Pytet_v0.3 Comments

김강희
khkim@ssu.ac.kr

소프트웨어 설계

- **대규모 프로젝트 수행시**
  - 요구 분석 → 시스템 설계 → SW 구조 설계 → SW 상세 설계 → UI 설계 → 프로그래밍 …
- **소규모(테트리스) 프로젝트 수행시**
  - **요구 분석** : 그 중요성은 아무리 강조해도 지나치지 않음
    - PC방 주인이 2인용 전투 테트리스 설계를 주문했다고 가정하자^^
  - **프로젝트 로드맵 설계** : 1인용 흑백 콘솔 테트리스 → 2인용 컬러 GUI 테트리스
  - **소스 트리 설계** : deterministic 요소(데이터 모델)과 non-deterministic 요소(외부 인터페이스)로 나누어서 설계
  - **데이터 모델 설계** : 시나리오들의 집합화(결과물: 순서도)와 시나리오 연산화(결과물: 객체간 연산 정의) 위주로 구상
  - **데이터 모델 코딩** : 단순 시나리오 → 시나리오 확장
    - 단순 시나리오 (객체 하나 + 시나리오 하나) → 확장 시나리오 코딩 (객체 다수 + 시나리오 다수)
    - 데이터 모델의 단순성을 추구해야 함

Tetris class 를 왜 만드나?

- tetris 게임 상태를 객체화함으로써 여러 객체를 생성할 수 있음
  - 즉, **2인용 테트리스 게임을 작성**할 수 있음
- class 작성시 고려사항들
  - **main.py 코드를 먼저!!! 간결하게 작성해야 함**
  - main 코드 작성자는 tetris.py 내부를 몰라도 게임 작성 가능해야 함
  - tetris class 를 외부에서 customize 할 수 있어야 함
    - 예1: setOfBlockArrays 를 교체할 수 있어야 함 → initSetOfBlockArrays()
    - 예2: tetris object 의 스크린 배열의 크기를 선택할 수 있어야 함 → Tetris(20, 15)
  - 동일 입력 시퀀스에 대해서 동일한 결과를 얻을 수 있어야 함
    - 예: tetris object 에 입력되는 블록 타입(난수)과 사용자 키를 외부에서 선택할 수 있어야 함
  - 기타 등등

main.py

```
1   from tetris import *
2   from random import *
3
4   def rotate(m_array):
5       size = len(m_array)
6       r_array = [[0] * size for _ in range(size)]
7
8       for y in range(size):
9           for x in range(size):
10              r_array[x][size-1-y] = m_array[y][x]
11
12      return r_array
13
```

// 정방 행렬을 표현하는 array 가 주어지면,
// 90도 회전시킨 array를 리턴함

main.py

```
14  def initSetOfBlockArrays():
15      arrayBlks = [ [ [ 0, 0, 1, 0 ],      # I shape
16                      [ 0, 0, 1, 0 ],
17                      [ 0, 0, 1, 0 ],
18                      [ 0, 0, 1, 0 ] ],
19                    [ [1, 0, 0],           # J shape
20                      [1, 1, 1],
21                      [0, 0, 0] ],
22                    [ [0, 0, 1],           # L shape
23                      [1, 1, 1],
24                      [0, 0, 0] ],
25                    [ [1, 1],              # O shape
26                      [1, 1] ],
27                    [ [0, 1, 1],           # S shape
28                      [1, 1, 0],
29                      [0, 0, 0] ],
30                    [ [0, 1, 0],           # T shape
31                      [1, 1, 1],
32                      [0, 0, 0] ],
33                    [ [1, 1, 0],           # Z shape
34                      [0, 1, 1],
35                      [0, 0, 0] ]
36                  ]
37
38      nBlocks = len(arrayBlks)
39      setOfBlockArrays = [[0] * 4 for _ in range(nBlocks)]
40
41      for idxBlockType in range(nBlocks):
42          temp_array = arrayBlks[idxBlockType]
43          setOfBlockArrays[idxBlockType][0] = temp_array
44          for idxBlockDegree in range(1,4):
45              temp_array = rotate(temp_array)
46              setOfBlockArrays[idxBlockType][idxBlockDegree] = temp_array
47
48      return setOfBlockArrays
```

// 4차원 배열 [7][4][x][x] 를 생성하여
// setOfBlockArrays 이름으로 리턴함

main.py

```
50  if __name__ == "__main__":
51      setOfBlockArrays = initSetOfBlockArrays()
52
53      Tetris.init(setOfBlockArrays)
54      board = Tetris(20, 15)
55
56      idxBlockType = randint(0, 6)
57      key = '0' + str(idxBlockType)
58      board.accept(key)
59      board.printScreen()
60
61      while True:
62          key = input('Enter a key from [ q (quit), a (left), d (right), s (down), w (rotate), \' \' (drop) ] : ')
63
64          if key != 'q':
65            state = board.accept(key)
66            board.printScreen()
67
68            if state == TetrisState.NewBlock:
69                idxBlockType = randint(0, 6)
70                key = '0' + str(idxBlockType)
71                state = board.accept(key)
72                if state == TetrisState.Finished:
73                    board.printScreen()
74                    print('Game Over!!!')
75                    break
76                board.printScreen()
77          else:
78            print('Game aborted...')
79            break
80
81      print('Program terminated...')
82
83  ### end of pytet.py
```

// 4차원 배열 [7][4][x][x] 를 생성

// Tetris class 에 4차원 배열 [7][4][x][x] 를 전달

// 20 x 15 크기의 Tetris object 생성

// 0..6 사이의 난수 생성

// 생성된 난수를 문자 코드로 변환 (예: 0 → '0')

// board object 에 최초 출현할 블록 타입(key)를 지정

// board object 의 oScreen 를 화면에 출력

// state 가 NewBlock 상태이면

// state 가 Finished 상태이면

tetris.py

```
1  from matrix import *
2  from enum import Enum
3
4  class TetrisState(Enum):  // 테트리스 게임 상태들을 3가지 상수들로 정의하는 클래스
5      Running = 0
6      NewBlock = 1
7      Finished = 2
8  ### end of class TetrisState():
9
10 class Tetris():           // Tetris class 정의 시작
11     nBlockTypes = 0       // self.XXX 변수들: 객체 소속 변수들 (상태 변수들, 동적 변수들)
12     nBlockDegrees = 0
13     setOfBlockObjects = 0
14     iScreenDw = 0    # larget enough to cover the largest block
15
16     @classmethod
17     def init(cls, setOfBlockArrays):   // Tetris.XXX 변수들: class 소속 변수들 (정적 변수들)
18         Tetris.nBlockTypes = len(setOfBlockArrays)
19         Tetris.nBlockDegrees = len(setOfBlockArrays[0])
20         Tetris.setOfBlockObjects = [[0] * Tetris.nBlockDegrees for _ in range(Tetris.nBlockTypes)]
21         arrayBlk_maxSize = 0
22         for i in range(Tetris.nBlockTypes):
23             if arrayBlk_maxSize <= len(setOfBlockArrays[i][0]):
24                 arrayBlk_maxSize = len(setOfBlockArrays[i][0])
25         Tetris.iScreenDw = arrayBlk_maxSize     # larget enough to cover the largest block
26                                                 // iScreenDw 값을 setOfBlockArrays 에서 추출함!
27         for i in range(Tetris.nBlockTypes):
28             for j in range(Tetris.nBlockDegrees):
29                 Tetris.setOfBlockObjects[i][j] = Matrix(setOfBlockArrays[i][j])
30         return              // Tetris.setOfBlockObjects 라는 28개 object 들의 2차원 배열 생성
31
```

tetris.py

```
32      def createArrayScreen(self):  // iScreen 를 U 컵 모양으로 초기화하기 위한 배열 생성
33          self.arrayScreenDx = Tetris.iScreenDw * 2 + self.iScreenDx
34          self.arrayScreenDy = self.iScreenDy + Tetris.iScreenDw
35          self.arrayScreen = [[0] * self.arrayScreenDx for _ in range(self.arrayScreenDy)]
36          for y in range(self.iScreenDy):
37              for x in range(Tetris.iScreenDw):
38                  self.arrayScreen[y][x] = 1
39              for x in range(self.iScreenDx):
40                  self.arrayScreen[y][Tetris.iScreenDw + x] = 0
41              for x in range(Tetris.iScreenDw):
42                  self.arrayScreen[y][Tetris.iScreenDw + self.iScreenDx + x] = 1
43
44          for y in range(Tetris.iScreenDw):
45              for x in range(self.arrayScreenDx):
46                  self.arrayScreen[self.iScreenDy + y][x] = 1
47
48          return self.arrayScreen
49
50      def __init__(self, iScreenDy, iScreenDx): // 생성자 함수
51          self.iScreenDy = iScreenDy
52          self.iScreenDx = iScreenDx
53          self.idxBlockDegree = 0
54          arrayScreen = self.createArrayScreen()
55          self.iScreen = Matrix(arrayScreen)
56          self.oScreen = Matrix(self.iScreen)
57          self.justStarted = True                    // justStarted 플래그는 accept 함수가
58          return                                     // 1회 호출되고 나면 False 로 변경해야 함
59
```

tetris.py

```
60      def accept(self, key):
61          self.state = TetrisState.Running
62
63          if key >= '0' and key <= '6':
64              if self.justStarted == False:
65                  self.deleteFullLines()
66              self.iScreen = Matrix(self.oScreen)
67              self.top = 0
68              self.left = Tetris.iScreenDw + self.iScreenDx//2 - 2
69              self.idxBlockType = int(key)
70              self.idxBlockDegree = 0
71              self.currBlk = Tetris.setOfBlockObjects[self.idxBlockType][self.idxBlockDegree]
72              self.tempBlk = self.iScreen.clip(self.top, self.left, self.top+self.currBlk.get_dy(), self.left+self.currBlk.get_dx())
73              self.tempBlk = self.tempBlk + self.currBlk
74              self.justStarted = False
75              print()
76
77              if self.tempBlk.anyGreaterThan(1):
78                  self.state = TetrisState.Finished
79              self.oScreen = Matrix(self.iScreen)
80              self.oScreen.paste(self.tempBlk, self.top, self.left)
81              return self.state
82          elif key == 'q':
83              pass
84          elif key == 'a': # move left
85              self.left -= 1
86          elif key == 'd': # move right
87              self.left += 1
88          elif key == 's': # move down
89              self.top += 1
90          elif key == 'w': # rotate the block clockwise
91              self.idxBlockDegree = (self.idxBlockDegree + 1) % 4
92              self.currBlk = Tetris.setOfBlockObjects[self.idxBlockType][self.idxBlockDegree]
93          elif key == ' ': # drop the block
94              while not self.tempBlk.anyGreaterThan(1):
95                      self.top += 1
96                      self.tempBlk = self.iScreen.clip(self.top, self.left, self.top+self.currBlk.get_dy(), self.left+self.currBlk.get_dx())
97                      self.tempBlk = self.tempBlk + self.currBlk
98          else:
99              print('Wrong key!!!')
```

상수 제거할 필요 있음!

변수 초기화 및 첫번째 블록 출력

키입력에 대한 do 동작

상수 제거할 필요 있음!

tetris.py

tempBlk 생성

```
101         self.tempBlk = self.iScreen.clip(self.top, self.left, self.top+self.currBlk.get_dy(), self.left+self.currBlk.get_dx())
102         self.tempBlk = self.tempBlk + self.currBlk
103
104         if self.tempBlk.anyGreaterThan(1):   ## 벽 충돌시 undo 수행
105             if key == 'a': # undo: move right
106                 self.left += 1
107             elif key == 'd': # undo: move left
108                 self.left -= 1
109             elif key == 's': # undo: move up
110                 self.top -= 1
111                 self.state = TetrisState.NewBlock
112             elif key == 'w': # undo: rotate the block counter-clockwise
113                 idxBlockDegree = (idxBlockDegree - 1) % 4
114                 self.currBlk = Tetris.setOfBlockObjects[self.idxBlockType][idxBlockDegree]
115             elif key == ' ': # undo: move up
116                 self.top -= 1
117                 self.state = TetrisState.NewBlock
118
119             self.tempBlk = self.iScreen.clip(self.top, self.left, self.top+self.currBlk.get_dy(), self.left+self.currBlk.get_dx())
120             self.tempBlk = self.tempBlk + self.currBlk
121
122         self.oScreen = Matrix(self.iScreen)
123         self.oScreen.paste(self.tempBlk, self.top, self.left)
124
125         return self.state
126
```

키입력에 대한 undo 동작

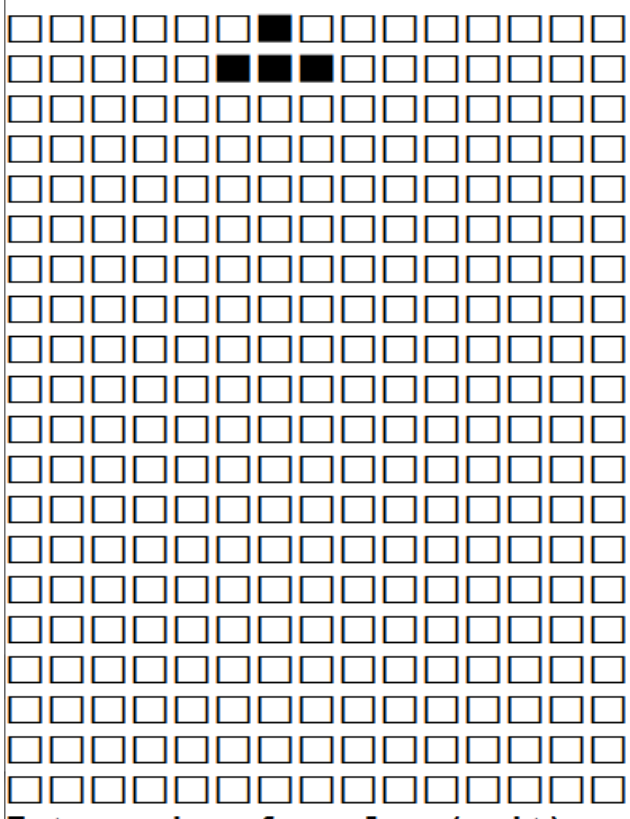
oScreen 출력

tetris.py

```
127     def printScreen(self):                          // oScreen 내용을 화면에 출력
128         array = self.oScreen.get_array()
129
130         for y in range(self.oScreen.get_dy()-Tetris.iScreenDw):
131             for x in range(Tetris.iScreenDw, self.oScreen.get_dx()-Tetris.iScreenDw):
132                 if array[y][x] == 0:
133                     print("□", end='')
134                 elif array[y][x] == 1:
135                     print("■", end='')
136                 else:
137                     print("XX", end='')
138             print()
139
140
141     def deleteFullLines(self):                      // 과제: 줄지우기 함수 작성
142         return
143
144 ### end of class Tetris():
```

pytet_v0.3 실행 화면

```
misyslab@LAPTOP-Q2R29N8I:~/sweng-2021/pytet_v0.3$ ll
합계 28
drwxr-xr-x 3 misyslab misyslab 4096  3월 18 09:17 ./
drwxr-xr-x 5 misyslab misyslab 4096  3월 18 09:17 ../
drwxr-xr-x 2 misyslab misyslab 4096  3월 18 09:02 __pycache__/
-rw-r--r-- 1 misyslab misyslab 2608  3월 18 09:01 main.py
-rw-r--r-- 1 misyslab misyslab 3921  3월 18 01:02 matrix.py
-rw-r--r-- 1 misyslab misyslab 5884  3월 18 09:03 tetris.py
misyslab@LAPTOP-Q2R29N8I:~/sweng-2021/pytet_v0.3$ python3 main.py
```



```
Enter a key from [ q (quit), a (left), d (right), s (down), w (rotate), ' ' (dro
p) ] : _
```

pytet_v0.1 → v0.3 작성시 고려사항들

```
16  ###
17  ### initialize variables
18  ###
19  arrayBlk = [ [ 0, 0, 1, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 1, 0 ] ]
20
21  ### integer variables: must always be integer!
22  iScreenDy = 15
23  iScreenDx = 10
24  iScreenDw = 4
25  top = 0
26  left = iScreenDw + iScreenDx//2 - 2
27
28  newBlockNeeded = False
29
30  arrayScreen = [
31      [ 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1 ],
32      [ 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1 ],
33      [ 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1 ],
```

// iScreenDy → 객체 소속 변수
// iScreenDx → 객체 소속 변수
// iScreenDw → 클래스 소속 변수
// top → 객체 소속 변수
// left → 객체 소속 변수

// newBlockNeeded → 임시 변수

// arrayScreen → 임시 변수

```
46      [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ],
47      [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ],
48      [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ],
49      [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ] ]
50
51  ###
52  ### prepare the initial screen output
53  ###
54  iScreen = Matrix(arrayScreen)
55  oScreen = Matrix(iScreen)
56  currBlk = Matrix(arrayBlk)
57  tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx())
58  tempBlk = tempBlk + currBlk
59  oScreen.paste(tempBlk, top, left)
60  draw_matrix(oScreen); print()
61
```

// iScreen → 객체 소속 변수
// oScreen → 객체 소속 변수
// currBlk → 객체 소속 변수
// tempBlk → 임시 변수

→ while loop 안의 newBlockNeeded 조건문의 body 코드와 동일함

pytet_v0.1 → v0.3 작성시 고려사항들

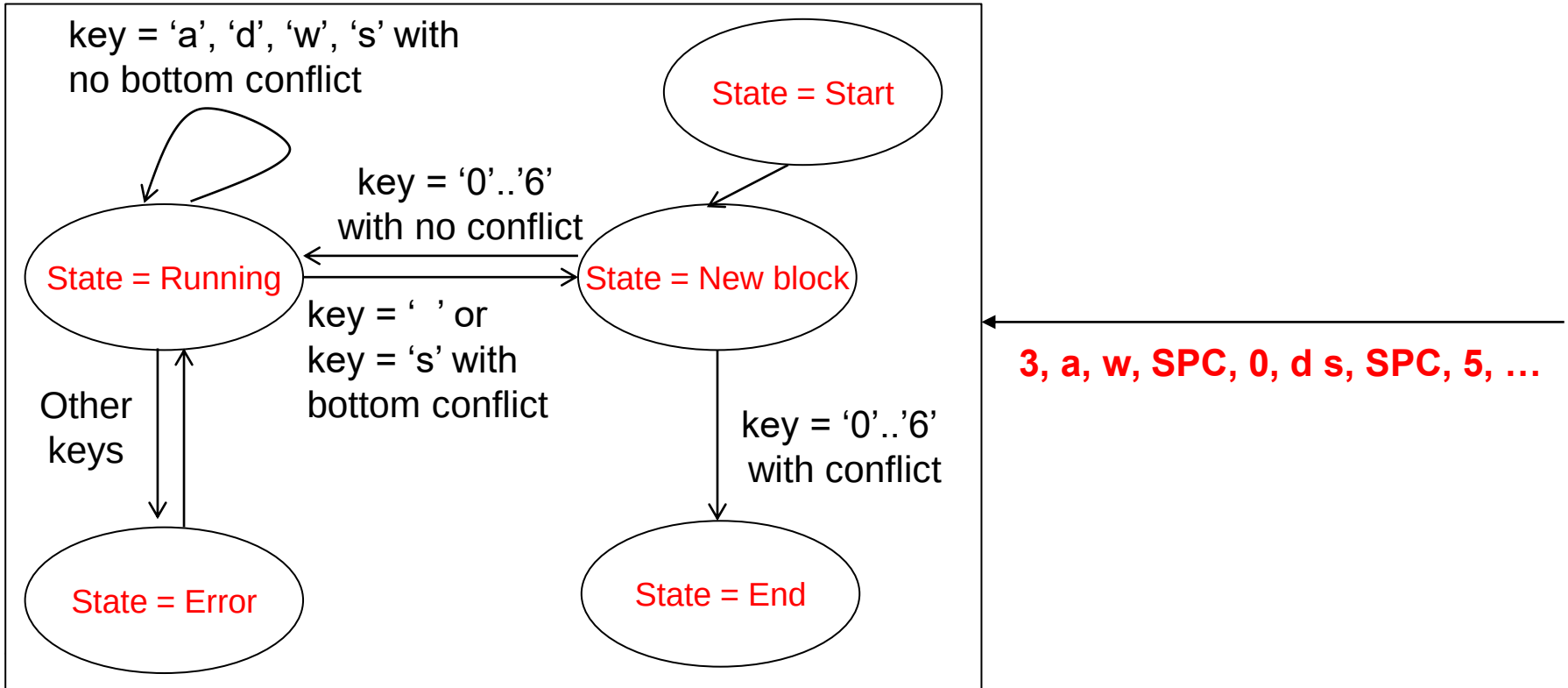
```
62  ###
63  ### execute the loop
64  ###
65
66  while True:
67      key = input('Enter a key from [ q (quit), a (left), d (right), s (down), w (rotate), \' \' (drop) ] : ')
68      if key == 'q':
71      elif key == 'a': # move left
73      elif key == 'd': # move right
75      elif key == 's': # move down
77      elif key == 'w': # rotate the block clockwise
80      elif key == ' ': # drop the block
83      else:
86
87      tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx())
88      tempBlk = tempBlk + currBlk
89      if tempBlk.anyGreaterThan(1):
90          if key == 'a': # undo: move right
92          elif key == 'd': # undo: move left
94          elif key == 's': # undo: move up
97          elif key == 'w': # undo: rotate the block counter-clockwise
99          elif key == ' ': # undo: move up
101
102         tempBlk = iScreen.clip(top, left, top+currBlk.get_dy(), left+currBlk.get_dx())
103         tempBlk = tempBlk + currBlk
104
105     oScreen = Matrix(iScreen)
106     oScreen.paste(tempBlk, top, left)
107     draw_matrix(oScreen); print()
108
109     if newBlockNeeded:
124
```

accept 함수

while loop 의 시작 부분으로 이동함

pytet_v0.1 → v0.3 작성시 고려사항들

- 테트리스 게임의 상태 기계 모델
  - key 값과 idxBlockType 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



[부록] 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

코딩 지침들

- **기능 위주의 코딩을 먼저 한다.**
- **불필요한 조건문들을 없앤다.**
- **반복되는 코드 조각을 없앤다.**
- **클래스를 작성하기 전에 클래스 사용법(main 코드)를 먼저 작성한다.**
  - 객체/클래스 소속 변수들을 main 코드에서 직접 참조하면 안 된다.
  - 클래스 함수들은 입력과 출력 위주로 최소 개수만 작성한다.
- **클래스 내부에서 사용하는 데이터는 외부에서 customize 할 수 있게 한다.**
  - 생성자 함수들의 인자 또는 클래스 함수들의 인자로 전달한다.
- **클래스 내부에 hardcoded constant 들을 제거한다.**
  - 0 또는 1 이외의 상수들은 그 존재 이유를 모두 의심한다.
- **클래스를 상태 기계로 이해한다**(util 클래스는 제외).
  - 동일 입력 시퀀스에 대해서 동일 출력 시퀀스를 얻을 수 있어야 한다.