

4주차 과제

staudy 할래?

스터디 내용

1. 선택문

2. 반복문

3. 과제-JUnit 5 학습하세요

-인텔리J, 이클립스, VS Code에서 JUnit 5로 테스트 코드 작성하는 방법에 익숙해 질 것.

4. 과제1.live-study 대시보드를 만드는 코드를 작성하세요.

깃헙 이슈 1번부터 18번까지 댓글을 순회하며 댓글을 남긴 사용자를 체크 할 것.

참여율을 계산하세요. 총 18회에 중에 몇 %를 참여했는지 소숫점 두자리까지 보여줄 것

-Github 자바 라이브러리를 사용하면 편리합니다.

깃헙 API를 익명으로 호출하는데 제한이 있기 때문에 본인의 깃헙 프로젝트에 이슈를 만들고 테스트를 하시면 더 자주 테스트할 수 있습니다.

5. LinkedList를 구현하세요.

정수를 저장하는 ListNode 클래스를 구현하세요.

-ListNode add(ListNode head, ListNode nodeToAdd, int position)를 구현하세요.

-ListNode remove(ListNode head, int positionToRemove)를 구현하세요.

-boolean contains(ListNode head, ListNode nodeTocheck)를 구현하세요.

스터디 내용

6. 과제3.Stack을 구현하세요.

-int 배열을 사용해서 정수를 저장하는 Stack을 구현하세요.

-void push(int data)를 구현하세요.

-int pop()을 구현하세요.

7. 과제4. 앞서 만든 ListNode를 사용해서 Stack을 구현하세요.

-ListNode head를 가지고 있는 ListNodeStack 클래스를 구현하세요.

-void push(int data)를 구현하세요.

-int pop()을 구현하세요.

8. 과제5. Queue를 구현하세요.

배열을 사용해서 한번

-ListNode를 사용해서 한번.

1. 선택문(Switch/case)

예문)

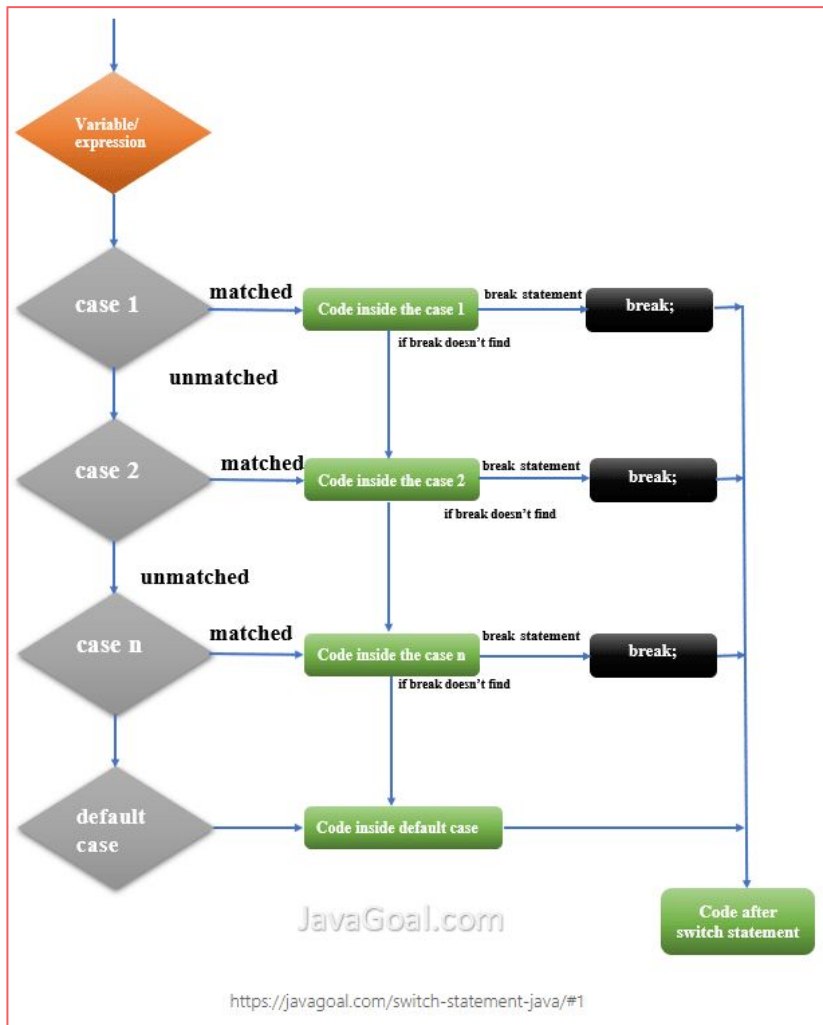
```
int num = 31;

switch (num % 3) {
    case 0:
        System.out.println(num + "을 3으로 나눈 나머지는 0입니다.");
        break;
    case 1:
        System.out.println(num + "을 3으로 나눈 나머지는 1입니다.");
        break;
    default:
        System.out.println(num + "을 3으로 나눈 나머지는 2입니다.");
        break; }
```

break문: java의 break문은 실행 순서를 중단하기 위해 매우 일반적으로 사용됩니다. break 문은 루프와 switch문에 사용됩니다.

continue문: break를 써주지 않으면 default로 다음 케이스를 실행

이후 자바 version의 switch문 업데이트



2. 반복문(for)

어떤 조건이 만족하는 동안 같은 내용을 계속해서 반복한다.

1. for 2. forEach 3. while 4. do while

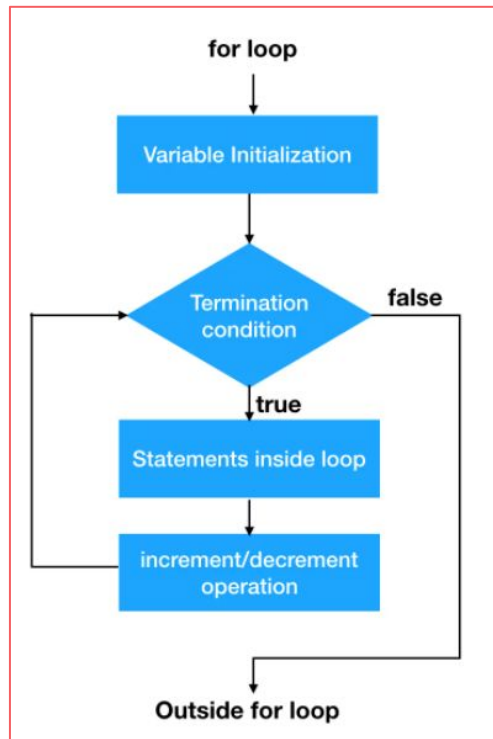
for문 부터 살펴보자

for (초기화식; 조건부; 증감부){
 조건이 참인경우 실행부}

```
for (int i = 0; i < 10; i++){  
    System.out.println(i+"번 반복완료");  
}
```

1. 초기화식, 조건문, 증감문을 반드시 작성할 필요는 없다.
2. 초기화식, 조건문, 증감문은 얼마든지 확장해서 구현할 수 있다.
3. 증감문에 반드시 ++/-- 같은 연산자를 사용할 필요는 없다.
4. 조건문에 사용하는 변수가 있는 경우, 이 변수가 꼭 초기화식에서 선언한 변수일 필요는 없다.

0번 반복완료
1번 반복완료
2번 반복완료
3번 반복완료
4번 반복완료
5번 반복완료
6번 반복완료
7번 반복완료
8번 반복완료
9번 반복완료



2-1. 반복문 활용

```
7 public static void main(String[] args) {  
8     int i;  
9     for (i = 0; i <= 10; i+=2){  
10         System.out.println(i+"번 반복완료");  
11     }  
12     System.out.println("최종 i: " + i);  
13 }  
14 }
```

Problem | @ Javado | Declarat | Progres | Console

<terminated> Solution [Java Application] C:\Users\user\Desktop\java\

0번 반복완료
2번 반복완료
4번 반복완료
6번 반복완료
8번 반복완료
10번 반복완료
최종 i: 12

```
9     for (int i = 0, j = 5; i < 10 && j < 12; i+=2, j++){  
10         System.out.println("i : " + i + " j : " + j );  
11     }
```

Problems | @ Javadoc | Declaration | Progress | Console

<terminated> Solution [Java Application] C:\Users\user\Desktop\java\OOPSW\jdk1.8.0_71\

i : 0 j : 5
i : 2 j : 6
i : 4 j : 7
i : 6 j : 8
i : 8 j : 9

2-2. 반복문 활용

```
8      int i = 0, j = 10;
9      for ( ; i < 10 && j < 12; ){
10         System.out.println("i : " + i + " j : " + j ) ;
11         i++;
12         j++;
13     }
14
```

<terminated> Solution [Java Applet]
i : 0 j : 10
i : 1 j : 11

다음과같이 초기화식, 증감부는 생략이 가능하지만 세미콜론은 생략하면 안된다.

*** 심지어 다음과 같이 작성해도 잘 동작 한다.*

```
1 package me.xxxelppa.study.week04;
2
3 public class Exam_021 {
4     public static void main(String[] args) {
5         System.out.println("===== 피보나치 수열 =====");
6         for(int cnt = 0, bf = 0, af = 1; cnt++ < 30; System.out.print(cnt == 1 ? "1\t" : (af += bf) + (cnt % 1
7     )
8 }
```

===== 피보나치 수열 =====									
1	1	2	3	5	8	13	21	34	55
89	144	233	377	610	987	1597	2584	4181	6765
10946	17711	28657	46368	75025	121393	196418	317811	514229	832040

2-3.반복문(forEach)

```
public static void main(String[] args) {  
    int[] list = new int[]{1,2,3,4,5,6,7,8,9,0};  
    int sum = 0;  
    for (int i = 0; i < list.length; i++){  
        sum += list[i];  
    }  
    System.out.println(sum); //45  
}
```



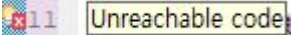
```
public static void main(String[] args) {  
    int[] list = new int[]{1,2,3,4,5,6,7,8,9,0};  
    int sum = 0;  
    for (int tmp : list){  
        sum += tmp;  
    }  
    System.out.println(sum); //45  
}
```

```
for (type var : iterate) {  
    body-of-loop  
}
```

위 **iterate**는 루프를 돌릴 객체이고
이 객체에서 한개씩 순차적으로
var에 대입되어 **for**문을 수행하게
된다. **iterate**부분에 들어가는
타입은 루프를 돌릴수 있는 형태인
배열 및 **ArrayList**등이 가능하다

2-4. 반복문(while)

```
public static void main(String[] args) {  
    for ( ; ; ){  
        System.out.println("for");  
    }  
    while (true){  
        System.out.println("while");  
    }  
}
```



```
public static void main(String[] args) {  
    int sum = 0;  
    int i = 1;  
    while (sum < 100){  
        sum += 10;  
        if (sum % 3 == 0){  
            continue;  
        }  
        System.out.println(i + "번째 sum : " + sum);  
        i++;  
    }  
}
```

```
1번째 sum : 10  
2번째 sum : 20  
3번째 sum : 40  
4번째 sum : 50  
5번째 sum : 70  
6번째 sum : 80  
7번째 sum : 100  
8번째 실행 중단
```

for 무한루프와 while 무한루프를 만들었다.
이 경우 while 무한루프에 도달할 수 없어
unreachable code 오류가 발생했다.

이때 **break;** 를 활용해 줄 수 있다.

그럼 continue 활용 예제를 보자

2-5. 반복문(do-while)

```
public static void main(String[] args) {  
    do{  
        System.out.println("while 반복문의 실행 조건이 false로 판별되어도");  
        System.out.println("do 블록을 무조건 한 번은 실행합니다.");  
    }while (false);  
}
```

```
while 반복문의 실행 조건이 false로 판별되어도  
do 블록을 무조건 한 번은 실행합니다.
```

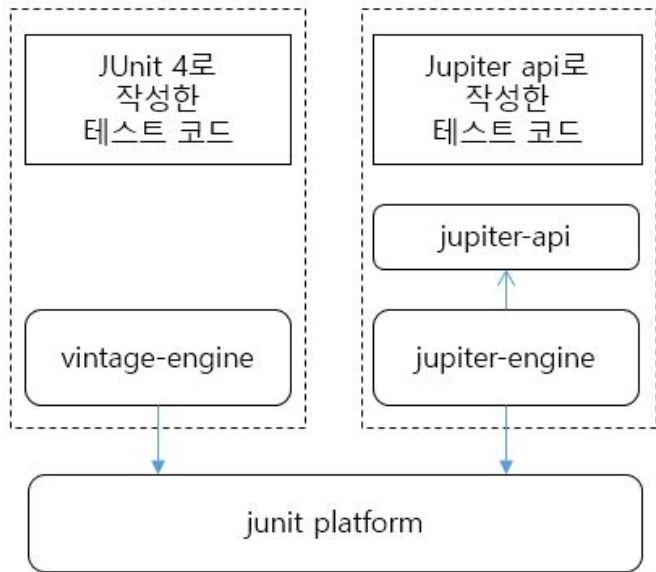
while 문의 조건부를 확인하는 것처럼 똑같이 동작하지만, 차이가 있다면 **while**의 조건이 참/거짓 여부에 상관없이 무조건 한 번은 실행해야 하는 내용이 있을 경우 사용할 수 있는 반복문이다.

3. JUnit5 학습하기

JUnit 5를 보면 JUnit 4에 비해 중요한 차이점은 다음과 같다.

JUnit 4가 단일 jar였던 것에 비해, JUnit 5는 크게 JUnit Platform, JUnit Jupiter, JUnit Vintage 모듈로 구성되어 있다. 테스트 실행을 위한 API와 테스트 작성자를 위한 API 모듈이 분리되어 있다.

<모듈 개요>



JUnit Platform은 테스트를 발견하고 테스트 계획을 생성하는 TestEngine 인터페이스를 정의하고 있다. Platform은 TestEngine을 통해서 테스트를 발견하고, 실행, 결과를 보고한다.

TestEngine의 실제 구현체는 별도 모듈로 존재한다.

이 모듈 중 하나가 jupiter-engine이다. 이 모듈은 jupiter-api를 사용해서 작성한 테스트 코드를 발견하고 실행한다. jupiter API는 JUnit5에 새롭게 추가된 테스트 코드용 API로서, 개발자는 Jupiter API를 사용해서 테스트 코드를 작성할 수 있다.

기존에 JUnit 4 버전 테스트코드는 vintage-engine 모듈을 사용한다 만약 테스트코드 작성을 위한 새로운 API를 창안한다면, 그 API에 알맞은 엔진 모듈을 함께 구현해서 제공하면 JUnit Platform 수정없이 새로 창안한 테스트 API를 실행, 결과를 리포팅할 수 있다

3-1. JUnit

JUnit이란?

자바용 단위 테스트 작성을 위한 산업 표준 프레임워크다.

외부 테스트 프로그램(케이스)을 작성하여 **System.out**으로 번거롭게 디버깅 불필요

프로그램 테스트 시 걸릴 시간도 관리

오픈 소스이며 플러그인 형태로 **Eclipse**에 포함

하나의 **jar** 파일이 전부이며 사용법도 간단

개발이 어느 정도 진행되면 프로그램 단위 테스트 반드시 실행

JDK 1.4에서 추가된 **assertXXX**를 사용하여 **Test**를 진행

테스트 결과를 단순한 텍스트로 남기는 것이 아니라 **Test** 클래스로 남기어,

개발자에게 테스트 방법 및 클래스의 **History**를 넘겨줄 수도 있음

3-2. Junit5 환경 셋팅에 애먹는중..

<https://www.youtube.com/watch?v=WDIVRAxYDEc>

4. live-study 대시보드를 만드는 코드 작성

-깃헙 이슈 1번부터 18번까지 댓글을 순회하며 댓글을 남긴 사용자를 체크 할 것.-참여율을 계산하세요. 총 18회에 중에 몇 %를 참여했는지 소숫점 두자리까지 보여줄 것

-Github 자바 라이브러리 를 사용하면 편리합니다.

깃헙 API를 익명으로 호출하는데 제한이 있기 때문에 본인의 깃헙 프로젝트에 이슈를 만들고 테스트를 하시면 더 자주 테스트할 수 있습니다.

GitHub API for Java

GitHub API for Java

Git Hub API for Java

[Introduction](#)
[Download](#)
[Source code](#)
[Mailing List](#)

Guides

- ▼ [GitHub App Auth Flow](#)
 - [JWT Authentication](#)
 - [App Installation Token](#)
- [Working with organizations](#)

References

[Javadoc](#)

Project Documentation

- ▼ [Project Information](#)
 - [Dependencies](#)
 - [Dependency Information](#)

What is this?

This library defines an object oriented representation of the GitHub API. By "object oriented" we mean there are classes that correspond to the domain model of GitHub (such as `GHUser` and `GHRepository`), operations that act on them as defined as methods (such as `GHUser.follow()`), and those object references are used in favor of using string handle (such as `GHUser.isMemberOf(GHOrganization)` instead of `GHUser.isMemberOf(String)`)

The library supports both github.com and GitHub Enterprise.

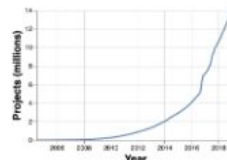
Most of the GitHub APIs are covered

MYNREPOSITORY

Search for groups, artifacts, categories

Search

Indexed Artifacts (18.6M)



Popular Categories

[Aspect Oriented](#)
[Actor Frameworks](#)
[Application Metrics](#)
[Build Tools](#)
[Bytecode Libraries](#)
[Command Line Parsers](#)
[Cache Implementations](#)
[Cloud Computing](#)
[Code Analyzers](#)
[Collections](#)
[Configuration Libraries](#)

Home » [org.kohsuke](#) » [github-api](#)

Kohsuke

GitHub API For Java

GitHub API for Java

License

MIT

Categories

GitHub API

Tags

github api client

Used By

96 artifacts

Central (99)

Spring Plugins (3)

Jenkins Releases (1)

Version	Repository	Usages	Date
1.117	Central	3	Nov, 2020
1.116	Central	10	Aug, 2020
1.115	Central	5	Jul, 2020
1.114	Central	2	Jun, 2020

[Home](#) » [org.kohsuke](#) » [github-api](#) » [1.116](#)



GitHub API For Java » 1.116

GitHub API for Java

License	MIT
Categories	GitHub API
HomePage	https://github-api.kohsuke.org/
Date	(Aug 13, 2020)
Files	jar (447 KB) View All
Repositories	Central
Used By	96 artifacts

Note: There is a new version for this artifact

New Version [1.117](#)

[Maven](#) [Gradle](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#)

[Buildr](#)

```
<!--  
https://mvnrepository.com/artifact/org.kohsuke/github-  
api -->  
<dependency>  
  <groupId>org.kohsuke</groupId>  
  <artifactId>github-api</artifactId>  
  <version>1.116</version>  
</dependency>
```


5. LinkedList를 구현하세요

- 정수를 저장하는 ListNode 클래스를 구현하세요.
- ListNode add(ListNode head, ListNode nodeToAdd, int position)를 구현하세요.
- ListNode remove(ListNode head, int positionToRemove)를 구현하세요.
- boolean contains(ListNode head, ListNode nodeTocheck)를 구현하세요.

<https://blog.naver.com/hsm622/222159930944>

5. LinkedList를 구현하세요

```
java.util
```

```
Class LinkedList<E>
```

```
java.lang.Object
```

```
java.util.AbstractCollection<E>
```

```
java.util.AbstractList<E>
```

```
java.util.AbstractSequentialList<E>
```

```
java.util.LinkedList<E>
```

LinkedList 개념 정리: <https://coding-factory.tistory.com/552>

6. stack을 구현하세요