7
자료구조: 배열

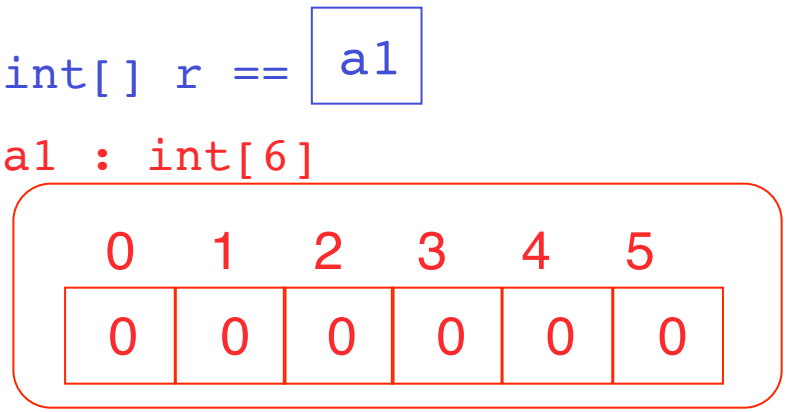
배열이 필요한 이유

- 6명의 학생의 점수를 받아서 각 학생에게 최고점과 얼마나 차이가 나는지 알려주려고 한다.
  - int score1, score2, score3, score4, score5, score6;
  - int high_score = score1;
  - if(score2 > high_score) high_score = score2;
  - ...
  - if(score6 > high_score) high_score = score6;

- 이렇게 하면 좋겠는데
  - for(i=2; i<=6; i++)
    { if(score $_i$ > high_score) high_score = score $_i$; }

배열

- 정의
  - 동일한 타입의 자료(원소)를 고정된 개수만큼 갖고 있는 자료 구조
- 타입에 [ ]를 붙이면 배열 타입이 된다.
  - `int[]`, `String[]`
- Java의 배열은 객체이다.
  - new로 생성한다.
  - `int[] r = new int[6];`



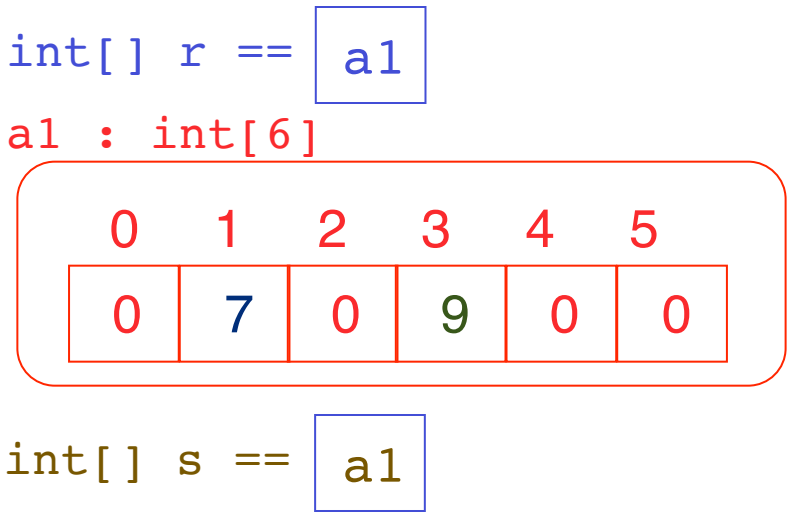
배열 다루기

```
int[] r = new int[6];
int x = 6;

r[1] = 7;
r[3] = r[x-5] + 2;
int[] s = r;
```

인덱스(index)는 임의의 정수 계산식 가능

같은 배열을 가리킴



배열의 초기화

- 배열의 원소들은 생성될 때 초기화된다.
  - 필드 변수 초기화 되는 것과 동일
    - int: 0, double: 0.0, boolean: false, 객체타입: null
- 생성 및 사용자 초기화
  - `int[] r = {1, 2, 4, 8, 16, 32};`

루프를 사용하여 초기화하기

```
int[] r = new int[12];

r[0] = 1; r[1] = 1;
for (int i=2; i<r.length; i=i+1) {
  r[i] = r[i-1] + r[i-2];
}

vs

int[] r = { 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 };
```

배열의 크기는 length 필드에 저장됨

메소드에 인수로 전달, 결과로 반환

- 배열의 내용이 뒤집힌 배열을 반환하는 메소드를 작성하여라.

```
int[] reverse(int[] r) {
   int size = r.length;
   int[] answer = new int[size];
   for(i=0; i<size; i++) {
      answer[size-1-i] = r[i];
   }
   return answer;
}
```

배열이 복사되어 전달되는 것이 아니라 주소가 전달되는 것이다.

배열이 복사되어 반환되는 것이 아니라 주소가 반환되는 것이다.

예제, 단순치환 암호

- 공백과 알파벳 소문자로 구성된 문자열을 변환표에 따라 암호화하고 복호화하는 프로그램을 작성해라.
  - 편의상 공백=0, 'a'=1, ..., 'z'=26 코드를 사용하겠다.

- 변환표는 다음과 같이 만든다.
  - code[0] = seed
  - code[i] = (code[i-1]+4) % 27

명세: class TranslateTable

생성자	
`TranslateTable(int seed)`	seed를 사용하여 암호화 테이블, 복호화 테이블 구축
속성	
`int[] encode`	암호화 테이블
`int[] decode`	복호화 테이블
메소드	
`encode(char c): char` `decode(char c): char`	문자를 암호화/복호화
`encode(String s): String` `decode(String s): String`	문자열을 암호화/복호화

변환표 생성

```
class TranslateTable {

    private int[] encode;
    private int[] decode;

    public TranslateTable(int seed) {
        encode = new int[27];
        decode = new int[27];
        encode[0] = seed;
      decode[seed] = 0;
        for(int i=1; i<27; i++) {
            int new_code = (encode[i-1]+4) % 27;
            encode[i] = new_code;
            decode[new_code] = i;
        }
    }
```

문자 코드 간 변환

```
private int c2i(char c) {
   if(c==' ') return 0;
   if('a'<=c && c<='z') return c-'a'+1;
   throw new RuntimeException("c2i: invalid character " + c);
}

private char i2c(int i) {
   if(i==0) return ' ';
   if(0<i && i<27) return (char)('a'+i-1);
   throw new RuntimeException("i2c: invalid code " + i);
}
```

암호화

```
    public char encode(char c) {
       return i2c(encode[c2i(c)]);
    }

    public String encode(String s) {
       String answer = "";
    for(int i=0; i<s.length(); i++)
           answer = answer + encode(s.charAt(i));
       return answer;
    }

  // 복호화도 유사하게 작성할 수 있다.
}
```

구동 클래스

```
import javax.swing.*;

class TranslateString {
    public static void main(String[] args) {
       TranslateTable m = new TranslateTable(1);
       String original = JOptionPane.showInputDialog
       ("암호화할 문장을 입력하세요.");
       String encoded = m.encode(original);
       String decoded = m.decode(encoded);
       JOptionPane.showMessageDialog(null,
       "원본: " + original + "\n암호화: " + encoded +
       "\n복호화: " + decoded);
    }
}
```

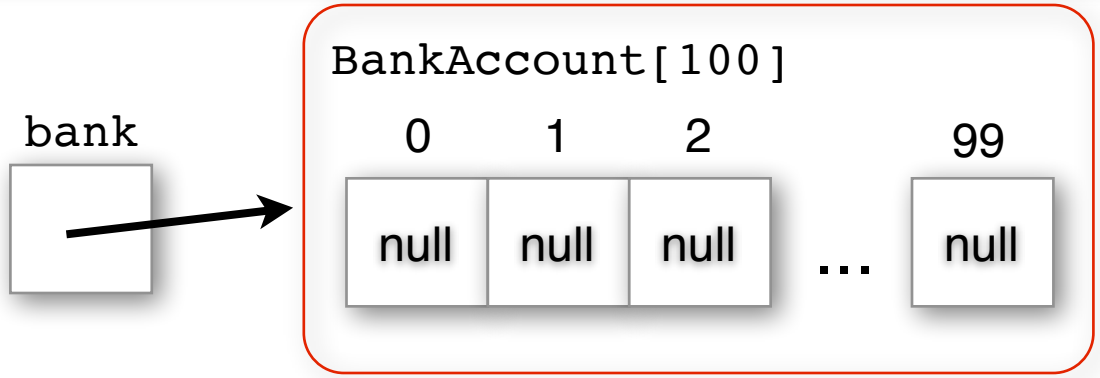
예제, 계좌 관리 프로그램

- 6장에서 다루었던 계좌 관리 프로그램을 두 계좌가 아닌 100 계좌를 관리할 수 있도록 하자.

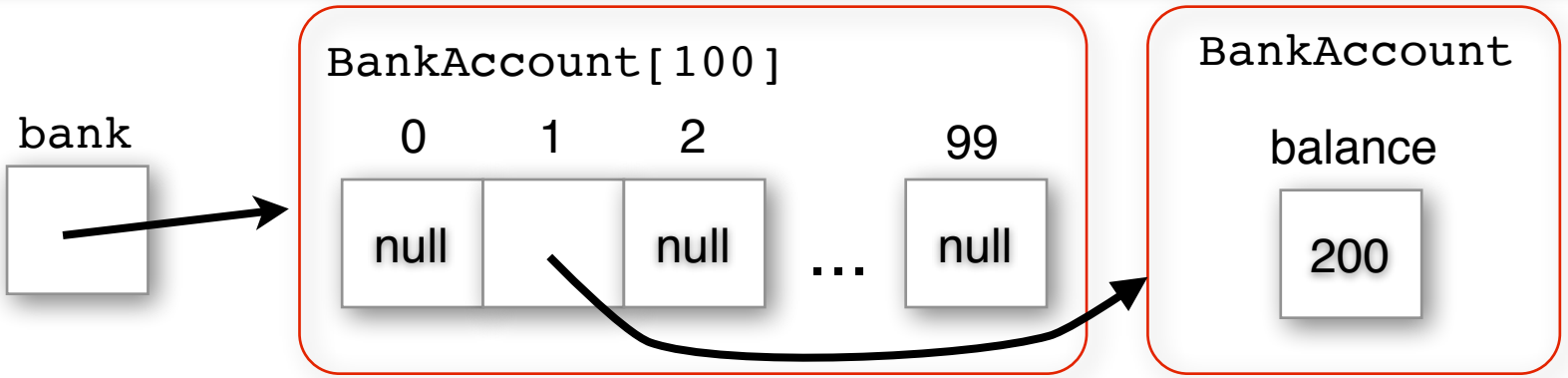
- 객체타입 배열 가능
  - `BankAccount[] bank = new BankAccount[100];`
  - 100개의 계좌가 만들어진 것이 아니라, 100개의 계좌를 저장할 수 있는 배열이 생성된 것.
  - `bank[75] = new BankAccount(200);`
  - 이제 2.00$가 들어 있는 계좌가 생성된 것이다.

계좌 배열

```
BankAccount[] bank = new BankAccount[100];
```




```
bank[1] = new BankAccount(200);
```



객체 배열 다루기

- bank[1].deposit(600);
  - bank[1] 객체에 deposit 메시지를 보낸다.
- bank[1] = null;
  - bank[1] 계좌를 없앤다.
- 객체 배열의 속성 상 모든 원소가 다 있어야 하는 것은 아님

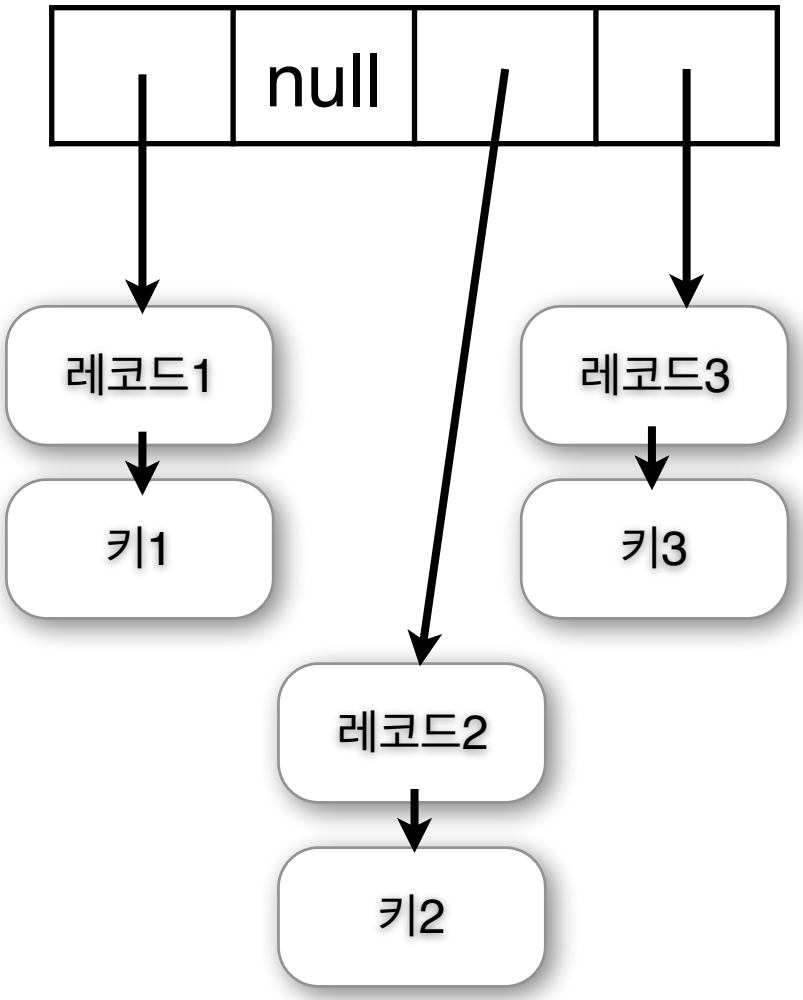
설계 예제, 데이터베이스

- 데이터베이스 (database)
  - 정보를 많이 모아둔 것
  - 예, 도서관의 소장도서 정보, 학교의 학생정보, 회사의 매출 정보 등

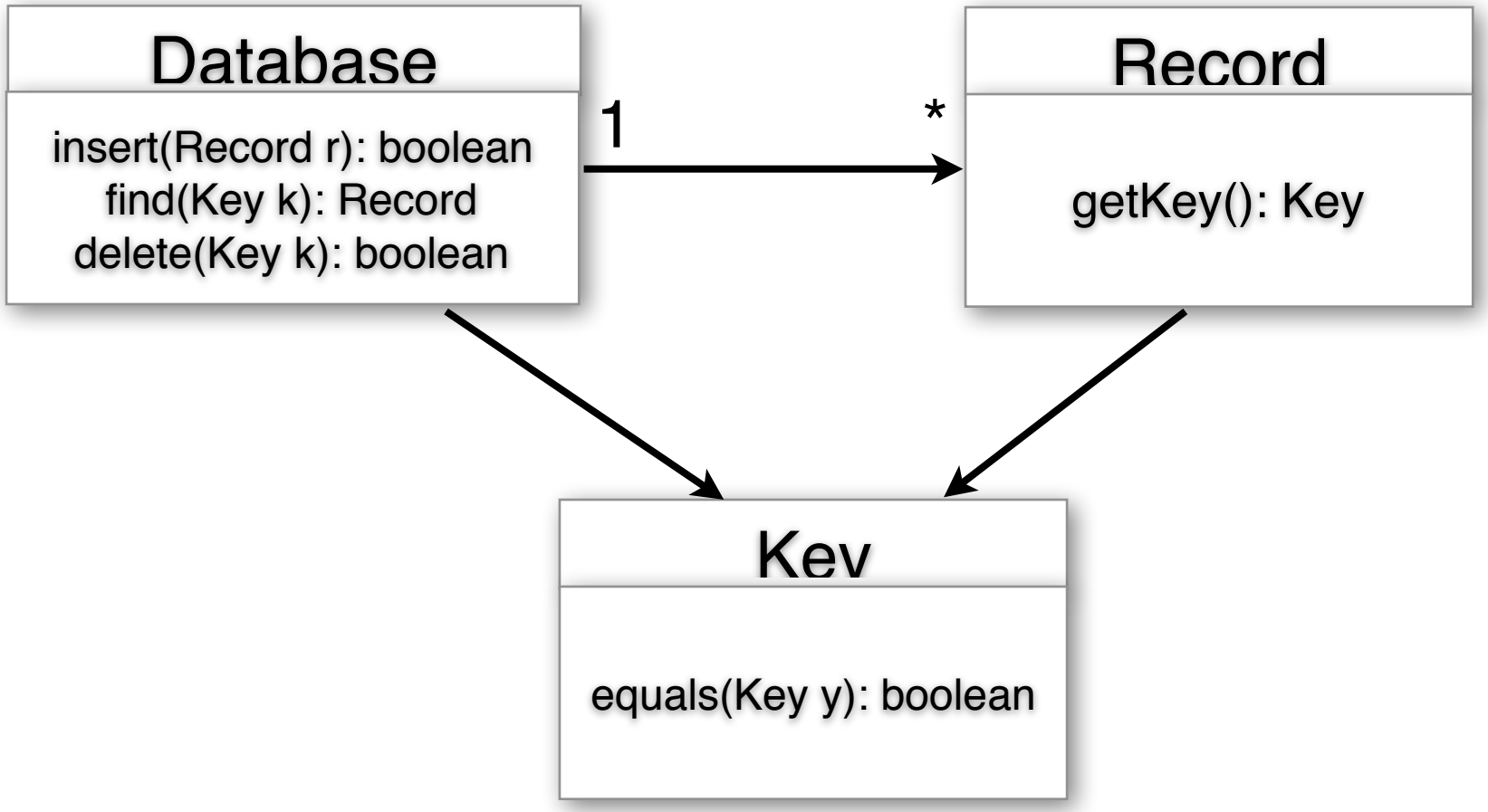
- 레코드 (record)
  - 데이터베이스에서 저장하는 정보의 한 단위
  - 키(key)를 통해 레코드들을 구별한다.

Java에서 단순 데이터베이스 만들기

- 키는 객체다.
- 레코드는 객체다. 키 객체를 갖고 있어야 한다.
- 데이터베이스는 레코드의 "배열"로 볼 수 있다. 레코드를 넣고, 찾고, 없앨 수 있어야 한다.
  - insert: 레코드를 데이터베이스에 추가
  - find: 키를 통해 레코드를 검색
  - delete: 키를 통해 레코드를 검색하여 삭제



소프트웨어 구조



클래스 다이어그램 간선 종류에 대한 자세한 정보: https://en.wikipedia.org/wiki/Class_diagram

명세

`class Database`	레코드를 저장하는 컨테이너
메소드	
`insert(Recrd r): boolean`	r을 데이터베이스에 추가한다. 성공하면 true, 아니면 false를 반환한다.
`find(Key k): Record`	k 키를 가지는 레코드를 찾는다. 실패하면 null을 반환한다.
`delete(Key k): boolean`	k 키를 가지는 레코드를 삭제한다. 성공하면 true, 실패하면 false를 반환한다.

`class Record`	데이터베이스의 자료 단위
메소드	
`keyOf(): Key`	레코드의 키를 반환한다.

`class Key`	레코드의 식별자, 키
메소드	
`equals(Key m): boolean`	자기와 m을 비교한다. 같으면 true, 틀리면 false를 반환한다

필드, 생성 메소드

```
public class Database {

  private Record[] base;
    private int NOT_FOUND = -1;

  public Database (int initial_size) {
       if (initial_size <= 0)
          initial_size = 1;
       base = new Record[initial_size];
    }
```

위치 찾기 메소드

```
private int findLocation(Key k)
{
    for (int i=0; i<base.length; i++)
        if(base[i]!=null && base[i].getKey().equals(k))
            return i;
    return NOT_FOUND;
}
private int findEmpty()
{
    for (int i=0; i<base.length; i++)
        if(base[i]==null)
            return i;
    return NOT_FOUND;
}
```

find & delete

```
public Record find(Key k) {
    int index = findLocation(k);
    if(index != NOT_FOUND)
        return base[index];
    else
        return null;
}
public boolean delete(Key k) {
    int index = findLocation(k);
    if(index != NOT_FOUND) {
        base[index] = null;
    return true;
  }
  else
        return false;
}
```

insert

```
public boolean insert(Record r) {
    if(findLocation(r.getKey()) != NOT_FOUND)
        return false;
    int index = findEmpty();
    if(index != NOT_FOUND)
        base[index] = r;
    else {
        Record[] temp = new Record[base.length * 2];
        for(int i=0; i<base.length; i++)
            temp[i] = base[i];
        temp[base.length] = r;
        base = temp;
    }
    return true;
}}
```

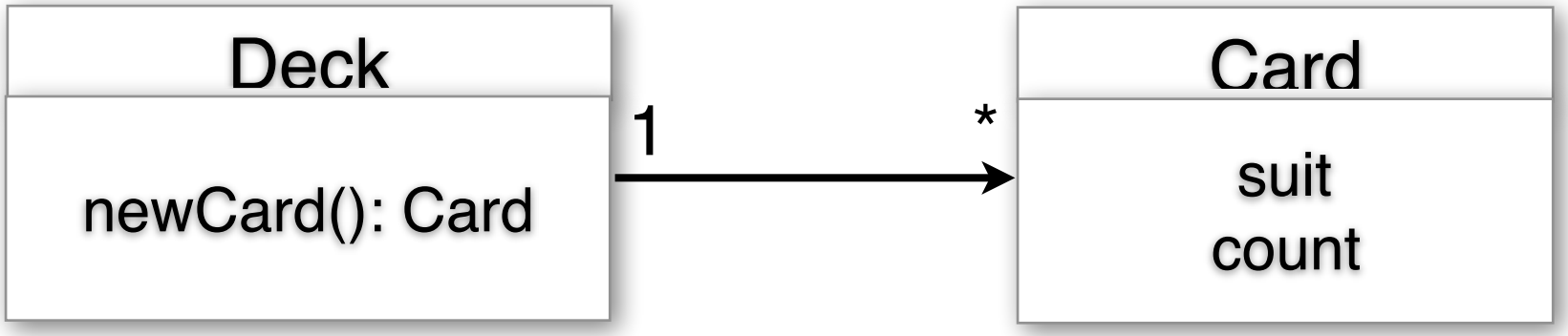
설계 예제, 카드 게임

- 카드 통에서 카드를 한 장씩 주는 프로그램을 작성해 보자.

- 카드
  - 모양(suit): 다이아몬드(diamonds), 하트(hearts), 클로버(clubs), 스페이드 (spades)
  - 숫자: A(1), 2~10, 잭(11), 여왕(12), 왕(13)

- 카드 통 (deck)
  - 카드의 배열

소프트웨어 구조



명세

class CardDeck	카드 통
속성	
private Card[] deck	남은 카드를 갖고 있다.
메소드	
newCard(): Card	카드 한 장을 준다. 통이 비었을 때는 null을 반환한다.
moreCards(): boolean	남은 카드가 있는지 반환한다.

class Card	카드
속성	
private suit: String	모양
private int count;	숫자
메소드	
getSuit(): String	모양 반환
getCount(): int	숫자 반환

카드

```
public class Card {
    public static String SPADES = "spades";
    public static String HEARTS = "hearts";
    public static String DIAMONDS = "diamonds";
    public static String CLUBS = "clubs";

    public static int ACE = 1;
    public static int JACK = 11;
    public static int QUEEN = 12;
    public static int KING = 13;
    public static int SIZE_OF_ONE_SUIT = 13;

    private String suit;
    private int count;

    public Card(String s, int c)
        { suit = s; count = c; }
    public String getSuit() { return suit; }
    public int getCount() { return count; }
}
```

카드 통

```
public class CardDeck {
    private int card_count; // 남은 카드 수
    private Card[] deck = new Card[4*Card.SIZE_OF_ONE_SUIT];
    // 불변식: deck[0]...deck[card_count-1]에는 카드가 있다.

  private void createSuit(String which_suit) {
    for(int i=1; i<=Card.SIZE_OF_ONE_SUIT; i++) {
            deck[card_count] = new Card(which_suit, i);
            card_count++;
        }
    }
    public CardDeck() {
        createSuit(Card.SPADES); createSuit(Card.HEARTS);
        createSuit(Card.CLUBS); createSuit(Card.DIAMONDS);
    }
```

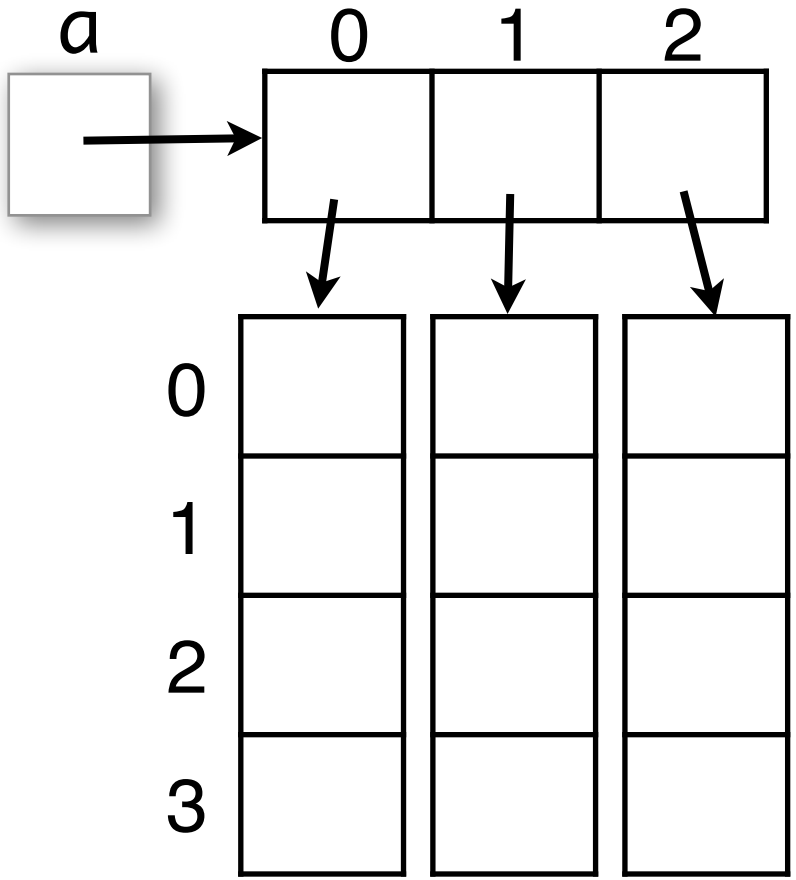
카드 통에서 카드 꺼내 주기

```
    public Card newCard() {
        Card next_card = null;
        if(card_count != 0 ) {
            int index = (int)(Math.random() * card_count);
            next_card = deck[index];
      // 카드를 뽑은 위치부터 앞으로 당겨 준다.
            for(int i=index+1; i<card_count; i++)
                deck[i-1] = deck[i];
            card_count--;
        }
        return next_card;
    }

    public boolean moreCards() { return card_count > 0; }
}
```

2차원 배열

- 정의: 배열을 원소로 하는 배열
- 문법
  - 생성: `int[][] a = new int[3][4];`
  - 열(column)의 수: `a.length`
  - 행(row)의 수: `a[0].length`



0,0	1,0	2,0
0,1	1,1	2,1
0,2	1,2	2,2
0,3	1,3	2,3

Java에서는 배열의 배열로 구현된다.

예, 대선 통계


```
int[ ][ ] election = new int[3][4];

for(int j=0; j<4; j++) {
    int votes = 0;
    for(int i=0; i<3; i++)
        votes = votes + election[i][j];
    System.out.println
       ("기호" + (j+1) + "번은 " +
        votes + "표 받았습니다.");
}

for(int  i=0; i<3; i++) {
    int votes = 0;
    for(int j=0; j<4; j++)
        votes = votes + election[i][j];
    System.out.println
       ((i+1) + " 지역은 " + votes +
        "표 행사했습니다.");
}
```

들쭉날쭉 배열 (Ragged Array)

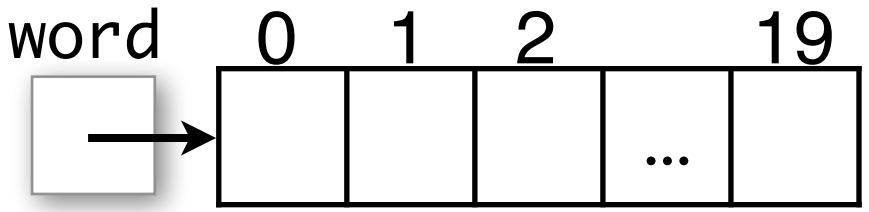
- 이차원 배열이 배열의 배열이기 때문에, 각 원소 배열의 크기를 다르게 할 수 있다.

이차원 배열의 첫 크기만 지정하면 원소 배열이 생성되지 않는다.

```
int max_words = 20;
char[][] word = new char[max_words][];
int count = 0;
boolean processing = true;
while (processing && count < max_words) {
    String s = JOptionPane.showInputDialog("Please type a word: ");
    if (s == null) processing = false;
    else {
        word[count] = new char[s.length()];
        for (int i=0; i<s.length(); i++)
            word[count][i] = s.charAt(i);
        count++;
    }
}
```

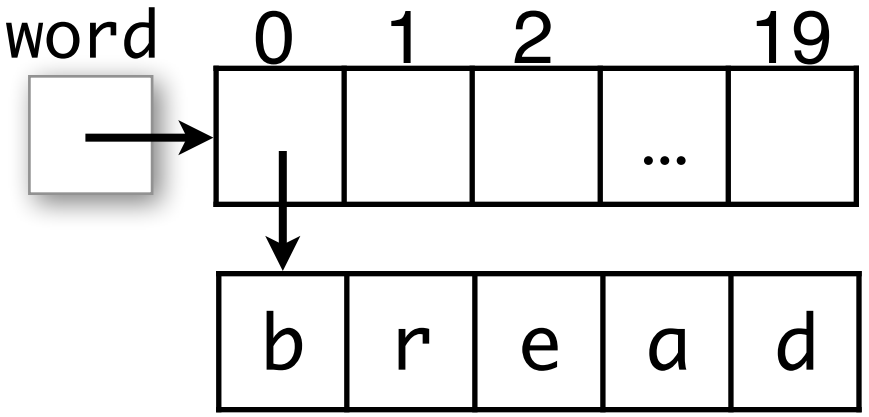
실행

```
char[][] word = new char[max_words][];
```



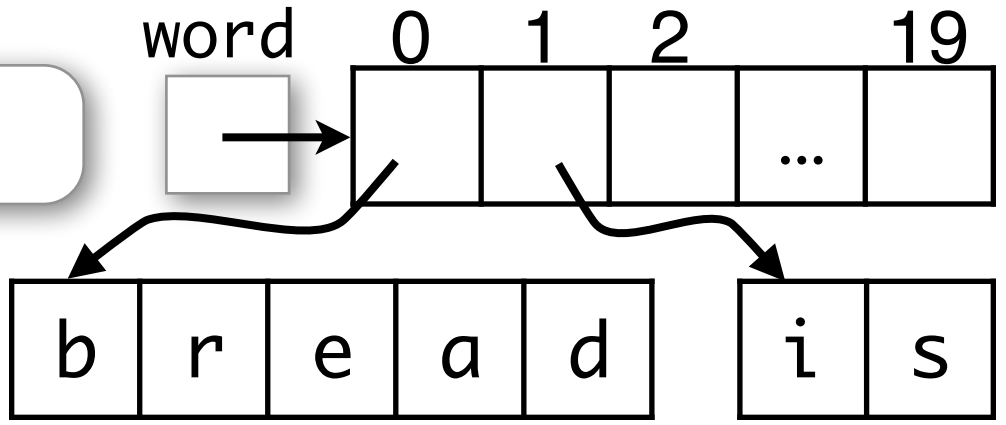

```
word[count] = new char[s.length()];
```

s="bread"

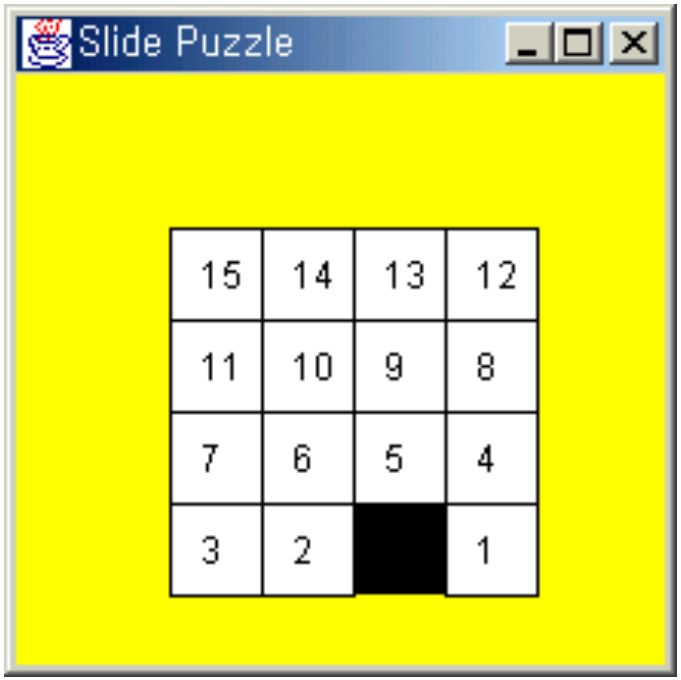


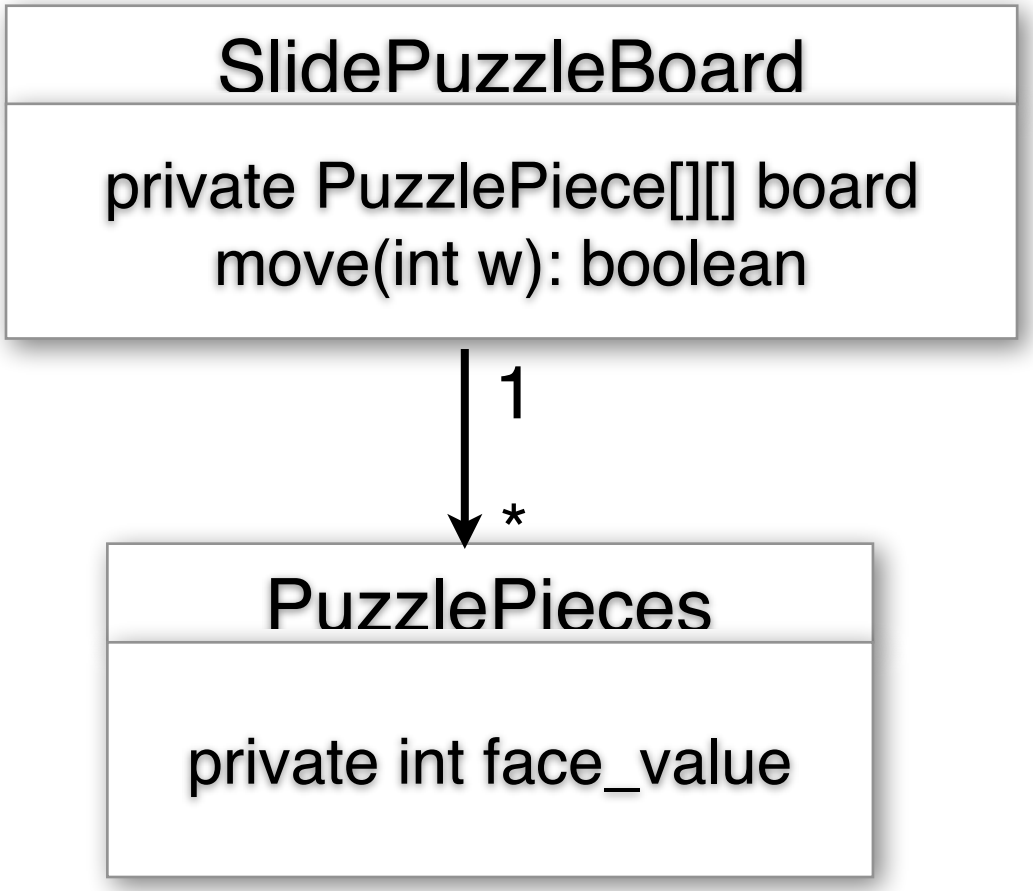

```
word[count] = new char[s.length()];
```

s="is"



예제, 퍼즐 게임


Slide Puzzle
15 14 13 12
11 10 9 8
7 6 5 4
3 2 1


SlidePuzzleBoard
private PuzzlePiece[][] board
move(int w): boolean
1
*
PuzzlePieces
private int face_value

퍼즐 조각

```
public class PuzzlePiece {
    private int face_value;
    public PuzzlePiece(int value) { face_value = value; }
    public int valueOf() { return face_value; }
}
```

퍼즐 판

```
public class SlidePuzzleBoard {
    private PuzzlePiece[][] board;
    int empty_row, empty_col, size;
     public SlidePuzzleBoard(int s) {
        size = s;  empty_row = 0;  empty_col = 0;
        board = new PuzzlePiece[s][s];
        for (int i = 0; i < size; i++ )
            for (int j = 0; j < size ; j++)
                board[i][j] = new PuzzlePiece(i * size + j);
        board[empty_row][empty_col] = null;
    }
    public boolean move(int w) {
        int row = 0; int col = 0;
        for (int i = 0 ; i < size; i++)
            for (int j = 0 ; j < size; j++)
                if (board[i][j] != null && board[i][j].valueOf() == w)
                    { row = i; col = j; }
        if (Math.abs(empty_row - row) == 1 || Math.abs(empty_col - col) == 1) {
            board[empty_row][empty_col] = board[row][col];
            board[row][col] = null; empty_row = row; empty_col = col;
            return true;
        }
        return false;    }}
```