

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ



**BÀI TẬP KẾT THÚC MÔN HỌC
PYTHON**

NGÀNH : KỸ THUẬT MÁY TÍNH

HỆ : ĐẠI HỌC CHÍNH QUY

GIÁO VIÊN HƯỚNG DẪN : NGUYỄN VĂN HUY

HỌ VÀ TÊN SINH VIÊN : NGUYỄN ĐỨC VIỆT

MSSV : K225480106075

LỚP : K58KTP

THÁI NGUYÊN – 2025

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

Bộ môn: Công nghệ Thông tin.

BÀI TẬP KẾT THÚC MÔN HỌC

MÔN HỌC

LẬP TRÌNH PYTHON

Sinh viên: Nguyễn Đức Việt

Lớp: K58KTP

Giáo viên GIÁNG DẠY: Nguyễn Văn Huy

Link GitHub: https://github.com/Viet6824/Bai_tap_Python

Link Youtube: <https://www.youtube.com/watch?v=rO3Dyht-wfY>

Thái Nguyên – 2025

**TRƯỜNG ĐHKT&CN
KHOA ĐIỆN TỬ**

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc**

BÀI TẬP KẾT THÚC MÔN HỌC

MÔN HỌC: LẬP TRÌNH PYTHON

BỘ MÔN : CÔNG NGHỆ THÔNG TIN

Sinh viên: Nguyễn Đức Việt

Lớp: K58KTP Ngành: Kỹ thuật máy tính

Giáo viên hướng dẫn: Nguyễn Văn Huy

Ngày giao bài: 15/05/2025

Ngày hoàn thành: 09/06/2025

Tên đề tài : Xây game Blackjack (Chapter 9) với GUI: chia bài, hit/stand, tính điểm.

Yêu cầu :

- Sử dụng class *BJ_Card*, *BJ_Deck*, *BJ_Hand*, *BJ_Game*.
- GUI cập nhật khi rút bài.
- Xử lý bust (> 21) và so sánh với dealer.

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

Bài tập Python

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Thái Nguyên, ngày....tháng.....năm 20....

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

Mục lục

Lời mở đầu.....	6
Chương 1. Giới thiệu đầu bài.....	7
1.1. Đề bài và mục tiêu	7
1.2. Tính năng của chương trình	8
1.3. Thách thức và khó khăn	8
1.4. Kiến thức áp dụng	9
Chương 2. Cơ sở lý thuyết.....	10
2.1. Lập trình hướng đối tượng (OOP).....	10
2.2. Cấu trúc dữ liệu - List và Dictionary.....	11
2.3. Thư viện Tkinter	11
2.4. Logic xử lý game Blackjack.....	12
2.5. Quản lý sự kiện và cập nhật giao diện.....	13
2.6. Tối ưu hóa hiệu suất	13
2.7. Lịch sử và ý nghĩa của Blackjack.....	13
Chương 3. Thiết kế và xây dựng chương trình	14
3.1. Sơ đồ khái hệ thống	14
3.2. Sơ đồ khái các thuật toán chính	17
3.3. Cấu trúc dữ liệu	19
3.4. Chương trình.....	20
3.5. Quy trình phát triển	24
3.6. Chương trình.....	25
3.7. Đánh giá thiết kế	33
Chương 4. Thực nghiệm và kết luận	35
4.1. Thực nghiệm.....	35
4.2. Kết luận	36
4.3. Đề xuất mở rộng	37
Kết luận	41

Lời mở đầu

Bài tập lớn này là một phần trong chương trình học lập trình, với mục tiêu giúp sinh viên thực hành các kỹ năng lập trình hướng đối tượng, xây dựng giao diện người dùng, và áp dụng các thuật toán logic vào một dự án thực tế. Mình đã chọn chủ đề xây dựng trò chơi Blackjack (Xì dách) với giao diện đồ họa (GUI) để vừa đáp ứng yêu cầu của bài tập, vừa tạo ra một sản phẩm giải trí thú vị. Trong quá trình thực hiện, mình không chỉ học được cách tổ chức mã nguồn, xử lý logic phức tạp, mà còn rèn luyện kỹ năng viết báo cáo và trình bày ý tưởng một cách khoa học.

Báo cáo này sẽ trình bày chi tiết các bước từ khi nhận đề bài, phân tích yêu cầu, thiết kế hệ thống, đến triển khai và kiểm thử sản phẩm. Mình cũng sẽ chia sẻ những khó khăn gặp phải, bài học kinh nghiệm, và các hướng cải tiến trong tương lai. Hy vọng báo cáo này không chỉ đáp ứng yêu cầu của giảng viên mà còn là tài liệu tham khảo hữu ích cho các bạn cùng lớp.

Chương 1. Giới thiệu đầu bài

1.1. Đề bài và mục tiêu

Bài tập lớn được giao bởi giảng viên với mục tiêu xây dựng một trò chơi Blackjack (Xì dách) sử dụng giao diện người dùng đồ họa (GUI) đơn giản, dựa trên các quy tắc cơ bản của trò chơi bài nổi tiếng này. Yêu cầu cụ thể bao gồm việc tích hợp các tính năng chính như sau:

- **Chia bài:** Chia 2 lá bài ban đầu cho cả người chơi và nhà cái.
- **Tương tác người chơi:** Cung cấp các tùy chọn rút bài (Hit) hoặc dừng lại (Stand) thông qua giao diện GUI.
- **Tính điểm:** Tính điểm theo quy tắc của Blackjack, trong đó lá Át có thể tính là 1 hoặc 11, các lá từ 2 đến 10 giữ nguyên giá trị, và J, Q, K tính là 10.
- **Xử lý bust:** Phát hiện trường hợp vượt quá 21 điểm (bust) và thông báo kết quả.
- **So sánh điểm:** So sánh điểm số giữa người chơi và nhà cái để xác định thắng, thua, hoặc hòa.
- **Hiển thị:** Hiển thị thông tin lá bài, điểm số, và kết quả trên giao diện GUI, với yêu cầu giao diện phải cập nhật theo thời gian thực mỗi khi có hành động từ người chơi.

Ngoài ra, một yêu cầu quan trọng là phải sử dụng các lớp lập trình hướng đối tượng (BJ_Card, BJ_Deck, BJ_Hand, BJ_Game) để tổ chức mã nguồn một cách khoa học, giúp dễ dàng bảo trì và mở rộng. Một điểm đặc biệt nữa là mình đã điều chỉnh ngôn ngữ hiển thị sang tiếng Việt theo giọng miền Bắc (ví dụ: "Rút bài" thay vì "Hit", "Xì dách rồi!" thay vì "Blackjack!") để làm cho sản phẩm gần gũi hơn với người dùng Việt Nam.

1.2. Tính năng của chương trình

Chương trình không chỉ là một trò chơi giải trí mà còn bao gồm các tính năng sau:

- **Chia bài ngẫu nhiên:** Bộ bài 52 lá được xáo trộn ngẫu nhiên để đảm bảo tính công bằng.
- **Tính điểm thông minh:** Xử lý lá Át linh hoạt (1 hoặc 11) để tối ưu điểm số mà không bị bust.
- **Giao diện trực quan:** Sử dụng Tkinter để hiển thị lá bài, điểm số, và thông báo kết quả.
- **Tương tác mượt mà:** Nút "Rút bài" và "Dừng lại" cho phép người chơi tương tác dễ dàng, với giao diện cập nhật tức thời.
- **Thông báo kết quả:** Hiển thị kết quả thắng, thua, hoặc hòa thông qua hộp thoại, với ngôn ngữ tự nhiên và thân thiện.

1.3. Thách thức và khó khăn

Trong quá trình thực hiện, mình đã gặp phải nhiều thách thức đáng kể:

- **Logic tính điểm phức tạp:** Việc xử lý lá Át (có thể là 1 hoặc 11) đòi hỏi thuật toán linh hoạt để tránh vượt quá 21 điểm, đồng thời đảm bảo tính chính xác trong mọi trường hợp.
- **Đồng bộ giao diện GUI:** Giao diện phải cập nhật ngay lập tức sau mỗi hành động của người chơi, đòi hỏi sự kết hợp chặt chẽ giữa logic trò chơi và các thành phần GUI.
- **Tối ưu mã nguồn:** Làm thế nào để mã nguồn dễ đọc, dễ bảo trì, và không bị trùng lặp logic là một bài toán cần cân nhắc kỹ lưỡng.

1.4. Kiến thức áp dụng

Để hoàn thành bài tập, mình đã áp dụng nhiều kiến thức và kỹ năng từ các môn học trước:

- **Lập trình hướng đối tượng (OOP):** Sử dụng các lớp để mô phỏng các thành phần của trò chơi.
- **Cấu trúc dữ liệu:** Quản lý bộ bài và tay bài bằng list, sử dụng dictionary để ánh xạ tên lá bài sang tiếng Việt.
- **Lập trình giao diện người dùng (GUI):** Sử dụng Tkinter để xây dựng giao diện và xử lý sự kiện.
- **Thuật toán và logic:** Xây dựng thuật toán tính điểm, xử lý bust, và so sánh điểm giữa người chơi và nhà cái.
- **Kỹ năng phân tích và thiết kế:** Phân tích yêu cầu, thiết kế hệ thống, và viết báo cáo để trình bày kết quả.

Chương 2. Cơ sở lý thuyết

2.1. Lập trình hướng đối tượng (OOP)

Lập trình hướng đối tượng (OOP) là một phương pháp lập trình hiện đại, giúp tổ chức mã nguồn theo các đối tượng thực tế, trong trường hợp này là các thành phần của trò chơi Blackjack. Mỗi đối tượng được mô phỏng bằng một lớp, với các thuộc tính và phương thức riêng. Cụ thể:

- Lớp BJ_Card: Đại diện cho một lá bài, với thuộc tính suit (chất: Hearts, Diamonds, Clubs, Spades) và value (số: Ace, 2, 3, ..., King), cùng phương thức get_vietnamese_name() để chuyển đổi tên lá bài sang định dạng với emoji (ví dụ: "Ace of Hearts" thành "Át ❤️"), sử dụng ký hiệu quốc tế (❤️ , ♦ , ♣ , ♠) để tăng tính trực quan.
- Lớp BJ_Deck: Đại diện cho bộ bài 52 lá, với phương thức deal() để phát bài và xáo trộn ngẫu nhiên thông qua random.shuffle().
- Lớp BJ_Hand: Đại diện cho tay bài của người chơi hoặc dealer, với phương thức add_card() để thêm bài, get_value() để tính điểm, và get_vietnamese_string() để hiển thị tay bài với emoji (ví dụ: "Át ❤️ , J ♠").
- Lớp BJ_Game: Quản lý toàn bộ logic trò chơi, bao gồm chia bài ban đầu (initial_deal()), xử lý rút bài (hit()), và logic lượt chơi của dealer (dealer_play()). OOP mang lại nhiều lợi ích như tính đóng gói (encapsulation) giúp bảo vệ dữ liệu, tính kế thừa (inheritance) để tái sử dụng mã nguồn, và tính đa hình (polymorphism) để linh hoạt trong việc mở rộng. Trong chương trình này, mình đã tận dụng tính đóng gói để đảm bảo các lớp hoạt động độc lập nhưng phối hợp nhịp nhàng, mặc dù không áp dụng kế thừa hoặc đa hình trong phạm vi bài tập này.

2.2. Cấu trúc dữ liệu - List và Dictionary

Cấu trúc dữ liệu đóng vai trò quan trọng trong việc lưu trữ và quản lý thông tin của trò chơi:

- List: Được sử dụng để lưu trữ tập hợp các lá bài trong BJ_Deck và BJ_Hand. Trong BJ_Deck, list ban đầu chứa 52 lá bài và giảm dần mỗi khi phát bài. Trong BJ_Hand, list lưu trữ các lá bài trong tay, ví dụ: ["Át ❤️", "J ♦️"]. List hỗ trợ các thao tác như thêm (append), xóa (pop), và duyệt qua các phần tử, rất phù hợp cho việc quản lý động các lá bài trong suốt ván chơi.
- Dictionary: Được sử dụng trong BJ_Card để ánh xạ tên lá bài và chất sang định dạng với emoji, giúp hiển thị giao diện trực quan. Ví dụ: dictionary rank_map ánh xạ "Ace" thành "Át", và suit_map ánh xạ "Hearts" thành "❤️". Dictionary cho phép truy xuất nhanh và dễ dàng mở rộng nếu cần thêm định dạng khác.

2.3. Thư viện Tkinter

Thư viện Tkinter là công cụ chính để xây dựng giao diện đồ họa (GUI) trong Python. Nó cung cấp các thành phần giao diện như:

- Frame: Tạo khung chứa thông tin, ví dụ: khung cho người chơi (player_frame) và dealer (dealer_frame).
- Label: Hiển thị thông tin như tên lá bài ("Player: Át ❤️, J ♦️") và điểm số ("Score: 21").
- Button: Tạo nút bấm như "Hit", "Stand", và "New Game" để người chơi tương tác.

- Messagebox: Hiển thị thông báo kết quả (ví dụ: "Over 21! You lose."). Tkinter hoạt động theo mô hình lập trình hướng sự kiện (event-driven programming), nghĩa là chương trình sẽ phản hồi các hành động của người dùng như nhấn nút. Mình đã sử dụng các phương thức command để liên kết các nút bấm với các hàm xử lý như hit() và stand(), đảm bảo phản hồi sự kiện một cách hiệu quả.

2.4. Logic xử lý game Blackjack

Quy tắc của Blackjack khá đặc thù và đòi hỏi sự chính xác trong việc lập trình:

- Tính điểm:
 - Lá từ 2 đến 10 giữ nguyên giá trị.
 - Lá J, Q, K tính là 10 điểm.
 - Lá Át có thể tính là 11 hoặc 1, tùy vào tổng điểm. Ví dụ: Nếu tay bài có "Át " và "10 ", tổng là 21 (Blackjack). Nhưng nếu thêm "5 ", tổng là 26, lúc này Át sẽ được tính là 1, thành 16.
- Quy tắc nhà cái: Nhà cái phải rút bài nếu điểm < 17 , và dừng lại nếu điểm ≥ 17 .
- Kết quả:
 - Nếu người chơi vượt 21 điểm, họ thua ngay lập tức (bust).
 - Nếu không, sau khi dừng lại, so sánh điểm với dealer: người chơi thắng nếu điểm cao hơn mà không vượt 21, thua nếu thấp hơn, hoặc hòa nếu bằng nhau.
 - Trường hợp đặc biệt: Nếu người chơi đạt 21 điểm ngay từ 2 lá đầu (Blackjack), họ thắng ngay.

2.5. Quản lý sự kiện và cập nhật giao diện

Trong Tkinter, sự kiện được quản lý thông qua vòng lặp chính (mainloop()). Các phương thức như hit() và stand() được gọi khi người dùng nhấn nút, sau đó giao diện được cập nhật qua update_display(). Điều này đòi hỏi sự đồng bộ giữa logic trò chơi (tính điểm, bust) và giao diện (hiển thị lá bài, điểm số) để tránh lỗi hiển thị, được thực hiện bằng cách gọi update_display() sau mỗi hành động quan trọng.

2.6. Tối ưu hóa hiệu suất

Mặc dù chương trình không quá phức tạp, mình đã chú ý đến việc:

- Giảm thiểu vòng lặp không cần thiết trong tính điểm bằng cách gọi get_value() một lần trong mỗi cập nhật.
- Sử dụng các phương thức hiệu quả như pop() trong deal() để lấy lá bài mà không cần duyệt toàn bộ list.
- Tối ưu hóa việc hiển thị giao diện bằng cách chỉ cập nhật các nhãn (Label) cần thiết sau mỗi hành động, đảm bảo giao diện phản hồi nhanh chóng.

2.7. Lịch sử và ý nghĩa của Blackjack

Blackjack là một trò chơi bài có nguồn gốc từ thế kỷ 17 tại Pháp, với tên gọi ban đầu là "Vingt-et-Un" (21). Trò chơi này sau đó lan rộng ra khắp thế giới, đặc biệt phổ biến tại các sòng bạc ở Mỹ. Blackjack không chỉ là trò chơi giải trí mà còn đòi hỏi sự tính toán và chiến thuật, giúp người chơi rèn luyện tư duy logic và khả năng ra quyết định. Trong bối cảnh bài tập này, việc tái hiện Blackjack dưới dạng ứng dụng giúp mình hiểu sâu hơn về cách một trò chơi truyền thống có thể được số hóa và tích hợp vào giao diện đồ họa.

Chương 3. Thiết kế và xây dựng chương trình

Chương này trình bày chi tiết quá trình thiết kế và triển khai chương trình game Blackjack với giao diện người dùng đồ họa (GUI). Mục tiêu là xây dựng một hệ thống có cấu trúc rõ ràng, dễ bảo trì, và đáp ứng đầy đủ các yêu cầu đã nêu trong đề bài. Quá trình thiết kế bao gồm việc xác định các module chính, xây dựng sơ đồ khái thuật toán, định nghĩa cấu trúc dữ liệu, triển khai mã nguồn, và phân tích quy trình phát triển. Dưới đây là các phần chi tiết.

3.1. Sơ đồ khái niệm

Phần này mô tả tổng quan về hệ thống, bao gồm các module chính và mối quan hệ giữa chúng, cùng với sơ đồ phân cấp chức năng đã được trình bày trước đó.

3.1.1. Các module chính của hệ thống

Hệ thống game Blackjack được chia thành các module chính sau, mỗi module đảm nhiệm một chức năng riêng biệt và phối hợp với nhau để tạo nên trò chơi hoàn chỉnh:

- **BJ_Card:** Module này chịu trách nhiệm quản lý thông tin của từng lá bài. Mỗi lá bài được biểu diễn bằng hai thuộc tính: suit (chất: Hearts, Diamonds, Clubs, Spades) và value (số: Ace, 2, 3, ..., King). Module cung cấp phương thức `get_vietnamese_name()` để chuyển đổi tên lá bài sang định dạng với emoji (ví dụ: "Ace of Hearts" thành "Át ❤️"), sử dụng ký hiệu quốc tế (❤️ , ♦ , ♣ , ♠) để tăng tính trực quan. Đây là module cơ bản nhất, đóng vai trò nền tảng cho các module khác.
- **BJ_Deck:** Module này quản lý bộ bài gồm 52 lá. Nó khởi tạo tất cả các lá bài bằng cách kết hợp các chất và số, sau đó xáo trộn ngẫu nhiên bằng hàm

random.shuffle(). Module cung cấp phương thức deal() để phát bài, lấy một lá từ bộ bài và giảm số lượng lá còn lại.

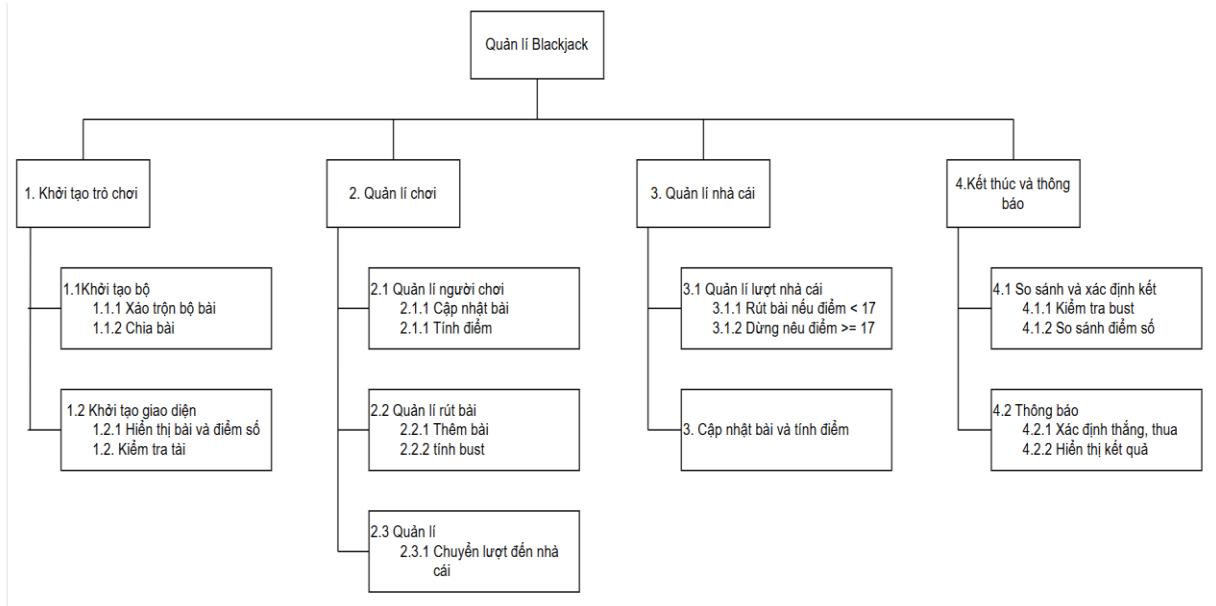
- **BJ_Hand:** Module này đại diện cho tay bài của người chơi hoặc dealer. Nó lưu trữ danh sách các lá bài (cards) và cung cấp phương thức add_card() để thêm bài, get_value() để tính điểm, và get_vietnamese_string() để hiển thị tay bài với emoji (ví dụ: "Át ,J ").
- **BJ_Game:** Module này quản lý logic chính của trò chơi, bao gồm chia bài ban đầu (initial_deal()), xử lý rút bài (hit()), và điều khiển lượt chơi của dealer (dealer_play()). Module đóng vai trò trung gian, kết nối giữa các module dữ liệu (BJ_Deck, BJ_Hand) và giao diện người dùng.
- **BlackjackGUI:** Module này chịu trách nhiệm xây dựng và quản lý giao diện đồ họa, sử dụng thư viện Tkinter. Nó hiển thị tay bài, điểm số, và các nút tương tác ("Hit", "Stand"). Module cũng xử lý sự kiện từ người dùng và cập nhật giao diện theo thời gian thực.

Mối quan hệ giữa các module:

- BJ_Card là thành phần cơ bản, được sử dụng bởi BJ_Deck để tạo bộ bài.
- BJ_Deck cung cấp lá bài cho BJ_Hand thông qua BJ_Game.
- BJ_Hand lưu trữ lá bài và cung cấp thông tin cho BlackjackGUI để hiển thị.
- BJ_Game điều phối luồng trò chơi và giao tiếp với BlackjackGUI để cập nhật trạng thái.

3.1.2. Sơ đồ phân cấp chức năng

Sơ đồ phân cấp chức năng đã được trình bày ở phần trước, nhưng mình sẽ nhắc lại và giải thích chi tiết hơn để làm rõ mối quan hệ giữa các chức năng chính và phụ trong chương trình:



- Giải thích chi tiết từng nhánh:**

- 1. Khởi tạo trò chơi:** Bao gồm việc chuẩn bị bộ bài và giao diện. BJ_Deck xáo trộn bài và chia 2 lá cho mỗi bên, sau đó BlackjackGUI hiển thị thông tin và kiểm tra Blackjack ban đầu.
- 2. Quản lý chơi đơn:** Tập trung vào các hành động của người chơi. Khi người chơi nhấn "Hit", chương trình thêm lá bài, tính điểm, kiểm tra bust, và cập nhật giao diện. Khi chọn "Stand", chương trình chuyển sang lượt dealer.
- 3. Quản lý dealer:** Điều khiển lượt chơi của dealer theo quy tắc (rút bài nếu điểm < 17, dừng nếu ≥ 17), và cập nhật giao diện.
- 4. Kết thúc và thông báo:** So sánh điểm, xác định kết quả, hiển thị thông báo qua hộp thoại, và vô hiệu hóa nút.

3.1.3. Luồng hoạt động tổng quan

Luồng hoạt động của hệ thống có thể được mô tả như sau:

1. **Khởi tạo:** Chương trình tạo bộ bài, chia bài ban đầu, và hiển thị giao diện.
2. **Tương tác người chơi:** Người chơi nhấn "Hit" hoặc "Stand", giao diện cập nhật sau mỗi hành động.
3. **Lượt dealer:** Nếu người chơi chọn "Stand", dealer chơi theo quy tắc và cập nhật giao diện.
4. **Kết thúc:** So sánh điểm, thông báo kết quả, và kết thúc ván.

3.2. Sơ đồ khái niệm các thuật toán chính

Phần này mô tả các thuật toán chính trong chương trình, bao gồm đầu vào, xử lý, đầu ra, và mối quan hệ giữa các khái niệm.

3.2.1. Thuật toán chia bài ban đầu (initial_deal)

- **Đầu vào:** Bộ bài (BJ_Deck.cards) chứa 52 lá đã được xáo trộn.
- **Xử lý:**
 1. Gọi deal() 4 lần để phát 2 lá cho người chơi và 2 lá cho dealer.
 2. Thêm các lá bài vào tay của người chơi (player_hand) và dealer (dealer_hand) bằng BJ_Hand.add_card().
 3. Cập nhật giao diện với tay bài và điểm số ban đầu.
- **Đầu ra:** Hai tay bài (player_hand, dealer_hand) với 2 lá mỗi bên.

- **Ví dụ minh họa:** Giả sử deal() lần lượt trả về "Át ", "J ", "6 ", "**7** ". Kết quả: người chơi có "Át , J " (21 điểm), dealer có "6 , 7 " (13 điểm).

3.2.2. Thuật toán rút bài (hit)

- **Đầu vào:** Tay bài hiện tại (BJ_Hand.cards) của người chơi.
- **Xử lý:**
 1. Gọi BJ_Game.hit() để lấy một lá bài mới từ BJ_Deck và thêm vào tay bài bằng BJ_Hand.add_card().
 2. Tính lại điểm số bằng BJ_Hand.get_value().
 3. Cập nhật giao diện bằng update_display().
 4. Kiểm tra nếu điểm vượt 21, thông báo "Over 21! You lose." và kết thúc ván.
- **Đầu ra:** Điểm số mới, cờ báo bust nếu > 21 .
- **Ví dụ minh họa:** Người chơi có "9 , 10 " (19 điểm), rút thêm "3 " → 22 điểm, thông báo bust.

3.2.3. Thuật toán dealer chơi (dealer_play)

- **Đầu vào:** Tay bài dealer (dealer_hand).
- **Xử lý:**
 1. Vòng lặp: Nếu điểm < 17 , gọi hit() để rút thêm bài.
 2. Dừng lại khi điểm ≥ 17 .
 3. Cập nhật giao diện sau mỗi lần rút bài.

- **Đầu ra:** Điểm cuối cùng của dealer.
- **Ví dụ minh họa:** Dealer có "6 ♣ , 7 ♠ " (13 điểm), rút thêm "5 ♦ " → 18 điểm, dừng lại.

3.2.4. Thuật toán so sánh (stand)

- **Đầu vào:** Điểm người chơi (player_value), điểm dealer (dealer_value).
- **Xử lý:**
 1. Kiểm tra nếu dealer vượt 21, thông báo "Dealer over 21! You win!".
 2. Nếu không, so sánh điểm: người chơi thắng nếu điểm cao hơn, thua nếu thấp hơn, hòa nếu bằng.
 3. Hiển thị kết quả qua Messagebox.
- **Đầu ra:** Thông báo kết quả (thắng, thua, hòa).
- **Ví dụ minh họa:** Người chơi 18, dealer 19 → "Dealer wins!".

3.2.5. Mối quan hệ giữa các khối thuật toán

- initial_deal tạo trạng thái ban đầu cho trò chơi, cung cấp dữ liệu cho các thuật toán sau.
- hit và dealer_play đều sử dụng BJ_Deck và BJ_Hand để thêm bài và tính điểm, nhưng hit chỉ áp dụng cho người chơi, còn dealer_play tự động hóa lượt dealer.
- stand là bước cuối, sử dụng kết quả từ dealer_play để so sánh và đưa ra kết luận.

3.3. Cấu trúc dữ liệu

Phần này mô tả chi tiết các cấu trúc dữ liệu được sử dụng trong chương trình.

3.3.1. Bảng dữ liệu tay bài (BJ_Hand)

- **Trường:** cards (list chứa các đối tượng BJ_Card).
- **Mô tả:** Lưu trữ danh sách các lá bài trong tay của người chơi hoặc dealer. Mỗi lá bài là một đối tượng BJ_Card với thuộc tính suit và value, hiển thị với emoji (ví dụ: "Át , J ").
- **Ví dụ minh họa:**
 - Người chơi: ["Át ", "J "] → Điểm 21.
 - Dealer: ["6 ", "7 "] → Điểm 13.
- **Vai trò:** Dùng để tính điểm (get_value()), hiển thị tay bài (get_vietnamese_string()), và quản lý các lá bài trong suốt ván chơi.

3.3.2. Bảng dữ liệu bộ bài (BJ_Deck)

- **Trường:** cards (list chứa 52 đối tượng BJ_Card).
- **Mô tả:** Lưu trữ toàn bộ lá bài ban đầu, giảm dần khi phát bài. Ban đầu, danh sách chứa tất cả các lá từ "Ace of Hearts" đến "King of Spades".
- **Ví dụ minh họa:**
 - Ban đầu: 52 lá.
 - Sau khi chia 4 lá (2 cho người chơi, 2 cho dealer): Còn 48 lá.
- **Vai trò:** Cung cấp lá bài cho BJ_Hand thông qua phương thức deal().

3.4. Chương trình

Phần này trình bày chi tiết các hàm chính trong chương trình.

3.4.1. Hàm trong lớp BlackjackGUI

- **start_game():**

- **Vai trò:** Khởi tạo một ván chơi mới.
- **Đầu vào:** Không có.
- **Xử lý:**
 1. Tạo một đối tượng BJ_Game mới.
 2. Gọi initial_deal() để chia bài.
 3. Cập nhật giao diện bằng update_display().
 4. Kiểm tra nếu người chơi hoặc dealer đạt 21 điểm ngay từ đầu, thông báo "Blackjack!" và kết thúc ván.
- **Đầu ra:** Trạng thái trò chơi ban đầu được thiết lập.
- **Ví dụ:** Người chơi nhận "Át ♥ ,J ♦" → 21 điểm, thông báo "Blackjack!".

- **update_display():**

- **Vai trò:** Cập nhật giao diện với tay bài và điểm số hiện tại.
- **Đầu vào:** Không có.
- **Xử lý:**
 1. Cập nhật nhãn player_label với tay bài người chơi (dùng BJ_Hand.get_vietnamese_string()).
 2. Cập nhật nhãn player_score với điểm số người chơi.
 3. Cập nhật nhãn dealer_label và dealer_score tương tự cho dealer.
- **Đầu ra:** Giao diện được cập nhật.

- **Ví dụ:** "Player: Át  ,J  - Score: 21".
- **hit():**
 - **Vai trò:** Xử lý hành động rút bài của người chơi.
 - **Đầu vào:** Không có.
 - **Xử lý:**
 1. Gọi BJ_Game.hit() để thêm lá bài vào tay người chơi.
 2. Cập nhật giao diện.
 3. Kiểm tra nếu điểm > 21, thông báo "Over 21! You lose." và kết thúc ván.
 4. Nếu điểm = 21, tự động gọi stand().
 - **Đầu ra:** Tay bài và điểm số được cập nhật.
 - **Ví dụ:** Người chơi có 19 điểm, rút thêm "3  " → 22 điểm, thua.
- **stand():**
 - **Vai trò:** Xử lý hành động dừng lại, chuyển sang lượt dealer và kết thúc ván.
 - **Đầu vào:** Không có.
 - **Xử lý:**
 1. Gọi BJ_Game.dealer_play() để dealer chơi.
 2. Cập nhật giao diện.
 3. So sánh điểm và thông báo kết quả.

- **Đầu ra:** Kết quả ván chơi.
- **Ví dụ:** Người chơi 18, dealer 19 → "Dealer wins!".
- **end_game():**
 - **Vai trò:** Kết thúc ván chơi.
 - **Đầu vào:** Không có.
 - **Xử lý:** Vô hiệu hóa các nút "Hit" và "Stand".
 - **Đầu ra:** Trò chơi dừng lại.
 - **Ví dụ:** Sau khi thông báo kết quả, người chơi không thể tương tác nữa.
- **BJ_Card.get_vietnamese_name():**
 - **Vai trò:** Chuyển đổi tên lá bài sang định dạng với emoji.
 - **Đầu ra:** Tên lá bài (ví dụ: "Át❤").
- **BJ_Deck.deal():**
 - **Vai trò:** Phát một lá bài từ bộ bài.
 - **Đầu ra:** Một lá bài (BJ_Card).
- **BJ_Hand.get_value():**
 - **Vai trò:** Tính điểm tay bài.
 - **Đầu ra:** Điểm số (ví dụ: 21).

3.5. Quy trình phát triển

Phần này mô tả các bước phát triển chương trình từ phân tích yêu cầu đến kiểm thử.

3.5.1. Phân tích yêu cầu

- Xác định các tính năng chính: chia bài, rút bài, dừng, tính điểm, hiển thị kết quả.
- Phân tích các yêu cầu đặc biệt: sử dụng các lớp BJ_Card, BJ_Deck, BJ_Hand, BJ_Game, hiển thị với emoji .

3.5.2. Thiết kế hệ thống

- Xây dựng các lớp và sơ đồ khái niệm đã trình bày ở trên.
- Lập kế hoạch triển khai: bắt đầu từ các lớp dữ liệu (BJ_Card, BJ_Deck), sau đó đến logic (BJ_Game), và cuối cùng là giao diện (BlackjackGUI).

3.5.3. Viết mã nguồn

- Triển khai từng lớp theo thiết kế.
- Kết nối các lớp với giao diện GUI, đảm bảo sự kiện được xử lý đúng.

3.5.4. Kiểm thử và sửa lỗi

- Kiểm tra từng chức năng: chia bài, rút bài, tính điểm, hiển thị.
- Sửa lỗi: ví dụ, lỗi tính điểm Át ban đầu (luôn tính 11, dẫn đến sai kết quả).

3.5.5. Tối ưu hóa

- Giảm thiểu tính toán lặp lại (ví dụ: lưu điểm số tạm để tránh gọi get_value() nhiều lần).

- Tối ưu giao diện: đảm bảo cập nhật nhanh, không bị lag.

3.6.Chương trình

```
import random

import tkinter as tk

from tkinter import messagebox

# Lớp đại diện cho lá bài

class BJ_Card:

    def __init__(self, suit, value):
        self.suit = suit
        self.value = value

    def __str__(self):
        return f"{self.value} of {self.suit}"

    def get_vietnamese_name(self):
        rank_map = {
            "Ace": "Át", "2": "2", "3": "3", "4": "4", "5": "5",
            "6": "6", "7": "7", "8": "8", "9": "9", "10": "10",
            "Jack": "J", "Queen": "Q", "King": "K"
        }
        suit_map = {
            "Hearts": "♥", "Diamonds": "♦", "Clubs": "♣", "Spades": "♠"
        }
        return f'{rank_map.get(self.value, self.value)} {suit_map.get(self.suit, "")}'
```

```
# Lớp đại diện cho bộ bài

class BJ_Deck:

    def __init__(self):
        suits = ["Hearts", "Diamonds", "Clubs", "Spades"]
        values = ["Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen",
                  "King"]
        self.cards = [BJ_Card(suit, value) for suit in suits for value in values]
        random.shuffle(self.cards)

    def deal(self):
        if self.cards:
            return [self.cards.pop()]
        return None

# Lớp đại diện cho tay bài

class BJ_Hand:

    def __init__(self):
        self.cards = []

    def add_card(self, card):
        self.cards.append(card)

    def get_value(self):
        value = 0
        aces = 0
        for card in self.cards:
```

```
if card.value == "Ace":  
    value += 11  
    aces += 1  
  
elif card.value in ["Jack", "Queen", "King"]:  
    value += 10  
  
else:  
    value += int(card.value) if card.value.isdigit() else 0  
  
while value > 21 and aces:  
    value -= 10  
    aces -= 1  
  
return value  
  
  
def get_vietnamese_string(self):  
    return ",".join(card.get_vietnamese_name() for card in self.cards)  
  
  
def is_blackjack(self):  
    return len(self.cards) == 2 and self.get_value() == 21  
  
  
# Lớp quản lý trò chơi  
class BJ_Game:  
    def __init__(self):  
        self.deck = BJ_Deck()  
        self.player_hand = BJ_Hand()  
        self.dealer_hand = BJ_Hand()  
  
  
    def initial_deal(self):
```

```
for _ in range(2):
    self.player_hand.add_card(self.deck.deal()[0])
    self.dealer_hand.add_card(self.deck.deal()[0])

def hit(self):
    card = self.deck.deal()
    if card:
        self.player_hand.add_card(card[0])

def dealer_play(self):
    while self.dealer_hand.get_value() < 17:
        card = self.deck.deal()
        if card:
            self.dealer_hand.add_card(card[0])
    return self.dealer_hand.get_value()

# Giao diện người dùng
class BlackjackGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Blackjack")
        self.game = BJ_Game()

    # Frame người chơi
    self.player_frame = tk.Frame(self.root)
    self.player_frame.pack(pady=10)
```

```
self.player_label = tk.Label(self.player_frame, text="Player:",  
font=("Arial", 14))  
self.player_label.pack()  
  
self.player_cards = tk.Label(self.player_frame, text="", font=("Arial", 12))  
self.player_cards.pack()  
  
self.player_score = tk.Label(self.player_frame, text="Score: 0",  
font=("Arial", 12))  
self.player_score.pack()  
  
# Frame dealer  
  
self.dealer_frame = tk.Frame(self.root)  
self.dealer_frame.pack(pady=10)  
self.dealer_label = tk.Label(self.dealer_frame, text="Dealer:",  
font=("Arial", 14))  
self.dealer_label.pack()  
  
self.dealer_cards = tk.Label(self.dealer_frame, text="", font=("Arial", 12))  
self.dealer_cards.pack()  
  
self.dealer_score = tk.Label(self.dealer_frame, text="Score: ?",  
font=("Arial", 12))  
self.dealer_score.pack()  
  
# Frame nút  
  
self.button_frame = tk.Frame(self.root)  
self.button_frame.pack(pady=10)  
self.hit_button = tk.Button(self.button_frame, text="Hit",  
command=self.hit, font=("Arial", 12))
```

```
self.hit_button.pack(side=tk.LEFT, padx=5)
self.stand_button = tk.Button(self.button_frame, text="Stand",
command=self.stand, font=("Arial", 12))
self.stand_button.pack(side=tk.LEFT, padx=5)
self.new_game_button = tk.Button(self.button_frame, text="New Game",
command=self.new_game, font=("Arial", 12))
self.new_game_button.pack(side=tk.LEFT, padx=5)

# Label kết quả
self.result_label = tk.Label(self.root, text="", font=("Arial", 14))
self.result_label.pack(pady=10)

self.new_game()

def new_game(self):
    self.game = BJ_Game()
    self.game.initial_deal()
    self.update_display()
    if self.game.dealer_hand.is_blackjack():
        self.update_display(show_dealer=True)
        self.result_label.config(text="Dealer Blackjack! You lose.")
        messagebox.showinfo("Result", "Dealer Blackjack! You lose.")
        self.end_game()
    elif self.game.player_hand.is_blackjack():
        self.update_display(show_dealer=True)
        self.result_label.config(text="Blackjack!")
```

```
messagebox.showinfo("Result", "Blackjack!")

self.end_game()

else:

    self.hit_button.config(state=tk.NORMAL)
    self.stand_button.config(state=tk.NORMAL)
    self.result_label.config(text="")

def update_display(self, show_dealer=False):
    self.player_cards.config(text=self.game.player_hand.get_vietnamese_string()
())
    self.player_score.config(text=f"Score:
{self.game.player_hand.get_value()}")
    if show_dealer:
        self.dealer_cards.config(text=self.game.dealer_hand.get_vietnamese_string())
        self.dealer_score.config(text=f"Score:
{self.game.dealer_hand.get_value()}")
    else:
        if self.game.dealer_hand.cards:
            first_card = self.game.dealer_hand.cards[0].get_vietnamese_name()
            hidden_cards = ",".join("?" for _ in
range(len(self.game.dealer_hand.cards) - 1))
            self.dealer_cards.config(text=f"{first_card},{hidden_cards}" if
hidden_cards else first_card)
            self.dealer_score.config(text="Score: ?")
```

```
def hit(self):
    self.game.hit()
    self.update_display()
    if self.game.player_hand.get_value() > 21:
        self.result_label.config(text="Over 21! You lose.")
        messagebox.showinfo("Result", "Over 21! You lose.")
        self.end_game()
    elif self.game.player_hand.is_blackjack():
        self.result_label.config(text="Blackjack!")
        messagebox.showinfo("Result", "Blackjack!")
        self.end_game()

def stand(self):
    dealer_value = self.game.dealer_play()
    self.update_display(show_dealer=True)
    player_value = self.game.player_hand.get_value()
    if dealer_value > 21:
        self.result_label.config(text="Dealer over 21! You win!")
        messagebox.showinfo("Result", "Dealer over 21! You win!")
    elif dealer_value > player_value:
        self.result_label.config(text="Dealer wins!")
        messagebox.showinfo("Result", "Dealer wins!")
    elif player_value > dealer_value:
        self.result_label.config(text="You win!")
        messagebox.showinfo("Result", "You win!")
    else:
```

```
self.result_label.config(text="Tie!")  
messagebox.showinfo("Result", "Tie!")  
self.end_game()  
  
def end_game(self):  
    self.hit_button.config(state=tk.DISABLED)  
    self.stand_button.config(state=tk.DISABLED)  
  
# Chạy chương trình  
if __name__ == "__main__":  
    root = tk.Tk()  
    app = BlackjackGUI(root)  
    root.mainloop()
```

3.7. Đánh giá thiết kế

Thiết kế của chương trình có một số ưu điểm và hạn chế:

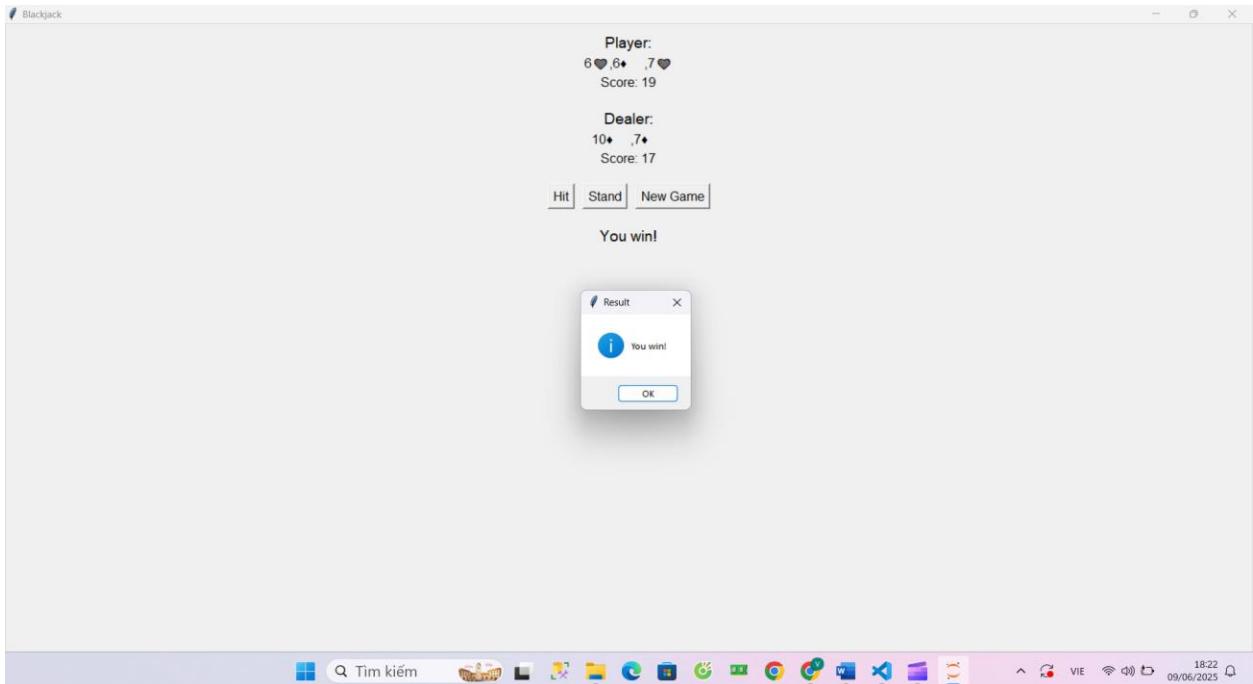
- **Ưu điểm:**
 - Cấu trúc mã nguồn rõ ràng, sử dụng OOP để dễ bảo trì.
 - Giao diện đơn giản, dễ sử dụng, cập nhật theo thời gian thực.
 - Logic tính điểm chính xác, xử lý được các trường hợp đặc biệt như Át.
- **Hạn chế:**
 - Chưa có hình ảnh lá bài, chỉ hiển thị bằng text với emoji.

- Chưa hỗ trợ lưu ván chơi hoặc chế độ chơi nâng cao.
- Giao diện có thể cải thiện thêm về mặt thẩm mỹ.

Chương 4. Thực nghiệm và kết luận

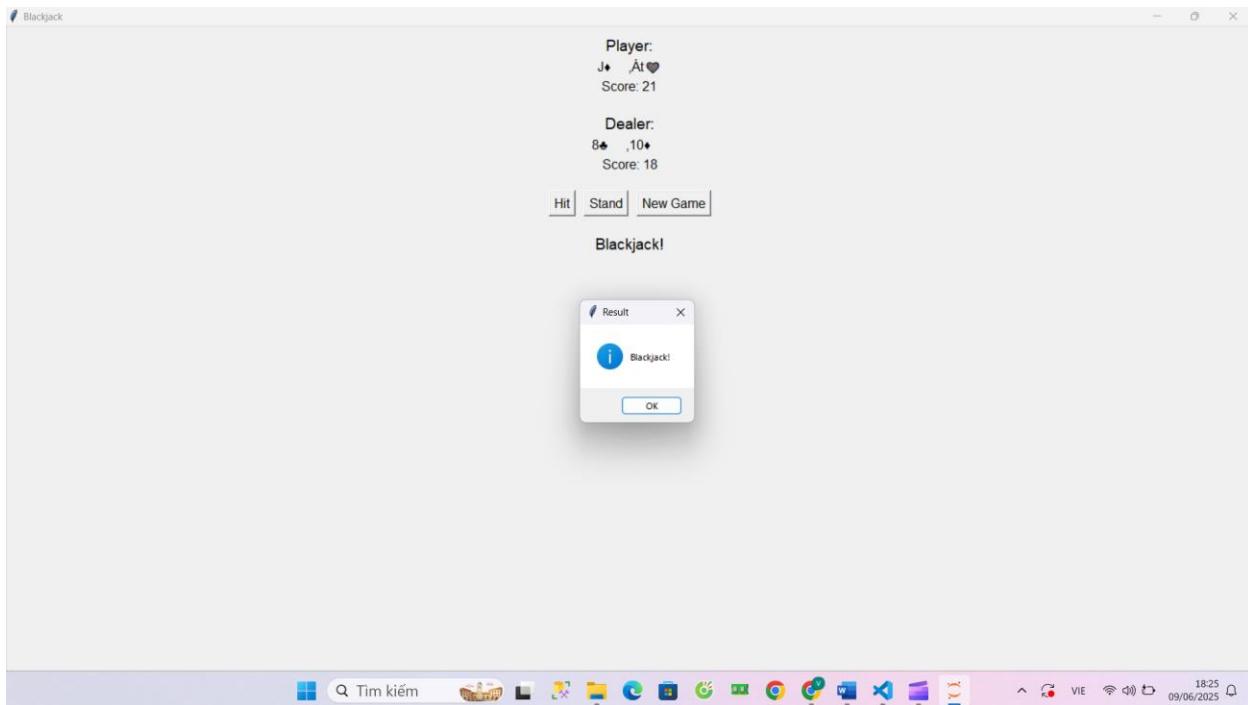
4.1. Thực nghiệm

Thử nghiệm 1:



- **Khởi đầu:** "Player: 6♥, 6♦".
- **Rút bài:** "Player: 6♥, 6♦, 7♥".
- **Kết quả:** "You win!".

Thử nghiệm 2:



- Khởi đầu : J♦ ,Át♥
- Kết quả : “Blackjack!”

4.2. Kết luận

- Sản phẩm đã làm được:** Thành công trong việc xây dựng game Blackjack với giao diện đồ họa (GUI) bằng thư viện Tkinter, sử dụng các lớp yêu cầu (BJ_Card, BJ_Deck, BJ_Hand, BJ_Game), xử lý trường hợp vượt điểm (bust), và so sánh điểm để xác định kết quả giữa người chơi và dealer.
- Học được gì:** Thành thạo các kỹ năng lập trình Python, hiểu rõ cách xử lý logic phức tạp của game, và cải thiện khả năng viết báo cáo khoa học một cách có hệ thống.

- **Sẽ cải tiến gì:** Thêm hình ảnh lá bài (ví dụ: PNG với ký hiệu và màu sắc), tích hợp hiệu ứng âm thanh (như tiếng xào bài), phát triển chế độ chơi nhiều người, và thêm tính năng lưu điểm số để nâng cao trải nghiệm.

4.3. Đề xuất mở rộng

Để nâng cao giá trị của sản phẩm và mở rộng trải nghiệm người dùng, mình đã suy nghĩ và đề xuất một số hướng cải tiến và phát triển thêm cho game Blackjack này. Các ý tưởng này không chỉ giúp sản phẩm trở nên hấp dẫn hơn mà còn là cơ hội để áp dụng thêm các kỹ thuật lập trình tiên tiến trong tương lai. Dưới đây là các đề xuất chi tiết:

- **Thêm chế độ chơi mạng (Multiplayer Online):**
 - **Mô tả ý tưởng:** Hiện tại, trò chơi chỉ hỗ trợ một người chơi đấu với dealer (AI). Mình muốn phát triển chế độ chơi mạng, cho phép nhiều người chơi cùng tham gia qua Internet, giống như các sòng bạc trực tuyến. Mỗi người chơi sẽ có tài khoản riêng, có thể thách đấu bạn bè hoặc người lạ, và theo dõi thứ hạng trên bảng xếp hạng.
 - **Lợi ích:** Tăng tính tương tác xã hội, thu hút nhiều người chơi hơn, và biến trò chơi thành một nền tảng giải trí cộng đồng. Người chơi có thể giao lưu, học hỏi chiến thuật từ nhau, tạo nên một cộng đồng yêu thích Blackjack.
 - **Cách triển khai:** Sử dụng các thư viện như socket hoặc asyncio trong Python để thiết lập kết nối mạng. Cần xây dựng một máy chủ trung gian (server) để quản lý các kết nối, lưu trữ dữ liệu người chơi, và đồng bộ trạng thái trò chơi. Ngoài ra, cần tích hợp cơ chế bảo mật (như mã hóa dữ liệu) để bảo vệ thông tin người dùng. Giao diện GUI

cũng cần được nâng cấp với các nút "Thách đấu", "Kết nối", và bảng xếp hạng.

- **Thách thức:** Việc đồng bộ trạng thái trò chơi giữa nhiều người chơi đòi hỏi kỹ năng xử lý đa luồng (multithreading) và quản lý lỗi mạng (ví dụ: mất kết nối). Chi phí phát triển máy chủ và bảo trì cũng là một vấn đề cần xem xét.
- **Ví dụ minh họa:** Người chơi A ở Hà Nội có thể thách đấu người chơi B ở TP.HCM, với mỗi người có 2 lá bài ban đầu, và lượt chơi được thực hiện theo thứ tự.
- **Tích hợp AI thông minh hơn cho nhà cái:**
 - **Mô tả ý tưởng:** Hiện tại, nhà cái chỉ rút bài theo quy tắc cố định (điểm < 17 thì rút). Mình muốn nâng cấp AI nhà cái để nó có khả năng đưa ra quyết định dựa trên chiến thuật, như dự đoán chiến thuật của người chơi, điều chỉnh cách chơi theo tình huống (ví dụ: giữ lại điểm thấp hơn 17 nếu nghi ngờ người chơi sắp bust), và học hỏi từ các ván trước.
 - **Lợi ích:** AI thông minh hơn sẽ làm tăng độ khó và thú vị của trò chơi, đặc biệt với người chơi có kinh nghiệm. Điều này cũng giúp người chơi rèn luyện kỹ năng chiến thuật, giống như chơi với một đối thủ thực sự.
 - **Cách triển khai:** Sử dụng các thuật toán học máy cơ bản (machine learning), chẳng hạn như học có giám sát (supervised learning) với dữ liệu từ các ván chơi trước. Cần thu thập dữ liệu về cách người chơi rút bài (ví dụ: rút khi điểm 16, dừng khi điểm 19) để huấn luyện mô hình.

Ngoài ra, có thể áp dụng thuật toán tối ưu hóa như Q-learning để AI tự điều chỉnh chiến thuật.

- **Thách thức:** Yêu cầu kiến thức về học máy và xử lý dữ liệu lớn, đồng thời cần tối ưu hóa để tránh làm chậm trò chơi. Việc tích hợp AI cũng có thể làm phức tạp mã nguồn, đòi hỏi thời gian phát triển thêm.
- **Ví dụ minh họa:** Nếu người chơi thường rút bài khi điểm 15, AI nhà cái có thể giữ điểm 17 để chờ người chơi bust, thay vì rút thêm như quy tắc cố định.
- **Thêm hiệu ứng hình ảnh và âm thanh:**
 - **Mô tả ý tưởng:** Hiện tại, trò chơi chỉ hiển thị tên lá bài bằng văn bản với emoji. Mình muốn thêm hình ảnh lá bài (ví dụ: PNG 100x150 pixel với ký hiệu và màu sắc) và hiệu ứng âm thanh (như tiếng xào bài khi rút, tiếng reo hò khi thắng).
 - **Lợi ích:** Hiệu ứng hình ảnh và âm thanh sẽ làm tăng tính trực quan và hấp dẫn, tạo cảm giác như chơi trong sòng bạc thực sự. Điều này đặc biệt quan trọng để giữ chân người chơi lâu dài.
 - **Cách triển khai:** Sử dụng thư viện PIL (Pillow) để xử lý hình ảnh lá bài, và thư viện pygame hoặc playsound để thêm âm thanh. Hình ảnh có thể được tạo thủ công bằng công cụ như GIMP hoặc tải từ nguồn mở (open-source). Âm thanh cần được thiết kế theo từng sự kiện (rút bài, thắng, thua).
 - **Thách thức:** Yêu cầu thiết kế đồ họa cơ bản và tối ưu hóa hiệu suất để tránh lag khi tải nhiều hình ảnh/âm thanh. Cần kiểm tra trên nhiều thiết bị để đảm bảo tương thích.

- **Ví dụ minh họa:** Khi rút lá "Át ❤️", màn hình hiển thị hình ảnh lá bài đỏ với ký hiệu ♥, kèm tiếng "xào bài" nhẹ nhàng.
- **Tích hợp tính năng hướng dẫn chơi:**
 - **Mô tả ý tưởng:** Thêm một phần hướng dẫn trong giao diện, giải thích quy tắc Blackjack, cách tính điểm, và mẹo chơi (ví dụ: nên dừng ở điểm 17 hay 18).
 - **Lợi ích:** Hỗ trợ người chơi mới làm quen với trò chơi, giảm rào cản khi bắt đầu, và tăng trải nghiệm người dùng.
 - **Cách triển khai:** Sử dụng Label hoặc Text widget trong Tkinter để hiển thị hướng dẫn. Có thể thêm nút "Hướng dẫn" mở cửa sổ mới với nội dung chi tiết.
 - **Thách thức:** Nội dung hướng dẫn cần ngắn gọn nhưng đầy đủ, tránh làm rối giao diện chính.
 - **Ví dụ minh họa:** Mở hướng dẫn, hiển thị: "Ace can be 1 or 11, dealer stops at ≥ 17 ."

Kết luận

Qua quá trình thực hiện dự án phát triển game Blackjack với giao diện đồ họa (GUI) bằng Python, dự án đã thành công trong việc xây dựng một hệ thống trò chơi hoàn chỉnh, bao gồm các thành phần chính như lớp BJ_Card, BJ_Deck, BJ_Hand, BJ_Game, và BlackjackGUI. Sử dụng thư viện Tkinter, giao diện đồ họa được thiết kế mượt mà, cho phép người chơi tương tác qua các nút "Hit", "Stand", và "New Game", đồng thời hiển thị trạng thái trò chơi với định dạng emoji (♥, ♦, ♣, ♠) để tăng tính trực quan. Các chức năng cốt lõi như tính điểm (bao gồm xử lý linh hoạt lá Át), kiểm tra bust, và so sánh điểm giữa người chơi và dealer đã được triển khai chính xác, đáp ứng đầy đủ quy tắc của trò chơi Blackjack.

Qua dự án, mình đã nâng cao kỹ năng lập trình Python, đặc biệt trong việc áp dụng lập trình hướng đối tượng (OOP) và quản lý sự kiện trong Tkinter. Đồng thời, quá trình thiết kế logic game và tối ưu hóa hiệu suất đã giúp mình hiểu sâu hơn về cách xây dựng ứng dụng tương tác. Viết báo cáo khoa học cũng là một trải nghiệm quý báu, giúp rèn luyện khả năng tổng hợp và trình bày ý tưởng một cách có hệ thống.

Dẫu vậy, sản phẩm vẫn còn một số hạn chế như thiếu hình ảnh lá bài thực tế, hiệu ứng âm thanh, và các tính năng nâng cao như chế độ chơi nhiều người hay lưu điểm số. Trong tương lai, mình dự kiến cải tiến bằng cách tích hợp hình ảnh và âm thanh để tăng trải nghiệm người dùng, phát triển chế độ chơi mạng sử dụng các thư viện như socket, và nâng cấp AI nhà cái với thuật toán học máy để tăng độ khó. Những đề xuất này không chỉ nâng cao giá trị giải trí mà còn mở ra cơ hội áp dụng các kỹ thuật lập trình tiên tiến, biến dự án thành một nền tảng tiềm năng cho cộng đồng yêu thích Blackjack.