Lecture4: Data Model Design 2

김강희
khkim@ssu.ac.kr

목 차

- 이론:
  - main() 위주로 작성된 코드를 Tetris 클래스로 구성
  - 상태 기계 (Finite State Machine)
  - enum 타입

- 실습:
  - v1: Tetris 클래스 작성
  - v2: Tetris 클래스 검증 (복수 객체)
  - v3: Tetris 클래스의 accept() 인자를 하나로 통합
  - v4: Tetris 클래스에 enum 타입 적용

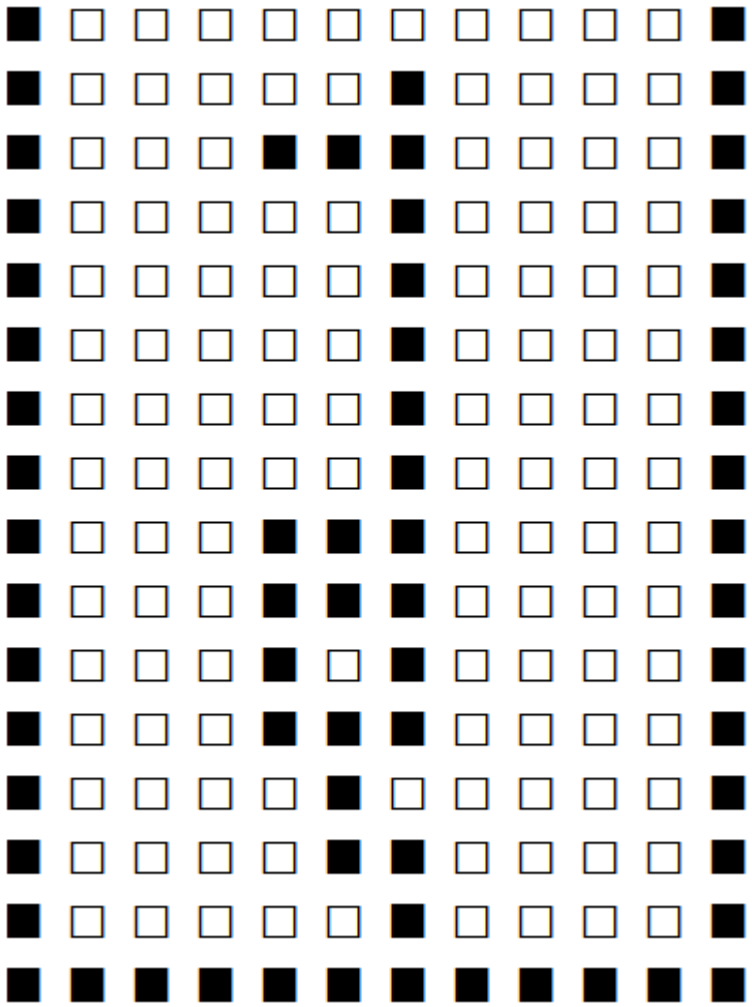
이번 Lecture에서 알아야 할 사항들

- 클래스 작성
  - Static member와 dynamic member의 구분
  - Static member 초기화와 dynamic member 초기화(생성자)
  - Private member와 public member의 구분
  - 객체의 동작을 "상태 기계"로 이해
  - Hardcoded constant 제거 (enum 타입 활용)
  - 디버깅에 유리한 legacy code 존속

Main.java (v1)

```
1  import java.io.BufferedReader;
2  import java.io.IOException;
3  import java.io.InputStreamReader;
4  import java.util.Random;
5
6  public class     Main {
7      private static int[][][][] setOfBlockArrays = { // [7][4][?][?]
163    private static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
164    private static String line = null;
165    private static int nKeys = 0;
166    private static char getKey() throws IOException {
181    public static void main(String[] args) throws Exception {
182        char key;
183        int idxType;
184        boolean newBlockNeeded;
185        Tetris.init(setOfBlockArrays);      // Tetris class의 static 변수들 초기화
186        Tetris board = new Tetris(15, 10); // board 객체의 dynamic 변수들 초기화
187        Random random = new Random();
188        idxType = random.nextInt(7);
189        board.accept('0', idxType);         // 새로운 idxType 값이 전달될 때 '0' 인자는 역할이 없는 인자임
190        board.printScreen(); System.out.println();// Tetris class의 printScreen()에서 oScreen 인자는 내부
191                                                  에서 접근하기로 함
192        while ((key = getKey()) != 'q') { // while 루프 구조는 Tetris class의 accept()에서 제거해야 함
193            newBlockNeeded = board.accept(key, idxType);
194            board.printScreen(); System.out.println();
195            if (newBlockNeeded) {
196                idxType = random.nextInt(7);
197                newBlockNeeded = board.accept('0', idxType); // 새로운 idxType 값이 전달될 때 '0' 인자는
198                board.printScreen(); System.out.println();    역할이 없는 인자임
199                if (newBlockNeeded) break; // Game Over!
200            }
201        }
202        System.out.println("Program terminated!");
203    }
204 }
```

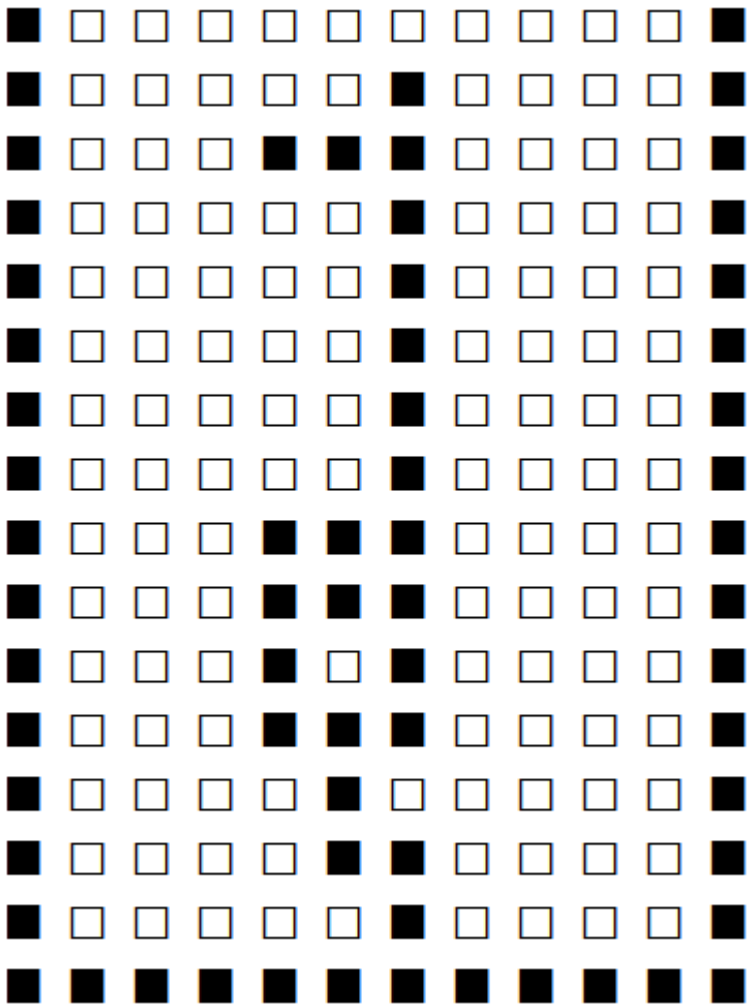
Main.java (v1)

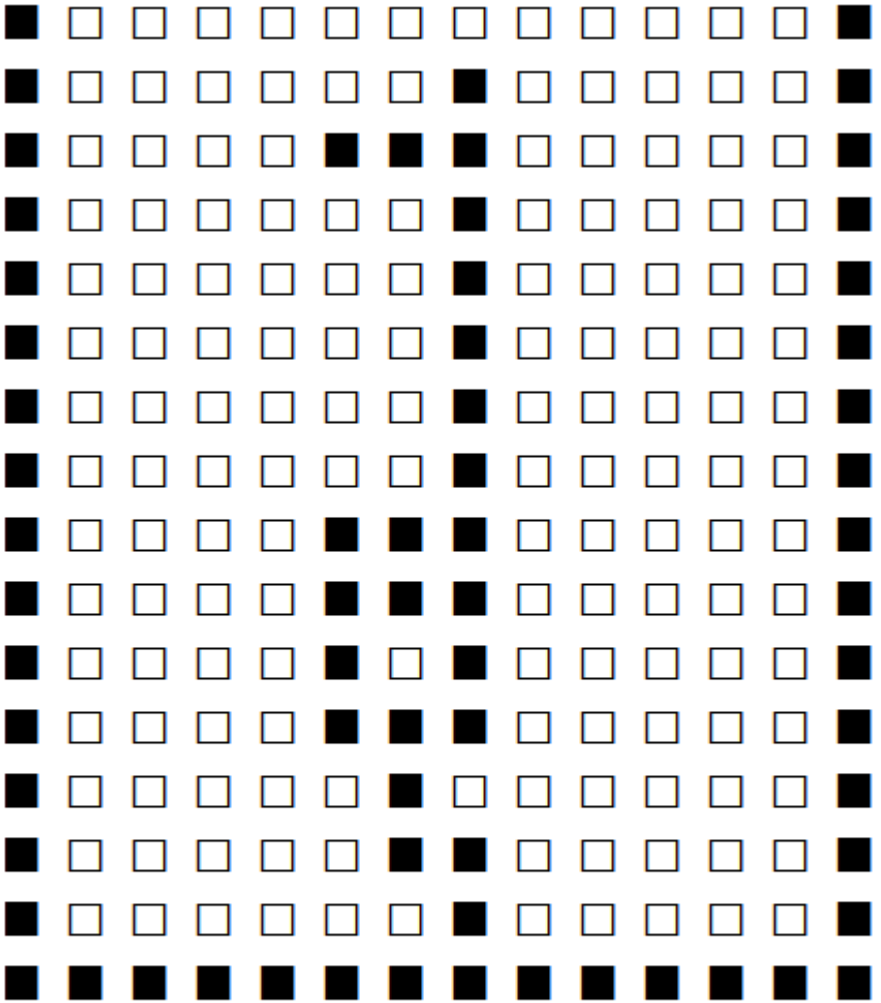


Main.java (v2)

```
  6 public class     Main {
  7     private static int[][][][] setOfBlockArrays = { // [7][4][?][?]
163     private static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
164     private static String line = null;
165     private static int nKeys = 0;
166     private static char getKey() throws IOException {
181     public static void main(String[] args) throws Exception {
182         char key;
183         int idxType;
184         boolean newBlockNeeded;
185         Tetris.init(setOfBlockArrays);
186         Tetris board = new Tetris(15, 10);  // board를 15x10 배열로 생성
187         Tetris board2 = new Tetris(15, 12); // board2를 15x12 배열로 생성
188         Random random = new Random();
189         idxType = random.nextInt(7);
190         board.accept('0', idxType);
191         board2.accept('0', idxType);  // board와 board2에 동일한 idxType 전달
192         board.printScreen(); System.out.println();
193         board2.printScreen(); System.out.println();
194
195         while ((key = getKey()) != 'q') {
196             newBlockNeeded = board.accept(key, idxType);
197             board2.accept(key, idxType);     // board와 board2에 동일한 key 전달
198             board.printScreen(); System.out.println();
199             board2.printScreen(); System.out.println();
200             if (newBlockNeeded) {
201                 idxType = random.nextInt(7);
202                 newBlockNeeded = board.accept('0', idxType);
203                 board2.accept('0', idxType); // board와 board2에 동일한 idxType 전달
204                 board.printScreen(); System.out.println();
205                 board2.printScreen(); System.out.println();
206                 if (newBlockNeeded) break; // Game Over!
207             }
208         }
209         System.out.println("Program terminated!");
210     }
211 }
```

Main.java (v2)





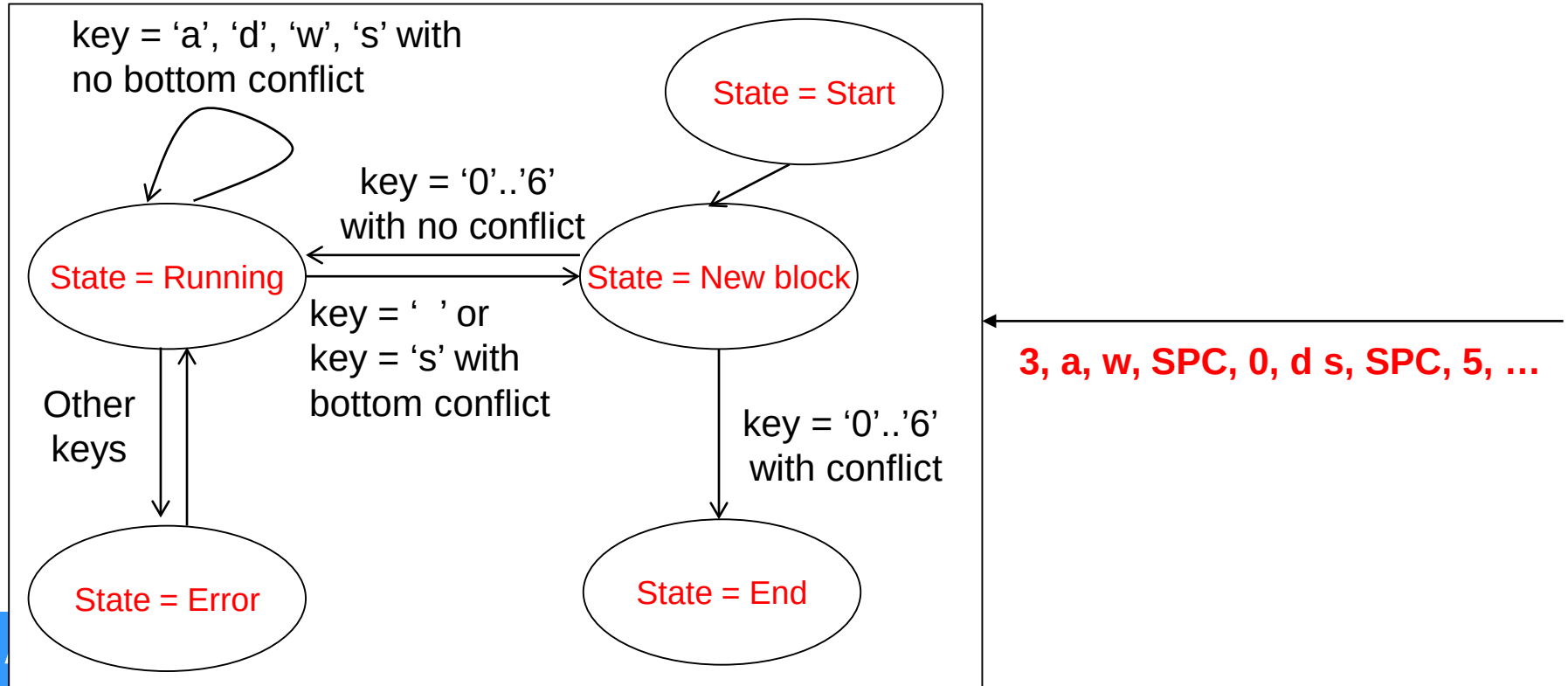
```
Program terminated!
```

상태 기계

- 상태 기계
  - 유한 상태 기계(finite-state machine, FSM) 또는 유한 오토마톤(finite automaton, 복수형: 유한 오토마타 finite automata)이라고 번역함
  - 컴퓨터 프로그램과 전자 논리 회로를 설계하는데 쓰이는 수학적 모델로서 간단히 상태 기계라고 부르기도 함
  - 유한 상태 기계는 유한한 개수의 상태를 가질 수 있는 오토마타, 즉 추상 기계라고 할 수 있음
  - 한 번에 오로지 하나의 상태만을 가지게 됨
  - 현재 상태(current state)란 현재 시간의 상태를 지칭함
  - 어떠한 사건(event)에 의해 한 상태에서 다른 상태로 변화할 수 있으며, 이를 전이(transition)이라 함
  - 유한 상태 기계는 현재 상태로부터 가능한 전이 상태와 이러한 전이를 유발하는 조건들의 집합으로서 정의됨

상태 기계

- 테트리스 게임의 상태 기계 모델
  - key 값과 idxType 값의 나열을 하나의 입력 시퀀스(input sequence)로 이해하고 Tetris 상태 기계는 Start, Running, New Block, End, Error 상태를 가진다고 이해할 수 있음
  - 동일한 입력 시퀀스에 대해서 상태 기계는 항상 동일한 상태를 가짐
  - Tetris 상태 기계의 입력을 하나의 변수 타입으로 통일하면 Tetris class 코드 상에 상태 기계 관점을 더 잘 표현할 수 있음



Main.java (v3)

```
1  import java.io.BufferedReader;
2  import java.io.IOException;
3  import java.io.InputStreamReader;
4  import java.util.Random;
5
6  public class     Main {
7      private static int[][][][] setOfBlockArrays = { // [7][4][?][?]
163    private static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
164    private static String line = null;
165    private static int nKeys = 0;
166    private static char getKey() throws IOException {
181    public static void main(String[] args) throws Exception {
182        char key;
183        boolean newBlockNeeded;
184        Tetris.init(setOfBlockArrays);
185        Tetris board = new Tetris(15, 10);
186        Random random = new Random();
187        key = (char) ('0' + random.nextInt(7));// idxType 정수값(0~6)을 ASCII 문자 코드('0'~'6')으로 변환
188        board.accept(key);
189        board.printScreen(); System.out.println();
190
191        while ((key = getKey()) != 'q') {
192            newBlockNeeded = board.accept(key);
193            board.printScreen(); System.out.println();
194            if (newBlockNeeded) {
195                key = (char) ('0' + random.nextInt(7));// idxType 정수값(0~6)을 ASCII 문자 코드
196                newBlockNeeded = board.accept(key);     ('0'~'6')으로 변환
197                board.printScreen(); System.out.println();
198                if (newBlockNeeded) break; // Game Over!
199            }
200        }
201        System.out.println("Program terminated!");
202    }
203 }
```

Enum 타입

- Enum 타입
  - 상수들의 집합은 enum 타입으로 선언할 수 있음 (클래스 선언과 유사)
  - 생성자의 인자로 고유 상수값을 지정함
  - 상수값을 확인하는 메쏘드를 제공하면 추후 enum 타입을 정수로 변환하기 편리함

```
enum TetrisState {
    Running(0), NewBlock(1), Finished(2); // 상수값마다 고유한 심볼 선언
    private final int value;                    // final 키워드 사용
    private TetrisState(int value) { this.value = value; }  // 생성자
    public int value() { return value; }                    // 값 확인 메쏘드
}
```

Main.java (v4)

```
1  import java.io.BufferedReader;
2  import java.io.IOException;
3  import java.io.InputStreamReader;
4  import java.util.Random;
5
6  public class     Main {
7      private static int[][][][] setOfBlockArrays = { // [7][4][?][?]
163    private static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
164    private static String line = null;
165    private static int nKeys = 0;
166    private static char getKey() throws IOException {
181    public static void main(String[] args) throws Exception {
182        char key;
183        TetrisState state;      // boolean newBlockNeeded → TetrisState state 변수로 변경
184        Tetris.init(setOfBlockArrays);
185        Tetris board = new Tetris(15, 10);
186        Random random = new Random();
187        key = (char) ('0' + random.nextInt(7));
188        board.accept(key);
189        board.printScreen(); System.out.println();
190
191        while ((key = getKey()) != 'q') {
192            state = board.accept(key);
193            board.printScreen(); System.out.println();
194            if (state == TetrisState.NewBlock) {
195                key = (char) ('0' + random.nextInt(7));
196                state = board.accept(key);
197                board.printScreen(); System.out.println();
198                if (state == TetrisState.Finished) break; // Game Over!
199            }
200        }
201        System.out.println("Program terminated!");
202    }
203 }
```

감사합니다!