하는 것이 편리할 것이다. 각각의 테스트클래스들은 setUp()이라는 메소드를 갖게 되며, 이러한 setUp()메소드는 테스트 설비를 구성하는데 사용된다는 사실에 주목하기 바란다. 이때, setUp()메소드는 각 테스트에 대한 추가적인 메소드를 갖는다.

기존의 테스트 메소드에 대하여, 수작업으로 동작을 수정하거나, 완전히 새롭게 수작업으로 테스트 메소드를 추가하기 위하여 편집하는 것은 바람직하다. 특히, 테스트 메소드의 이름을 "test"라는 접두어로 시작되게 만들어서 테스트 메소드임을 알 수 있도록 해야 한다.

JUnit테스트 작성에 관하여 보다 상세한 사항을 알아보기 위해서는 본 튜토리얼 문서의 종반부에 있는 JUnit관련 참고문헌을 읽어보기 바란다.

10. 테스트를 먼저 작성하기

요 약	구현하기 전에 테스트를 작성하기 위해서는, 수작업에 의해 테스트를 작성하거나 메소드 스템브(method stub)를 사용해야 한다.
------------	---------------------------------------------------------------------------

XP(eXtreme Programming)기법에 의하면, 테스트는 메소드를 구현하기전에 작성되어야 한다고 한다. BlueJ의 단위테스트 통합기능을 사용함으로써, 다음과 같이 2가지 방법으로 이와같은 문제를 해결할 수 있다.

첫 번째 방법으로서, 앞절에서 설명한바와 같이, 수작업에 의해 테스트를 작성한다. 테스트를 작성하고 JUnit에서 설명하고 있는 구현방식으로 작업한다.

두 번째 방법으로서, 참조클래스내에 메소드 스템브를 작성한다. 이러한 메소드 스템브의 리턴값은 임의의 값(dummy value)으로 하되, void형이 되어서는 않된다. 이와같이 작성한 후에, 상호작용방식의 기록기능을 이용하여 테스트를 생성할 수 있다. 이때, 완성된 구현 부분에 대한 기대결과값에 대응하는 점검사항을 작성하여 기입하도록 한다.

11. 다중클래스 테스트

요 약	"New Class..."를 이용(단, 클래스 유형은 "Unit Test"으로 설정)하여 비의존적인 형태의 테스트 클래스를 생성할 수 있다.
------------	--------------------------------------------------------------------------------

앞서 살펴보았던 예제들에서는 참조클래스에 첨부된 형태의 테스트 클래스가 사용되었다. 이와같이 테스트클래스를 참조클래스에 첨부하는 방법으로는 다른 클래스 타입에 대한 테스트에 테스트클래스를 사용할 수 없다.

경우에 따라서는, 여러개의 클래스들을 조합하여 테스트를 수행하도록 테스트클래스가 작성되어야 할 경우도 있다. 즉, 한개 클래스에만 논리적으로 연관시키지 않는 형태의 테스트 클래스를 작성할 수도 있다는 것이다. 이를 위해서, 테스트클래스들은 한개의 클래스에만 직접 첨부되지 않도록 해야 한다.

이를 위하여, "New Class..."기능을 이용하여 독립된 형태의 테스트클래스를 생성할 수 있으며, 클래스 생성시에 클래스의 유형을 "Unit Test"으로 선택해야 한다.

독립된 형태로 만들어진 테스트클래스들은 기존의 테스트클래스들과 동일한 방법으로 사용할 수 있다. 즉, 테스트설비 생성, 테스트메소드 작성, 그리고 테스트 시행 등이 가능하다.

12. 요약

1. BlueJ에서는 JUnit를 통합한 형태의 회귀테스트(regression testing)기능을 제공한다.
2. 테스트 도구들은 설정메뉴(preferences)에서 설정변경을 함으로써 활성화된다.
3. 클래스의 팝업메뉴에서 “테스트클래스생성”항목을 선택하여 테스트클래스를 생성한다.
4. 테스트클래스의 메뉴중에서 “테스트메소드생성(Create Test Method...)”를 선택하여 테스트메소드를 생성한다.
5. “Run Tests”버튼을 클릭하여 모든 테스트를 실행한다. 개별적으로 테스트를 수행하고자 할 경우에는 테스트 클래스의 메뉴에서 선택하여 실행시킨다.
6. 테스트결과 윈도우에는 테스트실행에 대한 요약과 더불어 실패한 경우의 세부사항까지 표시된다.
7. 테스트설비(test fixture)은, 테스트에 대한 시작지점으로 사용되는 객체들의 집합이다.
8. 테스트클래스에 대한 테스트설비를 생성하기 위하여, 필요시되는 객체들을 object bench에 생성하고, 테스트클래스의 메뉴중에서 “ObjectBench To Test Fixture” 항목을 선택한다.
9. 테스트 메소드는 테스트클래스의 소스코드에 직접 작성해 넣을 수 있다.
10. 구현하기 전에 테스트를 작성하기 위해서는, 수작업에 의해 테스트를 작성하거나 메소드 스티브(method stub)를 사용해야 한다.
11. "New Class..."를 이용(단, 클래스 유형은 “Unit Test”으로 설정)하여 비의존적인 형태의 테스트 클래스를 생성할 수 있다.