클래스 다이어그램 실습 문제 풀이

( 3/28 )

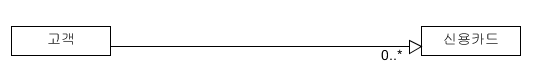
멀티미디어공학과

1191078

임인혁

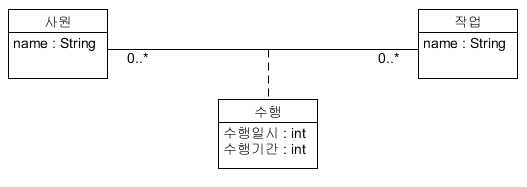
1. 다음 설명에 맞는 클래스 다이어그램을 작성하라.

|  |
| --- |
| * 고객은 여러 개의 신용카드를 소유할 수 있다. * 신용카드가 없는 고객도 있다. * 신용카드에는 어떤 고객 정보도 없다. |

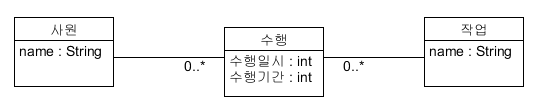


1. 다음 설명에 맞는 클래스 다이어그램을 연관 클래스를 사용하여 작성하라.

|  |
| --- |
| * 사원은 여러 개의 작업을 수행한다. * 한 작업은 여러 명의 사원에 의해 실행될 수 있다. * 작업 일시와 작업기간이 관리 되어져야 한다. |

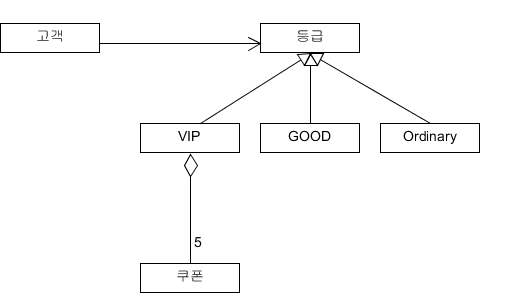


1. 문제 2를 연관 클래스를 사용하지 않고 클래스 다이어그램을 작성하라.



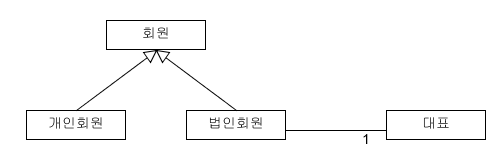
1. 다음 클래스 다이어그램의 설명으로 올바르지 않은 것은?

|  |
| --- |
| * 고객의 등급으로 VIP, Good, Ordinary 등급으로 나누어진다. * VIP 등급의 고객은 여러 개의 쿠폰을 가지고 있다. * VIP 등급의 고객은 다섯 장의 쿠폰이 발급된다. * 쿠폰을 발급받은 VIP 회원이 탈퇴하면 쿠폰은 다른 회원에게 양도 될 수 있다. |

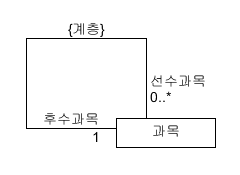


1. 다음 설명에 맞는 클래스 다이어그램을 작성하라.

|  |
| --- |
| * 회원은 법인회원과 개인회원으로 구분된다. * 법인회원의 대표는 한 명이다. |

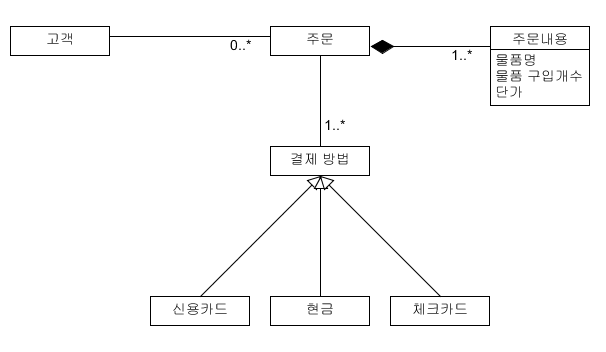


1. 한 과목은 여러 개의 선수 과목을 가질 수 있는 상황을 UML 클래스 다이어그램으로 모델링 하라

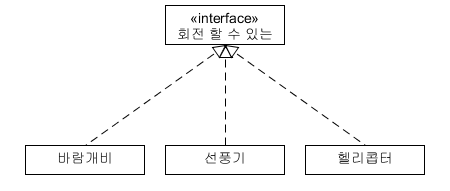


1. 다음 설명에 맞는 클래스 다이어그램을 작성하라.

|  |
| --- |
| * 고객은 여러 주문을 할 수 있다. * 한 주문에 대해 여러 방법으로 결제 할 수 있다. * 결제 방법으로 신용카드, 현금, 체크카드가 있다. * 각 주문은 고객이 구입한 여러 물품들에 대한 정보(물품명, 물품 구입개수, 단가)가 있다. |



1. “바람개비”, “선풍기”, “헬리콥터” 등은 회전할 수 있는 개체이다. 이 특성을 이용하여 이들을 클래스 다이어그램으로 모델링 하라.



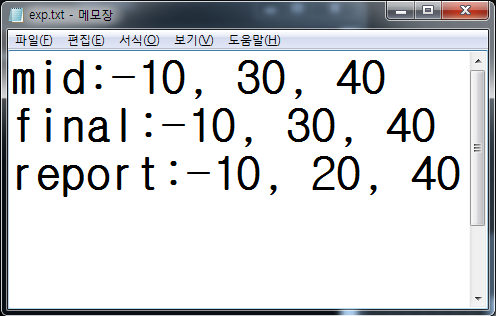
JUnit 실습문제

( 5/30 )

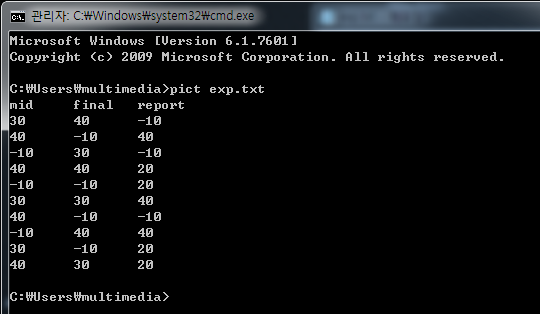
멀티미디어공학과

1191078

임인혁



[PICT 내용]



[결과창]

public class Grade {

public Grade(int middlescore, int finalscore, int reportscore){

rs = middlescore + finalscore + reportscore;

}

public char getresult() throws InvalidScopeException{

if(middlescore<0 || middlescore>35)throw new InvalidScopeException();

if(finalscore<0 || finalscore>35) throw new InvalidScopeException();

if(middlescore<0 || reportscore>30) throw new InvalidScopeException();

if(rs >= 80) grade = 'A';

else if(rs >= 70) grade = 'B';

else if(rs >= 60) grade = 'C';

else grade = 'F';

return grade;

}

}

위의 학점 프로그램 코드에서는 각각의 점수들을 합해서 Grade를 ‘A’와 ‘B’, ‘C’ 그리고 ‘F’로 구분 짓는 코드이다. 중간점수와 기말점수는 0~35점 이라는 조건을 만족하여야 하고, 과제점수는 0~30점 이라는 조건을 만족하여야 이 학점 프로그램이 정상적으로 돌아갈 수 있다.

@RunWith(value=Parameterized.class)

public class GradeTest {

@Parameters

public static Collection<Object[]> getParameters(){

return Arrays.asList(new Object[][]{

{'C', 30, 40, -10},

{'A', 40, -10, 40},

{'F', -10, 30, -10},

{'A', 40, 40, 20}

{'F', -10, -10, 20},

{'A', 30, 30, 40},

{'F', 40, -10, -10},

{'B', -10, 40, 40}

});

}

public GradeTest(char expected, int val1, int val2, int val3){

this.expected = expected;

this.val1 = val1;

this.val2 = val2;

this.val3 = val3;

}

@Test

public void test() throws Exception{

Grade grade = new Grade(val1, val2, val3);

grd = grade.getresult();

assertEquals(expected, grade.getresult());

}

}

{'C', 30, 40, -10},

{'A', 40, -10, 40},

{'F', -10, 30, -10},

{'A', 40, 40, 20},

{'F', -10, -10, 20},

{'A', 30, 30, 40},

{'F', 40, -10, -10},

{'B', -10, 40, 40}

PICT 프로그램에 설정된 결과 값을 돌리면 경우의 수를 체크해서 각각의 경우들을 다 설정 해 주는데 그 값을 점수에 반영 하고 점수들을 더해서 예상 된 결과를 예상 Grade 값에 넣어주고 Junit test를 하면 그에 적절한 결과가 출력 된다.

Selenium 개요와 프로젝트에 Selemnium 테스트 적용

( 6/12 )

멀티미디어공학과

1191078

임인혁

**목적**

Selenium은 다양한 브라우저와 다양한 테스트 작성 언어(Java, Ruby, Groovy, Python, PHP, and Perl)을 지원하는 웹 어플리케이션의 ‘자동화 테스트’를 위한 테스팅 프레임워크이다. 실제 브라우저를 가동하여 테스트하는 방식이기 때문에, 브라우저별로 UI 테스트를 실행하기 용이하다. Selenium WebDriver가 제공하는 Java API를 JUnit 테스트 코드에서 사용하면 손쉽게 브라우저 테스트 코드를 제작할 수 있다.

**설치과정**

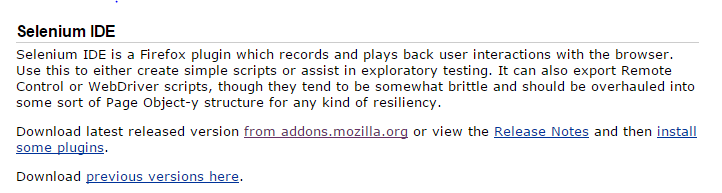
- Pom.xml 설정

<dependency>  
 <groupId>org.seleniumhq.selenium</groupId>  
 <artifactId>selenium-java</artifactId>  
 <version>2.47.1</version>  
 <scope>provided</scope>  
</dependency>

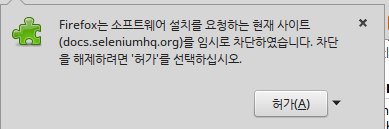
-Selenium IDE 설치

FireFox 설치 후

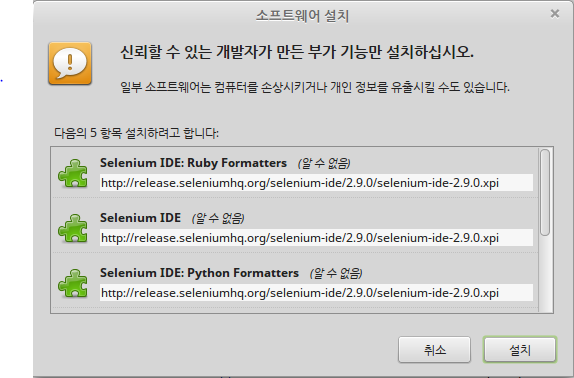
<http://docs.seleniumhq.org/download/> 접속 해서 Selenium IDE 다운로드



Selenium 허가

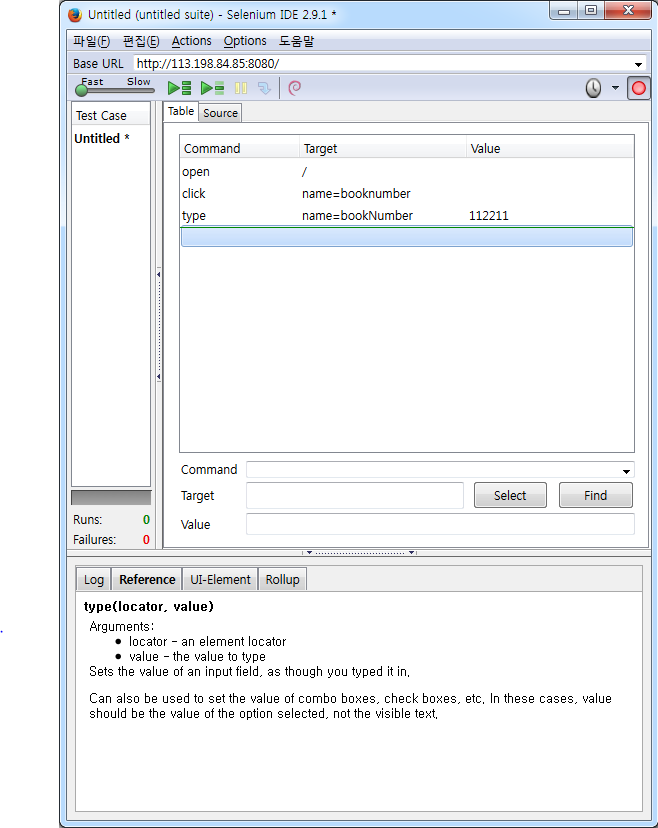


Selenium IDE 설치



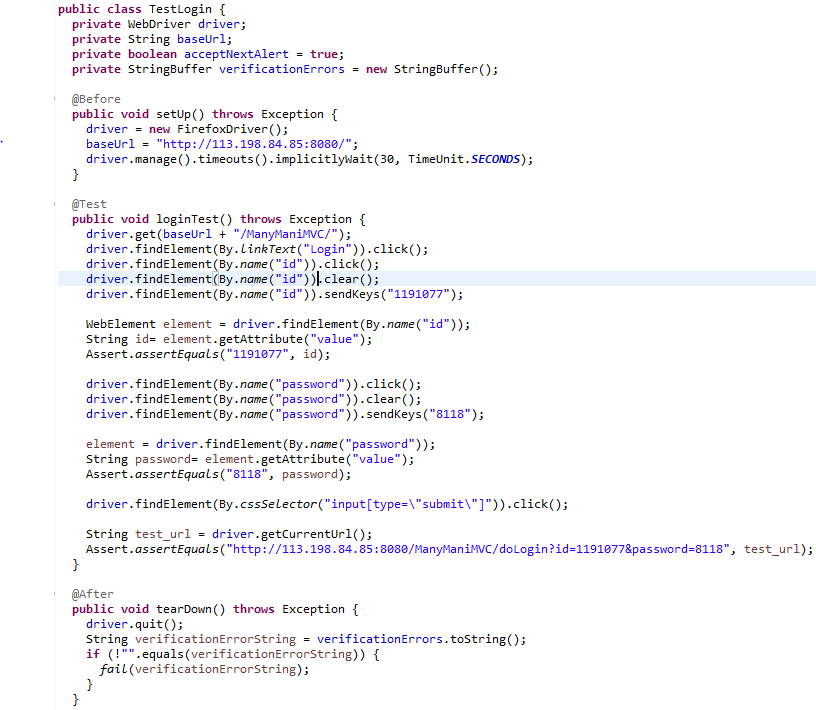
**동작 설명 및 코드**

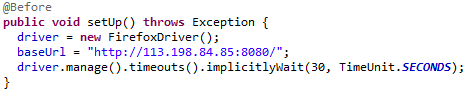
Selenium IDE 실행 후 시나리오 설정



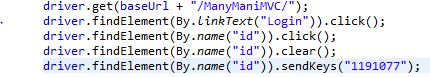
**Selenium TestLogin 코드**

TestLogin 코드





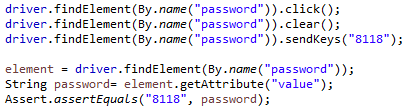
@Before에서 자동화 테스트를 실행하기 전 전처리 작업을 해줬습니다. Driver를 FireFox로 설정 해주고 baseUrl을 설정 했습니다. 그리고 @Test가 실행 되면서 자동화 테스트를 실행 합니다.



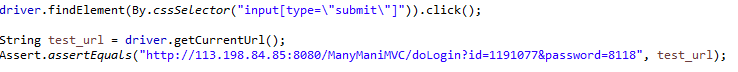
먼저 driver.get(baseUrl + “/ManyManiMVC/”)로 드라이버를 이동 시키고 driver에게 Login이라는 LinkText의 클릭 설정하고 driver에게 id라는 name을 찾아서 클릭한 후에 1191077을 입력하도록 했습니다.



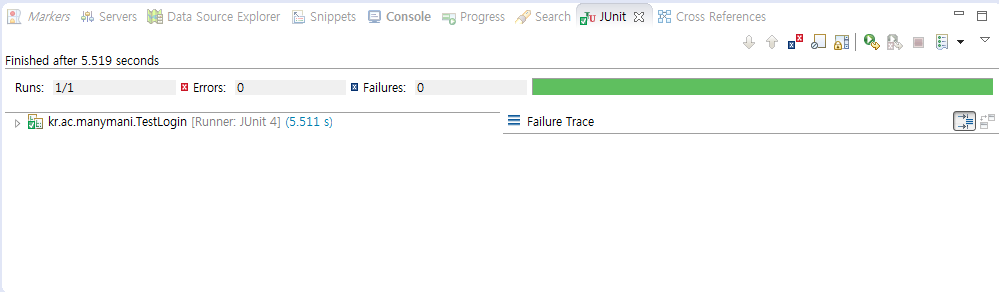
id라는 영역에 1191077이 제대로 입력 됐는지 확인하도록 했습니다.



Password를 입력하는 것도 id와 똑같이 했습니다.

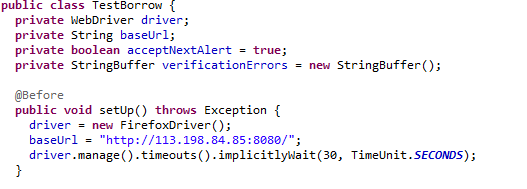


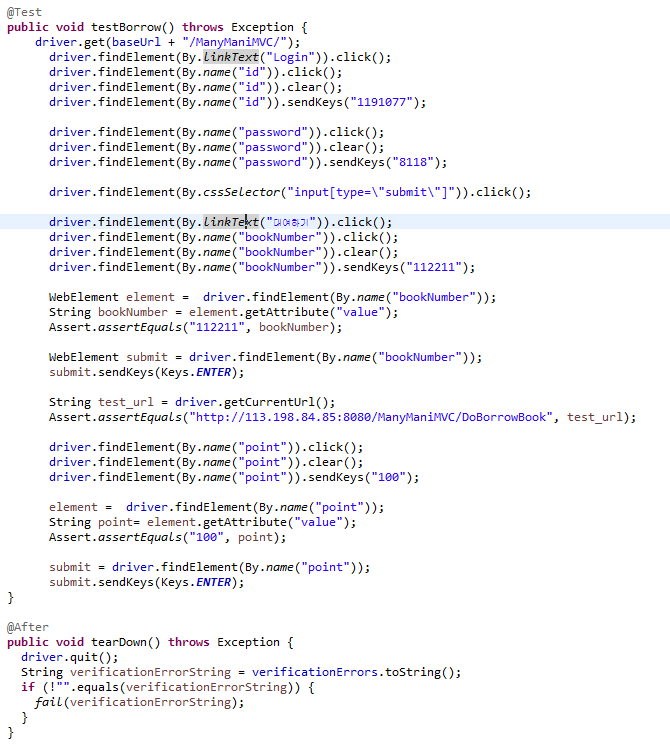
마지막으로 로그인이 성공 했을 경우 이동 되는 url을 확인하는 작업을 했습니다.

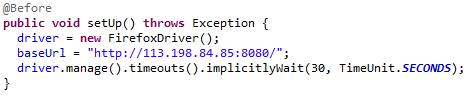


JUnit Test가 성공 했다는 것을 확인 했습니다.

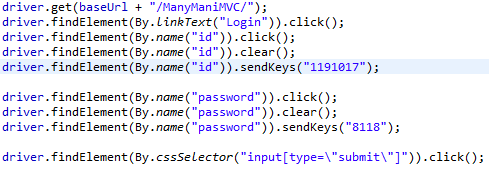
TestBorrow 코드



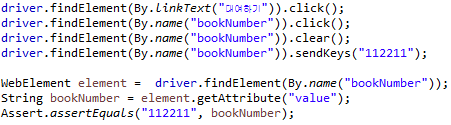




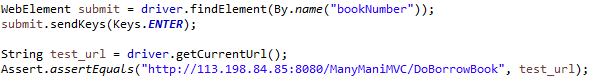
@Before에서 자동화 테스트를 실행하기 전 전처리 작업을 해줬습니다. Driver를 FireFox로 설정 해주고 baseUrl을 설정 했습니다. 그리고 @Test가 실행 되면서 자동화 테스트를 실행 합니다.



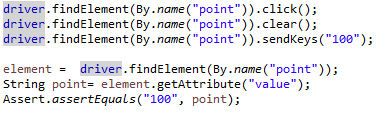
로그인을 설정했습니다.



대여하기라는 LinkText를 클릭하고 bookNumber를 찾아 클릭 한 후 sendKeys(“112211”)로112211을 입력하도록 설정 했습니다. 그리고 Assert.assertEquals(“112211”, bookNumber)로 bookNumber에 들어있는 값이 112211이 맞는지 확인했습니다.

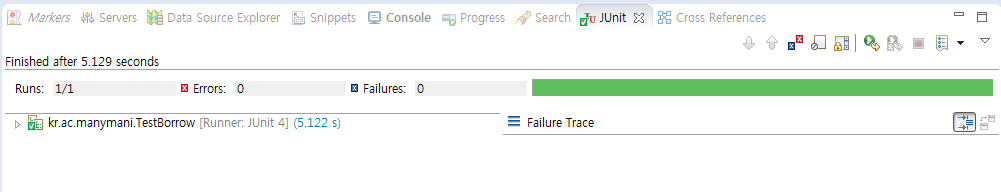


bookNumber를 입력했을 경우에 Enter키를 누르도록 설정하고 그 다음 url이 제대로 변경 됐는지 확인 했습니다.





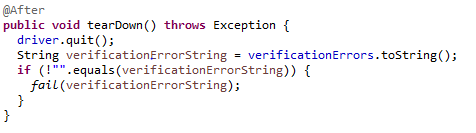
point라는 name을 찾아서 100을 입력하도록 했고 point라는 곳에 입력 된 값이 100이 맞는지 확인 했고 Enter키를 입력하도록 했습니다.

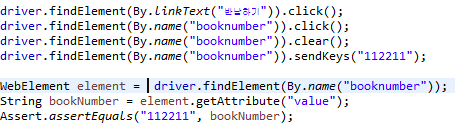


JUnit Test가 성공 했다는 것을 확인 했습니다.

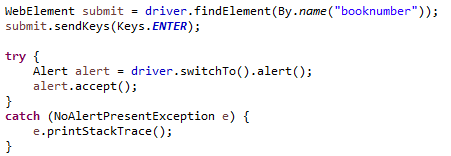
TestReturn 코드







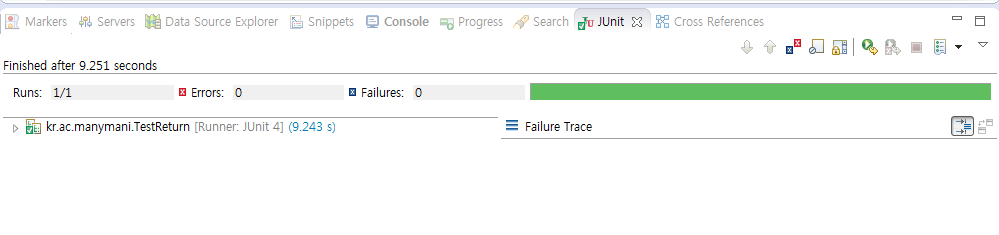
반납하기라는 LinkText를 찾아서 클릭한 후 booknumber라는 name을 찾아서 클릭하고 112211이라는 값을 입력하게 했습니다. 그리고 booknumber라는 곳에 입력 된 값이 112211이 맞는지 확인 하도록 설정했습니다.



입력 된 값을 보내기 위해서 booknumber에서 Enter키를 입력하도록 했고 반납이 성공 됐다는 alert 창이 나타나면 확인 버튼을 누르도록 설정 했습니다.

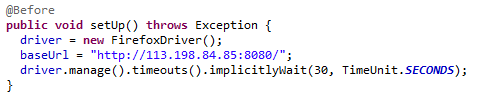


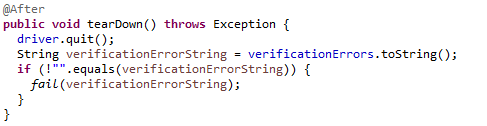
마지막으로 성공 했을 경우 url이 제대로 변경 됐다는 것을 확인 했습니다.

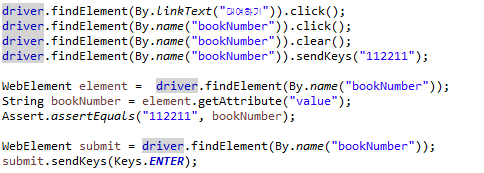


JUnit test가 성공 했다는 것을 확인 했습니다.

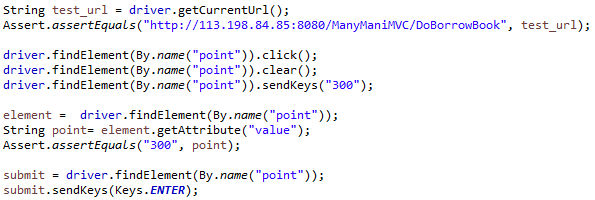
TestBorrowPointError 코드



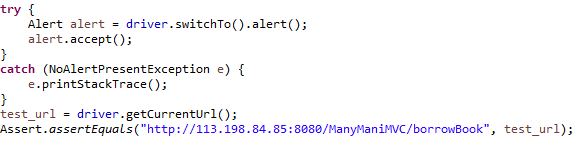
 



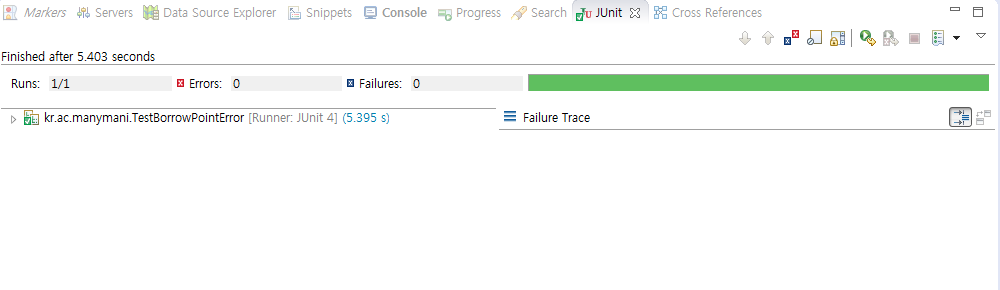
대여하기라는 LinkText를 찾아서 클릭 하도록 했고 bookNumber라는 name을 찾아서 클릭하고112211이라는 값을 입력하도록 했습니다. 그리고 bookNumber라는 곳에 112211이라는 값이 제대로 입력 됐는지 확인 했고 Enter키를 입력 하도록 설정 했습니다.



이동되는 url을 확인했고 point라는 name을 찾아서 클릭하고 포인트를 300 입력하도록 했습니다. 여기서 300이라는 포인트 값은 테스트 하는 사용자가 보유하고 있는 point보다 많은 포인트이며 보통의 시나리오와 다른 시나리오로 진행 됩니다.



포인트가 부족하다는 alert 창이 실행되면 확인 키를 누르게 했고 그 결과로 이동 된 url이 정확한지 확인 했습니다.



JUnit test 성공 화면입니다.