Tetris Class 만들기

숭실대학교
김강희 교수
(khkim@ssu.ac.kr)

순서

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

- 실습:
  - Tetris 클래스 작성
  - Tetris 클래스 검증 (복수 객체)

Tetris Class 를 만들자

- **이제 Tetris class 를 만들 준비가 되었다. 이유는?**
- Tetris 객체를 상태 기계로 이해하자
  - 입력은 key 값, 출력은 oScreen 이다.
  - object.accept(key) → object.oScreen
- Tetris 객체(데이터 모델)는 deterministic 해야 한다.
  - 동일한 입력 시퀀스에 대해서 동일한 출력 시퀀스를 얻어야 한다.
  - 그러자면, 난수 발생을 객체 외부에서 제공해야 한다.
  - object.accept(rand_num) → object.oScreen
- 하나의 상태 기계는 개념적으로 하나의 입력 함수를 가져야 한다.
  - accept(key)와 accept(rand_num)을 하나의 함수로 표현한다.
- static/dynamic 변수들을 정의하고 초기화 함수를 정의한다.
- private/public 변수들을 정의하고 초기화 함수를 정의한다.
- hardcoded constant 들을 제거하자 → enum type 사용

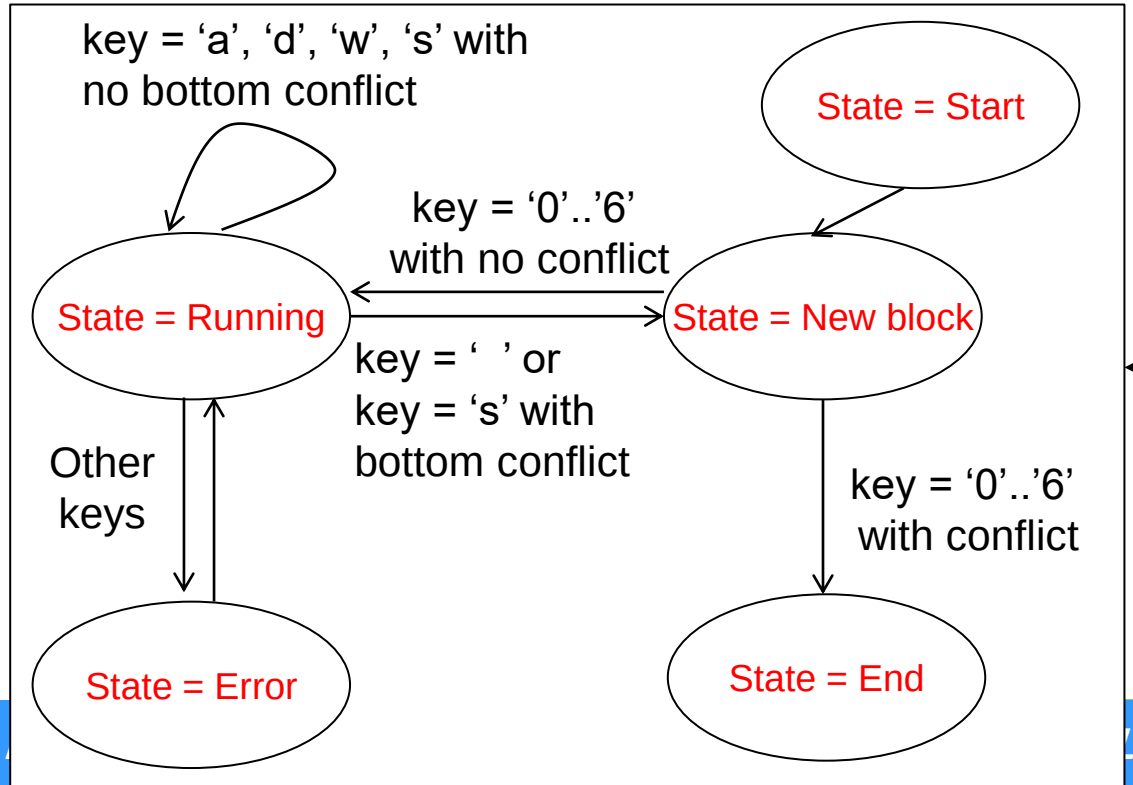
→ 위 조건들을 모두 만족하는 Tetris class 를 작성했다고 가정하고, 이를 사용하는 main 함수를 먼저 작성하자.

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 내부 상태를 가짐
  - Tetris 상태 기계의 입력을 하나의 변수 타입으로 통일하면 Tetris class 코드 상에 상태 기계 관점을 더 잘 표현할 수 있음



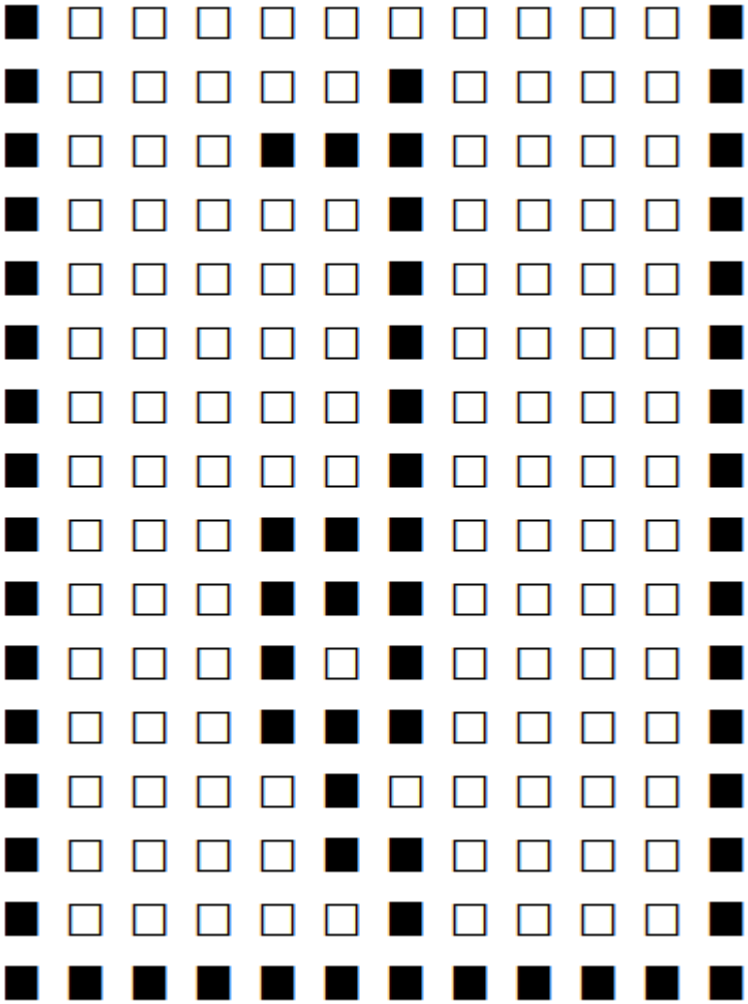
3, a, w, SPC, 0, d, s, SPC, 5, …

Main 함수 윤곽

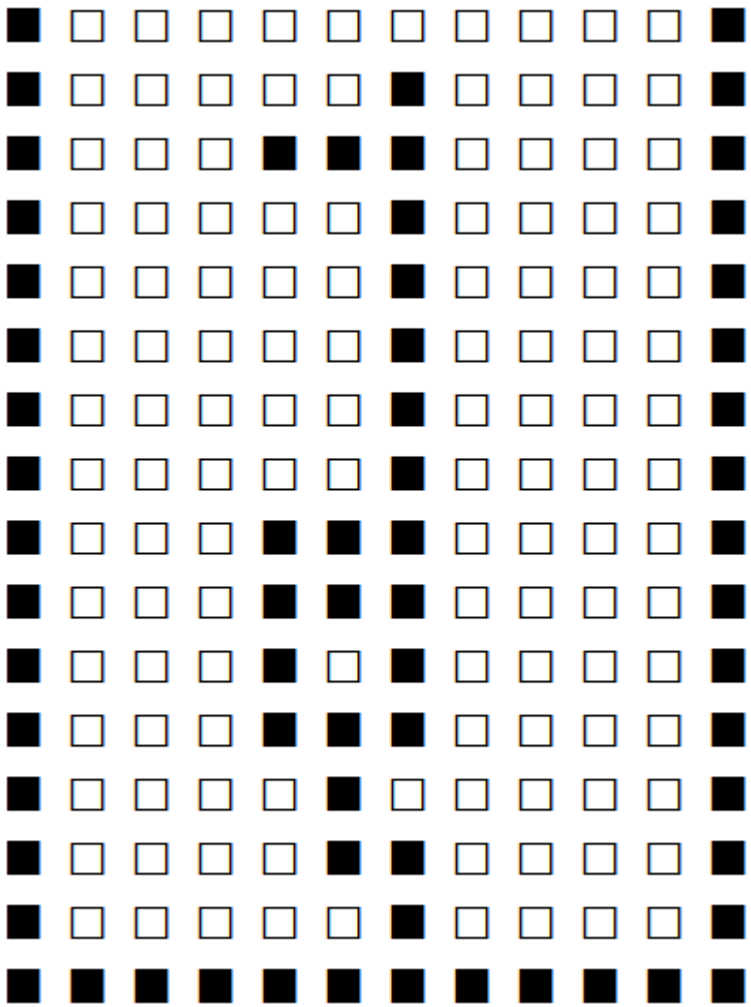
```
360  srand((unsigned int)time(NULL)); // init the random number generator
361
362  TetrisState state;
363  Tetris.init(setOfBlockArrays, MAX_BLK_TYPES, MAX_BLK_DEGREES); // static 변수들 초기화
364  Tetris *board = new Tetris(15, 10);                            // dynamic 변수들 초기화
365  key = (char) ('0' + rand() % MAX_BLK_TYPES);                   // rand_num 를 char 형으로 변환함
366  board->accept(key);
367  drawMatrix(board->get_oScreen(), SCREEN_DW); cout << endl;
368
369  while ((key = getch()) != 'q') {
370    state = board->accept(key);
371    drawMatrix(board->get_oScreen(), SCREEN_DW); cout << endl;
372    if (state == TetrisState.NewBlock) {
373      key = (char) ('0' + rand() % MAX_BLK_TYPES);
374      state = board->accept(key);
375      drawMatrix(board->get_oScreen(), SCREEN_DW); cout << endl;
376      if (state == TetrisState.Finished)
377        break;
378    }
379  }
380
381  delete board;
382  cout << "(nAlloc, nFree) = (" << Matrix::get_nAlloc() << ',' << Matrix::get_nFree() << ")" << endl;
383  cout << "Program terminated!" << endl;
384  return 0;
385  }
```

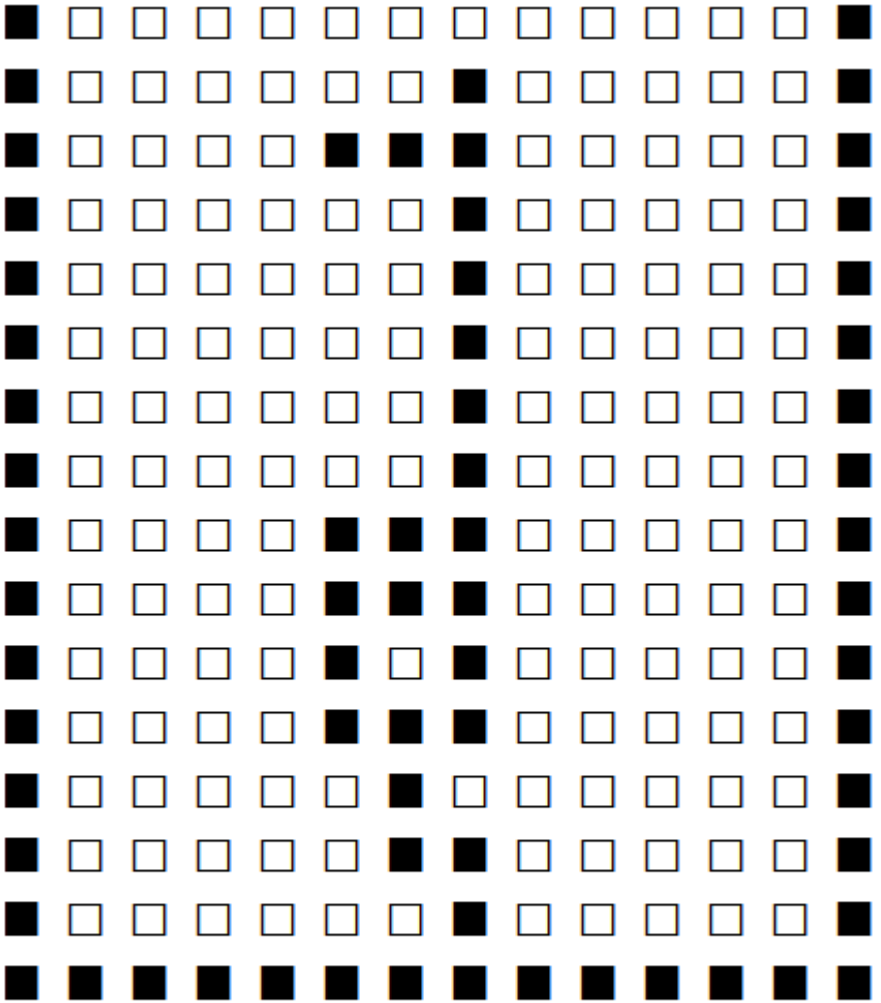
// 다음 **accept()**가 이동 **key** 를 원하는지, **rand_num** 을 원하는지 구분하기 위해 **accept()** 리턴값을 정의함

단일 객체



다중 객체: static/dynamic 변수 분리 확인





```
Program terminated!
```

남은 숙제

- 다음 조건을 만족하는 CTetris class를 Tetris class를 상속받아 작성할 것
  - Main.java 의 main 함수가 그대로 실행되어야 함. 단 Tetris라는 클래스 이름만 CTetris로 변경함
  - 빈칸은 0, 벽은 1로 표현할 것
  - main 함수의 setOfBlockArrays 배열에서
    - 테트리스 블록 1은 0 아닌 부분을 10으로 표현할 것
    - 테트리스 블록 2는 0 아닌 부분을 20으로 표현할 것
    - …
    - 테트리스 블록 7은 0 아닌 부분을 70으로 표현할 것
  - 각 테트리스 블록은 오른쪽 화면과 같이 서로 다른 글자 모양으로 출력되게 할 것