```
#-*- coding:utf-8 -*-

import tkinter as tk
import random
import pandas as pd

brickData = {'col':12, 'row':3, 'width':50, 'height':25, 'color1':'saddlebrown', 
'color2':'rosybrown'}
paddleData = {'color':'black', 'width':100, 'height':5, 'speed':15}
ballData = {'color': 'red', 'size':20, 'speed':2}

class BrickApp(tk.Tk):
    def __init__(self):
        tk.Tk.__init__(self)

        self.geometry('700x650')
        self.resizable(width=False, height=False)
        self.title('블록 깨기')

        self.playingFrame = PlayingFrame(self)
        self.playingFrame.pack()

        self.controlFrame = ControlFrame(self)
        self.controlFrame.pack()

        #self.playingFrame.startgame()

class PlayingFrame(tk.Frame):
    def __init__(self, parent):
        tk.Frame.__init__(self, parent)
        self.pack()

        self.shapeobjects = {}
        self.count = 0
        self.score = tk.Label(self, text='블록 갯수  : ' + str(self.count), padx=10, pady=10)
        self.score.pack()

        self.playground = tk.Canvas(self, background='white', width=600, height=550)
        self.playground.pack()

        self.playground.create_text(300, 200, font=('Arial', 14), text='게임을 시작하려면 시작 
버튼을 선택하세요!', justify=tk.CENTER)

        self.brick = None
        self.paddle = None
        self.ball = None


    def startgame(self):
        self.shapeobjects = {}
        self.playground.delete(tk.ALL)

        self.boundary = Boundary(self)          # 상단 및 좌우 벽에 대한 객체
        self.brick = Brick(self)
        self.paddle = Paddle(self)
        self.ball = Ball(self)

        self.bind('<Left>', self.arrowKeyPressed)
        self.bind('<Right>', self.arrowKeyPressed)
        self.focus_set()

        flag1 = False
        flag2 = False

        while not flag1 and not flag2:
            self.moveball()
            flag1 = self.checkboundary()
            flag2 = self.checkoverlap()
    def arrowKeyPressed(self, event):
        if event.keysym == 'Left' and self.paddle.position['x']>=5 :
            paddleData['speed'] = -5
        elif event.keysym=='Right' and self.paddle.position['x']+paddleData['width'] <= 595 :
            paddleData['speed'] = 5
        else :
            paddleData['speed'] = 0

        self.paddle.position['x'] += paddleData['speed']
        self.playground.move('paddle', paddleData['speed'], 0)
        self.playground.update()

    def moveball(self):
        self.playground.move('ball', self.ball.dx, self.ball.dy)
        self.playground.after(20)
        self.playground.update()


    # 삭제 대상 메쏘드 --> checkoverlap 메쏘드와 통합
    def checkboundary(self):
        endflag = False
        rightBound = 600 - ballData['size']
        bottomBound = 550

        newX = self.ball.position['x'] + self.ball.dx
        newY = self.ball.position['y'] + self.ball.dy

        if newX<0 :
            self.ball.position['x'] = 0
            self.ball.dx = -self.ball.dx
        elif newX>rightBound :
            self.ball.position['x'] = rightBound
            self.ball.dx = -self.ball.dx
        else :
            self.ball.position['x'] = newX

        if newY<0 :
            self.ball.position['y'] = 0
            self.ball.dy = -self.ball.dy
        elif newY >= bottomBound :
            self.playground.create_text(300, 200, font=('Arial', 14), text='게임 종료 !',
fill='red', justify=tk.CENTER)
            endflag = True
        else :
            self.ball.position['y'] = newY

        return endflag

    def checkoverlap(self):
        endflag = False
        tmp = self.playground.coords(self.ball.gameball)
        x1=tmp[0]
        y1=tmp[1]
        x2=tmp[2]
        y2=tmp[3]

        overlapped_list = self.playground.find_overlapping(x1, y1, x2, y2)

        for k,v in self.shapeobjects.items():
            if v in overlapped_list :
                self.ball.dy = -self.ball.dy

                if k != 'paddle':
                    self.playground.delete(self.shapeobjects[k])
                    self.count += 1

                    if len(self.shapeobjects.keys()) == 1 :
```

```
                    endflag = True

        self.score.config(text='블록 갯수  : ' + str(self.count))
        return endflag


# 구현 대상 클래스
class Boundary:
    def __init__(self, frame):

        # 상단, 좌우 벽에 대한 객체를 생성하여 playingframe의 shapeobjects에 추가
        # 볼 객체와의 overlapping여부를 확인할 때 사용




class Brick:
    def __init__(self, frame):
        for x in range(brickData['col']):
            for y in range(brickData['row']+1):
                name = 'brick'+str(x) + str(y)
                startpoint = [x*brickData['width'], 50+y*brickData['height']]
                endpoint = [(x+1)*brickData['width'], 50+(y+1)*brickData['height']]
                if (x+y)%2 == 0 :
                    frame.shapeobjects[name] = frame.playground.create_rectangle(startpoint[0],
startpoint[1], endpoint[0], endpoint[1],

outline='black', fill=brickData['color1'])
                else :
                    frame.shapeobjects[name] = frame.playground.create_rectangle(startpoint[0],
startpoint[1], endpoint[0], endpoint[1],

outline='black', fill=brickData['color2'])

class Paddle:
    def __init__(self, frame):
        self.position = {'x':250, 'y':540}

        frame.shapeobjects['paddle'] = frame.playground.create_rectangle(self.position['x'],
self.position['y'],
                                           self.position['x']+paddleData['width'],
self.position['y']+paddleData['height'],

                                           fill = paddleData['color'], tag='paddle')

class Ball:
    def __init__(self, frame):
        self.position = {}
        self.position['x'] = random.randint(0, 600)
        self.position['y'] = 150

        #self.dx = random.randint(-1, 1)*ballData['speed']
        #self.dy = random.randint(-1, 1)*ballData['speed']

        self.dx = 5 ; self.dy = 5

        self.gameball = frame.playground.create_oval(self.position['x'], self.position['y'],
                                                    self.position['x']+ballData['size'],
self.position['y']+ballData['size'],

                                                    fill = ballData['color'], tag='ball')


class ControlFrame(tk.Frame):
    def __init__(self, parent):
        tk.Frame.__init__(self, parent)
        self.pack()
        self.parent = parent
        self.playBtn = tk.Button(self, text='시 작', width=10, pady=5,
command=self.parent.playingFrame.startgame)
        self.playBtn.pack(side=tk.LEFT)

        self.exitBtn = tk.Button(self, text='종 료', width=10, pady=5, command=self.parent.quit)
        self.exitBtn.pack(side=tk.LEFT)


def main():
    myapp = BrickApp()
    myapp.mainloop()

main()
```