```
import tkinter as tk
import random
import copy

# Constants
GRID_SIZE = 4
CELL_SIZE = 100
PADDING = 10
BACKGROUND_COLOR = "#92877d"
EMPTY_CELL_COLOR = "#9e948a"
TILE_COLORS = {
    2: "#eee4da",
    4: "#ede0c8",
    8: "#f2b179",
    16: "#f59563",
    32: "#f67c5f",
    64: "#f65e3b",
    128: "#edcf72",
    256: "#edcc61",
    512: "#edc850",
    1024: "#edc53f",
    2048: "#edc22e",
}
TEXT_COLOR = "#776e65"
FONT = ("Helvetica", 40, "bold")

# Main 2048 Game Class
class Game2048:
    def _init_(self):
        self.window = tk.Tk()
        self.window.title("2048 Game")
        self.score = 0
        self.board = [[0] * GRID_SIZE for _ in range(GRID_SIZE)]
        self.history = []

        self.frame = tk.Frame(self.window, bg=BACKGROUND_COLOR)
        self.frame.grid(sticky="nsew")

        self.grid_cells = []
        for row in range(GRID_SIZE):
            row_cells = []
            for col in range(GRID_SIZE):
                cell_frame = tk.Frame(
                    self.frame, width=CELL_SIZE, height=CELL_SIZE, bg=EMPTY_CELL_COLOR
```

```
            )
            cell_frame.grid(row=row, column=col, padx=PADDING, pady=PADDING)
            label = tk.Label(self.frame, text="", bg=EMPTY_CELL_COLOR, font=FONT, width=4,
height=2)
            label.grid(row=row, column=col)
            row_cells.append(label)
        self.grid_cells.append(row_cells)

    # Game Controls
    self.window.bind("<Up>", self.move_up)
    self.window.bind("<Down>", self.move_down)
    self.window.bind("<Left>", self.move_left)
    self.window.bind("<Right>", self.move_right)
    self.window.bind("<u>", self.undo_move)  # 'u' for Undo

    self.start_game()
    self.window.mainloop()

  def start_game(self):
    # Initialize two random tiles on the board
    self.add_random_tile()
    self.add_random_tile()
    self.update_grid()

  def add_random_tile(self):
    empty_cells = [(i, j) for i in range(GRID_SIZE) for j in range(GRID_SIZE) if self.board[i][j]
== 0]
    if empty_cells:
      i, j = random.choice(empty_cells)
      self.board[i][j] = 2 if random.random() < 0.9 else 4

  def update_grid(self):
    for i in range(GRID_SIZE):
      for j in range(GRID_SIZE):
        value = self.board[i][j]
        if value == 0:
          self.grid_cells[i][j].configure(text="", bg=EMPTY_CELL_COLOR)
        else:
          self.grid_cells[i][j].configure(text=str(value), bg=TILE_COLORS.get(value,
"#ffcc00"))
    self.window.update_idletasks()

  def move_up(self, event):
    self.move(0, -1)
```

```
def move_down(self, event):
    self.move(0, 1)

def move_left(self, event):
    self.move(-1, 0)

def move_right(self, event):
    self.move(1, 0)

def move(self, dx, dy):
    if self.can_move(dx, dy):
        self.history.append(copy.deepcopy(self.board))
        self.board = self.merge_tiles(dx, dy)
        self.add_random_tile()
        self.update_grid()
        if self.check_win():
            self.show_game_over("You win!")
        elif not self.can_move_anywhere():
            self.show_game_over("Game Over!")

def merge_tiles(self, dx, dy):
    new_board = [[0] * GRID_SIZE for _ in range(GRID_SIZE)]
    for i in range(GRID_SIZE):
        row_or_col = [self.board[i][j] if dx == 0 else self.board[j][i] for j in range(GRID_SIZE)]
        merged = self.compress_and_merge(row_or_col)
        for j in range(GRID_SIZE):
            if dx == 0:
                new_board[i][j] = merged[j]
            else:
                new_board[j][i] = merged[j]
    return new_board

def compress_and_merge(self, line):
    new_line = [i for i in line if i != 0]
    for i in range(len(new_line) - 1):
        if new_line[i] == new_line[i + 1]:
            new_line[i] *= 2
            self.score += new_line[i]
            new_line[i + 1] = 0
    new_line = [i for i in new_line if i != 0]
    new_line.extend([0] * (GRID_SIZE - len(new_line)))
    return new_line
```

```
    def can_move(self, dx, dy):
        for i in range(GRID_SIZE):
            for j in range(GRID_SIZE):
                current = self.board[i][j]
                if current == 0:
                    continue
                ni, nj = i + dy, j + dx
                if 0 <= ni < GRID_SIZE and 0 <= nj < GRID_SIZE:
                    next_tile = self.board[ni][nj]
                    if next_tile == 0 or next_tile == current:
                        return True
        return False

    def can_move_anywhere(self):
        for dx, dy in [(0, 1), (1, 0), (0, -1), (-1, 0)]:
            if self.can_move(dx, dy):
                return True
        return False

    def check_win(self):
        for row in self.board:
            if 2048 in row:
                return True
        return False

    def show_game_over(self, message):
        game_over_frame = tk.Frame(self.frame, bg=BACKGROUND_COLOR)
        game_over_frame.place(relx=0.5, rely=0.5, anchor="center")
        tk.Label(game_over_frame, text=message, font=FONT,
bg=BACKGROUND_COLOR).pack()
        tk.Button(game_over_frame, text="Play Again", command=self.restart_game).pack()

    def restart_game(self):
        self._init_()

    def undo_move(self, event):
        if self.history:
            self.board = self.history.pop()
            self.update_grid()

# Running the Game
if _name_ == "_main_":
    Game2048()
```