HW1 Supplement

How to check the input

JAVA

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.regex.*;

public class Main {

    public static void main(String[] args) throws IOException {

        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        String input = bf.readLine();

        //input의 3번째 위치 혹은 4번째 위치에 "-" string이 들어올 때 까지 입력받기
        while(input.indexOf("-") != 2 && input.indexOf("-") != 3) {

            System.out.println("input again");
            input = bf.readLine();
        }

        //input string을 "-" string 을 기준으로 나누었을 때 생기는 split 배열의 길이 (배열의 갯수)가 3개일 때 까지 입력받기
        while(input.split( regex: "-").length != 3) {

            System.out.println("input again");
            input = bf.readLine();
        }

        //Pattern & Matcher - Regex를 사용하여 형식이 맞는지 판별하기 - 3가지 형식
        Pattern p = Pattern.compile("^\\d{2,3}-\\d{4}-\\d{4}");
        Matcher m = p.matcher( input: "010-0000-0000");
        boolean b = m.matches();
        System.out.println(b);

        boolean b2 = Pattern.compile("[a-zA-Z]+ [a-zA-Z]+").matcher( input: "asdfsa asdfasf").matches();
        System.out.println(b2);

        boolean b3 = Pattern.matches( regex: "\\d+",  input: "50293759823755");
        System.out.println(b3);
    }
}
```

Result:
true
true
true

Try & Catch

```cpp
#include <iostream>

using namespace std;

int main() {

    int a;

    try{
        cin >> a;
        if(a != 1) {

            throw 1;
        }
        if(a != 2) {

            throw 2;
        }
    }
    catch(int i) {

        cout << "error code : " << i << endl;
    }
    return 0;
}
```

1. When input = 2
- Result :
- “error code : 1”

2. When input = 1
- Result :
- “error code : 2”

3. When input = 3
- Result :
- “error code : 1”

(throws the first throw not the second one)

cpp

```cpp
#include <iostream>
#include <cstring>
#include <regex>

using namespace std;

int main() {

    int a;
    double b;
    char c;
    string d;

    try {
        cin >> d;

        if(d.length() != 12) {  //If the length of string d is not 5, throw 1
            throw 1;
        }
        if(d.find("-") != 2) {  //If the position of first "-"string of string d is not 2 (on third position)
            throw 2;
        }
        //If the position of first "-"string of substring of d starting from third is not 4
        // if d => "asdf-asdf-asdf"
        // d.substr(3) => "f-asdf-asdf"
        if(d.substr(3).find("-") != 4) {
            throw 3;
        }
```

```
        //c++ regex
        //use as below :
        // ^ : the start position of match string (not necessary when matching for multiple cases)
        // $ : the end position of match string (not necessary for the same reason)
        // \\d : any digit
        regex matchregex("^\\d\\d-\\d\\d\\d\\d-\\d\\d\\d\\d$");
        if(regex_match(d, matchregex))
        {
            //empty
        }
        else{
            throw 4;
        }
        // ( ) : grouping - the same concept with original parenthesis
        // ( a | b ) : a or b both is OKAY
        // \\d{4} : any four consecutive digits ex) 0187
        // (-\\d{4}){2} : format like  -1048-1984
        regex regex2("^(02|010)(-\\d{4}){2}");
        if(regex_match(d, regex2)) {}
        else {
            throw 5;
        }
        // [ ] : any character that match in the []  ex) [a-z] -> a, b, c all OKAY
        // a-z : every character okay between a ~ z
        // a-zA-Z : every character okay between a ~ z and A ~ Z
        // a-zA-Z0-9 : ... between a~z and A~Z and 0~9  ex) a OK A OK 0 OK
        // . : any character
        // + : 1 or more of characters   ex) \\d+  -> 010 OK  019589 OK 918501957195 OK  (empty) NOK
        // * : 0 or more of characters   ex) \\d*  -> (empty) OK 0 OK 00000 OK
        regex regex3("^[a-zA-Z]+_[a-zA-Z]+");
        if(regex_match(d, regex3)) {}
        else {
            throw 6;
        }

    }
    catch(int i) {

        cout << i << endl;
    }
    return 0;
}
```

Useful Sites

- https://regex101.com/
  - Regex tester site
- https://hamait.tistory.com/342
  - Regex grammar (in Korean)
- http://www.cplusplus.com/reference/regex/ECMAScript/
  - Regex grammar (in English)

March 30th, constraints changed.

1. print() – declare as virtual in Person class

* virtual 키워드

: virtual 키워드는 부모 클래스에서 함수를 가상으로 정의해 놓고, 실제로 구현은 자식 클래스에서 하고자 하는 경우에 사용하는 것으로 이해하면 편리하다. 이는 다형성을 이용하는데 있어서 반드시 활용하면 아주 유용하게 사용할 수 있을 것이다. virtual 키워드를 사용하기 위해서는 앞선 예에서 그냥 car.h에 Car 클래스 선언에서 beep 함수 앞에 virtual을 아래와 같이 넣으면 된다.

```
/* car.h */
#ifndef CAR_H
#define CAR_H

class Car {
    protected:
        int wheels;
        int price;
    public:
        Car();
        Car(int, int);
        ~Car();

        void setWheels(int);
        void setPrice(int);
        int getWheels(void);
        int getPrice(void);

        virtual void beep(void);

        friend class Engineer;
};
#endif
```

constraints changed

- Int phonebook -> string phonebook