

스프링 스터디 4주차 (2021.09.05)

스프링 핵심 원리 - 기본편

- 섹션 2. 스프링 핵심 원리 이해1 - 예제 만들기
- 섹션 3. 스프링 핵심 원리 이해2 - 객체 지향 원리 적용
- 섹션 4. 스프링 컨테이너와 스프링 빈



섹션 1에서 배운 이론들을 활용해 회원 구매 관리 예제 코드를 구현해 보았다!

1. 순수한 자바 코드로 구현
2. 객체 지향 원리를 적용 (SRP, OCP, DIP 고려)
3. IoC, DI, 스프링 코드 적용
4. 스프링 컨테이너 심화 적용

Test 코드 작성의 중요성

테스트 코드 작성 전에 psvm(public static void main)작성 후, 테스트 예시를 보여주시면서...



요즘 현대적인 어플리케이션 개발하려면,
테스트 코드는 선택이 아니라 필수로 아셔야 해요~
저희 팀에서도 거의 60퍼센트의 개발자 친구들이 테스트 코드를 개발하고 있습니다.
그만큼 테스트 코드의 작성은 중요합니다~~

그리고 "단위 테스트"에 대해서도 다시 한번 강조하신다.



이러한 테스트 코드 작성.. 뭔가 익숙한데..?
코딩 테스트의 테스트 케이스 검사!

```
테스트 1
    입력값 > "aabbaccc"
    기댓값 > 7
    실행 결과 > 테스트를 통과하였습니다.

테스트 2
    입력값 > "ababcdcdababcdcd"
    기댓값 > 9
    실행 결과 > 테스트를 통과하였습니다.

테스트 3
    입력값 > "abcabcdede"
    기댓값 > 8
    실행 결과 > 테스트를 통과하였습니다.

테스트 4
    입력값 > "abcabcbcabcdedededede"
    기댓값 > 14
```

다른 사람의 풀이 초기화 코드 실행 제출 후 채점하기

프로그래머스의 코드 실행 버튼 클릭 시 테스트 케이스 테스트 화면 (저 테스트 케이스만 맞아도 맞게 해줬으면...)

[문제 ref. <https://programmers.co.kr/learn/courses/30/lessons/60057>]

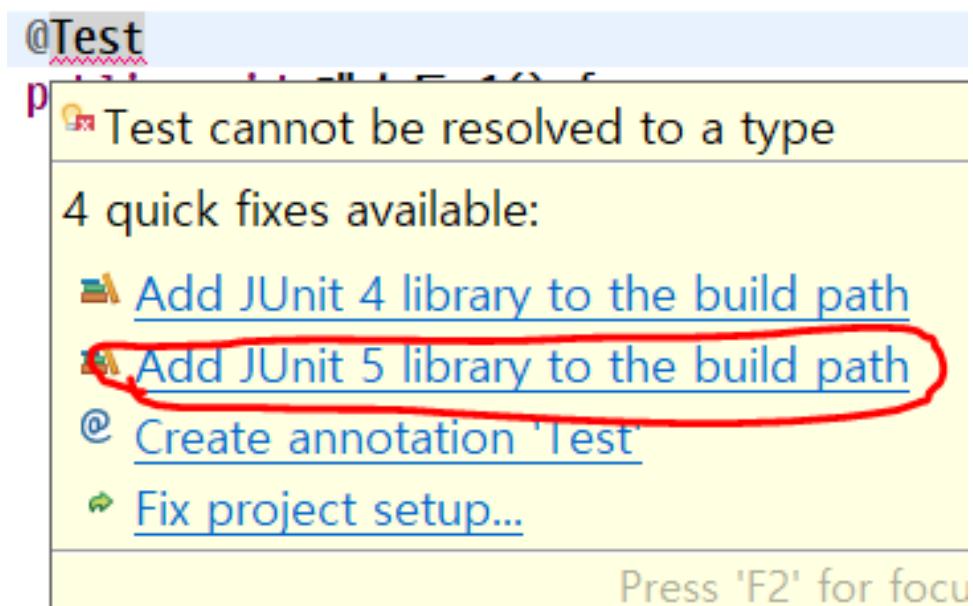
직접 입력하고 출력 결과를 하나하나 확인하는 방법도 좋지만, 테스트 코드를 작성해서 반복을 줄여 디버깅 시간을 절약하고, 처음에 잘 작성해 놓으면 테스트 케이스에서 오류가 날 확률이 적어지므로 신뢰성과 안정성을 높일 수 있다.

코딩 테스트에서 주어진 테스트 케이스들로 내 코드를 테스트하는 코드를 구현해보자!

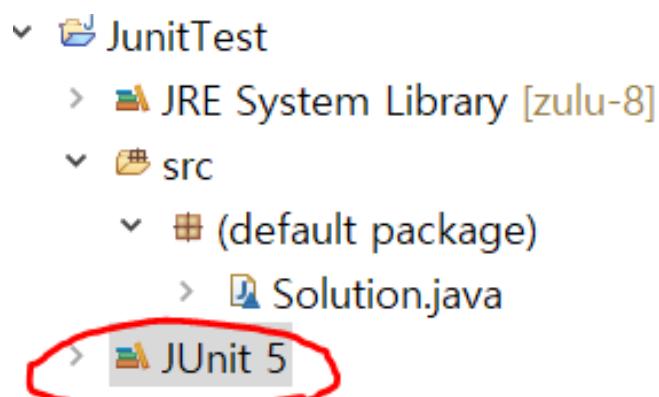
Eclipse에서의 Test 코드 작성

코딩 테스트에서 IDE로 많이 활용되는 **Eclipse**에서 진행했고, **JUnit5**를 활용했다!

JUnit 5 build path 추가



직접 build path를 추가해주는 방법도 있지만 @Test 어노테이션을 Solution 클래스 안에 작성하면 빨간줄이 그어지면서 quick fix가 가능하다.



이렇게 JUnit 5가 추가 되어서 JUnit 5를 활용할 수 있게 되었다!

Test 코드 작성

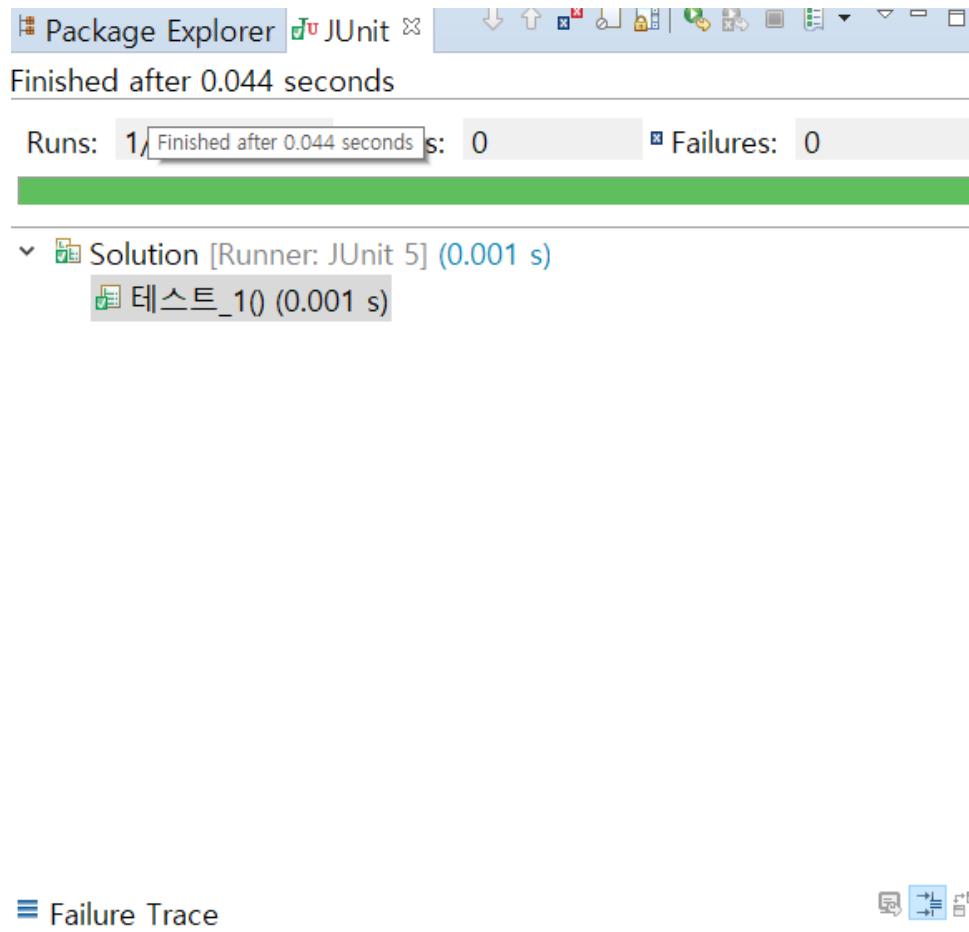
```
@Test
public void 테스트_1() {
    // given
    int answer = 7;

    // when
    String input = "aabbaccc";
    int myAnswer = solution(input);

    // then
    Assert.assertEquals(answer, myAnswer);
}
```

테스트 1 코드

간단하게 Solution class 안에서 테스트 코드 작성은 진행해 보았다. 세 부분으로 나눠서 첫 번째 테스트 케이스에 대한 내용을 테스트 했다.



JUnit 테스트 결과 화면

결과 화면의 모습으로, 테스트 케이스들을 더 추가 할 수도 있을 것이다.

작성 코드

```
import org.junit.Assert;
import org.junit.jupiter.api.Test;

class Solution {
    public int solution(String s) {
        int answer = s.length();

        for(int i=1;i<=s.length()/2;i++){
            // i개 단위로 잘라서 압축
            int zipLen = s.length();
            int cnt = 1;
            for(int j=i;j<=s.length()-i;j+=i){
                String prev = s.substring(j-i, j);
                String cur = s.substring(j, j+i);
            }
        }
    }
}
```

```

        if(prev.equals(cur)){
            // 이전 문자 단위와 같으면
            zipLen -= i;
            cnt++;
        }else if(cnt > 1){
            // 이전 문자 단위와 다르고
            // 압축중이었으면
            zipLen += Integer.toString(cnt).length();
            cnt = 1;
        }
    }

    // 압축중이었으면 계산했던 cnt값 반영
    if(cnt > 1) zipLen += Integer.toString(cnt).length();

    answer = Math.min(answer, zipLen);
}

return answer;
}

@Test
public void 테스트_1() {
    // given
    int answer = 7;

    // when
    String input = "aabbaccc";
    int myAnswer = solution(input);

    // then
    Assert.assertEquals(answer, myAnswer);
}

@Test
public void 테스트_2() {
    // given
    int answer = 9;

    // when
    String input = "ababcdcdababcdcd";
    int myAnswer = solution(input);

    // then
    Assert.assertEquals(answer, myAnswer);
}

@Test
public void 테스트_3() {
    // given
    int answer = 8;

    // when
    String input = "abcabcdede";
    int myAnswer = solution(input);

    // then
}

```

```

        Assert.assertEquals(answer, myAnswer);
    }

    @Test
    public void 테스트_4() {
        // given
        int answer = 14;

        // when
        String input = "abcabcabcabcdededede";
        int myAnswer = solution(input);

        // then
        Assert.assertEquals(answer, myAnswer);
    }

    @Test
    public void 테스트_5() {
        // given
        int answer = 17;

        // when
        String input = "xababcdcdababcdcd";
        int myAnswer = solution(input);

        // then
        Assert.assertEquals(answer, myAnswer);
    }

    @Test
    public void 통합_테스트() {
        테스트_1();
        테스트_2();
        테스트_3();
        테스트_4();
        테스트_5();
    }

    @Test
    public void 통합_테스트_한번에_입력() {
        Assert.assertEquals(7, solution("aabbaccc"));
        Assert.assertEquals(9, solution("ababcdcdababcdcd"));
        Assert.assertEquals(8, solution("abcabcdede"));
        Assert.assertEquals(14, solution("abcabcabcabcdededede"));
        Assert.assertEquals(17, solution("xababcdcdababcdcd"));
    }
}

```

Ref.

코딩테스트 연습 - 문자열 압축

데이터 처리 전문가가 되고 싶은 "어피치"는 문자열을 압축하는 방법에 대해 공부를 하고 있습니다. 최근에 대량의 데이터 처리를 위한 간단한 비손실 압축 방법에 대해 공부를 하고 있는데, 문자열에서 같은

👉 <https://programmers.co.kr/learn/courses/30/lessons/60057>

