
잠 실 개 발 자
Refactoring
M E E T U P

CONTENTS

1. 개발환경설정
2. 실습 코드 소개 (Restaurant Booking)
3. 테스트 코드 작성
4. 회고

잠 실 개 발 자
Refactoring
M E E T U P

배유미(삼성SDS)

SW 엔지니어링팀 /코드품질그룹

- 코드 품질 툴 개발
- clean code, refactoring 교육

youme.bae@samsung.com

실습가이드

오늘 할 일

Legacy Code Test Case 작성 - 1부

- **JUnit Test Coverage가 100%가 되도록 Test Case 작성**
- **작성한 Test Case Refactoring**

Legacy Code Test Case 작성하기– 2부

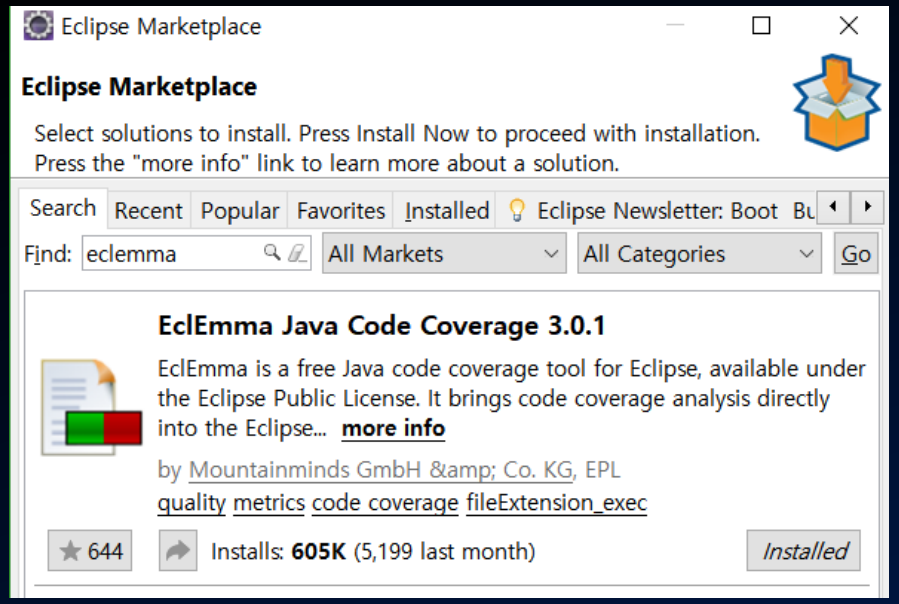
- **좀 더 똑똑하게 Test Case 작성하기**
- **Mock API 사용하기**
- **디자인 패턴 적용하여 Legacy Code Refactoring**

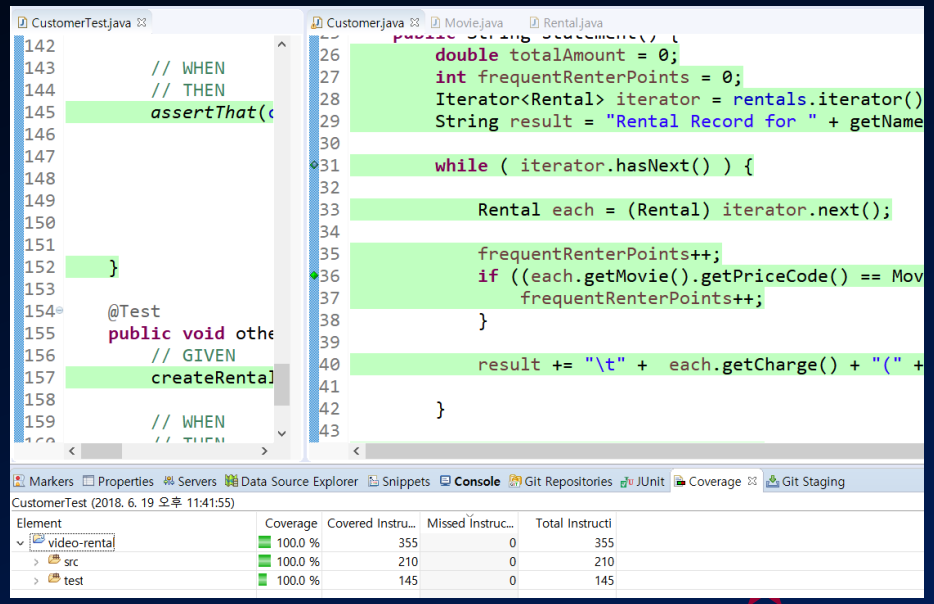
실습 환경 설정

테스트케이스 커버리지 : Eclemma

이클립스 - Help

- Eclipse Marketplace – **eclemma** – 설치 - 재기동





실습 환경 설정

소스 다운로드 : https://github.com/CODARI-JAMSIL/RestaurantBooking

Hamcrest 설정 (assertThat)

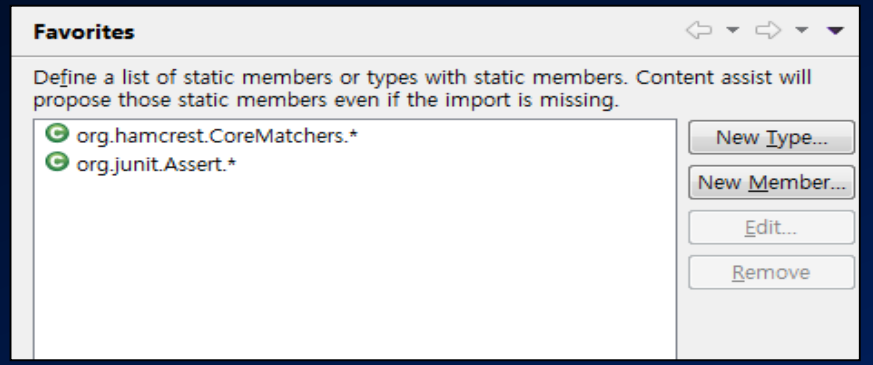
Static import 추가

Windows -> Preferences 메뉴

Java -> Editor -> Content Assist -> Favorites 항목

"New Type..." 버튼 : "**org.junit.Assert**" 및 "**org.hamcrest.CoreMatchers**" 추가

※ Junit dependency가 hamcrest-core dependency(version : 1.3)를 갖고 있기 때문에 별도 dependency 추가 불필요



실습 환경 설정

1. Git repository 연결
2. Zip 다운로드 & maven import

Legacy Code 소개

Restaurant Booking

레스토랑 예약 시스템

- BookingScheduler를 통해 시간대별 예약관리
- 예약은 정시에만 가능하다.
  → ex. 09:00(0), 09:03(x)
- 시간대별 수용가능 인원을 정할 수 있다.
  → 모든 시간대에 동일한 인원수 적용
- 일요일은 예약이 불가하다.
  → ex. '20180916(일)'에 '20180917(월)' 이용 예약 불가
  → ex. '20180917(월)'에 '20180923(일)' 이용 예약 가능
- 예약완료 시 SMS발송
- 이메일 주소가 있는 경우는 메일 발송

코드를 살펴 보자.

- BookingScheduler
- Customer
- Schedule
- MailSender
- SmsSender

이제 Legacy Code의 Test Code를 작성해보자.

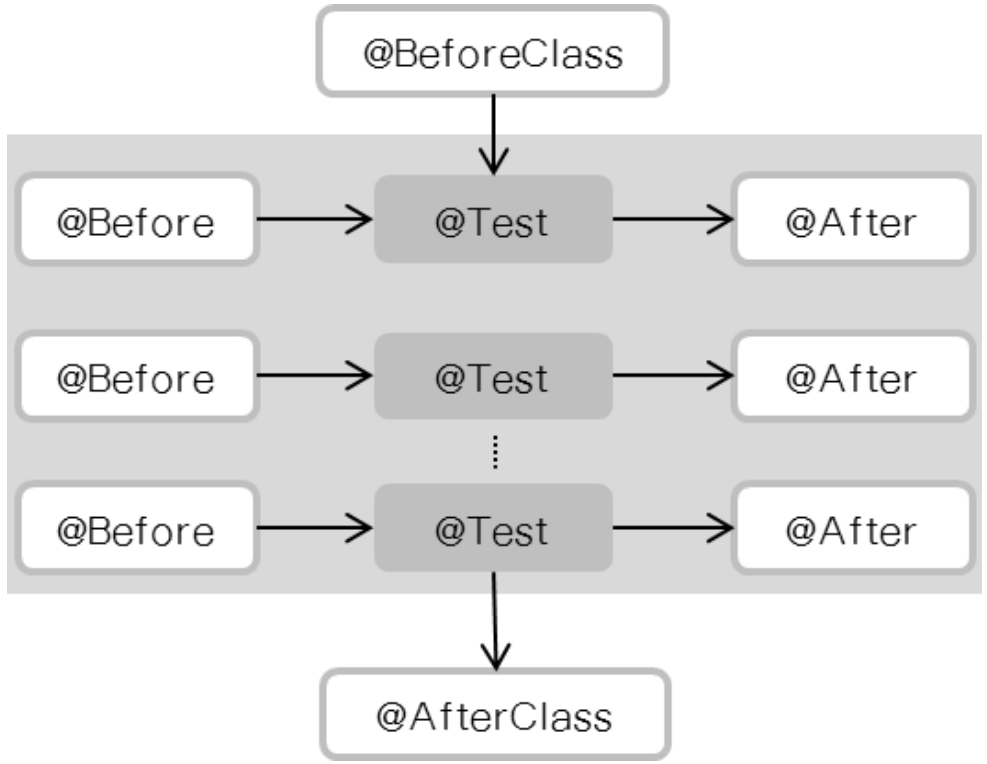
Junit 테스트 메소드 구조

```
public class PersonTest {

  @Test
   public void testGetDisplayName() {
     // arrange
     Person person = new Person("Gogh", "Vincent");

     // act
     String displayName = person.getDisplayName();

     // assert
     assertThat("Vincent Gogh", is(displayName));
   }
}
```

1. '@Test' method annotation
2. Arrange : 준비, setup
3. Act : 테스트 할 메소드 실행
4. Assert : 결과 검증

Assert 구문

Junit 실행 구조




```java
@BeforeClass
public static void setUpEnvironment() {
    System.out.println("BeforeClass");
}
```

```java
@Before
public void setUp() {
    System.out.println("- Before");
}

@Test
public void testForSample() {
    System.out.println("- Sample Test");
}

@After
public void tearDown() {
    System.out.println("- After");
}
```

```java
@AfterClass
public static void tearDownEnvironment() {
    System.out.println("AfterClass");
}
```

[실습] Test Case 1

- 예약은 정시에만 가능하다
  - 정시가 아닌 경우 예외 발생 테스트

```
@Test(expected=RuntimeException.class)
public void throwAnExceptionWhenBookingTimeIsNotOnTheHour()  throws Exception {

    // arrange
    BookingScheduler bookingScheduler = new BookingScheduler(3);
    DateTime dateTime = new DateTime(2018, 9, 17, 3, 55);
    int numberOfPeople = 1;
    Customer customer = new Customer("name", "010-1111-1111");
    Schedule schedule = new Schedule(dateTime , numberOfPeople , customer );

    // act
    bookingScheduler.addSchedule(schedule);
}
```

[실습] Test Case 2

- 예약은 정시에만 가능하다
  - 정시인 경우 스케쥴 추가 성공 테스트

```
@Test
public void scheduleIsAddWhenBookingTimeIsOnTheHour() throws Exception {

    // arrange
    BookingScheduler bookingScheduler = new BookingScheduler(3);
    DateTime dateTime = new DateTime(2018, 9, 17, 3, 00);
    int numberOfPeople = 1;
    Customer customer = new Customer("name", "010-1111-1111");
    Schedule schedule = new Schedule(dateTime , numberOfPeople ,
    customer );

    // act
    bookingScheduler.addSchedule(schedule);

    // assert
    assertThat(bookingScheduler.hasSchedule(schedule), is(true));
}
```

[실습] Test Code Refactoring 1

```
private static final int CAPACITY = 3;
private static final int NUMBER_OF_PEOPLE = 1;
private static final DateTimeFormatter dateTimeFormatter = DateTimeFormat.forPattern("YYYY/MM/dd HH:mm");
private static final DateTime NOT_ON_THE_HOUR = dateTimeFormatter.parseDateTime("2018/09/17 03:55");
private static final DateTime ON_THE_HOUR = dateTimeFormatter.parseDateTime("2018/09/17 03:00");
private static final Customer CUSTOMER = new Customer("name", "010-1111-1111");

private BookingScheduler bookingScheduler = new BookingScheduler(CAPACITY);


@Test(expected=RuntimeException.class)
public void throwAnExceptionWhenBookingTimeIsNotOnTheHour() throws Exception {
	// arrange
	Schedule schedule = new Schedule(NOT_ON_THE_HOUR , NUMBER_OF_PEOPLE , CUSTOMER );

	// act
	bookingScheduler.addSchedule(schedule);
}

@Test
public void scheduleIsAddWhenBookingTimeIsOnTheHour() throws Exception {
	// arrange
	Schedule schedule = new Schedule(ON_THE_HOUR , NUMBER_OF_PEOPLE , CUSTOMER);

	// act
	bookingScheduler.addSchedule(schedule);

	// assert
	assertThat(bookingScheduler.hasSchedule(schedule), is(true));
}
```

[실습] Test Case 3-1

- **시간대별 인원제한이 있다.**
- 같은 시간대에 Capacity를 초과했을 경우 예외 발생 테스트

```
@Test(expected=RuntimeException.class)
public void throwAnExceptionWhenCapacityPerHourIsOver() {
    // arrange
    List<Schedule> schedules = new ArrayList<>();
    Schedule fullSchedules = new Schedule(ON_THE_HOUR, CAPACITY, CUSTOMER);
    schedules.add(fullSchedules );
    bookingScheduler.setSchedules(schedules );
    Schedule newSchedule = new Schedule(ON_THE_HOUR , NUMBER_OF_PEOPLE ,
    CUSTOMER );

    // act
    bookingScheduler.addSchedule(newSchedule);
}
```

[실습] Test Case 3-2

- **시간대별 인원제한이 있다.**
- 같은 시간대에 Capacity를 초과했을 경우 예외 발생 테스트
- try~catch로 변경하여 Exception Message 체크

```
@Test
public void throwAnExceptionWhenCapacityPerHourIsOver() {
	// arrange
	List<Schedule> schedules = new ArrayList<>();
	Schedule fullSchedules = new Schedule(ON_THE_HOUR, CAPACITY, CUSTOMER);
	schedules.add(fullSchedules );
	bookingScheduler.setSchedules(schedules );
	Schedule newSchedule = new Schedule(ON_THE_HOUR , NUMBER_OF_PEOPLE , CUSTOMER );

	try {
		// act
		bookingScheduler.addSchedule(newSchedule);
		fail();
	} catch (RuntimeException e) {
		// assert
		assertThat(e.getMessage(),
				is("Number of people is over restaurant capacity per hour"));
	}

}
```

[실습] Test Case 4

- **시간대별 인원제한이 있다.**
- 시간대가 다르면 Capacity가 차 있어도 스케쥴 추가 성공 테스트

```
@Test
public void scheduleCanBeAddedWhenCapasityOfDifferentHourIsFull() {
      // arrange
      List<Schedule> schedules = new ArrayList<>();
      Schedule fullSchedules = new Schedule(ON_THE_HOUR, CAPACITY, CUSTOMER);
      schedules.add(fullSchedules );
      bookingScheduler.setSchedules(schedules );

      DateTime differentHour = ON_THE_HOUR.plusHours(1);
      Schedule newSchedule = new Schedule(differentHour  , NUMBER_OF_PEOPLE , CUSTOMER );

      // act
      bookingScheduler.addSchedule(newSchedule);

      // assert
      assertThat(bookingScheduler.hasSchedule(newSchedule), is(true));
}
```

[실습] Test Code Refactoring 2

Junit 실행구조

```
 private List<Schedule> schedules = new ArrayList<>();

 @Before
 public void setUp() {
            bookingScheduler.setSchedules(schedules );
 }
```

```
@Test
public void scheduleCanBeAddedWhenCapasityOfDifferentHourIsFull() {
      // arrange
      Schedule fullSchedules = new Schedule(ON_THE_HOUR, CAPACITY, CUSTOMER);
      schedules.add(fullSchedules );

      DateTime differentHour = ON_THE_HOUR.plusHours(1);
      Schedule newSchedule = new Schedule(differentHour  , NUMBER_OF_PEOPLE , CUSTOMER );

      // act
      bookingScheduler.addSchedule(newSchedule);

      // assert
      assertThat(bookingScheduler.hasSchedule(newSchedule), is(true));
}
```

[실습] Test Case 5

- “예약완료 시 SmsSender는 무조건 발송” 테스트 케이스 작성

```java
@Test
public void sendSmsToCustomerWhenScheduleIsAdded() {

	// arrange
	Schedule schedule = new Schedule(ON_THE_HOUR  , NUMBER_OF_PEOPLE , CUSTOMER );
	TestableSmsSender testableSmsSender = new TestableSmsSender();
	bookingScheduler.setSmsSender(testableSmsSender);

	// act
	bookingScheduler.addSchedule(schedule);

	// assert
	assertThat(testableSmsSender.isSmsSenderIsCalled(), is(true));
}
```

```java
package com.sds.cleancode.restaurant;

public class TestableSmsSender extends SmsSender {

	private boolean smsSenderIsCalled;

	public TestableSmsSender() {
		// TODO Auto-generated constructor stub
	}

	@Override
		public void send(Schedule schedule) {
		this.smsSenderIsCalled = true;
	}

		public boolean isSmsSenderIsCalled() {
		return smsSenderIsCalled;
	}
}
```

[실습] Test Case 6

- **"email이 있는 경우에만 EmailSender 발송" 테스트 케이스 작성**

해봅시다!!

* Testable Class

1. TestableMailSender Class작성 (extends MailSender)
2. isSendMailMethodIsCalled Method 작성

* Test Case

1. Email이 있는 Customer 생성
   private static final Customer CUSTOMER_WITH_EMAIL
   = new Customer("fakename", "010-1111-1111", abcd@gmail.com);
2. BookingScheduler의 setMailSender 메소드로 TestableMailSender 주입
3. Assert : sendMailMethodIsCalled 값 확인

[실습] Test Code Refactoring 3

Junit 실행구조

```
private TestableSmsSender testableSmsSender = new TestableSmsSender();
private TestableMailSender testableMailSender = new TestableMailSender();

@Before
public void setUp() {
      bookingScheduler.setSchedules(schedules );
      bookingScheduler.setSmsSender(testableSmsSender);
      bookingScheduler.setMailSender(testableMailSender);
}
```

```
@Test
public void sendEmailToCustomerWithEmailAddressWhenScheduleIsAdded() {

      // arrange
      Schedule schedule = new Schedule(ON_THE_HOUR  , NUMBER_OF_PEOPLE ,
      CUSTOMER_WITH_EMAIL );

      // act
      bookingScheduler.addSchedule(schedule);

      // assert
      assertThat(testableMailSender.isSendMailIsCalled(), is(true));
}
```

[다음 실습 안내] 일요일 예약 부분 주석제거

- **bookingScheduler의 주석 처리부분 해제 후 커버리지 100% 달성**

```
// throw an exception on Sunday
DateTime now = new DateTime();
if(now.getDayOfWeek() == DateTimeConstants.SUNDAY){
            throw new RuntimeException("Booking system is not available on sunday");
}
```

'New DateTime()' extract to method

```
// throw an exception on sunday
DateTime now = getNow();
if(now.getDayOfWeek() == DateTimeConstants.SUNDAY){
            throw new RuntimeException("Booking system is not available on sunday");
}


public DateTime getNow() {
            return new DateTime();
}
```

[실습] Test Case 7

- 현재 날짜가 일요일인 경우 예약불가 exception" 테스트 작성

```
@Test
public void throwAnExceptionOnSunday(){

	// arrange
	BookingScheduler bookingScheduler = new SundayBookingScheduler(CAPACITY);
	Schedule schedule = new Schedule(ON_THE_HOUR  , NUMBER_OF_PEOPLE , CUSTOMER );

	try {
		// act
		bookingScheduler.addSchedule(schedule);
		fail();
	} catch (RuntimeException e) {
		// assert
		assertThat(e.getMessage(), is("Booking system is not available on sunday"));
	}

}
```

```
public class SundayBookingScheduler extends BookingScheduler {

	public static DateTimeFormatter dateTimeFormatter = DateTimeFormat.forPattern("YYYY/MM/dd HH:mm");

	public SundayBookingScheduler(int capacityPerHour) {
		super(capacityPerHour);
	}

	@Override
	public DateTime getNow() {
		return dateTimeFormatter.parseDateTime("2018/09/16 03:00");
	}
}
```

[실습] Test Code Refactoring 4

- **TestableBookingScheduler로 변경**

```java
@Test
public void scheduleIsAddedWhenAnydayExceptSunday(){
	// arrange
	BookingScheduler bookingScheduler = new TestableBookingScheduler(CAPACITY, MONDAY);
	Schedule schedule = new Schedule(ON_THE_HOUR  , NUMBER_OF_PEOPLE , CUSTOMER );

	// act
	bookingScheduler.addSchedule(schedule);

	// assert
	assertThat(bookingScheduler.hasSchedule(schedule), is(true));

}
```

```java
public class TestableBookingScheduler extends BookingScheduler {

	private DateTime date;

	public TestableBookingScheduler(int capacityPerHour, DateTime date) {
		super(capacityPerHour);
		// TODO Auto-generated constructor stub
		this.date = date;
	}

	@Override
	public DateTime getNow() {
		// TODO Auto-generated method stub
		return date;
	}
}
```

Mock 이란?

넓은 의미

Mock Object는 **실제 객체의 행동을 의도한 방향으로 흉내 내도록 설계 된 객체**이다. 프로그래머는 일반적으로 다른 어떤 객체의 행동을 테스트하기 위해서 만든다.

(출처 : https://en.wikipedia.org/wiki/Mock_object 번역)

마치며


Thank You!


회고

별첨1. Assert

Assert

Assert 메소드	설명
assertTrue(condition)	True인지 검사
assertFalse(condition)	False인지 검사
assertNull	Null인지 검사
assertNotNull	Null이 아닌지 검사
assertEquals(expected, actual)	두 객체의 equals 결과가 참인지 또는 두 primitive 값이 같은지 검사
assertSame(expected, actual)	두 객체가 동일 객체인지 검사 (== 연산)
assertNotSame(exepcted, actual)	두 객체가 다른 객체인지 검사
assertArrayEquals(expected, actual)	두 개의 배열의 equals 값이 참인지 검사
fail(message)	테스트 케이스를 실패시킴
assertThat(condition)	해당 객체가 주어진 조건을 만족하는지 검사

assertThat(Object actual, Matcher matcher)

- Readable and more useful than assertEqual

Matcher

- assertThat("Simple Text", is("Simple Text"));
- assertThat(calculatedTax, is(not(thirtyPercent)));
- assertThat(phdStudentList, hasItem(DrJohn));

...

• 편집 관련

단축키	기능	단축키	기능
Ctrl-Z	Undo	Ctrl-Y	Redo
Ctrl-D	한 줄 삭제	Alt-Up/Down	줄 이동
Ctrl-/	Line comment (toggle)	Ctrl-Shift-F	소스 format
Ctrl-Shift-/	Block comment	Ctrl-Shift-₩	Block uncomment
Ctrl-Shift-O	Import 정리	Ctrl-Shift-M	특정 import 추가
Alt-Shift-Up/Down	Block 선택	Ctrl-Alt-J	라인 합치기

• 리팩터링 관련

단축키	기능	단축키	기능
Alt-Shift-T	Refactor 메뉴 호출	Alt-Shift-S	Source 메뉴 호출
Alt-Shift-R	Rename	Alt-Shift-V	Move
Alt-Shift-C	Change Method Signature	Alt-Shift-M	Extract Method
Alt-Shift-L	Extract Local Variable	Alt-Shift-I	Inline

• 탐색 관련

단축키	기능	단축키	기능
Ctrl-L	특정 line으로 이동	Ctrl-Shift-P	매칭되는 괄호로 이동
Ctrl-O	빠른 개요 보기	Ctrl-T	빠른 상속 구조 보기
Alt-Left/Right	뒤로/앞으로 이동	Ctrl-F/Ctrl-H	찾기 및 검색
Ctrl-Shift-R	리소스 찾기	Ctrl-Q	최종 수정 위치 이동

• 윈도우 관련 및 기타

단축키	기능	단축키	기능
Ctrl-N	새파일	Ctrl-M	에디터 최대화 (toggle)
Ctrl-W	창 닫기	Ctrl-Shift-L	단축키 리스트
Ctrl-1	빠른 수정	Ctrl-Space	Content Assist
Ctrl-Shift-Space	파라미터 힌트 보기	Alt-Shift-X, T	JUnit Test 실행
Ctrl-F11	Run	Ctrl-Shift-F11	Coverage 실행

• 템플릿

단축키	기능
sysout 입력 후 Ctrl-Space	System.out.println(); 자동 완성
try 입력 후 Ctrl-Space	try-catch 문 자동 완성
for 입력 후 Ctrl-Space	For 문 자동 완성