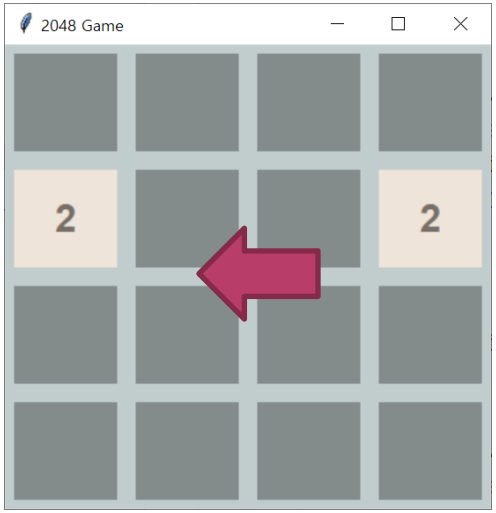
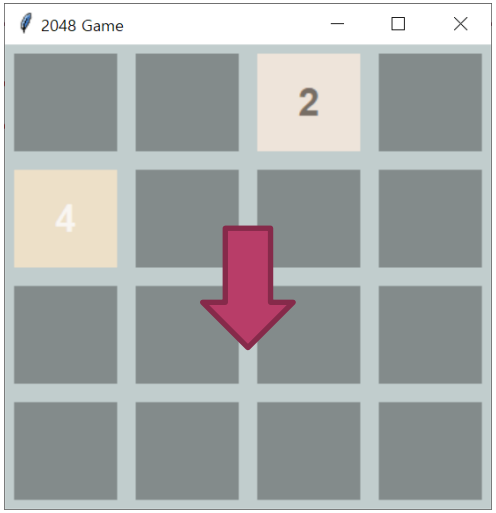
2048

담당교수 : 김영식

2048


2048 Game
2
2


2048 Game
2
4


2048 Game
2
4
2


2048 Game
2
2
4
2


2048 Game
2
4
4
2


2048 Game
2
4
4
2
2

2048


2048 Game
4
32 2
128 4 2
32 2


2048 Game
4
32 2 2
128 4
32 2 2


2048 Game
4 2
32 4
128 4
32 4


2048 Game
4
32 2
128 4
32 8 2


2048 Game
4 2 2
32 4
128 8 2
32


2048 Game
4 4
32 4
128 8 2
2 32

class Game2048

```
from tkinter import *
from tkinter import messagebox
import random

class Game2048:
    # 숫자 배경 색 사전
    bg_color = {
        2: '#eee4da', 4: '#ede0c8', 8: '#edc850',
        16: '#edc53f', 32: '#f67c5f', 64: '#f65e3b',
        128: '#edcf72', 256: '#edcc61', 512: '#f2b179',
        1024: '#f59563', 2048: '#edc22e', }
    # 숫자 색 사전
    color = {
        2: '#776e65',  4: '#f9f6f2', 8: '#f9f6f2',
        16: '#f9f6f2', 32: '#f9f6f2', 64: '#f9f6f2',
        128: '#f9f6f2', 256: '#f9f6f2', 512: '#776e65',
        1024: '#f9f6f2', 2048: '#f9f6f2',   }

    # . . .
```

def __init__

```
    def __init__(self,size):
        self.n = size
        self.window = Tk()
        self.window.title('2048 Game')
        self.gameArea = Frame(self.window,bg='azure3')
        self.gridCell = [[0]*self.n for _ in range(self.n)]
        self.compress = False
        self.merge = False
        self.moved = False
        self.end = False
        self.won = False
        self.score = 0
        self.board = [ ]

        # . . .
Game2048(4)
```

def __init__

```
    def __init__(self,size):
        # . . .
        for r in range(self.n):
            rows = [ ]
            for c in range(self.n):
                l = Label(self.gameArea, text='', bg='azure4', \
                                font=('arial',22,'bold'), width=4,height=2)
                l.grid(row=r,column=c,padx=7,pady=7)
                rows.append(l)
            self.board.append(rows)
        self.gameArea.pack()

        self.random_cell()
        self.random_cell()
        self.paintGrid()
        self.window.bind('<Key>',self.link_keys)
        self.window.mainloop()
Game2048(4)
```

def link_keys

```
def link_keys(self,event):
    if self.end or self.won:
        return
    self.compress = False
    self.merge = False
    self.moved = False
    key = event.keysym
    if key == 'Up':
        # …
    elif key == 'Down':
        # …
    elif key == 'Left':
        self.compressGrid()
        self.mergeGrid()
        self.moved = self.compress or self.merge
        self.compressGrid()
    elif key == 'Right':
        # …
    else:
        pass
    self.paintGrid()

     # . . .
```

def link_keys

```
def link_keys(self,event):
    # . . .
    flag = 0
    for r in range(self.n):
        for c in range(self.n):
            if (self.gridCell[r][c] == 2048):
                flag = 1
                break
    if flag == 1:
        self.won = True
        messagebox.showinfo('2048', 'Yon won!!')
        return

    for r in range(self.n):
        for c in range(self.n):
            if (self.gridCell[r][c] == 0):
                flag = 1
                break
    if not (flag or self.can_merge()):
        self.end = True
        messagebox.showinfo('2048', 'Game Over!!')
    if self.moved:
        self.random_cell()
    self.paintGrid()
```

def random_cell

```
def random_cell(self):
    cells = [ ]
    for r in range(self.n):
        for c in range(self.n):
            if self.gridCell[r][c] == 0:
                cells.append((r,c))
    curr = random.choice(cells)
    (r,c) = curr
    self.gridCell[r][c] = 2
```

def paintGrid

```
def paintGrid(self):
    for r in range(self.n):
        for c in range(self.n):
            if self.gridCell[r][c] == 0:
                self.board[r][c].configure(text='',bg='azure4')
            else:
                self.board[r][c].configure(text=str(self.gridCell[r][c]),
                                           bg=self.bg_color[self.gridCell[r][c]],
                                           fg=self.color[self.gridCell[r][c]])
```

def compressGrid

```
def compressGrid(self):
    self.compress = False
    temp = [[0]*self.n for _ in range(self.n)]
    for r in range(self.n):
        cnt = 0
        for c in range(self.n):
            if self.gridCell[r][c] != 0:
                temp[r][cnt] = self.gridCell[r][c]
                if cnt != c:
                    self.compress = True
                cnt += 1
    self.gridCell = temp
```

def mergeGrid

```
def mergeGrid(self):
    self.merge = False
    for r in range(self.n):
        for c in range(self.n-1):
            if self.gridCell[r][c] == self.gridCell[r][c+1] and \
                                      self.gridCell[r][c] != 0:
                self.gridCell[r][c] *= 2
                self.gridCell[r][c+1] = 0
                self.score += self.gridCell[r][c]
                self.merge = True
```