*Exam0110
*Exam0120
*java.util.Date 클래스
new java.sql.Date(밀리초) => 날짜 정보를 다루는 물건을 생성
설계도에 따라 만드는 방법

java.sql.Date.valueof(날짜문자열) => 날짜 정보를 다루는 물건을 생성
날짜 문자열의 형식은 반드시 yyyy-mm-dd 여야 한다.
자판기처럼 바로 만드는 방법

두 방법 모두 알고 있어야함
첫 번째 방법은 밀리초를 주고 두 번째 방법은 날짜 문자열을 입력해야한다.

*java.lang.Integer 클래스
Scanner를 통해 값을 뽑는 방법
int no = Scanner.nextInt();
리턴 값은 int 데이터형이다.

만약에 Scanner.nextLine(); 은 100 이라는 문자열을 리턴한다.
문자열을 다시 바꾸는 방법은 Integer.parseInt() 안에 문자열 화 된 값을 집어넣으면 int 데이터로 변환하여 리턴한다.

*배열
배열을 쓰는 이유는 몰까ㅇㅅㅇ..?
여러 명의 회원정보를 입력받아서 출력할 경우
같은 종류의 같은 값을 여러번 입력 받아야한다.
ex) 이름, 성적 정보
그럴 때, 배열이 없는 상태에서 처리한다면
5명의 회원 정보를 입력하려면

```
System.out.println("[회원]");
    int no1, no2, no3, no4, no5;
    String name1, name2, name3, name4, name5;
    String email1, email2, email3, email4, email5;
```

String password1, password2, password3, password4, password5;
String photo1, photo2, photo3, photo4, photo5;
String tel1, tel2, tel3, tel4, tel5;
Date registeredDate1,registeredDate2, registeredDate3, registeredDate4, registeredDate5;

ㄴ 이 문자열과 Scanne 을 이용한 입력을 5번씩 전부 수기로 입력해야한다.
숫자가 더 커질 경우 불편하기 때문에 배열을 사용한다.

즉, 배열은 같은 종류의 메모리를 쉽게 만드는 방법.

- 문법
메모리종류[] 메모리이름 = new 메모리종류[개수]
데이터타입[] 변수명 = new 데이터타입[개수];
ex) int[] arr = new int[5];

- C언어 스타일
데이터타입 변수명[] = new 데이터타입[개수];
ex) int arr[] = new int[5];

배열의 개수는 int 타입의 최대 값과 같다.
즉 2147483647 개 이다.

배열 선언문 분석:
arr1
- 배열 메모리의 주소를 담는 변수이다.
- 이렇게 메모리의 주소를 보관하는 변수를 '레퍼런스'라 부른다.

new int[5]
- new 명령은 사용할 메모리를 확보하는 명령이다.
- 즉 연속된 5 개의 int 타입 메모리를 준비하라는 명령이다.
- 사용할 메모리를 확보한 후 그 메모리의 찾아 갈 수 있도록 시작 주소를 리턴한다.
- 이렇게 값을 저장하기 위해 확보된 메모리를 "인스턴스"라 부른다.

new 명령은 메모리를 확보하는 명령이다.
- 리턴 값은 확보된 메모리의 시작 주소이다.

배열 레퍼런스만 따로 선언할 수 있다.
int[] arr1;

이 때 아직 레퍼런스는 배열의 메모리를 알지 못하기 때문에 사용할 수 없다.
arr1[0] = 100; // 컴파일 오류!

* 배열 초기화
일반 변수와 달리 new 명령으로 확보된 메모리는 종류에 상관없이 기본 값으로 자동 초기화 된다.
따라서 배열 메모리 또한 생성되는 순간 기본 값으로 자동 초기화 된다.
- 정수 배열(byte[], short[], int[], long[]) : 0
- 부동소수점 배열(float[], double[]) : 0.0
- 논리 배열(boolean[]) : false
- 문자 배열(char[]) : '\u0000'
- 주소 변수(Object[]) : null

*배열 선언과 초기화
1) 배열 선언 + 초기화
데이터타입[] 변수명 = new 데이터타입[]{값, 값, 값};
- 배열 메모리를 초기화시킬 때는 배열 개수를 지정해서는 안된다.
- 배열을 초기화시키는 값의 개수 만큼 메모리가 만들어진다.
- 즉 다음은 값 개수만큼 int 메모리가 3개가 생성된다.
ex) int[] arr = new int[]{10, 20, 30};

다음과 같이 new 명령을 생략할 수 있다.
데이터타입[] 변수명 = {값, 값, 값};
ex) int[] arr = {10, 20, 30};

2) 배열 선언 후 따로 배열 초기화 문장 실행
데이터타입[] 변수명;
변수명 = new 데이터타입[]{값, 값, 값};
ex)
int[] arr1;

arr1 = new int[]{10, 20, 30};
-변수를 선언한 후 따로 배열을 초기화시킬 때는 new 명령을 생략할 수 없다.

*인스턴스와 가비지 메모리
한 번 생성한 레퍼런스에 다른 배열의 인스턴스를 넣는 것이 가능하다.
기존에 레퍼런스가 보관하고 있던 주소는 잃어버렸기 때문에 이전 배열 인스턴스는 사용할 수 없다.
이렇게 주소를 잃어버려 사용할 수 없는 인스턴스(메모리)를 '가비지(garbage)'라 부른다.

int[] arr1;
arr1 = new int [5];
arr1[0] =100;

이렇게 한 번 생성한 레퍼런스에

arr1=new int[]{200,200,200}
다른 배열의 인스턴스를 넣으면
앞에서 입력한 arr1[0]=100;의 값은 잃어버리고 그 사용할 수 없는 인스턴스(100) 값은 가비지가 된다.

*가비지(garbage)
주소를 잃어 버려 사용할 수 없는 메모리
특정 조건이 되면 가비지 수집기(garbage collector)에 의해 메모리 해제된다.
메모리 해제? 다른 용도로 사용할 수 있도록 표시한다는 의미다.

가비지 수집 조건 = 가비지 컬렉터가 동작할 때
 메모리가 부족할 때
운영체제로부터 메모리를 추가로 받기 전에 먼저 기존에 사용하던 메모리에서 가비지를 제거한다.
CPU가 한가할 때
24시간 365일 내내 실행하는 서버 프로그램인 경우, 실행 중간에 CPU가 한가할 때 가비지를 제거한다.

프로그램(JVM)을 종료하면 JVM 사용한 메모리를 운영체제가 모두 회수한다.

*가비지를 강제로 지우도록 명령하는 방법
자바는 없다.

C언어 => free(메모리주소);

C++ 언어 => delete 객체주소;
요즘 언어의 트랜드는 사용하지 않는 메모리를 개발자가 직접 해제하는 것이 아니라 VM이 해제하는 것이다.
예) JavaScript, C#, Python, PHP, Go, Java 등 요즘 언어의 트랜드는 VM으로 실행하는 것이다.
왜? 직접 기계어로 전환되면 메모리를 관리를 자동으로 수행할 수 없다.

*가비지 컬렉터를 강제로 실행하는 방법
없다
단 원래 계획보다 가능한 빨리 실행하라고 독촉하는 방법은 있다.

System.gc() 메서드 호출

그런데 바로 실행할 지 나중에 실행할 지 그 시점을 보장하지는 않는다.

*반복문
count = count +1;

앞 count가 메모리라면 뒤의 count는 메모리에 들어있는 값이다
연산문을 읽을 때 뒤에서 앞으로 읽어야한다.
0+1을 앞에 있는 count에 넣는다.

while문 시작할 때 기본 변수 값은 0으로 준다
```
int count=0;
while(count < 5) {
sysout("Hello");
count = count + 1;
}
```

* 조회용으로만 쓰는 상수는 보통 대문자로 적는다.
상수는 언더바로 구분한다. ex) final int MAX_LENGTH =5;

*while 반복문을 출력할 때 출력을 멈추더라도 MAXLENGTH만큼 값을 출력한다.
때문에 size라는 변수를 while 문에 추가하여 입력한 정보 만큼의 양만 출력되도록 한다.
count변수가 아닌 size 변수를 준다.

```
for (int i =0; i < MAX_LENGTH; i = i +1 ) {
    size = size +1;

    System.out.print("번호? ");
    no[i] = Integer.parseInt(keyboardScan.nextLine());
    System.out.print("이름? ");
    name[i] = keyboardScan.nextLine();
    System.out.print("이메일? ");
    email[i] = keyboardScan.nextLine();
    System.out.print("암호? ");
    password[i] = keyboardScan.nextLine();
    System.out.print("사진? ");
    photo[i] = keyboardScan.nextLine();
    System.out.print("전화? ");
    tel[i] = keyboardScan.nextLine();
    registeredDate[i] = new Date();
    System.out.println();

    System.out.print("계속 입력하시겠습니까 (y/N)");
    String input = keyboardScan.nextLine();
    if (input.equals("N")) {
      break;
    }
```

이 때 키보드 커맨드는 보통 기본 값을 대문자로 저장한다
(y/N) = N이 기본값이라는 뜻.
때문에 if (input.equals("N") || input.equals("")) {
    break;
}

라고 해야한다.

여기서 input.equals("N")에서 input.equalsIgnoreCase("N") 라고 고치면 소문자 n도 입력 가능. 대소문자 구분 하지 않는 매서드

*while과 for와 if
while(조건) {
조건이 참인 동안 반복
}

for (변수 선언; 조건; 증감){
조건이 참인 동안 반복
}

if (조건) 문장;

*연산자 (operator) 와 피연산자(operand)
i =i +1;

i,i,1 - 피연산자
=,+ - 연산자