

제1장

프로그래밍의 기초

과목 개요

- C 언어의 문법을 어느 정도 알고 있으나
- 실제로 프로그래밍 경험이 거의 없거나 능숙하지 않은 ...
- 보통의 자료구조 수업보다 낮은 수준에서 시작해서
- Java 언어의 문법 + 기본적인 프로그래밍 스킬 + 자료구조를 동시에 학습
- **학습 방법**
 - 프로그래밍 경험이 거의 없는 경우 ➔ 처음부터 시작
 - Java 언어는 처음이지만 C 프로그래밍에는 어느 정도 능숙한 경우 ➔ 1.3절에서 시작
 - Java 사용 경험이 있지만 클래스, 객체 등의 개념을 다시 학습하고 싶은 경우 ➔ 2장에서 시작
 - Java 언어에 능숙한 경우 ➔ 3장에서 시작

1.1 시작하기: 변수, 배열, 그리고 반복문

Hello.java

Java 프로그램은 하나 혹은 그 이상의 클래스로 구성된다. 이 프로그램은 Hello라는 이름의 하나의 클래스로 이루어졌다.

```
public class Hello  
{  
    public static void main(String [] args)  
    {  
        System.out.println("Hello, Java!");  
    }  
}
```

main 메서드는 프로그램 실행이 시작되는 곳이다.

화면에 문자열을 출력한다. `System.out.println`은 C언어의 `printf`문에 해당한다. 출력하고 싶은 문자열을 껍따옴표("")로 묶는다.

Eclipse Tip: 자동완성 기능을 제공한다. 단축키는 `Ctrl-Space`이다.

Hello.java

```
public class Hello  
{  
    public static void main(String [] args)  
    {  
        System.out.println("Hello, Java!");  
    }  
}
```

Java에서 클래스 이름은 대문자로 시작하고, 여러 단어로 구성될 경우 각 단어를 대문자로 시작하는 것이 관습이다.

public과 static 키워드에 대해서는 나중에 공부한다 .

main 메서드는 항상 이렇게 선언된다. 당분간은 그냥
그려려니 하자.

겹따옴표로 둘러싸인 문자들은 하나의
String 데이터로 취급된다.

들여쓰기(indentation)를 체계적으로 하는 것은 매우
중요하다.

Eclipse Tip: 코드 블록을 선택하고 **Ctrl-i**를 누르면
자동으로 들여쓰기를 정리해준다. 항상 이렇게 하는 습
관을 들이자.

Code1.java

```
/*
 * Code1.java
 * Defining and assigning values to fields and local variables.
 */
```

```
class Code1 {
    public static int num;
```

```
    public static void main(String [] args) {
```

```
        int anotherNum = 5;
        num = 2;
```

```
        // print out some information
```

```
        System.out.println(num + anotherNum);
```

```
        System.out.println("num: " + num);
```

```
        System.out.println("anotherNum: " + anotherNum);
    }
```

```
}
```

변수는 `main` 메서드 내부에 선언될 수도 있고 외부에 선언될 수도 있다. 하지만 클래스 외부에 선언될수는 없다. 메서드 외부에 선언된 변수가 반드시 `static`이어야하는 것은 아니지만 당분간은 이렇게 `static` 변수에 대해서만 다룬다.

메서드 내부에 선언된 변수는 그 메서드 내에서만 사용 가능하고, 메서드 외부에 선언된 변수는 그 클래스 전체에서 사용될 수 있다.

"`num:`"는 문자열(string)이고, `num`은 정수이다. Java에서 +의 양쪽 중 하나가 문자열이면 다른쪽도 문자열로 해석하여 두 문자열을 합친다. 따라서 "`num:`" + `num`은 "`num: 2`"라는 하나의 문자열이

변수의 타입

타입 이름	설명
byte	정수
short	정수
→ int	정수
long	정수
float	소수
→ double	소수
char	문자 하나
→ boolean	참 또는 거짓
→ String	문자열
그밖의 타입들	* 나중에 배울 것임

} **프리미티브 타입**

- 변수는 데이터를 보관하는 장소(memory)
- 변수는 사용하기 전에 선언되어야 한다. 변수의 선언이란 “이름”과 “타입”을 정해주는 것
- 변수는 “적용 범위(scope)”를 가짐

메서드(블록) 내부에 선언된 변수는 그 메서드(블록) 내부에서만 사용 가능

메서드 외부에 (클래스 내부) 선언된 변수는 클래스 내에서 사용 가능

Code2.java

```
import java.util.Scanner;
```

import문은 C언어의 include와 유사하다. 즉 라이브러리에 포함된 기능들을 사용할 수 있게 해준다.

```
class Code2 {
```

```
    public static void main(String [] args) {
        int number = 123;

        Scanner keyboard = new Scanner(System.in);

        System.out.print("Please enter an integer: ");

        int input = keyboard.nextInt();

        if(input == number) {
            System.out.println("Numbers match! :-)");
        } else {
            System.out.println("Numbers do not match! :-((");
        }
    }
}
```

Eclipse Tip:

초보자의 경우 뭘 import해야 하는지 알기 어렵다. import문 없이 먼저 코드를 작성한 후 Eclipse의 메뉴 “Source>OrganizeImports”를 실행하면 자동으로 필요한 import문을 생성해 준다. 단축키는 “Ctrl-Shift-O”이다.

Code2.java

```
import java.util.Scanner;  
  
class Code2 {  
  
    public static void main(String [] args) {  
        int number = 123;  
  
        Scanner keyboard = new Scanner(System.in);  
  
        System.out.print("Please enter an integer: ");  
  
        int input = keyboard.nextInt();  
  
        if(input == number) {  
            System.out.println("Numbers match! :-)");  
        } else {  
            System.out.println("Numbers do not match! :-(");  
        }  
    }  
}
```

키보드로 부터 데이터를 입력받기 위해서는 이렇게 `Scanner`라는 객체를 만든다. `System.in`은 키보드를 의미한다.

키보드로 부터 하나의 정수를 입력받아 변수 `input`에 저장한다.

Code3.java

```
import java.util.Scanner;
```

```
class Code3 {  
    public static void main(String [] args) {  
        String str = "Hello, world";  
        String input = null;  
        Scanner keyboard = new Scanner(System.in);  
        System.out.print("Please type a string: ");  
  
        input = keyboard.next(); ——————  
  
        if(str == input){  
            System.out.println("Strings match! :-)"); ——————  
        } else {  
            System.out.println("Strings do not match! :-(");  
        }  
    }  
}
```

next() 메서드는 하나의 문자열을 읽어준다. 이외에도 실수를 읽을 때는 nextDouble(), 한 라인을 통채로 읽을 때는 nextLine() 등이 있다.

두 String을 비교한다. 의도한 대로 되는지
각자 테스트해본다.

Code3.java

```
import java.util.Scanner;

class Code3 {
    public static void main(String [] args) {
        String str = "Hello, world";
        String input = null;
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Please type a string: ");

        input = keyboard.next();

        if(str.equals(input)) {
            System.out.println("Strings match! :-)");
        } else {
            System.out.println("Strings do not match! :-( ");
        }
    }
}
```

String간의 비교는 ==연산자로 할 수 없고 equals 메서드를 사용한다. 이 문장은 input.equals(str)이라고 할 수도 있다. 결과는 true혹은 false이다.

Code4.java

```
import java.util.Scanner;  
class Code4 {  
    public static void main(String [] args) {  
        String name = null;  
        int age;  
        String gender;  
  
        Scanner keyboard = new Scanner(System.in);  
        System.out.print("Please type your name and your age and your gender: ");  
  
        name = keyboard.next();  
        age = keyboard.nextInt();  
        gender = keyboard.next();  
  
        if (gender.equals("male"))  
            System.out.println(name + ", you're " + age + " years old man.");  
        else if (gender.equals("female"))  
            System.out.println(name + ", you're " + age + " years old woman.");  
        else  
            System.out.println("Hmm... interesting.");  
    }  
}
```

연속해서 키보드로 부터 읽는다.

이렇게 if ~ else if ~ else
와 같이 연속해서 비교를 할 수 있다.

Code5.java

```
class Code5 {  
    public static void main(String [] args) {  
        // declare the array  
        int [] grades; ——————  
  
        // allocate memory for 5 indices  
        grades = new int[5]; ——————  
  
        // assign some values to the array  
        grades[0] = 100;  
        grades[1] = 76;  
        grades[2] = 92;  
        grades[3] = 95;  
        grades[4] = 14;  
  
        // print out each value  
        System.out.println(grades[0]);  
        System.out.println(grades[1]);  
        System.out.println(grades[2]);  
        System.out.println(grades[3]);  
        System.out.println(grades[4]);  
    }  
}
```

Java에서 배열을 선언하는 방법은 C언어와 조금 다르다. 배열을 사용하기 위해서는 먼저 이렇게 배열을 선언한다.

그런 다음 이렇게 배열의 크기를 지정하면서 생성한다. 실제 배열이 만들어지는 시점은 여기이다.

이 두 라인은 다음과 같이 한 라인으로 만들 수도 있다.

int[] grades = new int [5];

배열의 각 칸에 데이터를 저장하고, 저장된 데이터를 읽기 위해서 []사용한다. 배열의 인덱스는 0부터 시작한다.

Code6.java

```
class Code6 {  
    public static void main(String [] args) {  
  
        int [] grades;  
  
        grades = new int[5];  
  
        grades[0] = 100;  
        grades[1] = 76;  
        grades[2] = 92;  
        grades[3] = 95;  
        grades[4] = 14;  
  
        for(int i = 0; i < grades.length; i++) {  
            System.out.println("Grade "+(i+1)+": "+grades[i]);  
        }  
    }  
}
```

for 반복문을 사용하면 동일한 일을 반복하는 것을 효과적으로 표현할 수 있다.

이 위치에 선언된 변수 i의 적용범위
(scope)는 이 for문에 한정된다.

Java의 배열은 length라는 기능을 제공한다. grades.length는 5이다. length는 배열의 크기이지 배열에 저장된 데이터의 개수는 아니다.

Code7.java

```
class Code7 {  
    public static void main(String [] args) {  
        int[] grades = new int[5];  
  
        grades[0] = 100;  
        grades[1] = 76;  
        grades[2] = 92;  
        grades[3] = 95;  
        grades[4] = 14;  
  
        int i = 0;  
        while(i<grades.length) {  
            System.out.println("Grade "+(i+1)+": "+grades[i]);  
            i++;  
        }  
    }  
}
```

while은 for와 함께 대표적인 반복문의 하나이다.

모든 for문은 while문으로 바꿀 수 있고, 그 역도 마찬가지이다. 일반적으로 반복의 횟수가 정해져 있을 경우 for문을, 그렇지 않을 경우 while문을 선호한다.

Code8.java

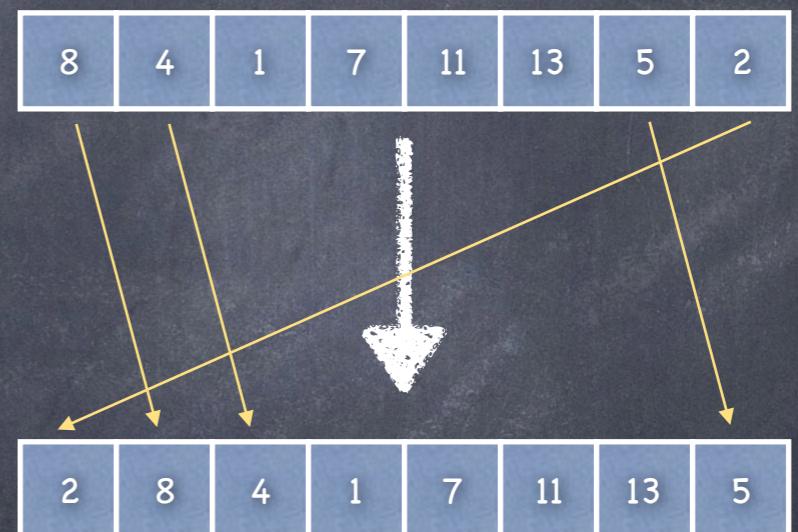
```
class Code8 {  
    public static void main(String [] args) {  
        Scanner kb = new Scanner(system.in);  
        int n = kb.nextInt();  
        int [] data = new int [n];  
  
        for (int i=0; i<n; i++)  
            data[i] = kb.nextInt();  
        kb.close();  
  
        int sum = 0, max = data[0];  
        for ( int i=0; i<n; i++) {  
            sum += data[i];  
            if (data[i] > max) max = data[i];  
        }  
        System.out.println("The sum is " + sum + " and the maximum is " + max);  
    }  
}
```

사용자로부터 n개의 정수를 입력받은 후 합과 최대값을 구하여 출력하는 코드이다.

Code9.java

```
class Code9 {  
    public static void main(String[] args) {  
        Scanner kb = new Scanner(system.in);  
        int n = kb.nextInt();  
        int [] data = new int [n];  
        for (int i=0; i<n; i++)  
            data[i] = kb.nextInt();  
        kb.close();  
  
        int tmp = data[n-1];  
        for (int i=n-2; i>=0; i--)  
            data[i+1] = data[i];  
        data[0] = tmp;  
  
        for (int i=0; i<n; i++)  
            System.out.println(data[i]);  
    }  
}
```

n개의 정수를 입력받아 순서대로 배열에 저장한다. 그런 다음 모든 정수들을 한 칸씩 오른쪽으로 shift하라. 마지막 정수는 배열의 첫 칸으로 이동하라.



Code10.java

```
class Code10 {  
    public static void main(String[] args) {  
        for ( int n=2; n<=100000; n++ ) {  
            boolean isPrime = true;  
            for (int i=2; i*i<=n && isPrime; i++) {  
                if (n%i==0)  
                    isPrime = false;  
            }  
            if (isPrime)  
                System.out.println(n);  
        }  
    }  
}
```

1~100000 사이의 모든 소수들을 찾아서 출력하는
프로그램이다.

각각의 정수 n 에 대해서 이 **for** 문을 돌면서 2보다 크거나 같은 약수가 있는지 검사한다. 하나라도 약수가 있다면 이미 소수가 아니므로 더이상 검사할 필요가 없다. 변수 **isPrime**이 어떤 역할을 하는지 잘 생각해보라.

Code11.java

```
class Code11 {  
    public static void main(String[] args) {  
        Scanner kb = new Scanner(system.in);  
        int n = kb.nextInt();  
        int [] data = new int [n];  
        for (int i=0; i<n; i++)  
            data[i] = kb.nextInt();  
        kb.close();  
        int count = 0;  
        for ( int i=0; i<n-1; i++) {  
            for ( int j=i+1; j<n; j++ ) {  
                if (data[i] == data[j] )  
                    count++;  
            }  
        }  
        System.out.println(count);  
    }  
}
```

사용자로부터 먼저 정수의 개수 n을 입력받는다. 이어서 n개의 정수를 입력받아 순서대로 배열에 저장한다. 그런 다음 중복된 정수쌍의 개수를 카운트하여 출력하라. 예를 들어 n=6이고 입력된 정수들이 2, 4, 2, 4, 5, 2이면 중복된 정수쌍은 (2,2), (2,2), (2,2), (4,4)로 모두 4쌍이다.

Code12.java

```
class Code12 {  
    public static void main(String[] args) {  
        Scanner kb = new Scanner(system.in);  
        int n = kb.nextInt();  
        int [] data = new int [n];  
        for (int i=0; i<n; i++)  
            data[i] = kb.nextInt();  
        kb.close();  
  
        int maxSum = 0;  
        for ( int i=0; i<n; i++ ) {  
            for (int j=i; j<n; j++) {  
                int sum = 0;  
                for (int k=i; k<=j; k++)  
                    sum += data[k];  
                if (sum > maxSum)  
                    maxSum = sum;  
            }  
        }  
        System.out.println("The max sum is " + maxSum);  
    }  
}
```

n개의 정수를 입력받아 배열에 저장한다. 이들 중에서 0개 이상의 연속된 정수들을 더하여 얻을 수 있는 최대값을 구하여 출력하는 프로그램을 작성하라.



28

Code13.java

```
class Code13 {  
    public static void main(String[] args) {  
        /* 지금까지와 동일하게 n개의 정수를 입력받는다. */  
        int maxPrime = 0;  
        for (int i=0; i<n; i++) {  
            for (int j=i; j<n; j++) {  
  
                int val = 0;  
                for (int k=i; k<=j; k++)  
                    val = val * 10 + data[k];  
  
                boolean isPrime = true;  
                for (int p = 2; p<val/2 && isPrime; p++) {  
                    if (val % p == 0) isPrime = false;  
                }  
  
                if (isPrime && val > 1 && val > maxPrime)  
                    maxPrime = val;  
            }  
            if (maxPrime > 0)  
                System.out.println("The max prime number is " + maxPrime);  
            else  
                System.out.println("No prime number exists.");  
        }  
    }  
}
```

n개의 음이 아닌 한 자리 정수를 입력받아 배열에 저장한다. 이들 중에서 1개 이상의 연속된 정수들을 합하여(?) 얻을 수 있는 소수 들 중에서 최대값을 구하여 출력하는 프로그램을 작성하라.

data[i]...data[j]를 하나의 정수로 환산한다.

1	9	4	0	7	1	3	6	2	3
---	---	---	---	---	---	---	---	---	---

713으로 해석

Code14.java

```
class Code14 {  
    public static void main(String[] args) {  
        Scanner kb = new Scanner(system.in);  
        int n = kb.nextInt();  
        int [] data = new int [n];  
  
        for (int i=0; i<n; i++)  
            data[i] = kb.nextInt();  
        kb.close();  
    }  
}
```

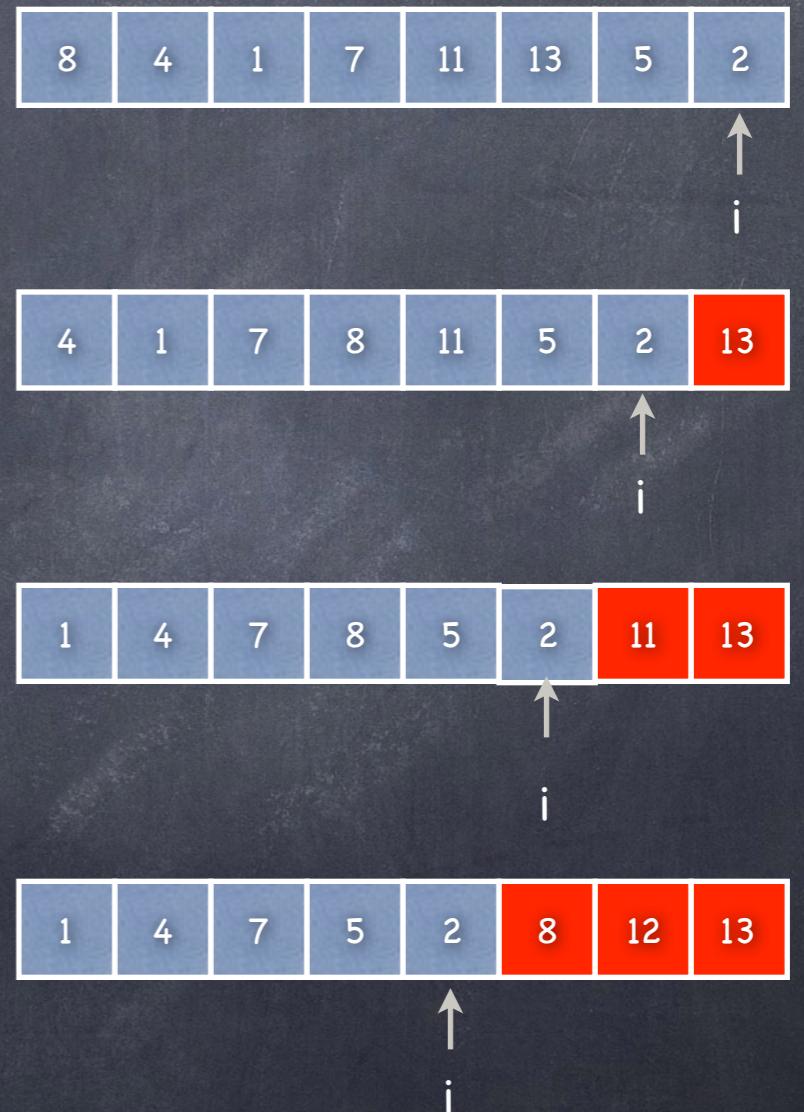
사용자로부터 n개의 정수를 입력받은 후 오름차순으로 정렬(sort)하여 출력하는 코드이다.

Code14.java (continued)

버블정렬(bubblesort) 알고리즘

```
// bubble sort
for ( int i=n-1; i>0; i-- ) {
    for ( int j=0; j<i; j++ ) {
        if ( data[j] > data[j+1] ) {
            swap {
                int tmp = data[j];
                data[j] = data[j+1];
                data[j+1] = tmp;
            }
        }
    }
    System.out.println("Sorted data:");
    for ( int i=0; i<n; i++ )
        System.out.println(data[i]);
}
```

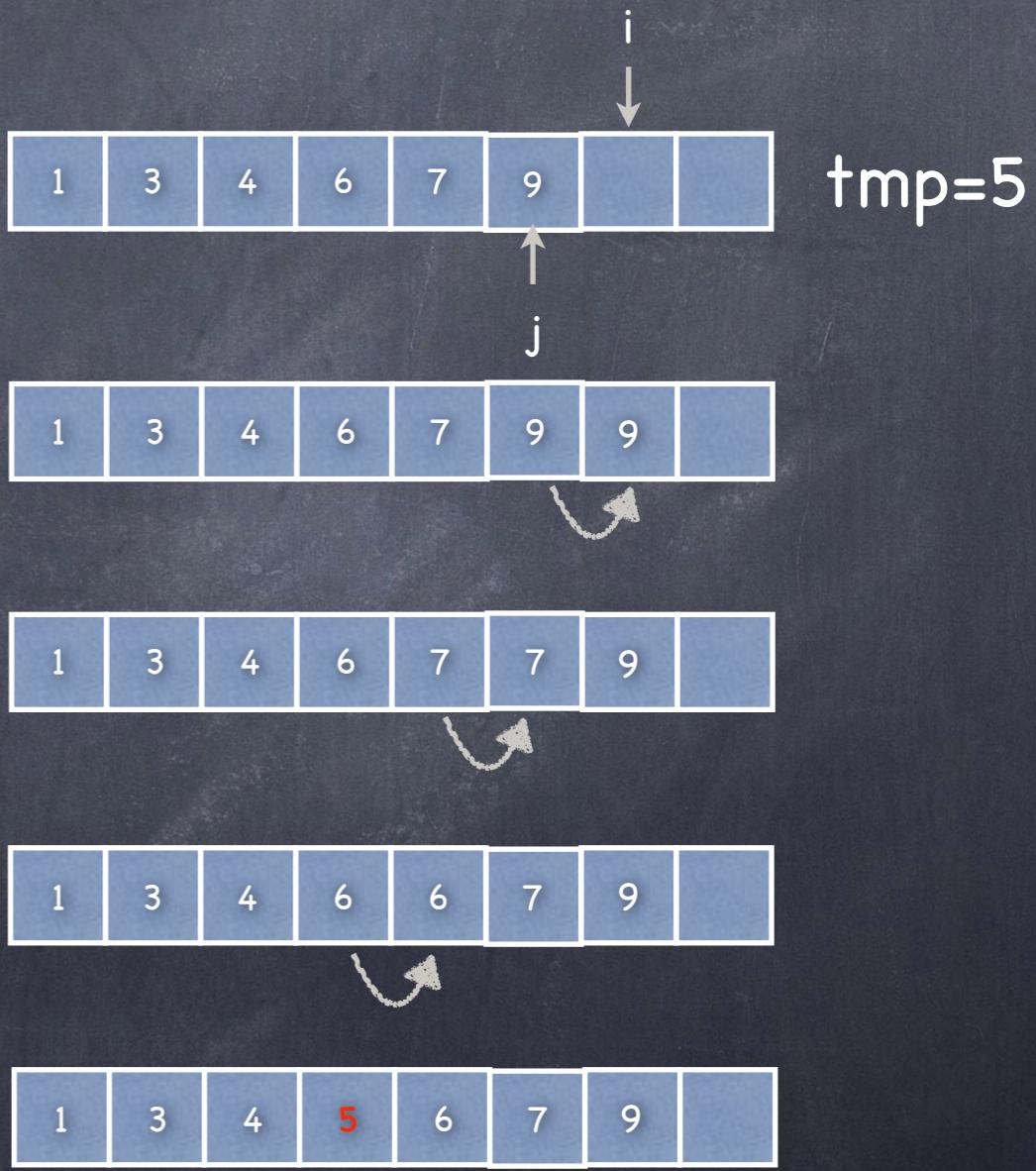
swap {
 int tmp = data[j];
 data[j] = data[j+1];
 data[j+1] = tmp;
}
data[0] ~ data[i] 중에서
최대값을
data[i] 위치로 몰아가는 일



Code15.java

```
class Code15 {  
    public static void main(String[] args) {  
        Scanner kb = new Scanner(system.in);  
        int n = kb.nextInt();  
        int [] data = new int [n];  
  
        for (int i = 0; i < n; i++) {  
            int tmp = kb.nextInt();  
            int j = i-1;  
            while ( j>=0 && data[j] > tmp) {  
                data[j+1] = data[j];  
                j--;  
            }  
            data[j+1] = tmp;  
  
            for (int k=0; k<=i; k++)  
                System.out.print(data[k] + " ");  
            System.out.println();  
        }  
        kb.close();  
    }  
}
```

사용자로부터 n개의 정수를 입력받는다. 정수가 하나씩 입력될 때마다 현재까지 입력된 정수들을 오름차순으로 정렬하여 출력하라.



Task Set #1

- 1 ~ 100000 사이의 정수들 중 소수의 개수를 세어 출력하는 프로그램을 작성하라.
- 1 ~ 100000 사이의 정수들 중에 가장 작은 50개의 소수를 찾아 그 합을 구하여 출력하는 프로그램을 작성하라. 만약 소수의 개수가 50개 미만이라면 모든 소수의 합을 출력하라.
- 사용자로부터 n 개의 정수를 받아서 그 중 모든 짝수를 더한 값에서 모든 홀수를 뺀 값을 계산해 출력하는 프로그램을 작성하라. 입력 형식은 먼저 n 의 값이 주어지고 이어서 n 개의 정수들이 주어진다. 예를 들어 $n=5$ 이고 입력 정수들이 8, 1, 6, 9, 12라면 답은 $26-10=16$ 이다.
- 사용자로부터 하나의 양의 정수 n 을 입력받는다. 그런 다음 다음과 같이 n 줄을 출력하는 프로그램을 작성하라.

1
1 2
1 2 3
1 2 3 4

$n=4$ 인 경우의 예

- 사용자로부터 n 개의 정수를 받아서 그 중 최대값에서 최소값을 뺀 값을 계산해 출력하는 프로그램을 작성하라. 입력 형식은 3번과 동일하다.

Task Set #1

6. 사용자로부터 하나의 양의 정수 n 을 입력받는다. n 보다 크거나 같으면서 가장 작은 2의 거듭제곱수를 찾아서 출력하는 프로그램을 작성하라. 예를 들어 $n=12$ 이면 출력은 $2^4=16$ 이다.
7. 사용자로부터 n 개의 정수를 입력받은 후 그 중 서로 다르면서 차이가 가장 작은 두 수를 찾아 출력하는 프로그램을 작성하라. 입력 형식은 먼저 n 의 값이 주어지고 이어서 n 개의 정수들이 주어진다. 예를 들어 $n=5$ 이고 정수들이 8, 1, 6, 9, 6이라면 이 중 서로 다르면서 차이가 최소인 두 수는 8과 9이다.
8. 사용자로부터 n 개의 정수를 입력받아 크기순으로 정렬한 후 중복된 수를 제거하고 출력하는 프로그램을 작성하라. 입력 형식은 먼저 n 의 값이 주어지고 이어서 n 개의 정수들이 주어진다. 예를 들어 $n=8$ 이고 입력된 정수들이 4 7 4 12 4 10 9 7이었다면 출력은 4 7 9 10 12이다.
9. 사용자로부터 n 개의 정수를 입력받아 배열에 저장한 후 짝수 인덱스에 저장된 수는 그대로 두고 홀수 인덱스에 저장된 수들은 오름차순으로 정렬하여 출력하는 프로그램을 작성하라. 예를 들어 입력된 정수들이 1 7 4 12 5 10 9 7이라면 출력은 1 7 4 7 5 10 9 12이다.

Task Set #1

10. 사용자로부터 하나의 양의 정수 n 을 입력받고 이어서 n 개의 정수가 입력된다. 그런 다음 다시 하나의 정수 K 가 입력된다. n 개의 정수들 중에서 2개의 정수를 선택해 그 합이 K 가 되는 경우의 수를 카운트하는 프로그램을 작성하라. 단 같은 정수를 중복 선택하는 것은 허용하지 않는다.
11. 입력으로 두 행렬($p \times q$ 행렬과 $q \times r$ 행렬)을 받아서 두 행렬을 곱하는 프로그램을 작성하라. 입력의 형식은 다음과 같다. 먼저 첫 줄에는 첫 번째 행렬의 크기 p 와 q 가 주어지고, 이어지는 p 줄에는 각 줄마다 q 개의 정수가 주어진다. 다음으로 두 번째 행렬의 크기 q 와 r 가 주어지고, 마찬가지로 이어지는 q 줄에는 각 줄마다 r 개의 정수가 주어진다. 두 행렬의 곱을 구하여 출력한다.

3 4 // $p=3, q=4$

12 4 8 3
-5 7 0 7
9 -3 4 7 7

4 2 // $q=4, r=2$

2 8
9 12
0 45
18 -61

1.2 메서드 호출과 프로그램의 기능적 분할

Code16.java

```
class Code16 {  
    public static void main(String[] args) {  
        Scanner keyboard = new Scanner(System.in);  
        System.out.println("Please enter two integers and press Enter.");
```

입력으로 두 정수 a와 b를 받아 a의 b
승을 계산한다.

```
int a = keyboard.nextInt();  
int b = keyboard.nextInt();
```

4. 메서드 power가 return문으로 넘겨준 값을 받
아서 변수 result에 저장한다.

```
int result = power(a,b);  
System.out.println("The result is " + result);  
keyboard.close();
```

1. 메서드 power를 호출하면서 매개변
수로 정수 a와 b의 값을 전낸다.

```
}
```

```
public static int power(int n, int m) {  
    int result = 1;  
    for ( int i=0; i<m; i++ )  
        result *= n;  
    return result;  
}
```

2. 메서드 power는 매개변수로 두 개의 정수를 전내 받
으며 각각을 m과 n이라고 이름 짓는다.

3. 메서드 power는 계산 결과, 즉 변수 result의 값을
return문을 이용해 자신을 호출한 이에게 넘겨준다.

Code17.java

```
class Code17 {  
    public static void main(String[] args) {  
        for ( int i=2; i<=100000; i++ ) {  
            if (isPrime(i))  
                System.out.println(i);  
        }  
    }  
  
    public static boolean isPrime(int k) {  
        if (k<2)  
            return false;  
        for (int i=2; i*i<=k; i++) {  
            if (k%i==0)  
                return false;  
        }  
        return true;  
    }  
}
```

1~100000 사이의 소수를 찾아 출력한다.

Code18.java

사용자로부터 n개의 정수를 입력받은 후 오름차순으로 정렬(sort)하여 출력하는 코드이다.

```
class Code18 {  
    public static void main(String[] args) {  
        Scanner kb = new Scanner(System.in);  
        int n = kb.nextInt();  
        int [] data = new int [n];  
        for (int i=0; i<n; i++)  
            data[i] = kb.nextInt();  
        kb.close();  
    }  
}
```

Code18.java (continued)

```
bubbleSort(data, n);
System.out.println("Sorted data:");
for ( int i=0; i<n; i++)
    System.out.println(data[i]);
}

public static void bubbleSort(int [] data, int n) {
    for ( int i=n-1; i>0; i--) {
        for ( int j=0; j<i; j++ ) {
            if (data[j] > data[j+1]) {
                int tmp = data[j];
                data[j] = data[j+1];
                data[j+1] = tmp;
            }
        }
    }
}
```

1. 메서드 bubbleSort를 호출하면서 매개변수로 배열 data와 n의 값을 전낸다.

2. 메서드 bubbleSort는 매개변수로 하나의 배열과 하나의 정수를 전내 받으며 각각을 data와 n이라고 이름 짓는다.

3. 메서드 bubbleSort는 아무 것도 return하지 않는다. 하지만 이 메서드가 한 일은 배열 data에 이미 반영되어 있다.

Code18_2.java

```
class Code18_2 {  
    public static void main(String[] args) {  
        // exactly the same with Code18  
    }  
  
    public static void bubbleSort(int [] data, int n) {  
        for ( int i=n-1; i>0; i--) {  
            for ( int j=0; j<i; j++ ) {  
                if ( data[j] > data[j+1] )  
                    swap(data[j], data[j+1]);  
            }  
        }  
    }  
  
    public static void swap(int a, int b) {  
        int tmp = a;  
        a= b;  
        b = tmp;  
    }  
}
```

두 정수 data[j]와 data[j+1]을 swap하기 위해 메서드 swap을 호출하였다.

매개변수로 건내받은 두 정수를 swap하였다.

what's wrong?

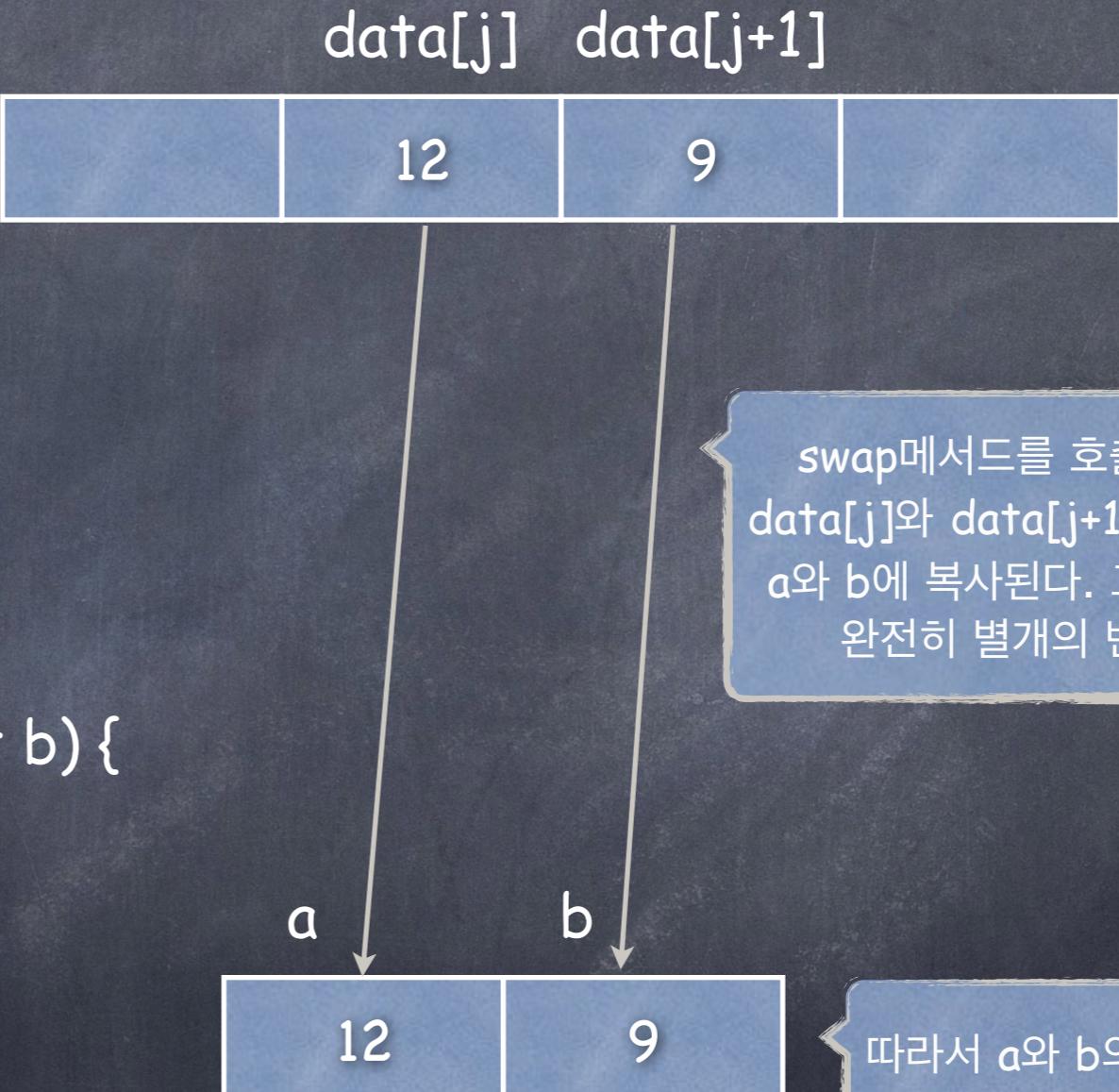
값에 의한 호출 (call by value)

호출문:

```
swap(data[j], data[j+1]);
```

호출된 메서드:

```
public static void swap(int a, int b) {  
    int tmp = a;  
    a = b;  
    b = tmp;  
}
```



값에 의한 호출: 배열

호출문:

```
bubbleSort(data, n);
```

...

1. 매개변수로 배열 data를 넘긴다.

3. 배열 data의 정렬이 이루어졌다.

호출된 메서드:

```
public static void bubbleSort(int [] data, int n) {  
    for ( int i=n-1; i>0; i--) {  
        for ( int j=0; j<i; j++ ) {  
            if (data[j] > data[j+1]) {  
                int tmp = data[j];  
                data[j] = data[j+1];  
                data[j+1] = tmp;  
            }  
        }  
    }  
}
```

2. 배열 data[j]의 내용을 변경한다.

매개변수

타입 이름	설명
byte	정수
short	정수
int	정수
long	정수
float	소수
double	소수
char	문자 하나
boolean	참 또는 거짓
String	문자열
그밖의 타입들	* 나중에 배울 것임

프리미티브 타입

- 프리미티브 타입의 매개변수는 호출된 메서드에서 값을 변경하더라도 호출한 쪽에 영향을 주지 못한다. 이것은 “값에 의한 호출”이기 때문이다.
- 배열의 값은 호출된 메서드에서 변경하면 호출한 쪽에서도 변경된다.
- 비일관성?

Code19.java

데이터 파일은 프로젝트 폴더에 있어야 한다.

```
class Code19 {  
    public static void main(String[] args) {  
        Scanner inFile = new Scanner(new File("input.txt"));  
        String [] name = new String [1000];  
        String [] number = new String [1000];  
        int count=0;  
        while ( inFile.hasNext() ) {  
            name[count] = inFile.next();  
            number[count] = inFile.next();  
            count++;  
        }  
        inFile.close();  
  
        for ( int i=0; i<count; i++ )  
            System.out.println("Name: " + name[i] + ", Phone: " + number[i]);  
    }  
}
```

input.txt라는 파일로 부터 사람들의 이름과 전화번호 쌍을 입력받아 배열에 저장하고 출력한다.

System.in 대신에 데이터 파일의 이름을 지정하고 **File**을 만든 후 그 파일에 대한 **Scanner**를 만든다.

hasNext() 메서드는 파일의 끝에 도달했는지 검사한다.

input.txt 파일의 예

```
John 0326547123  
James 0108237462  
Kwon 0511122874  
Lee 82104443642  
Kim 0102873468  
Park 0102783456
```

Code19.java (continued)

The screenshot shows a Java code editor with the following code:

```
public static void main(String[] args) {
    Scanner inFile = new Scanner(new File("input.txt"));
    String [] name = ... ! Add throws declaration
    String [] number ... ! Surround with try/catch
    int count=0;
    while (inFile.has
        name[count] =
        number[count]
        count++;
    }
    inFile.close();

    for ( int i=0; i<
        System.out.pr
    }
}
```

A code completion tooltip is displayed over the line `new Scanner(new File("input.txt"));`. It contains the following suggestions:

- ... ! Add throws declaration
- ... ! Surround with try/catch
- import java.io.File;
- import java.io.FileNotFoundException;
- import java.util.Scanner;
- ...
- public static void main(String[] args) {
- Scanner inFile;
- try {
- inFile = new Scanner(new File("input.txt"));
- } catch (FileNotFoundException e) {
- // TODO Auto-generated catch block
- e.printStackTrace();
- }

At the bottom right of the tooltip, there is a message: "Press 'Tab' from proposal table or click for focus".

- 위 그림과 같이 오류가 발생한다. 빨간 x표시에 마우스를 올려놓고 잠시 기다리면 “Unhandled exception type FileNotFoundException”이라는 메시지가 나타난다.
- 이것은 만약 “input.txt”라는 이름의 파일이 없다면 어떻게 할 것인지를 지정해주지 않았기 때문에 발생한 오류이다.
- 빨간 x표시를 마우스로 클릭하면 위 그림과 같은 메시지가 나온다.

Code19.java (continued)

The screenshot shows a Java code editor with the following code:

```
public static void main(String[] args) {
    Scanner inFile = new Scanner(new File("input.txt"));
    String [] name = ... ! Add throws declaration
    String [] number ... ! Surround with try/catch
    int count=0;
    while (inFile.has
        name[count] =
        number[count]
        count++;
    }
    inFile.close();

    for ( int i=0; i<
        System.out.pr
    }
}
```

A code completion tooltip is displayed over the line `String [] name =`. It contains two suggestions:

- `! Add throws declaration`
- `! Surround with try/catch`

The tooltip also includes imports and other parts of the code:

```
...
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
...
public static void main(String[] args) {
    Scanner inFile;
    try {
        inFile = new Scanner(new File("input.txt"));
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

At the bottom right of the tooltip, there is a message: `Press 'Tab' from proposal table or click for focus`.

- 빨간 x표시를 마우스로 클릭하면 위 그림과 같은 메시지가 나온다.
- 위 그림은 두 가지 해결 방법이 있음을 알려준다. 첫째는 “Add throws declaration”이고 둘째는 “Surround with try/catch”이다.
- 두번째 해결방법인 “Surround with try/catch”를 마우스로 클릭한다.

Code19.java (continued)

- Eclipse가 자동으로 아래 그림과 같이 try/catch문을 생성해 준다.

```
public static void main(String[] args) {  
    Scanner inFile;  
    try {  
        inFile = new Scanner(new File("input.txt"));  
    } catch (FileNotFoundException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
    String [] name = new String [1000];  
    String [] number = new String [1000];  
    int count=0;  
    while (inFile.hasNext()) {  
        name[count] = inFile.next();  
        number[count] = inFile.next();  
        count++;  
    }  
    inFile.close();  
  
    for ( int i=0; i<count; i++ )  
        System.out.println("Name: " + name[i] + ", Phone: " + number[i]);  
}
```

하지만 여전히 오류가 있다. 이 오류들은 이 부분을 try{ ... } 문 내부로 옮겨 주면 해결된다.

Java 문법: 변수를 초기화하지 않고 사용하는 것은 오류이다.

Code19.java (final)

```
class Code19 {  
    public static void main(String[] args) {  
        int count=0;  
        String [] name = new String [1000];  
        String [] number = new String [1000];  
  
        try {  
            Scanner inFile = new Scanner(new File("input.txt"));  
            while (inFile.hasNext()) {  
                name[count] = inFile.next();  
                number[count] = inFile.next();  
                count++;  
            }  
            inFile.close();  
        } catch (FileNotFoundException e) {  
            System.out.println("No data file exists.");  
            return;  
        }  
  
        for ( int i=0; i<count; i++ )  
            System.out.println("Name: " + name[i] + ", Phone: " + number[i]);  
    }  
}
```

적절한 메시지를 출력하고 **return**문 혹은
System.exit(0)으로 프로그램을 종료한다.

Code20.java

```
class Code20 {  
    static String [] name;  
    static String [] number;  
    static int count = 0;  
  
    public static void main(String[] args) {  
        name = new String [1000];  
        number = new String [1000];  
  
        try {  
            Scanner inFile = new Scanner(new File("input.txt"));  
            while (inFile.hasNext()) {  
                name[count] = inFile.next();  
                number[count] = inFile.next();  
                count++;  
            }  
            inFile.close();  
        } catch (FileNotFoundException e) {  
            System.out.println("No data file exists.");  
            return;  
        }  
    }  
}
```

input.txt 파일로부터 이름과 전화번호 쌍을 입력받은
후 이름의 알파벳 순서로 정렬하여 출력한다.

입력받은 이름과 전화번호들을 저장할 배열 name과 number, 그리고 개수
count는 main메서드만이 아니라 bubbleSort 메서드에서 사용된다. 그래
서 main 메서드 외부에 선언하였다. 이렇게 하나의 메서드가 아니라 클래스
전체에서 사용될 데이터는 클래스의 멤버로 만드는 것이 좋다.

Code20.java (continued)

```
public static void bubbleSort() {  
    for ( int i=0; i<count; i++ )  
        System.out.println(name[i] + "s phone number is " + number[i]);  
}  
  
public static void bubbleSort() {  
    for ( int i=count-1; i>0; i-- ) {  
        for ( int j=0; j<i; j++ ) {  
            if (name[j] > name[j+1]) {  
                String tmpName = name[j];  
                name[j] = name[j+1];  
                name[j+1] = tmpName;  
  
                String tmpNumber = number[j];  
                number[j] = number[j+1];  
                number[j+1] = tmpNumber;  
            }  
        }  
    }  
}
```

name, number, 그리고 count가 클래스의 멤버이므로 매개변수로 넘겨줄 필요가 없다.

두 이름에 대해 알파벳순의 순서를 이렇게 비교할 수 있을까?
문자열 간의 사전식 순서는 `String`클래스가 제공하는 `compareTo` 메서드를 이용하여 검사한다. 두 문자열 `str1`과 `str2`에 대해서 `str1.compareTo(str2)`는 `str1`이 빠르면 음수, 같으면 0, 크면 양수를 반환한다. 따라서 이 문장은

`if (name[j].compareTo(name[j+1]) > 0)`

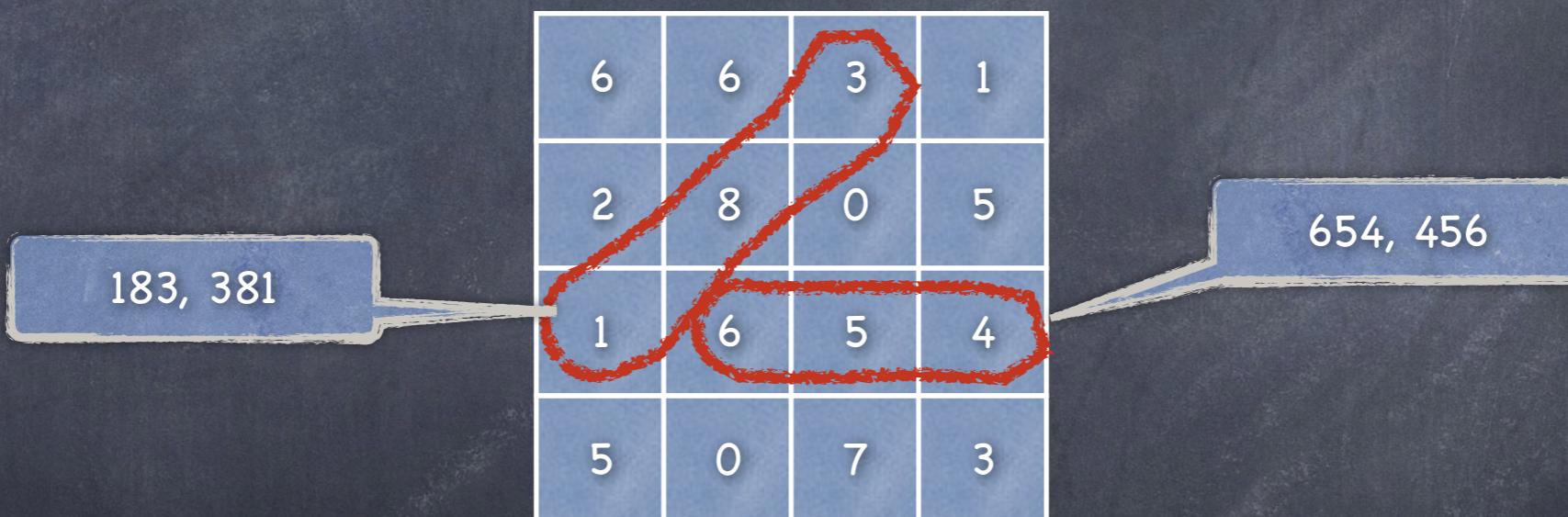
으로 바꿔어야 한다. 이 경우 사람 이름을 비교하는 것으로 대소문자를 구분하지 않는 것이 좋을 것이다. 대소문자 구분없이 비교할 경우

`if (name[j].compareToIgnoreCase(name[j+1]) > 0)`

이름을 swap할 때 번호도 함께 swap해줘야 한다.

2차원 배열에서 소수 찾기

- 입력으로 $n \times n$ 개의 음이 아닌 한자리 정수가 그림과 같이 주어진다. 이 정수들 중 가로, 세로, 대각선의 8방향으로 연속된 숫자들을 합쳐서(?) 만들 수 있는 모든 소수를 찾아서 나열하는 프로그램을 작성하라. 중복된 수를 출력해도 상관없다.



컴퓨터는 빠르다.
뭔가 교묘한 방법을 찾으려 하지 말고
무식하지만 논리적으로 가장 명료한
방법을 먼저 찾아라. 그런 다음 어떻게 개선할 수 있는지를 고민하라.

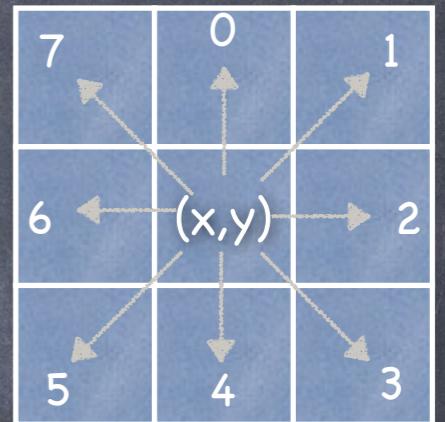
명료하고 단순하게...

모든 가능한 수열들에 대해서
정수값으로 환산하라;
만약 그것이 소수이면
출력한다;

6	6	3	1
2	8	0	5
1	6	5	4
5	0	7	3

모든 가능한 수열들에 대해서

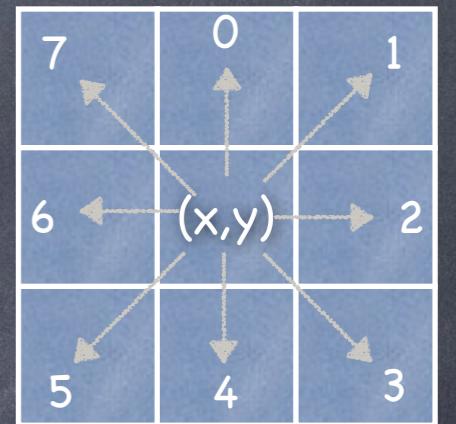
- 하나의 수열은 (시작점, 방향, 길이)에 의해서 정의된다.
- 방향은 0~7번까지의 수로 표현하자.



임의의 시작점 (x, y)
에 대해서 8가지 방향이 있음.
(각 방향을 0~7번 방향이라고 부르자.)

모든 가능한 수열들에 대해서

```
for (int x=0; x<n; x++) {  
    for (int y=0; y<n; y++) {  
        for (int dir=0; dir<8; dir++) {  
            for (int length=1; length<=n; length++) {  
                int value = computeValue(x, y, dir, length);  
                if (value != -1 && isPrime(value))  
                    System.out.println(value);  
            }  
        }  
    }  
}
```



시작위치 (x,y) 에서 dir 방향으로 길이가 $length$ 인 수열을 정수로 환산하여 반환하라. 만약 그런 수열이 존재하지 않으면 -1을 반환하라.

수열을 정수로 환산하기

6	6	3	1
2	8	0	5
1	6	5	4
5	0	7	3

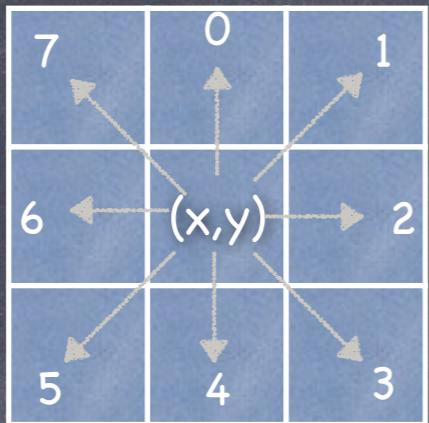
$$5 = 5$$

$$5 * 10 + 6 = 56$$

$$56 * 10 + 0 = 560$$

$$560 * 10 + 1 = 5601$$

int computeValue(int x, int y, int dir, int len)



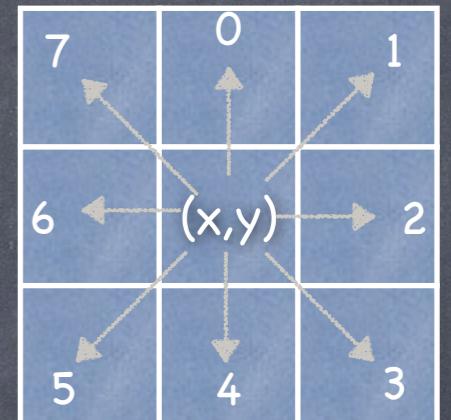
int computeValue(int x, int y, int dir, int len)
위치 (x,y)에서 dir 방향으로 연속된 len개의 digit로 만 들어지는 정수를 반환. 그런 값이 없으면 -1을 반환

```
public static int computeValue(int x, int y, int dir, int len) {  
    int value=0;  
    for (int i=0; i<len; i++) {  
        int digit = getDigit(x, y, dir, i);  
        if (digit== -1)  
            return -1;  
        value = value*10+digit;  
    }  
    return value;  
}
```

시작위치 (x,y)에서 dir 방향으로 i칸 떨어진 자리에 있는 digit를 반환한다. 만약 그런 자리가 존재하지 않으면 -1을 반환 하라.

int getDigit(int x, int y, int dir, int k)

```
public static int getDigit(int x, int y, int dir, int k) {  
    int newX = x, newY = y;  
    switch (dir) {  
        case 0: newY -= k; break;  
        case 1: newX += k; newY -= k; break;  
        case 2: newX += k; break;  
        case 3: newX += k; newY += k; break;  
        case 4: newY += k; break;  
        case 5: newX -= k; newY += k; break;  
        case 6: newX -= k; break;  
        case 7: newX -= k; newY -= k; break;  
    }  
    if (newX < 0 || newX >= n || newY < 0 || newY >= n)  
        return -1;  
    return grid[newX][newY];  
}
```



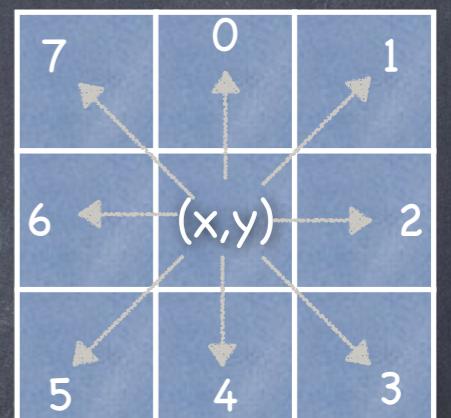
좀더 영악한 getDigit(int x, int y, int dir, int k)

- offsetX[dir]과 offsetY[dir]은 임의의 위치에서 dir방향으로 한 칸 움직였을 때 좌표값의 증감분

```
int [] offsetX = {0, 1, 1, 1, 0, -1, -1, -1};  
int [] offsetY = {-1, -1, 0, 1, 1, 1, 0, -1};
```

- 따라서 (x,y) 에서 dir방향으로 k칸 떨어진 위치의 좌표는

```
(x+k*offsetX[dir], y+k*offsetY[dir])
```



Code21.java

```
class Code21 {  
    static int n;  
    static int [][] grid;  
  
    public static void main(String[] args) {  
        try {  
            Scanner inFile = new Scanner(new File("data.txt"));  
            n = inFile.nextInt();  
            grid = new int [n][n];  
            for ( int i=0; i<n; i++)  
                for (int j=0; j<n; j++)  
                    grid[i][j] = inFile.nextInt();  
            inFile.close();  
        }  
    }  
}
```

입력은 **data.txt** 파일로 부터 읽는다.

data.txt 파일의 예

```
4  
6 6 3 1  
2 8 0 5  
1 6 5 4  
5 0 7 3
```

Code21.java (continued)

```
for (int x=0; x<n; x++) {  
    for (int y=0; y<n; y++) {  
        for (int dir=0; dir<8; dir++) {  
            for (int len=1; len<=n; len++) {  
                int val = computeValue(x, y, dir, len);  
                if (val != -1 && isPrime(val))  
                    System.out.println(val);  
            }  
        }  
    }  
}  
} catch (FileNotFoundException e) {  
    e.printStackTrace();  
}  
}
```

이것이 이 문제에 대한 최선의 방법은 아니다. 조금만 생각해 보면 계산에 있어서 수많은 중복(redundancy)이 존재함을 알수 있다. 하지만 논리적으로 가장 간명한 방법임은 틀림없다. 이렇게 가장 간명한 방법을 먼저 생각하고, 거기에 어떤 계산의 중복이 있는지 파악한 후, 그것을 제거하는 방향으로 생각해보는 것은 하나의 훌륭한 전략이다.

Code21.java (continued)

```
static int [] offsetX = {0, 1, 1, 1, 0, -1, -1, -1};  
static int [] offsetY = {-1, -1, 0, 1, 1, 1, 0, -1};  
  
public static int getDigit(int x, int y, int dir, int k )  
{  
    int newX = x + k*offsetX[dir];  
    int newY = y + k*offsetY[dir];  
    if ( newX<0 || newX>=grid.length || newY<0 || newY>=grid.length )  
        return -1;  
    else  
        return grid[newX][newY];  
}  
  
public static int computeValue(int x, int y, int dir, int len) {  
    // at page 50  
}  
  
public boolean isPrime(int k) {  
    ...  
}  
}    // end of class Code21
```

Task Set #2

13. 데이터 파일 `input.txt`에 사람들의 이름과 나이가 아래 그림과 같은 형식으로 저장되어 있다. 단 이름은 하나의 영문 단어라고 가정한다. 이 데이터 파일을 읽은 후 사람들을 나이순으로 정렬하여 출력하는 프로그램을 작성하라. 단 나이가 같은 경우에는 이름의 알파벳 순으로 정렬되어야 한다.

John 24
James 24
Tom 32
Jackson 37
Park 21
Lee 21

`input.txt`의 예

Lee 21
Park 21
James 24
John 24
Tom 32
Jackson 37

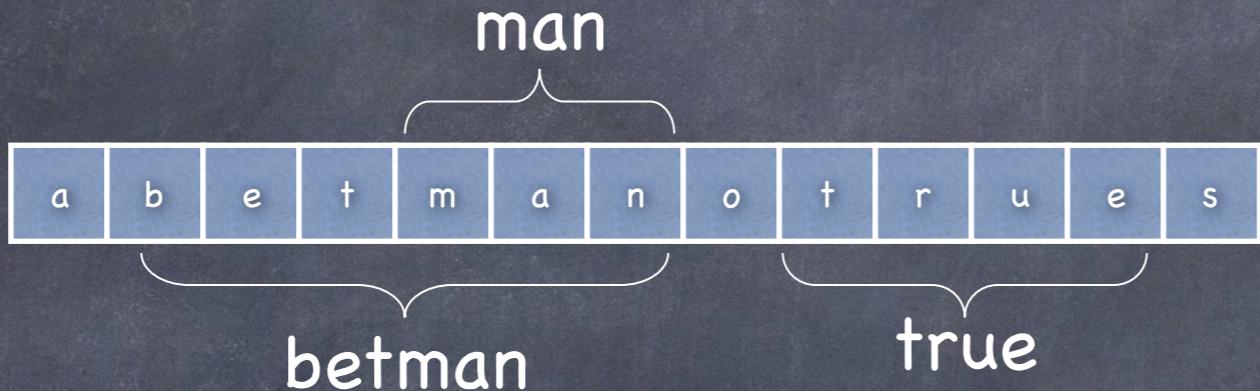
출력 예

14. 입력으로 정수의 개수 n 과 이어서 n 개의 정수들이 주어진다. 이 중 오름차순 혹은 내림차순으로 정렬되어 있는 가장 긴 구간을 찾아서 그 구간의 길이를 출력하는 프로그램을 작성하라. 예를 들어 $n=13$ 이고 입력 정수들이 2, 3, 15, 15, 13, 12, 11, 9, 0, -1, 2, -3, 5 라면 밑줄친 구간이 가장 긴 구간이고 그 길이는 8이다.

15. 서로 다른 수들로 구성된 수열 a_1, a_2, \dots, a_n 이 있다. 어떤 수가 자신과 인접한 두 수 (즉, 바로 전의 수와 바로 다음의 수)보다 작을 때 그 수를 **local minimum**이라고 부른다. 단 a_1 의 경우에는 a_2 보다 작으면, a_n 의 경우에는 a_{n-1} 보다 작으면 **local minimum**이다. 어떤 수열에서 **local minimum**이 최대 1개일 때 그 수열은 “**오목하다**”라고 부르기로 하자. 입력으로 하나의 수열이 주어질 때 가장 길이가 긴 오목한 부분 수열을 찾아서 출력하는 프로그램을 작성하라. 예를 들어 입력 수열이 1, 3, 7, 6, 12, 11, 9, 0, -1, 2, -3, 5 라면 가장 긴 오목한 부분 수열은 12, 11, 9, 0, -1, 2이다. 입력 형식은 먼저 n 의 값이 주어지고 이어서 n 개의 정수가 주어진다.

Task Set #2

16. 파일 `input.txt`에 여러 개의 단어들이 한 줄에 하나씩 주어진다. 이 파일을 읽어서 배열에 저장한다. 키보드로부터 문자의 개수 n 과 이어서 n 개의 문자들을 입력받아 문자 배열에 저장한다. 이 배열에서 파일에 등장하는 모든 단어들을 찾아 출력하는 프로그램을 작성하라.



and
betman
mad
man
true
...

`input.txt`의 예

17. 아래 그림과 같이 0 혹은 1로 채워진 $n \times n$ 그리드가 주어진다. “1만으로 채워진” 가장 큰 정사각형을 찾아 그 면적을 출력하는 프로그램을 작성하라. 입력 형식은 먼저 n 의 값이 주어지고 이어서 $n \times n$ 개의 0 혹은 1의 값이 한 칸씩 띄워져서 한 줄에 n 개씩 주어진다.

0	1	1	0	0	1
1	0	1	1	1	0
0	1	1	1	1	1
1	0	1	1	1	1
0	1	1	0	0	0
1	1	0	1	1	1

1만으로 채워진 가장 큰 정사각
형. 면적은 9이다.

Task Set #2

18. *Code21.java*의 문제에서 대각방향은 제외하고 가로, 세로로만 인접한 수로 구성되는 모든 소수를 찾아 출력하는 프로그램을 작성하라. 단, 각 행이나 열의 끝과 시작이 연결되어 있어서 원형이라고 가정한다. 가령 옆의 그림에서 511도 찾아야 한다. 입력은 *input18.txt* 파일에서 받아들이고 입력 파일의 첫 줄에 그리드의 크기 n 이 주어진다.

6	6	3	1
1	1	0	5
1	6	5	4
5	0	7	3

19. 입력파일 *input19.txt*에 먼저 정수의 개수 n 과 이어서 n 개의 정수가 주어진다. 이 정수들을 입력된 순서대로 아래의 왼쪽 그림과 같이 $n \times n$ 그리드의 대각선에 저장한다. 아래의 그림은 $n=4$ 이고 입력 정수가 6, 1, 5, 3인 경우의 예이다. 이제 그리드의 대각선 위쪽의 칸들을 모두 채운다. 규칙은 $grid[i][j]$ 에는 $grid[i+1][j] - grid[i][j-1]$ 의 절대값을 저장한다. 즉 자신의 “바로 왼쪽 값과 바로 아래쪽 값의 차이의 절대값”이다. 이런 식으로 대각선 위쪽의 모든 칸을 채우고, 대각선의 아래쪽은 그냥 0으로 채워서 화면으로 출력하는 프로그램을 작성하라.

A 4x4 grid with axes labeled 'x 증가' (increasing x) pointing down and 'y 증가' (increasing y) pointing right. The grid contains the following values:

6			
	1		
		5	
			3

A 4x4 grid showing the state after filling the diagonal and the row above it. The top row contains 6, 5, 1, 4 and the bottom row contains 0, 1, 4, 2. All other cells are empty.

6	5	1	4
	1	4	
		5	2
			3

The final 4x4 grid with all values filled according to the rules. The top row is 6, 5, 1, 1. The second row is 0, 1, 4, 2. The third and fourth rows are both 0, 0, 5, 2 and 0, 0, 0, 3 respectively.

6	5	1	1
0	1	4	2
0	0	5	2
0	0	0	3

1.3 문자열 (String)과 리스트 다루기

인덱스 메이커

- 입력으로 하나의 텍스트 파일을 읽는다 (`sample.txt`).
- 텍스트 파일에 등장하는 모든 단어들의 목록을 만들고, 각 단어가 텍스트 파일에 등장하는 횟수를 센다. 단, 단어 개수는 100,000개 이하라고 가정한다.
- 사용자가 요청하면 단어 목록을 하나의 파일로 저장한다.
- 사용자가 단어를 검색하면 그 단어가 텍스트 파일에 몇 번 등장하는지 출력한다.

실행 예

```
$ read sample.txt          // 텍스트 파일 sample.txt를 읽고 인덱스를 만든다.  
$ find heaven              // heaven이라는 단어가 몇번 나오는지 출력한다.  
The word "heaven" appears 13 times.  
$ saveas index.txt         // 인덱스를 index.txt라는 파일로 저장한다.  
$ find java                 
The word "java" does not appear.  
$ exit
```

Code22.java

자료구조부터 만들어 보자. 대부분의 프로그램에서 제일 먼저 생각해야 할 것은 데이터를 저장할 자료구조를 정의하는 것이다.

```
class Code22 {
```

```
    static String [] words = new String [100000];
```

```
    static int [] count = new int [100000];
```

```
    static int n;
```

단어들의 목록을 저장할 배열

각 단어의 등장 횟수를 저장

저장된 단어의 개수

Code22.java (Continued)

```
public static void main(String[] args) {
    Scanner kb = new Scanner(System.in);
    while(true) { _____
        System.out.print("$ ");
        String command = kb.next();
        if (command.equals("read")) {
            String fileName = kb.next();
            makeIndex(fileName);
        }
        else if (command.equals("find")) {
            String keyword = kb.next();
            int index = findWord(keyword);
            if (index != -1)
                System.out.println("The word " + keyword + " appears " + count[index] + " times.");
            else
                System.out.println("The word " + keyword + " doesn't appear.");
        }
    }
}
```

while문을 한번 돌때마다 사용자로부터 하나의 명령을 받아 수행한다. 언제 **while**문을 종료할지가 미리 정해져 있지 않을 경우 이렇게 무한 루프를 만들고, 종료할 조건이 만족되면 (이 예에서는 사용자가 **exit** 명령을 입력하면) **break**문으로 루프를 빠져나오는 것은 하나의 전형적인 스타일이다.

findWord는 단어 **keyword**를 배열 **words**에서 검색하여 찾으면 배열 **index**를, 못 찾으면 -1을 반환한다.

Code22.java (Continued)

```
else if (command.equals("saveas")) {  
    String fileName = kb.next();  
    saveAs(fileName);  
}  
else if (command.equals("exit"))  
    break;  
}  
kb.close();  
}
```

Code22.java (Continued)

```
private static void makeIndex(String fileName) {
    try {
        Scanner theFile = new Scanner(new File(fileName));
        while(theFile.hasNext()) {
            String word = theFile.next();
            addWord(word);
        }
        theFile.close();
    } catch (FileNotFoundException e) {
        System.out.println("File not exist !");
    }
}
```

Code22.java (Continued)

```
private static void addWord(String word) {  
    int index = findWord(word);  
    if (index > -1) {  
        count[index]++;  
    }  
    else {  
        words[n] = word;  
        count[n] = 1;  
        n++;  
    }  
}
```

```
private static int findWord(String keyword) {  
    for (int i=0; i<n; i++)  
        if (words[i].equals(keyword))  
            return i;  
    return -1;  
}
```

이렇게 데이터들을 순차적으로 검색하는 것을 순차검색(sequential search) 알고리즘이라고 부른다.

Code22.java (Continued)

```
private static void saveAs(String fileName) {  
    try {  
        PrintWriter out = new PrintWriter(new FileWriter(fileName));  
        for (int i=0; i<n; i++) {  
            out.println(words[i] + " " + count[i]);  
        }  
        out.close();  
    } catch (IOException e) {  
        System.out.println("Save failed. Don't ask me why!");  
    }  
} // end of class Code22
```

문제점

- 소수점, 쉼표 등의 특수기호가 단어에 포함된다.
- 숫자 등이 단어로 취급된다.
- 대문자와 소문자가 다른 단어로 취급된다.
- 단어들이 알파벳 순으로 정렬되면 좋겠다.

String 클래스 기본 메서드

문자열 동일성 검사	boolean equals(String)	String str1 = "java"; String str2 = "Java"; boolean equals = str1.equals(str2);
문자열 사전식 순서	int compareTo(String)	String str1 = "absolute"; String str2 = "base"; int result = str1.compareTo(str2);
문자열 길이	int length()	String str = "abcdef"; int length = str.length();
특정 위치의 문자	char charAt(int)	String str = "ABCDEFG"; char ch = str.charAt(2);
지정한 문자의 위치 검색	int indexOf(char)	String str = "abcdef"; int index = str.indexOf("d");
지정된 범위의 부분 문자열	String substring(int, int)	String str = "ABCDEF"; String substr = str.substring(0, 2);

단어의 앞뒤에 붙은 특수문자 제거하기

```
public static String trimming(String str)
{
    if (str==null || str.length()<=0 )
        return null;
    int i = 0, j = str.length()-1;
    while (i<str.length() && !Character.isLetter(str.charAt(i)))
        i++;
    while (j>=0 && !Character.isLetter(str.charAt(j)))
        j--;
    if (i<=j)
        return str.substring(i, j+1);
    else
        return null;
}
```

“-?*sdh-fg+=(
 ↑
 i=4 ↑
 j=9

Code23.java

```
private static void makeIndex(String fileName) {
    try {
        Scanner theFile = new Scanner(new File(fileName));
        while(theFile.hasNext()) {
            String word = theFile.next();
            String trimmed = trimming(word);
            if (trimmed != null)
                addWord(trimmed.toLowerCase());
        }
        theFile.close();
    } catch (FileNotFoundException e) {
        System.out.println("File not exist !");
    }
}
```

자료구조, main, save 메서드는 Code22와 동일하다. 메서드 trimming을 추가하고, makeIndex, addWord, findWord 메서드를 수정한다.

Code23.java (Continued)

```
private static void addWord(String word) {  
    int index = findWord(word);  
    if (index > -1) {  
        count[index]++;  
    }  
    else {  
        int i=n-1;  
        for (; i>=0 && words[i].compareToIgnoreCase(word)>0; i--) {  
            words[i+1] = words[i];  
            count[i+1] = count[i];  
        }  
        words[i+1] = word;  
        count[i+1] = 1;  
        n++;  
    }  
}
```

단어들을 알파벳순으로 정렬하기 위해서는 일단 모든 단어들을 읽어서 인덱스를 만든 후에 한번에 정렬할 수도 있고, 항상 정렬된 상태를 유지하도록 삽입하는 방법이 있다. 여기서는 후자의 방법으로 해본다.

Task Set #3

20. *Code23*은 단어의 앞뒤에 붙은 특수문자들을 제거하지만 중간에 등장하는 특수문자들은 제거하지 못한다. *Code23*에서 먼저 단어의 앞뒤에 붙은 특수문자와 숫자들을 제거한 후 남은 단어에 대해서 중간에 영어 알파벳, hyphen (-), 그리고 underscore(_) 이외의 다른 문자가 포함되면 인덱스에 포함하지 말고 버리도록 프로그램을 수정하라. 또한 길이가 2 이하인 단어들도 모두 버린다.
21. 20번 문제에서 작성한 코드에 *search*명령을 추가한다. *search* 명령을 실행하면 사용자가 지정한 단어를 접두어로 포함하는 모든 단어와 등장 횟수를 찾아 출력한다. 예를 들어 어떤 단어가 *the*를 접두어로 가지고 있는지 알려면 일단 그 단어는 길이가 3 이상이어야 하고, 그 단어의 앞 3글짜를 빼어내면 *the*가 되어야 한다. 어떤 단어의 일부분만 빼어내기 위해서는 *substring* 메서드를 사용한다.

```
$ search the  
their 3  
them 5  
there 2
```

22. 어떤 단어가 단어 내의 문자들의 순서만 바꾸어서 다른 단어와 동일해질 수 있을 때 그 단어는 다른 단어의 *anagram*이라고 부른다. 예를 들어 *orchestra*는 *carthorse*의 *anagram*이다. 입력으로 키보드로 부터 두 단어를 받아서 서로 *anagram* 관계인지 검사하는 프로그램을 작성하라. 다음과 같이 작동해야 한다.

```
$ test orchestra carthorse  
yes  
$ test power person  
no  
$ exit
```

Task Set #3

23. *Code20*에서 사용자로 부터 이름과 전화번호 쌍들을 입력받아 저장하고, 이름의 알파벳 순으로 정렬하여 출력하는 일을 하였다. 이 프로그램을 다음과 같이 수정하라. 우선 입력을 키보드가 아니라 `input.txt`라는 이름의 파일로 부터 받는다. 사람 수는 최대 1000명이라고 가정한다. *Code20*에서 사람 이름은 항상 한 단어라고 가정하였다. 이름이 여러 단어로 구성될 수 있도록 수정한다. 이때 이름과 전화번호를 구분하기 위한 구분자로 ‘|’ 문자를 사용한다. 아래는 입력 파일의 예이다. 입력 파일에는 단어의 앞뒤로 불필요한 공백이 있을 수 있다. 출력에서는 불필요한 공백들을 모두 제거해야 한다.

입력 파일의 예: `directory.txt`

```
Hong Gildong | 010-7624-4867  
Lee Sang Ho | 051-475-3845  
David Lee | 32875628234  
Junho Kim | 2346923479  
Park Hyung-Woo | 02-4574-4857
```

출력 예: 화면으로 출력

```
David Lee | 32875628234  
Hong Gildong | 010-7624-4867  
Junho Kim | 2346923479  
Lee Sang Ho | 051-475-3845  
Park Hyung-Woo | 02-4574-4857
```