CSED101. Programming & Problem solving
Fall 2023
Programming Assignment #4
(75 points)

정의동 (justicedong@postech.ac.kr)

■ **제출 마감일: 2023.12.14 23:59**

■ **개발 환경: Python 3.x**

■ **제출물**

- .py 소스 코드 (assn4_model.py, assn4_app.py)
  - ➢ 프로그램의 소스 코드에 채점자의 이해를 돕기 위한 주석을 반드시 붙여주세요.
- 보고서 파일 (.docx, .hwp 또는 .pdf; assn4.docx, assn4.hwp 또는 assn4.pdf)
- 보고서는 AssnReadMe.pdf를 참조하여 작성하시면 됩니다.
- 명예 서약 (Honor code): 표지에 다음의 서약을 기입하여 제출해 주세요: "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." 보고서 표지에 명예 서약이 기입되어 있지 않은 과제는 제출되지 않은 것으로 처리됩니다.
- 작성한 소스 코드와 보고서 파일은 PLMS를 통해 제출해 주세요.

■ **주의 사항**

- 구문 오류(Syntax Error)가 발생하거나 실행이 되지 않는 과제는 0점으로 채점됩니다.
- 제출 기한보다 하루 늦게 제출된 과제는 최종 20%, 이틀 늦게 제출된 과제는 최종 40% 감점됩니다. 제출 기한보다 사흘 이상 늦으면 제출 받지 않습니다 (0점 처리). 늦은 제출시 PLMS에 기록된 최종 수정일시를 기준으로 감점합니다.
- 각 문제의 제한 조건과 요구 사항을 반드시 지켜 주시기 바랍니다.
- 이번 과제는 추가 기능 구현과 관련된 추가 점수가 따로 없습니다.
- 보고서 작성 시, 참고 링크도 포함해주세요. 부정 행위 적발 시 0점 처리됩니다.
- 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 "POSTECH 전자컴퓨터공학부 부정행위 정의"를 따릅니다 (PLMS의 본 과목 공지사항에 등록된 글 중, 제목이 [document about cheating]인 글에 첨부되어 있는 disciplinary.pdf를 참조할 것.)

■ Problem: 지뢰찾기

(목적)

(1) Python tkinter 라이브러리를 이용하여 GUI 프로그래밍을 익힙니다.
(2) 클래스 정의 및 인스턴스 생성을 익힙니다.
(3) 클래스 상속 및 메서드 오버 라이딩을 익힙니다.

(설명)

지뢰 찾기는 기억력과 추리력을 요하는 간단한 게임으로, 지뢰를 피해 모든 빈 칸을 찾는 것이 이 게임의 목표입니다. 보드의 각 칸에는 지뢰가 있는 칸과 없는 칸으로 이루어지며, 플레이어는 마우스 클릭을 통해서 보드와 상호작용하며 지뢰가 없는 구역만 밝혀야 합니다. 사용자가 만약 지뢰가 없는 칸을 클릭할 경우에는 인접한 8칸(자신을 중심으로 한 3x3 구역)에 대하여 지뢰의 개수를 숫자를 통해서 힌트를 줍니다. 이를 기반한 추리를 통해서 지뢰가 없는 모든 칸을 밝혀야 합니다. 아래 그림에서 볼 수 있듯이 파란 네모 안에 3은 자신과 인접한 8칸(빨간 네모 안의 영역)에 3개의 지뢰가 있는 것을 나타냅니다.

[게임 규칙]

(1) 플레이어는 두 가지 동작(마우스 우 클릭, 좌 클릭)을 통해서 보드와 상호 작용할 수 있습니다.

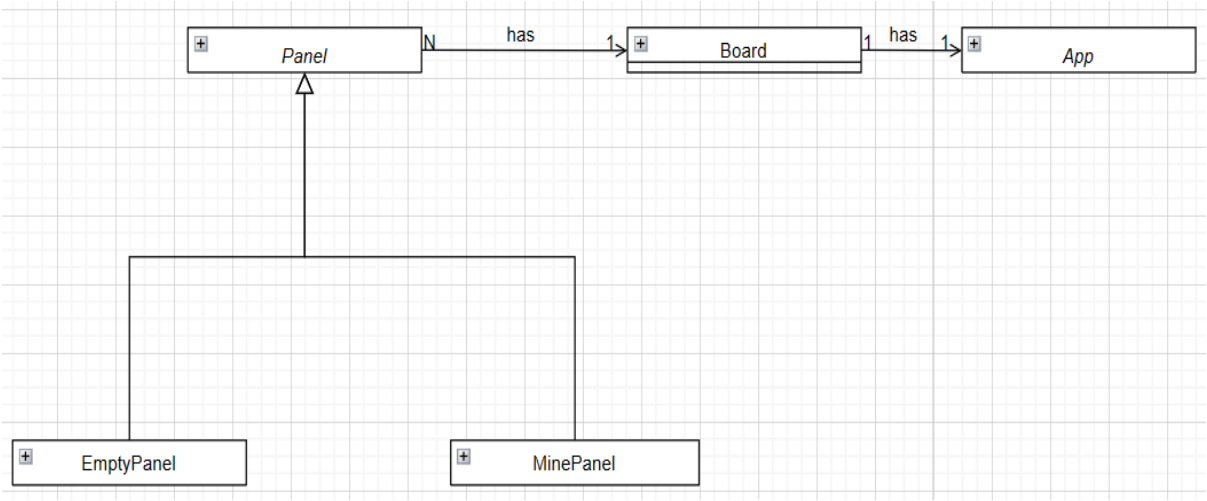
(2) 플레이어가 마우스 우 클릭을 수행했을 때 기대할 수 있는 결과는 다음과 같습니다.
   A. 이미 밝혀진 칸인 경우, 아무 일도 일어나지 않습니다.
   B. 아직 밝혀지지 않은 칸인 경우
      - 아직 깃발이 없는 경우, 깃발을 통해서 마킹을 할 수 있습니다. 깃발의 경우 플레이의 부수적인 도구로 모든 지뢰가 있는 위치를 기억하는 것은 플레이에 부담을 줄 수 있기 때문에 플레이 시에 지뢰를 표시하는 도구로 사용할 수 있습니다. (표시할 수 있는 깃발의 개수의 제한은 없습니다.)
      - 깃발이 있는 경우, 해당 깃발을 지울 수 있습니다.

(3) 플레이어가 마우스 좌 클릭을 수행했을 때 기대할 수 있는 결과는 다음과 같습니다.
   A. 깃발로 표시된 칸인 경우, 아무 일도 일어나지 않습니다.
   B. 이미 밝혀진 칸인 경우, 아무 일도 일어나지 않습니다.
   C. 아직 밝혀지지 않은 칸인 경우
      - 해당 위치가 지뢰라면 게임은 종료되고 플레이어는 패배합니다.
      - 해당 위치가 빈 칸이라면 인접한 8칸의 지뢰의 수를 표시합니다. 만약, 인접한 8칸의 지뢰의 수가 0인 경우, 해당 과정을 각 8 칸에서 다시 반복합니다.

(4) 플레이어가 지뢰를 제외한 모든 빈 칸을 밝혔다면, 플레이어는 승리합니다.

(구현 목표)

(1) 총 4개의 클래스를 통해서 지뢰 찾기를 구현합니다. 클래스를 구현할 때에는 아래에서 제시한 변수 및 메서드들을 반드시 사용하여 프로그램을 구현해야 합니다. 아래 명시된 변수 이름, 메서드 이름 및 메서드의 매개변수 및 리턴 값은 변경 불가능 합니다. 이외에 필요한 변수 및 메서드는 추가로 정의해서 사용할 수 있으며, 추가한 내용에 대해서는 소스코드와 보고서에 설명을 포함하여 작성해야 합니다.



A. Panel

지뢰 찾기 게임의 각 칸에 해당하는 class입니다. 아래에 주어진 모든 변수, 함수를 구현해야 합니다.

① [변수] isRevealed : 해당 panel이 밝혀진 상태인지를 나타내는 bool type 변수로, instance를 생성 시 반드시 False로 초기화 합니다.

② [변수] hasFlag : 해당 panel이 flag를 보유하고 있는지를 나타내는 bool type 변수로, instance를 생성 시 반드시 False로 초기화 합니다.

③ [함수] toggleFlag(self) : 해당 panel의 hasFlag를 toggle합니다. 즉, 현재 hasFlag가 True라면 실행 후 False가 되고, False라면 실행 후 True가 됩니다.

④ [함수] unveil(self) : 해당 panel을 밝혀진 상태로 변경됩니다.
* Hint: isRevealed의 값이 True가 되어야 합니다.

B. EmptyPanel

지뢰가 존재하지 않는 칸을 의미하는 class로 Panel을 상속해야 합니다. 아래에 주어진 모든 함수를 구현해야 합니다.

① [변수] numOfNearMines: 해당 panel과 인접한 mine의 수를 저장하는 num type 변수로, instance를 생성 시 반드시 0으로 초기화 합니다.

② [함수] addNumOfNearMines(self): 해당 panel의 numOfNearMines의 값을 1 증가시킵니다.

③ [함수] unveil(self): 부모인 Panel의 unveil을 수행하고, 이때 인접한 mine의 수를 return합니다.

C. MinePanel

지뢰가 존재하는 칸을 의미하는 class로 Panel을 상속해야 합니다. 아래에 주어진 모든 함수를 구현해야 합니다.

① [함수] unveil(self): 부모인 Panel의 unveil을 수행하고, -1을 return합니다.

D. Board

지뢰 찾기 게임에 사용되는 모든 기능을 포함하는 class입니다. 여기서는 앞서 만든 Panel class들을 활용하여 아래에 주어진 모든 함수를 구현해야 합니다.

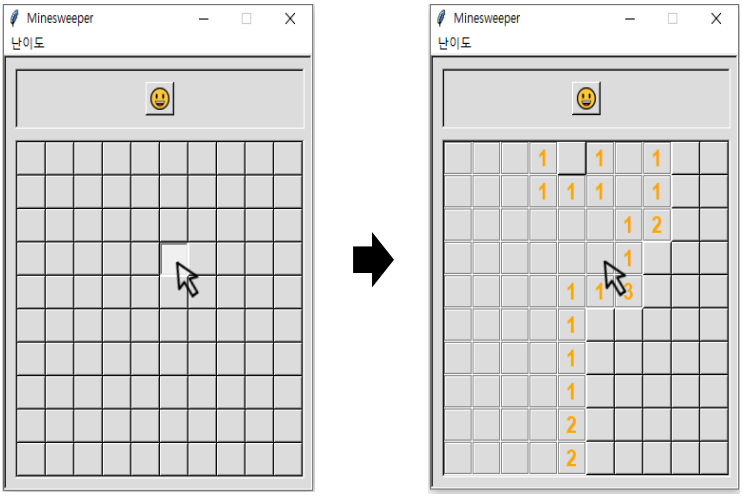
① [변수] panels : 2차원 리스트로 Panel instance를 저장합니다. 이는 reset 함수가 호출될 때마다 초기화됩니다. (따라서, instance 생성 시에 반드시 특정 값으로 초기화할 필요는 없습니다.)

② [함수] reset(self, numMine, height, width): Board를 초기화합니다. 이 과정에서 panels에 height크기만큼의 행과 width만큼의 열을 가지는 2차원 리스트를 생성합니다. 또한, numMine으로 주어진 숫자만큼 mine을 random한 위치에 분포시킵니다. Mine이 위치한 곳에는 MinePanel이 들어가고, Mine이 없는 위치에는 EmptyPanel이 들어가도록 합니다. 마지막으로, 각 EmptyPanel에는 인접한 Panel에 위치한 Mine의 수만큼 numOfNearMines가 들어가도록 합니다. 이때, EmptyPanel 의 변수인 numOfNearMines 변수에 직접 접근하여 값을 변경해서는 안되고, addNumOfNearMines 함수만을 이용하여 구현하도록 합니다.
(해당 지뢰 찾기 게임에서는 Board의 사이즈와 지뢰의 개수가 난이도에 따라서 유동적으로 변화할 수 있습니다. 이 함수는 그 상황마다 Board를 Random하게 초기할 때 사용할 수 있습니다.)

③ [함수] getNumOfRevealedPanels(self): 현재 board에 밝혀져 있는 Panel의 개수를 return합니다.

④ [함수] unveil(self, y, x): panels의 y행 x열에 위치한 Panel을 밝힙니다. 만약, 밝혀낸 Panel이 지뢰라면, -1을 return합니다. 그렇지 않다면, 아무것도 return하지 않습니다. 또한, 만약 밝혀낸 Panel의 numOfNearMines의 값이 0이라면, 인접한 8칸 중에 지뢰가 없는 칸에 대해서 이 과정을 반복합니다. 만약 깃발이 존재하는 칸에도 반복되었고, 해당 칸이 지뢰가 아닌 경우 해당 칸의 깃발을 제거하고, 밝혀 냅니다. 그렇지 않고 지뢰가 있는 경우 깃발을 유지합니다.
* Hint: 아래 그림에서 알 수 있듯이 한 번의 y행 x열의 Panel을 밝히는 행위가 얼마나 많은 Panel을 밝히게 될지는 알 수 없습니다. 따라서, 별도의 자료

구조(stack 등)을 이용하여 차례대로 밝혀내거나 재귀 함수를 활용하여 이 문제를 해결할 수 있습니다.
(해당 함수는 Board의 특정 구역(y행 x열)을 선택하였을 때의 결과를 panels에 반영하는 기능을 합니다. 따라서, GUI에서는 해당 구역의 선택 시 이 함수를 호출하고, panels의 최종 결과를 그대로 화면에 보여주기만 하면 됩니다.)



⑤ [함수] toggleFlag(self, y, x): panels의 y행 x열에 위치한 Panel의 flag를 toggle합니다. 즉, 현재 flag가 존재한다면 실행 후 없어지고, 없다면 실행 후 존재하게 됩니다.

⑥ [함수] checkReveal(self, y, x): panels의 y행 x열에 위치한 Panel이 밝혀져 있는지 확인합니다. 밝혀져 있다면 True를 return하고, 그렇지 않다면 False를 return합니다.

⑦ [함수] checkFlag(self, y, x): panels의 y행 x열에 위치한 Panel에 flag가 있는지 확인합니다. flag가 있다면, True를 return하고, 그렇지 않다면 False를 return합니다.

⑧ [함수] checkMine(self, y, x): panels의 y행 x열에 위치한 Panel에 mine이 있는지 확인합니다. mine이 있다면, True를 return하고, 그렇지 않다면 False를 return합니다.

⑨ [함수] getNumOfNearMines(self, y, x): panels의 y행 x열에 위치한 Panel에 인접한 mine의 수를 return합니다. 이는 해당 Panel이 EmptyPanel임을 가정으로 합니다.

E. App

GUI 구현 코드를 담당합니다. 전체 GUI를 감싸는 Frame를 상속하고, 각 widget에 Board에 존재하는 함수를 적절하게 사용하여 구현합니다. 쉽게 구현하기 위해서는 board가 변경될 때마다 전체 판을 다시 그리는 것도 좋은 방법이 될 수 있습니다. 별도로 제시하는 method는 없고, 자율적으로 해당 class를 적절히 채워 아래 (2)에서 제시하는 GUI 요구 사항을 만족시켜야 합니다. (추가한 변수 및 메서드에 대한 설명을

반드시 주석 및 보고서에 추가해야 합니다.)

(2) 해당 시스템에서 요구하는 GUI는 다음과 같습니다.

A. 전체는 Header 부분과 Body 부분으로 나뉩니다.

B. Header 부분에는 초기화 버튼이 존재하며, 가로 방향에서 중앙에 위치합니다.

① 좌 클릭 시에 Body 부분의 모든 내용이 초기화 합니다.

② 평상 시에는 웃는 모양이 표기되고, 패배 시에는 해골, 성공 시에는 선글라스를 낀 캐릭터가 노출됩니다.

C. Body 부분에는 버튼으로 이루어진 지뢰 찾기 보드가 주어집니다. (처음 크기는 10x10으로 고정합니다.)

① 각 버튼은 클릭이 가능해야 하고, 이미 드러난 부분은 시각적으로 명확하게 구분이 가야 합니다.

② 밝혀진 빈 칸 중에서 인근 지뢰의 수가 0인 칸은 0을 표기하지 않고, 비워 둡니다.

③ 좌 클릭 시에 해당 위치에 해당하는 값이 밝혀집니다(숫자, 빈칸, 폭탄 중 하나). 이미 밝혀졌거나 깃발이 존재하는 경우 아무 일도 일어나지 않습니다.

④ 우 클릭 시에 해당 위치에 깃발을 꽂거나 제거할 수 있습니다. 이미 있다면 제거하고, 없다면 꽂을 수 있다. 이미 밝혀진 칸의 경우 아무 일도 일어나지 않습니다.

⑤ 패배한 경우 존재하는 모든 칸을 밝힙니다. 아래의 두 번째 그림과 같이 모든 칸이 밝혀져야 합니다.

⑥ 구현에 필요한 이미지 파일(웃는 사람, 해골, 선글라스, 폭탄, 깃발)은 imgs 폴더 안에 모두 존재하니 이를 사용하시기 바랍니다.

⑦ 이외의 디테일한 디자인은 평가 항목에 들어가지 않습니다.

Ex) 왼쪽부터 플레이 중, 패배 시, 성공 시의 결과

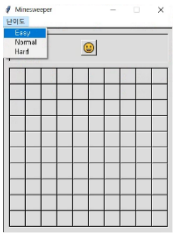
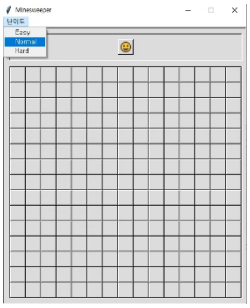
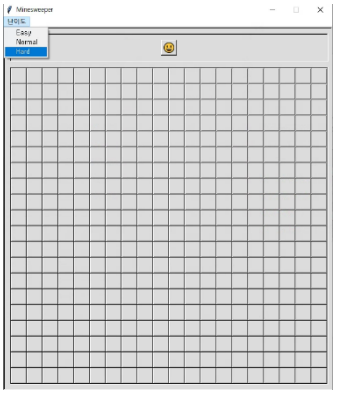
D. 상단 난이도 메뉴를 통해서 Easy, Normal, Hard로 바꿀 수 있도록 구현합니다. 난이도마다 지뢰의 개수와 보드의 크기가 변하도록 합니다.

① Easy : 10개의 지뢰, 10x10 board

② Normal : 30개의 지뢰, 15x15 board

③ Hard : 50개의 지뢰, 20x20 board

Ex) 상단에 난이도라는 메뉴가 존재하고, 왼쪽부터 해당 난이도에 따른 보드의 크기 변화를 보입니다. (Easy, Normal, Hard 순)

(주의 사항)

(1) 다시 한 번 강조하지만, 디자인은 점수에 반영되지 않습니다. 과도한 시간을 투자하지 마시기 바랍니다.

(2) 해당 과제에서는 두 개의 파일(model.py, app.py)이 존재하며, 두 개 모두 제출을 해야 합니다. 그 중 app.py에서는 GUI가 동작하지 않더라도, model.py의 Board, Panel, EmptyPanel, MinePanel의 구현만으로도 점수를 부여합니다.

[참고]

(1) Demo 동영상

https://drive.google.com/file/d/1cgOAmF7E-QWR1bSH3zkReoaXHUbBKH5R/view?usp=sharing

(2) GUI 참고 코드

구현에 도움이 될 수 있는 GUI code입니다. 하지만, 정답이 아니니 편한대로 구현하시면 됩니다.

https://github.com/euidong/minesweeper/blob/v0.0.1/gui_test.py