

## 1. 삼각형 결정조건

사용자로부터 입력받은 값을 이용하여 삼각형을 만들고, 해당 삼각형이 직삼각형과 정삼각형 중에 어떤 것에 해당 되는지 검사하는 프로그램을 작성하세요.

문제의 요구 사항은 다음과 같습니다.

- (1) 사용자에게 3개의 정수형 변수를 받을 것(자연수를 받는다고 가정)
- (2) 3개의 정수로 삼각형을 만들 시, 삼각형이 성립되는지 검사
- (3) 3개의 정수로 삼각형을 만들 수 있을 시, 해당 삼각형이 직삼각형과 정삼각형 중에 어떤 것에 해당되는지 판단

코드 작성 시 요구 사항은 다음과 같습니다.

- (1) lib Package를 만들고 내부에 Triangle class를 정의할 것
- (2) Triangle class의 멤버 메소드는 다음과 같음

TriangleCondition	
Return type	Boolean
Input parameter	int a, int b, int c
Description	입력받은 정수 a, b, c 가 삼각형 결정 조건과 성립하는 지 - 삼각형의 각 변을 a, b, c라고 할 때, $a + b > c$ $b + c > a$ $c + a > b$ 이 3개의 부등식을 모두 만족할 경우, 삼각형이 성립한다.
Return value	성립 시 : True 성립하지 않을 시 : False

RightTriangle	
Return type	Boolean
Input parameter	int a, int b, int c
Description	입력받은 정수 a, b, c 가 직각삼각형 조건과 성립하는 지 판단 - c 가 가장 긴 변이고, 나머지 두 변을 a와 b라고 할 때, $a^2 + b^2 = c^2$
Return value	성립 시 : True 성립하지 않을 시 : False

EquilateralTriangle	
Return type	Boolean
Input parameter	int a, int b, int c
Description	입력받은 정수 a, b, c 가 정삼각형이 성립하는 지 판단 - 세 변의 길이가 모두 같다.
Return value	성립 시 : True 성립하지 않을 시 : False

(3) Main method에서는 Triangle Class 내에 선언한 메소드를 활용하여 다음과 같이 출력

```
Input Three Integer >> 7
5
4
Three input numbers can make triangle
```

그림 1 삼각형을 이룰 수 있으나  
직각삼각형이나 정삼각형이 아닌 경우

```
Input Three Integer >> 3
7
10
Three input numbers can't make triangle
```

그림 2 삼각형을 이루지 못하는 경우

```
Input Three Integer >> 6
6
6
Three input numbers can make equilateral triangle
```

그림 3 삼각형을 이룰 수 있고, 직각삼각형인 경우

```
Input Three Integer >> 5
12
13
Three input numbers can make right triangle
```

그림 4 삼각형을 이룰 수 있고, 정삼각형인 경우

## 2. Mini-Sudoku

Sudoku는 모든 행과 열의 합이 같은 것을 말합니다.

사용자로부터 값을 입력받아서 3X3 Sudoku를 완성하는 프로그램을 작성하세요.

다음 빈칸에 조건에 맞는 식을 넣은 후 검증하는 코드를 작성하시오.

4	9	2
3		7
8	1	6

- (1) 해당 표를 2차원 배열로 선언
  - (2) 노란색으로 표기되어 있는 곳의 값은 사용자로부터 입력을 받음
  - (3) 사용자로부터 값을 입력받아 배열의 값을 다 채웠을 때, 다음의 조건을 만족하는지 검사하고, 모든 조건을 만족하면 정답으로 처리
- <정답 조건> : 모든 조건을 다 만족해야 정답으로 처리
- 1 ~ 9까지의 모든 수가 들어가 있어야 한다.
  - 각 행의 합이 모두 일치해야 한다.
  - 각 열의 합이 모두 일치해야 한다.

### 출력 예시

```
Input the number for Center >> 5
4 9 2
3 5 7
8 1 6
Gotcha!
```

그림 1 값이 정답일 경우

```
Input the number for Center >> 7
4 9 2
3 7 7
8 1 6
Wrong answer!
```

그림 2 값이 정답이 아닐 경우

### 3. 거리에 따른 택시 요금 계산

※ 해당 문제는 제공되는 기본 JAVA 코드를 수정하는 문제입니다. (Quiz3.java, Meter.java)

요금이 서로 다른 2가지 기본택시와 모범택시가 있습니다.

택시의 요금을 계산해주는 Meter 인터페이스에서 상속받아서 각각 보통택시와 모범택시의 요금을 계산해주는 Taxi 클래스와 Deluxe 클래스를 만들고, 이를 이용해서 요금을 계산해주는 프로그램을 작성하세요.

#### (1) Quiz3.java

- Scanner를 선언하고 사용자로부터 이동거리를 입력 받음
- taxi 와 deluxe 클래스의 fare 함수를 사용한 요금 계산
- taxi 와 deluxe 클래스의 get\_total\_fare 함수를 사용한 총 요금 출력

#### (2) Meter.java

- meter interface의 서브 클래스인 taxi와 deluxe 클래스를 정의
- taxi 클래스의 fare 메소드(meter interface의 fare 메소드 오버라이딩) 구현
  - 기본 택시의 요금을 계산하여 저장함
  - 1600m 까지는 기본 요금 부과 : 4800
  - 1600m 이후 131m당 요금(선결제 방식) : 100원
  - ex) 1800m를 이동했다면, 131m당 요금이 2번 발생하여 총 5000원 부과
- taxi 클래스의 getTotalReward 메소드(meter interface의 get\_total\_reward 메소드 오버라이딩) 구현
  - **return type : int**
  - 요금 반환
- deluxe 클래스의 fare 메소드(meter interface의 fare 메소드 오버라이드) 구현
  - 모범 택시의 요금을 계산하여 저장함
  - 3000m까지 기본 요금 : 6500
  - 151미터당 요금(선결제 방식) : 200원
  - ex)3200m를 이동했다면, 151m당 요금이 2번 발생하여, 6900원이 총 요금.
- deluxe 클래스에 meter에 선언된 getTotalReward 메소드 override
  - **return type : int**
  - 총 요금 반환

```
이동거리를 입력하세요(단위: m) >> 1800
기본 택시 요금은 5000 입니다.
모범 택시 요금은 6500 입니다.
```

출력 예시 1

```
이동거리를 입력하세요(단위: m) >> 3200
기본 택시 요금은 6100 입니다.
모범 택시 요금은 6900 입니다.
```

출력 예시 2

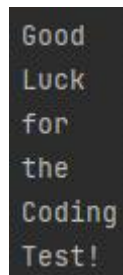
## 4. String Parser

※ 해당 문제는 제공되는 기본 JAVA 코드를 수정하는 문제입니다. (Quiz4.java)

String을 띄어쓰기 단위로 나누고 출력하는 프로그램을 작성하세요.

조건은 다음과 같습니다.

- 제공되는 input string인 "Good Luck for the Coding Test!"를 인자로 받음.
- 띄어쓰기 단위로 해당 String을 분리하고 분리된 모든 String을 ArrayList에 삽입
- ArrayList 의 들어있는 모든 값을 Iterator를 활용하여 출력



```
Good
Luck
for
the
Coding
Test!
```

결과 화면

## 5. Vector 정렬

수강한 수업 들의 이름과 학점을 출력하는 프로그램을 작성하세요.  
총 2개의 java 파일을 제출해야합니다.(Quiz5.java, StudentInfo.java)

### Quiz5.java

(1) main 메소드에 다음 정보를 학점을 2차원 String 배열로 선언하시오

Subject	Grade
Computer Network	A
Signal Processing	C
JAVA	A
C Language	A
Data Structure	B
Database	D

(2) StudyInfo class를 인자로 사용하는 Vector를 선언하고 데이터를 담으시오.

(3) 다음과 같은 메소드를 main함수가 있는 java에 선언하고 코드를 작성하시오.

printVector	
Return type	void
Input parameter	Vector
Description	Vector의 내부 인자들을 Loop를 활용하여 출력 (Iterator 사용 금지)
Return value	None

(4) 다음 메소드를 main 메소드가 있는 java 파일에 선언하고 코드를 작성하시오.

sortVector	
Return type	Vector
Input parameter	Vector
Description	Vector 내부의 인자들을 학점순으로 정렬 (과목명은 정렬하지 않아도 됨)
Return value	Vector

(5) 다음 메소드를 main 메소드가 있는 java 파일에 선언하고 코드를 작성하시오.

findVector	
Return type	int
Input parameter	Vector
Description	Vector 내에서 특정 과목명을 가지는 StudyInfo의 index를 찾아서 반환하는 메소드
Return value	과목명이 있을 경우 : index 과목이 없을 경우 : -1w

(6) Database 수업을 재수강하여 학점이 A로 변경되었을 경우, Vector를 재정렬하고 이를 출력하시오.

(7) 다음 설명을 보고, StudyInfo.java파일을 만들고 내부에 class 및 메소드를 작성하시오.

public class StudyInfo			
Variable		private String subject : 과목명 저장 private String grade : 학점 저장	
Method	생성자	Input parameter	String subject, String grade
		Description	StudyInfo Class 생성자 parameter 값을 class에 저장
	setGrade	Return type	void
		Input parameter	String grade
		Description	class 내의 grade 값 변경을 위한 메소드
	getSubject	Return type	String
		Input parameter	None
		Description	class의 subject 값을 가져오는 메소드
		Return value	return subject
	getGrade	Return type	String
		Input parameter	None
		Description	class의 grade 값을 가져오는 메소드
		Return value	return grade

```

=====Before Sort=====
Computer Network A
Signal Processing C
JAVA A
C language A
Data Structure B
Database D
=====

=====After Sort=====
Computer Network A
JAVA A
C language A
Data Structure B
Signal Processing C
Database D
=====

Retake Database Lecture and Get A
Computer Network A
JAVA A
C language A
Database A
Data Structure B
Signal Processing C

```

출력 예시

# 주요 함수

## String 생성자

생성자	설명
<code>String()</code>	빈 스트링 객체 생성
<code>String(char[] value)</code>	<code>char</code> 배열에 있는 문자들을 스트링 객체로 생성
<code>String(String original)</code>	매개변수로 주어진 문자열과 동일한 스트링 객체 생성
<code>String(StringBuffer buffer)</code>	매개변수로 주어진 스트링 버퍼의 문자열을 스트링 객체로 생성

## String 주요 메소드

메소드	설명
<code>char charAt(int index)</code>	<code>index</code> 인덱스에 있는 문자 값 리턴
<code>int codePointAt(int index)</code>	<code>index</code> 인덱스에 있는 유니코드 값 리턴
<code>int compareTo(String anotherString)</code>	두 스트링을 사전 순으로 비교하여 두 스트링이 같으면 0, 현 스트링이 <code>anotherString</code> 보다 먼저 나오면 음수, 아니면 양수 리턴
<code>String concat(String str)</code>	현재 스트링 뒤에 <code>str</code> 스트링을 덧붙인 새로운 스트링 리턴
<code>boolean contains(CharSequence s)</code>	<code>s</code> 에 지정된 문자들을 포함하고 있으면 <code>true</code> 리턴
<code>int length()</code>	스트링의 길이(문자 개수) 리턴
<code>String replace(CharSequence target, CharSequence replacement)</code>	<code>target</code> 이 지정하는 일련의 문자들을 <code>replacement</code> 가 지정하는 문자들로 변경한 스트링 리턴
<code>String[] split(String regex)</code>	정규식 <code>regex</code> 에 일치하는 부분을 중심으로 스트링을 분리하고, 분리된 스트링들을 배열로 저장하여 리턴
<code>String substring(int beginIndex)</code>	<code>beginIndex</code> 인덱스부터 시작하는 서브 스트링 리턴
<code>String toLowerCase()</code>	소문자로 변경한 스트링 리턴
<code>String toUpperCase()</code>	대문자로 변경한 스트링 리턴
<code>String trim()</code>	스트링 앞뒤의 공백 문자들을 제거한 스트링 리턴



## StringBuffer 생성자

생성자	설명
<code>StringBuffer()</code>	초기 버퍼의 크기가 16인 스트링 버퍼 객체 생성
<code>StringBuffer(charSequence seq)</code>	<code>seq</code> 가 지정하는 일련의 문자들을 포함하는 스트링 버퍼 생성
<code>StringBuffer(int capacity)</code>	지정된 초기 크기를 갖는 스트링 버퍼 객체 생성
<code>StringBuffer(String str)</code>	지정된 스트링으로 초기화된 스트링 버퍼 객체 생성

## StringBuffer 주요 메소드

메소드	설명
<code>StringBuffer append(String str)</code>	<code>str</code> 스트링을 스트링 버퍼에 덧붙인다.
<code>StringBuffer append(StringBuffer sb)</code>	<code>sb</code> 스트링 버퍼를 현재의 스트링 버퍼에 덧붙인다. 이 결과 현재 스트링 버퍼의 내용이 변한다.
<code>int capacity()</code>	스트링 버퍼의 현재 크기 리턴
<code>StringBuffer delete(int start, int end)</code>	<code>start</code> 위치에서 <code>end</code> 위치 앞까지 부분 문자열 삭제
<code>StringBuffer insert(int offset, String str)</code>	<code>str</code> 스트링을 스트링 버퍼의 <code>offset</code> 위치에 삽입
<code>StringBuffer replace(int start, int end, String str)</code>	스트링 버퍼 내의 <code>start</code> 위치의 문자부터 <code>end</code> 가 지정하는 문자 앞의 서브 스트링을 <code>str</code> 로 대체
<code>StringBuffer reverse()</code>	스트링 버퍼 내의 문자들을 반대 순서로 변경
<code>void setLength(int newLength)</code>	스트링 버퍼 내 문자열 길이를 <code>newLength</code> 로 재설정, 현재 길이보다 큰 경우 널 문자(' ')로 채우며 작은 경우는 기존 문자열이 잘린다.

## StringTokenizer 생성자

생성자	설명
StringTokenizer(String str)	str 스트링의 각 문자를 구분 문자로 문자열을 분리하는 스트링 토크나이저 생성
StringTokenizer(String str, String delim)	str 스트링과 delim 구분 문자로 문자열을 분리하는 스트링 토크나이저 생성
StringTokenizer(String str, String delim, boolean returnDelims)	str 스트링과 delim 구분 문자로 문자열을 분리하는 스트링 토크나이저 생성. returnDelims가 true이면 delim이 포함된 문자도 토큰에 포함된다.

## StringTokenizer 주요 메소드

메소드	설명
int countTokens()	스트링 토크나이저가 분리한 토큰의 개수 리턴
boolean hasMoreTokens()	스트링 토크나이저에 다음 토큰이 있으면 true 리턴
String nextToken()	스트링 토크나이저에 들어 있는 다음 토큰 리턴

## LinkedList 주요 메서드

메소드	설명
boolean add(E element)	ArrayList의 맨 뒤에 element 추가
void add(int index, E element)	인덱스 index 위치에 element 삽입
boolean addAll(Collection<? extends E> c)	컬렉션 c의 모든 요소를 ArrayList의 맨 뒤에 추가
void clear()	ArrayList의 모든 요소 삭제
boolean contains(Object o)	ArrayList가 지정된 객체를 포함하고 있으면 true 리턴
E elementAt(int index)	index 인덱스의 요소 리턴
E get(int index)	index 인덱스의 요소 리턴
int indexOf(Object o)	o와 같은 첫 번째 요소의 인덱스 리턴, 없으면 -1 리턴
boolean isEmpty()	ArrayList가 비어있으면 true 리턴
E remove(int index)	index 인덱스의 요소 삭제
boolean remove(Object o)	o와 같은 첫 번째 요소를 ArrayList에서 삭제
int size()	ArrayList가 포함하는 요소의 개수 리턴
Object[] toArray()	ArrayList의 모든 요소를 포함하는 배열 리턴

## Vector 주요 메소드

메소드	설명
<code>boolean add(E element)</code>	벡터의 맨 뒤에 <code>element</code> 추가
<code>void add(int index, E element)</code>	인덱스 <code>index</code> 에 <code>element</code> 를 삽입
<code>int capacity()</code>	벡터의 현재 용량 리턴
<code>boolean addAll(Collection&lt;? extends E&gt; c)</code>	컬렉션 <code>c</code> 의 모든 요소를 벡터의 맨 뒤에 추가
<code>void clear()</code>	벡터의 모든 요소 삭제
<code>boolean contains(Object o)</code>	벡터가 지정된 객체 <code>o</code> 를 포함하고 있으면 <code>true</code> 리턴
<code>E elementAt(int index)</code>	인덱스 <code>index</code> 의 요소 리턴
<code>E get(int index)</code>	인덱스 <code>index</code> 의 요소 리턴
<code>int indexOf(Object o)</code>	<code>o</code> 와 같은 첫 번째 요소의 인덱스 리턴, 없으면 <code>-1</code> 리턴
<code>boolean isEmpty()</code>	벡터가 비어 있으면 <code>true</code> 리턴
<code>E remove(int index)</code>	인덱스 <code>index</code> 의 요소 삭제
<code>boolean remove(Object o)</code>	객체 <code>o</code> 와 같은 첫 번째 요소를 벡터에서 삭제
<code>void removeAllElements()</code>	벡터의 모든 요소를 삭제하고 크기를 <code>0</code> 으로 만들
<code>int size()</code>	벡터가 포함하는 요소의 개수 리턴
<code>Object[] toArray()</code>	벡터의 모든 요소를 포함하는 배열 리턴

## Iterator 인터페이스 메소드

메소드	설명
<code>boolean hasNext()</code>	방문할 요소가 남아 있으면 <code>true</code> 리턴
<code>E next()</code>	다음 요소 리턴
<code>void remove()</code>	마지막으로 리턴된 요소 제거