**CODE BÀI TẬP LỚN**

**Blackjack.py**

import tkinter as tk

from tkinter import messagebox, simpledialog

from cards import Deck

from games import Game

import os

from PIL import Image, ImageTk

class BlackjackGUI(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.title("Blackjack")

self.configure(bg="white") # Đổi màu nền cửa sổ thành trắng

self.deck = Deck()

self.game = Game(self.deck)

# Thống kê

self.win\_count = 0

self.lose\_count = 0

self.draw\_count = 0

self.balance = 1000

self.bet = 0

self.history = []

# Style options

self.card\_bg = "white" # Đổi màu nền lá bài thành trắng

self.card\_fg = "#ffd369"

self.label\_fg = "#222831" # Đổi màu chữ cho dễ nhìn trên nền trắng

self.btn\_bg = "#222831"

self.btn\_fg = "#ffd369"

self.status\_fg = "#00adb5"

self.font\_main = ("Segoe UI", 14, "bold")

self.font\_card = ("Consolas", 16, "bold")

# Menu hướng dẫn

menubar = tk.Menu(self)

helpmenu = tk.Menu(menubar, tearoff=0)

helpmenu.add\_command(label="Luật chơi", command=self.show\_rules)

menubar.add\_cascade(label="Trợ giúp", menu=helpmenu)

self.config(menu=menubar)

# Frame Dealer

tk.Label(self, text="Dealer's Hand", font=self.font\_main, fg=self.label\_fg, bg="white").pack(pady=(10,0))

self.dealer\_frame = tk.Frame(self, bg="white")

self.dealer\_frame.pack(pady=5)

# Frame Player

tk.Label(self, text="Your Hand", font=self.font\_main, fg=self.label\_fg, bg="white").pack(pady=(20,0))

self.player\_frame = tk.Frame(self, bg="white")

self.player\_frame.pack(pady=5)

# Status

self.status = tk.Label(self, text="Game started. Hit or Stand?", font=("Segoe UI", 12, "bold"), fg=self.status\_fg, bg="white")

self.status.pack(pady=10)

# Thông tin cược và thống kê

self.info\_frame = tk.Frame(self, bg="white")

self.info\_frame.pack(pady=5)

self.balance\_label = tk.Label(self.info\_frame, text=f"Balance: ${self.balance}", font=("Segoe UI", 12), fg="#00adb5", bg="white")

self.balance\_label.grid(row=0, column=0, padx=10)

self.bet\_label = tk.Label(self.info\_frame, text=f"Bet: ${self.bet}", font=("Segoe UI", 12), fg="#ffd369", bg="white")

self.bet\_label.grid(row=0, column=1, padx=10)

self.score\_label = tk.Label(self.info\_frame, text=self.get\_score\_text(), font=("Segoe UI", 12), fg="#222831", bg="white")

self.score\_label.grid(row=0, column=2, padx=10)

# Buttons

self.button\_frame = tk.Frame(self, bg="white")

self.button\_frame.pack(pady=10)

self.bet\_button = tk.Button(self.button\_frame, text="Bet", command=self.place\_bet, font=self.font\_main, bg="#00adb5", fg="#222831", width=8, relief="raised", bd=3)

self.bet\_button.grid(row=0, column=0, padx=10)

self.hit\_button = tk.Button(self.button\_frame, text="Hit", command=self.hit, font=self.font\_main, bg=self.btn\_bg, fg=self.btn\_fg, width=8, relief="raised", bd=3, activebackground="#393e46", state=tk.DISABLED)

self.hit\_button.grid(row=0, column=1, padx=10)

self.stand\_button = tk.Button(self.button\_frame, text="Stand", command=self.stand, font=self.font\_main, bg=self.btn\_bg, fg=self.btn\_fg, width=8, relief="raised", bd=3, activebackground="#393e46", state=tk.DISABLED)

self.stand\_button.grid(row=0, column=2, padx=10)

self.restart\_button = tk.Button(self.button\_frame, text="Restart", command=self.restart, font=self.font\_main, bg="#ffd369", fg="#222831", width=8, relief="raised", bd=3, activebackground="#393e46")

self.restart\_button.grid(row=0, column=3, padx=10)

self.history\_button = tk.Button(self.button\_frame, text="History", command=self.show\_history, font=self.font\_main, bg="#393e46", fg="#ffd369", width=8, relief="raised", bd=3)

self.history\_button.grid(row=0, column=4, padx=10)

self.update\_display()

def get\_score\_text(self):

return f"Wins: {self.win\_count} Losses: {self.lose\_count} Draws: {self.draw\_count}"

def get\_card\_image(self, card):

rank\_map = {

'A': 'ace', 'J': 'jack', 'Q': 'queen', 'K': 'king',

'10': '10', '9': '9', '8': '8', '7': '7', '6': '6',

'5': '5', '4': '4', '3': '3', '2': '2'

}

suit\_map = {

'♠': 'spades', '♥': 'hearts', '♦': 'diamonds', '♣': 'clubs'

}

rank = rank\_map[card.rank]

suit = suit\_map[card.suit]

filename = os.path.join("cards", f"{rank}\_of\_{suit}.png")

if not os.path.exists(filename):

filename = os.path.join("cards", "back.png") # Ảnh mặc định nếu thiếu

img = Image.open(filename).resize((60, 90))

return ImageTk.PhotoImage(img)

def update\_display(self):

# Xóa các widget cũ

for widget in self.player\_frame.winfo\_children():

widget.destroy()

for widget in self.dealer\_frame.winfo\_children():

widget.destroy()

# Hiển thị bài người chơi

self.player\_card\_imgs = []

for card in self.game.player.cards:

img = self.get\_card\_image(card)

self.player\_card\_imgs.append(img)

tk.Label(self.player\_frame, image=img, bg=self.card\_bg).pack(side=tk.LEFT, padx=2)

# Hiển thị bài dealer

self.dealer\_card\_imgs = []

if self.bet == 0:

# Chưa đặt cược, ẩn toàn bộ bài dealer

for \_ in self.game.dealer.cards:

try:

back\_img = ImageTk.PhotoImage(Image.open(os.path.join("cards", "back.png")).resize((60, 90)))

except:

back\_img = None

self.dealer\_card\_imgs.append(back\_img)

tk.Label(self.dealer\_frame, image=back\_img, bg=self.card\_bg).pack(side=tk.LEFT, padx=2)

elif self.hit\_button['state'] == tk.NORMAL:

# Đang chơi, chỉ lật 1 lá đầu

if self.game.dealer.cards:

img = self.get\_card\_image(self.game.dealer.cards[0])

self.dealer\_card\_imgs.append(img)

tk.Label(self.dealer\_frame, image=img, bg=self.card\_bg).pack(side=tk.LEFT, padx=2)

for \_ in self.game.dealer.cards[1:]:

try:

back\_img = ImageTk.PhotoImage(Image.open(os.path.join("cards", "back.png")).resize((60, 90)))

except:

back\_img = None

self.dealer\_card\_imgs.append(back\_img)

tk.Label(self.dealer\_frame, image=back\_img, bg=self.card\_bg).pack(side=tk.LEFT, padx=2)

else:

# Kết thúc ván, lật toàn bộ bài dealer

for card in self.game.dealer.cards:

img = self.get\_card\_image(card)

self.dealer\_card\_imgs.append(img)

tk.Label(self.dealer\_frame, image=img, bg=self.card\_bg).pack(side=tk.LEFT, padx=2)

self.balance\_label.config(text=f"Balance: ${self.balance}")

self.bet\_label.config(text=f"Bet: ${self.bet}")

self.score\_label.config(text=self.get\_score\_text())

def hand\_str(self, hand):

return ' '.join(str(card) for card in hand.cards)

def place\_bet(self):

if self.bet > 0:

messagebox.showinfo("Bet", "Bạn đã đặt cược rồi!")

return

bet = simpledialog.askinteger("Bet", f"Balance: ${self.balance}\nNhập số tiền cược:", minvalue=1, maxvalue=self.balance)

if bet:

self.bet = bet

self.balance -= bet

self.status.config(text="Bet placed. Hit or Stand?", fg="#ffd369")

self.hit\_button.config(state=tk.NORMAL)

self.stand\_button.config(state=tk.NORMAL)

self.bet\_button.config(state=tk.DISABLED)

self.update\_display()

def hit(self):

self.game.hit(self.game.player)

self.update\_display()

if self.game.player.value() > 21:

self.end\_game("You busted!")

elif self.game.player.value() == 21:

self.end\_game("Blackjack!")

def stand(self):

while self.game.dealer.value() < 17:

self.game.hit(self.game.dealer)

self.reveal\_dealer()

p\_val = self.game.player.value()

d\_val = self.game.dealer.value()

if d\_val > 21 or p\_val > d\_val:

self.end\_game("You win!")

elif p\_val == d\_val:

self.end\_game("Draw!")

else:

self.end\_game("Dealer wins.")

def reveal\_dealer(self):

self.update\_display()

def end\_game(self, result):

self.reveal\_dealer()

self.status.config(text=result)

self.hit\_button.config(state=tk.DISABLED)

self.stand\_button.config(state=tk.DISABLED)

self.bet\_button.config(state=tk.NORMAL)

# Cập nhật thống kê và tiền

if result == "You win!":

self.win\_count += 1

self.balance += self.bet \* 2

elif result == "Draw!":

self.draw\_count += 1

self.balance += self.bet

else:

self.lose\_count += 1

self.history.append(f"Bet: ${self.bet} - {result} (Balance: ${self.balance})")

self.bet = 0

self.deck = Deck()

self.game = Game(self.deck)

self.update\_display()

messagebox.showinfo("Game Over", result)

def restart(self):

self.deck = Deck()

self.game = Game(self.deck)

self.status.config(text="Game started. Place your bet!", fg="#00adb5")

self.hit\_button.config(state=tk.DISABLED)

self.stand\_button.config(state=tk.DISABLED)

self.bet\_button.config(state=tk.NORMAL)

self.bet = 0

self.update\_display()

def show\_history(self):

if not self.history:

messagebox.showinfo("History", "Chưa có lịch sử ván chơi.")

else:

messagebox.showinfo("History", "\n".join(self.history[-10:]))

def show\_rules(self):

rules = (

"Luật chơi Blackjack:\n"

"- Mục tiêu: Có tổng điểm gần 21 nhất nhưng không vượt quá 21.\n"

"- J, Q, K tính 10 điểm. A tính 1 hoặc 11 điểm.\n"

"- Đặt cược, sau đó chọn Hit (rút bài) hoặc Stand (dừng).\n"

"- Nếu bạn vượt quá 21 điểm, bạn thua.\n"

"- Dealer rút đến khi >= 17 điểm.\n"

"- So điểm để xác định thắng/thua/hòa.\n"

"- Nếu thắng, bạn nhận lại gấp đôi tiền cược."

)

messagebox.showinfo("Luật chơi", rules)

if \_\_name\_\_ == "\_\_main\_\_":

app = BlackjackGUI()

app.mainloop()

**Cards.py**

import random

class Card:

RANKS = ['A'] + [str(n) for n in range(2, 11)] + ['J', 'Q', 'K']

SUITS = ['♠', '♥', '♦', '♣']

def \_\_init\_\_(self, rank, suit):

self.rank = rank

self.suit = suit

def \_\_str\_\_(self):

return f'{self.rank}{self.suit}'

def value(self):

if self.rank in ['J', 'Q', 'K']:

return 10

if self.rank == 'A':

return 11

return int(self.rank)

class Deck:

def \_\_init\_\_(self):

self.reset()

def reset(self):

self.cards = [Card(rank, suit) for suit in Card.SUITS for rank in Card.RANKS]

random.shuffle(self.cards)

def deal(self):

if not self.cards:

self.reset()

return self.cards.pop() if self.cards else None

**Games.py**

from cards import Deck, Card

class Hand:

def \_\_init\_\_(self):

self.cards = []

def add(self, card):

self.cards.append(card)

def value(self):

val = sum(card.value() for card in self.cards)

num\_aces = sum(1 for card in self.cards if card.rank == 'A')

while val > 21 and num\_aces:

val -= 10

num\_aces -= 1

return val

def \_\_str\_\_(self):

return ' '.join(str(card) for card in self.cards)

class Game:

def \_\_init\_\_(self, deck):

self.deck = deck

self.player = Hand()

self.dealer = Hand()

self.deal\_initial()

def deal\_initial(self):

for \_ in range(2):

card\_p = self.deck.deal()

card\_d = self.deck.deal()

if card\_p: self.player.add(card\_p)

if card\_d: self.dealer.add(card\_d)

def hit(self, hand):

card = self.deck.deal()

if card:

hand.add(card)

def reset(self):

self.deck.reset()

self.player = Hand()

self.dealer = Hand()

self.deal\_initial()