TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

BÀI TẬP LỚN

LẬP TRÌNH PYTHON

NGÀNH : KỸ THUẬT MÁY TÍNH

HỆ : ĐẠI HỌC CHÍNH QUY

THÁI NGUYÊN - 2025

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

BÀI TẬP LỚN

LẬP TRÌNH PYTHON

BỘ MÔN: CÔNG NGHỆ THÔNG TIN

GIÁO VIÊN GIẢNG DẠY : TS.NGUYỄN VĂN HUY

LỚP : K58KMT

SINH VIÊN THỰC HIỆN : TRƯƠNG VĂN QUYẾN

THÁI NGUYÊN – 2025

TRƯỜNG ĐHKTCN
KHOA ĐIỆN TỬ

CỘNG HOÀ 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: TRƯƠNG VĂN QUYẾN

Lớp : K58KTP.K01 *Ngành: Kỹ thuật máy tính*

Giáo viên hướng dẫn: Ts. Nguyễn Văn Huy

Ngày giao đề 20/05/2025 Ngày hoàn thành

Tên đề tài : Critter Caretaker với GUI

Yêu cầu :

- *Class Critter với attributes và methods.*
- *GUI: Entry tạo critter, Buttons: "Tạo", "Cho ăn", "Chơi", "Ngủ".*
- *Cập nhật trạng thái real-time.*
- *Bắt lỗi khi chưa tạo critter.*

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

Thái Nguyên, ngày….tháng…..năm 20....
GIÁO VIÊN HƯỚNG DẪN
(Ký ghi rõ họ tên)

LỜI MỞ ĐẦU

Trong thời đại công nghệ phát triển nhanh chóng, việc xây dựng các ứng dụng tương tác với người dùng thông qua giao diện đồ họa (GUI) ngày càng trở nên phổ biến và cần thiết. Bài tập lớn môn Lập trình Python với đề tài "Critter Caretaker với GUI" là một cơ hội tốt để sinh viên vận dụng các kiến thức đã học về lập trình hướng đối tượng, xử lý sự kiện, và thiết kế giao diện với thư viện Tkinter để phát triển một ứng dụng hoàn chỉnh.

Với cú pháp rõ ràng, dễ đọc và gần gũi với ngôn ngữ tự nhiên, Python đặc biệt phù hợp với những người mới bắt đầu học lập trình. Tuy nhiên, không dừng lại ở đó, Python còn được đánh giá cao bởi cộng đồng lập trình viên chuyên nghiệp, nhờ vào kho thư viện phong phú, khả năng mở rộng linh hoạt và khả năng tích hợp tốt với nhiều nền tảng công nghệ hiện đại. Từ việc phân tích dữ liệu trong khoa học dữ liệu, phát triển mô hình học máy trong trí tuệ nhân tạo, đến xây dựng các website chuyên nghiệp hay viết các công cụ tự động hóa, Python đều có thể đáp ứng hiệu quả.

Chính nhờ sự đa năng và thân thiện của mình, Python đã trở thành công cụ không thể thiếu trong chương trình giảng dạy của nhiều trường đại học và là nền tảng vững chắc để người học phát triển kỹ năng lập trình một cách toàn diện. Việc tìm hiểu và nắm vững Python sẽ mở ra nhiều cơ hội trong học tập, nghiên cứu và nghề nghiệp trong lĩnh vực công nghệ thông tin hiện đại.
Đề tài yêu cầu xây dựng một hệ thống tương tác với "critter" – một đối tượng mô phỏng thú cưng ảo – cho phép người dùng tạo mới, cho ăn, cho chơi hoặc cho ngủ. Thông qua quá trình phát triển chương trình, sinh viên không chỉ rèn luyện kỹ năng lập trình mà còn nâng cao khả năng phân tích, thiết kế hệ thống và xử lý ngoại lệ trong các tình huống thực tế.

Báo cáo này trình bày quá trình thiết kế, xây dựng và kiểm thử ứng dụng Critter Caretaker, từ cơ sở lý thuyết đến hiện thực hóa chương trình và đánh giá kết quả. Mục tiêu là tạo ra một sản phẩm vừa đáp ứng yêu cầu đề bài, vừa thể hiện được tư duy lập trình và tính sáng tạo cá nhân.

Em xin chân thành cảm ơn sự quan tâm giúp đỡ của Thầy **NGUYỄN VĂN HUY** đã giúp đỡ em hoàn thành đề tài này.

MỤC LỤC

CHƯƠNG I TỔNG QUAN ĐỀ TÀI 7
1.1.Định dướng đề tài. 7
1.2.Thách thức của bài toán: 7
CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 9
2.1. Lập trình hướng đối tượng với Python 9
2.2. Thư viện Tkinter và xử lý giao diện người dùng 9
2.2.1. Giới thiệu về Tkinter 9
2.2.2. Các thành phần chính của Tkinter 9
2.2.3. Xử lý sự kiện và tương tác người dùng 10
2.2.4 Ưu điểm của Tkinter 10
2.2.5 Hạn chế 10
2.3. Quản lý trạng thái đối tượng 10
2.3.1. Khởi tạo trạng thái ban đầu 10
2.3.2. Truy cập và thay đổi trạng thái 10
2.3.3. Quản lý trạng thái thông qua phương thức 11
CHƯƠNG 3 THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH 11
3.1. Sơ đồ khối hệ thống 11
3.3.2. Khởi tạo Critter với các chỉ số mặc định 12
3.3.3. Hiển thị giao diện GUI với các lựa chọn 12
3.3.4. Người dùng chọn hành động 12
3.3.5. Cập nhật trạng thái Critter 12
3.3.6. Hiển thị trạng thái mới lên GUI 12
3.3.7. Vòng lặp chăm sóc 12
3.2. Sơ đồ thuật toán chính 13
3.2.1. Thuật toán: Tạo Critter 14
3.2.2. Thuật toán: Chọn Critter 14
3.2.3. Thuật toán: Cho Feed/Play/Sleep 15

3.2.4. Thuật toán: Hiển thị trạng thái Critter 17
3.3. Cấu trúc dữ liệu 18
3.3.1. Bảng Critter 18
3.3.2. Danh sách Critter 18
3.4. Các hàm chính trong chương trình, ... 18
3.4.1. Hàm khởi tạo lớp Critter 18
3.4.2.Hàm hành động của Critter 19
3.4.3.Hàm xác định tâm trạng 20
3.4.4 Hàm tạo Critter 20
3.4.5. Hàm chọn Critter trong danh sách 21
4.1. Kết quả thực nghiệm 22
4.1.2. Chức năng hành động cho từng Critter và Tâm trạng của Critter 23
4.2. Sản phẩm đã làm đã làm được những gì 24
4.2.1.Tạo mới thú ảo Critter thành công 24
4.2.2.Giao tiếp với Critter qua các nút chức năng 25
4.2.3.Cập nhật trạng thái theo thời gian 25
4.2.4Hiển thị tâm trạng và các chỉ số rõ ràng 25
4.2.5.Giao diện đồ họa thân thiện, dễ thao tác 25
4.3.Những điều học được 26
4.4. Hướng cải tiến trong tương lai 26

CHƯƠNG I TỔNG QUAN ĐỀ TÀI

1.1.Định dướng đề tài.

Trong bài tập lớn cuối môn này, em được giao thực hiện đề tài số 4: **Xây dựng ứng dụng "Critter Caretaker" với giao diện đồ họa (GUI)**. Đây là một bài toán mô phỏng việc chăm sóc một sinh vật ảo (critter) thông qua các hành động như cho ăn, chơi, ngủ… thông qua giao diện người dùng do người lập trình thiết kế.
Chương trình sẽ cho phép người dùng:
Nhập tên để **tạo một Critter mới**.
Thực hiện các hành động tương tác với Critter như:

- **Cho ăn** (giảm mức độ đói – hunger),
- **Chơi** (giảm sự chán – boredom),
- **Ngủ** (đặt lại trạng thái Critter).

Hiển thị trạng thái hiện tại của Critter gồm mức độ đói và chán.
Cập nhật trạng thái mỗi khi người dùng thực hiện hành động.
Báo lỗi nếu người dùng cố gắng tương tác khi chưa tạo Critter.
Các tính năng chính:
Giao diện trực quan với **Tkinter** (nhập tên, các nút điều khiển, trạng thái).
Mô hình hóa Critter bằng **lập trình hướng đối tượng** (class, thuộc tính, phương thức).
Xử lý ngoại lệ khi chưa tạo Critter hoặc nhập tên rỗng.
Tự động thay đổi trạng thái hunger/boredom theo thời gian hoặc sau mỗi hành động (tùy chọn nâng cao).

1.2.Thách thức của bài toán:

Mặc dù đề tài không yêu cầu thuật toán phức tạp, nhưng việc triển khai một ứng dụng tương tác giữa người dùng và sinh vật ảo vẫn đặt ra một số thách thức nhất định trong quá trình lập trình. Để hoàn thành bài toán một cách hiệu quả, người thực hiện cần giải quyết tốt các khía cạnh sau:

- **Tổ chức class Critter hợp lý**, đảm bảo khả năng mở rộng và kiểm soát trạng thái.
- **Xây dựng GUI trực quan** và dễ sử dụng bằng Tkinter, kết nối logic với giao diện.
- **Quản lý trạng thái nội tại** của Critter sao cho phản ánh đúng quá trình tương tác của người dùng.
- **Xử lý ngoại lệ và đảm bảo tính ổn định** của chương trình khi người dùng nhập sai hoặc thao tác bất hợp lý.

Để hoàn thành đề tài "Critter Caretaker" với giao diện đồ họa, em đã vận dụng nhiều kiến thức đã học trong học phần, đặc biệt là các khái niệm về lập trình hướng đối

tượng trong Python. Việc xây dựng một lớp Critter với các thuộc tính như mức độ đói (hunger), mức độ chán (boredom), và các phương thức xử lý hành vi như ăn, chơi, ngủ đã giúp em hiểu rõ hơn về cách tổ chức dữ liệu và chức năng trong một đối tượng. Mỗi hành động được thực hiện trên Critter đều tác động đến trạng thái của nó, thể hiện rõ đặc điểm đóng gói và quản lý trạng thái nội tại – một nguyên tắc cơ bản của lập trình hướng đối tượng.

Bên cạnh đó, em đã áp dụng thư viện Tkinter để xây dựng giao diện đồ họa (GUI) cho chương trình. Các kiến thức về cách tạo cửa sổ, bố trí các thành phần giao diện như nút bấm (Button), ô nhập liệu (Entry), nhãn hiển thị (Label), cũng như cách gắn sự kiện (event binding) cho các nút chức năng, đã được em vận dụng để hoàn thiện phần giao diện của ứng dụng. Tkinter không chỉ giúp chương trình dễ sử dụng hơn mà còn là cầu nối giữa người dùng và logic xử lý của hệ thống.

Ngoài ra, việc xử lý lỗi và kiểm tra đầu vào cũng là một phần quan trọng trong quá trình phát triển ứng dụng. Em đã sử dụng các cấu trúc điều kiện và hộp thoại thông báo (messagebox) để kiểm soát tình huống khi người dùng chưa tạo Critter mà đã thao tác, hoặc khi nhập tên Critter không hợp lệ. Điều này giúp chương trình trở nên thân thiện và an toàn hơn khi sử dụng.

Cuối cùng, em cũng được rèn luyện kỹ năng quản lý trạng thái của đối tượng trong suốt vòng đời hoạt động. Mỗi hành động của người dùng đều làm thay đổi trạng thái của Critter, và các trạng thái này được cập nhật và hiển thị liên tục trên giao diện. Việc xây dựng một cơ chế cập nhật trạng thái đơn giản nhưng hiệu quả giúp em hiểu rõ tầm quan trọng của việc duy trì tính nhất quán trong thiết kế phần mềm.

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

2.1. Lập trình hướng đối tượng với Python

Lập trình hướng đối tượng (Object-Oriented Programming – OOP) là một phương pháp lập trình dựa trên ý tưởng mô hình hóa phần mềm bằng các "đối tượng", tức là các thực thể có trạng thái (dữ liệu) và hành vi (phương thức). Python là một ngôn ngữ lập trình hỗ trợ hoàn toàn các nguyên lý của lập trình hướng đối tượng, đồng thời cung cấp cú pháp đơn giản, dễ đọc, giúp người học tiếp cận OOP một cách tự nhiên và hiệu quả.

Trong Python, **lớp (class)** là một khuôn mẫu (template) để tạo ra các đối tượng. Một lớp định nghĩa các thuộc tính và phương thức mà đối tượng của lớp đó sẽ có. Việc khởi tạo một đối tượng từ lớp gọi là **tạo thể hiện (instance)**. Mỗi đối tượng có thể lưu trữ dữ liệu riêng biệt, đồng thời chia sẻ các hành vi chung được định nghĩa trong lớp.

2.2. Thư viện Tkinter và xử lý giao diện người dùng

Trong lĩnh vực phát triển ứng dụng desktop bằng Python, **Tkinter** là một trong những thư viện được sử dụng phổ biến nhất để xây dựng giao diện người dùng (GUI – Graphical User Interface). Đây là thư viện GUI tiêu chuẩn đi kèm với Python, được xây dựng dựa trên nền tảng Tk GUI toolkit, cho phép lập trình viên dễ dàng thiết kế các cửa sổ giao diện, nút bấm, hộp nhập liệu, menu và các thành phần đồ họa khác.

2.2.1. Giới thiệu về Tkinter

Tkinter cung cấp một cách tiếp cận đơn giản và trực quan để tạo ra các cửa sổ giao diện người dùng. Một ứng dụng cơ bản trong Tkinter thường gồm ba thành phần chính:

Cửa sổ chính (Main Window): Là cửa sổ giao diện đầu tiên xuất hiện khi ứng dụng chạy.

Widgets (Tiện ích giao diện): Là các thành phần như Button, Label, Entry, Frame, Menu,... dùng để hiển thị nội dung và tương tác với người dùng.

Sự kiện (Events): Là các hành động từ người dùng như nhấn nút, nhập dữ liệu, di chuyển chuột,... mà chương trình cần xử lý.

2.2.2. Các thành phần chính của Tkinter

Tkinter hỗ trợ nhiều widget phục vụ cho thiết kế giao diện:

Label: Hiển thị văn bản.Entry: Nhập dữ liệu một dòng.Text: Nhập dữ liệu nhiều dòng.Button: Nút bấm thực hiện hành động.

Checkbutton, Radiobutton: Chọn lựa tùy chọn.Listbox, Combobox: Hiển thị danh sách lựa chọn.Canvas: Vẽ hình học, ảnh, đồ thị.Frame: Tổ chức bố cục giao diện.Menu: Tạo thanh menu tương tác.

2.2.3. Xử lý sự kiện và tương tác người dùng

Tkinter sử dụng **mô hình hướng sự kiện**, tức là chương trình sẽ "chờ" người dùng thao tác (click, nhập liệu...) và sau đó thực hiện hành động tương ứng. Sự kiện thường được gán qua các hàm command= hoặc sử dụng phương thức bind() để gán hàm xử

2.2.4 Ưu điểm của Tkinter

Miễn phí, tích hợp sẵn trong Python.
Cú pháp dễ học, dễ hiểu.
Phù hợp với các ứng dụng nhỏ đến trung bình.
Có thể chạy trên nhiều hệ điều hành (Windows, macOS, Linux).

2.2.5 Hạn chế

Giao diện khá đơn giản, không hiện đại so với các framework như PyQt, Kivy.
Khó tùy biến khi làm ứng dụng phức tạp hoặc mang tính thẩm mỹ cao.

2.3. Quản lý trạng thái đối tượng

Trong lập trình hướng đối tượng (OOP), **trạng thái của đối tượng** đề cập đến **tập hợp các giá trị** mà các thuộc tính (biến) của đối tượng đang nắm giữ tại một thời điểm cụ thể. Việc quản lý trạng thái là một phần rất quan trọng trong việc xây dựng các chương trình hướng đối tượng, bởi vì **hành vi của đối tượng phụ thuộc trực tiếp vào trạng thái hiện tại của nó**.
Trong Python, trạng thái của một đối tượng thường được biểu diễn thông qua **các thuộc tính (attributes)** được định nghĩa bên trong lớp và gán cho từng thể hiện (instance). Trạng thái này có thể thay đổi trong quá trình thực thi chương trình thông qua các phương thức (methods).

2.3.1. Khởi tạo trạng thái ban đầu

Khi một đối tượng được khởi tạo từ lớp, trạng thái ban đầu thường được thiết lập thông qua phương thức __init__(), còn gọi là **constructor**. Đây là nơi mà các biến thể hiện (instance variables) được gán giá trị ban đầu.

2.3.2. Truy cập và thay đổi trạng thái

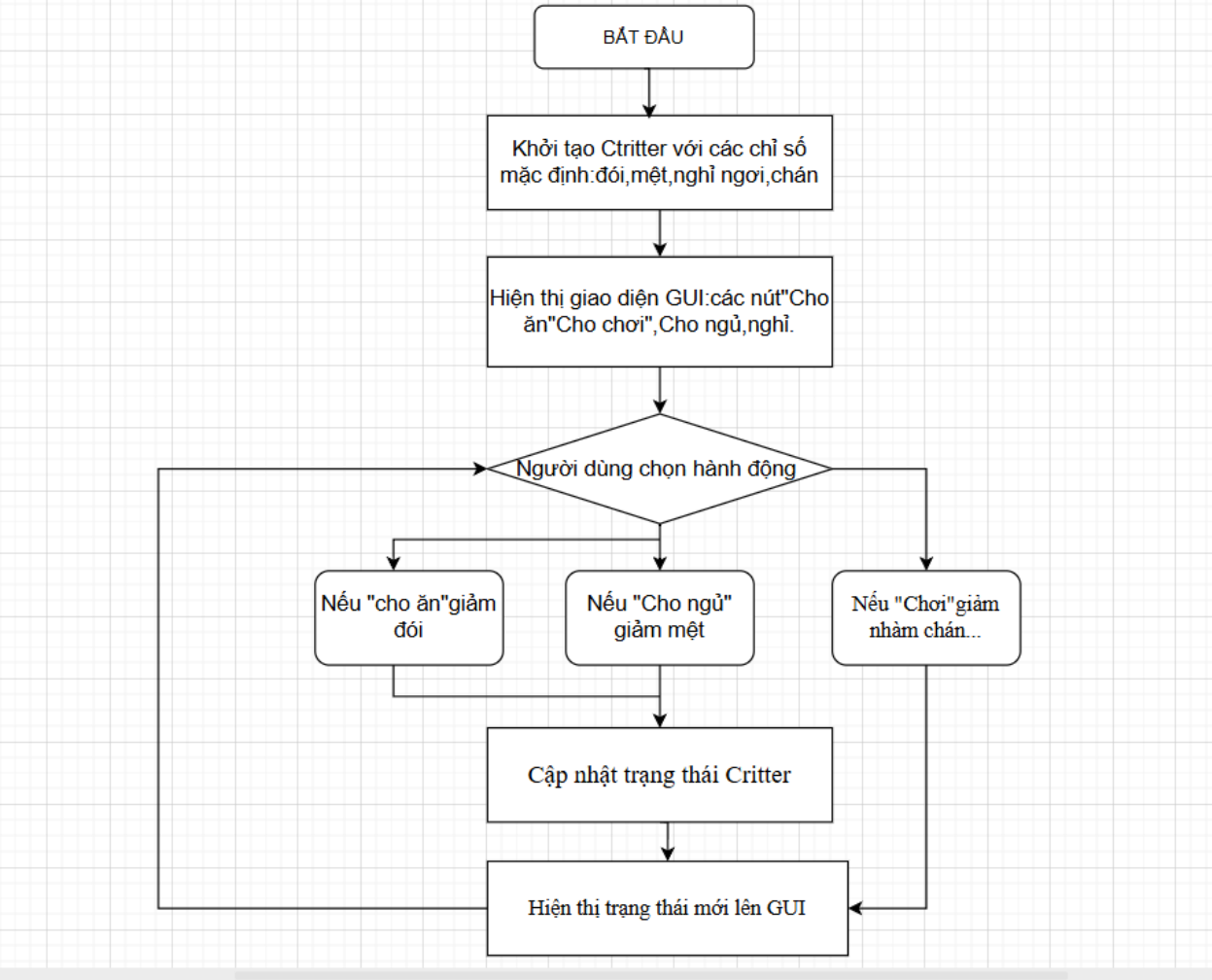
Trạng thái của đối tượng có thể được truy cập hoặc thay đổi bằng cách sử dụng cú pháp self.ten_bien bên trong lớp, hoặc doi_tuong.ten_bien bên ngoài lớp.
python
Việc thay đổi trạng thái nên được thực hiện thông qua các phương thức của lớp để đảm bảo tính an toàn và tuân thủ nguyên tắc đóng gói (encapsulation**)**.

2.3.3. Quản lý trạng thái thông qua phương thức

Việc điều khiển thay đổi trạng thái nên được thực hiện thông qua các **phương thức công khai (public methods)**, qua đó ta có thể thêm kiểm tra tính hợp lệ, ghi log, hoặc thực hiện các tác vụ liên quan khác.

CHƯƠNG 3 THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

3.1. Sơ đồ khối hệ thống



Hình 3.1.Sơ đồ khối hệ thống

3.3.1. Bắt đầu chương trình

- Khi ứng dụng được khởi động, hệ thống sẵn sàng tiếp nhận tương tác từ người dùng.
- Không gian giao diện người dùng được khởi tạo, nhưng chưa có sinh vật (Critter) nào tồn tại.

3.3.2. Khởi tạo Critter với các chỉ số mặc định

Hệ thống cho phép người dùng tạo một Critter mới.
Khi người dùng nhấn nút "Tạo Critter", hệ thống sẽ khởi tạo một đối tượng Critter với các chỉ số mặc định:

- **Đói (hunger)**: mức độ đói của sinh vật, khởi tạo ở giá trị trung bình (ví dụ: 5/10).
- **Mệt (fatigue)**: mức độ mệt mỏi, cũng khởi tạo ở giá trị mặc định.
- **Nhàm chán (boredom)**: biểu thị mức độ cần được chơi đùa hoặc tương tác.

3.3.3. Hiển thị giao diện GUI với các lựa chọn

Sau khi tạo Critter, giao diện người dùng sẽ hiện các nút tương tác:

- **Cho ăn**: để giảm chỉ số đói.
- **Cho ngủ**: để giảm chỉ số mệt.
- **Chơi**: để giảm chỉ số nhàm chán.

Các nút này là các hành động chủ đạo mà người dùng có thể thực hiện để chăm sóc sinh vật Critter.

3.3.4. Người dùng chọn hành động

Người dùng có thể nhấn bất kỳ nút nào để thực hiện hành động tương ứng.
Hệ thống sẽ kiểm tra lựa chọn của người dùng và phản hồi phù hợp:

- Nếu chọn **Cho ăn** → hệ thống gọi hàm xử lý giảm đói.
- Nếu chọn **Cho ngủ** → hệ thống xử lý giảm mệt.
- Nếu chọn **Chơi** → hệ thống giảm mức độ nhàm chán.

3.3.5. Cập nhật trạng thái Critter

Sau mỗi hành động, hệ thống cập nhật lại trạng thái nội tại của Critter:
Ví dụ: sau khi ăn, chỉ số hunger giảm, nhưng có thể fatigue hoặc boredom tăng nhẹ (nếu mô phỏng phức tạp hơn).
Việc cập nhật giúp trạng thái luôn phản ánh đúng tình trạng hiện tại của sinh vật.

3.3.6. Hiển thị trạng thái mới lên GUI

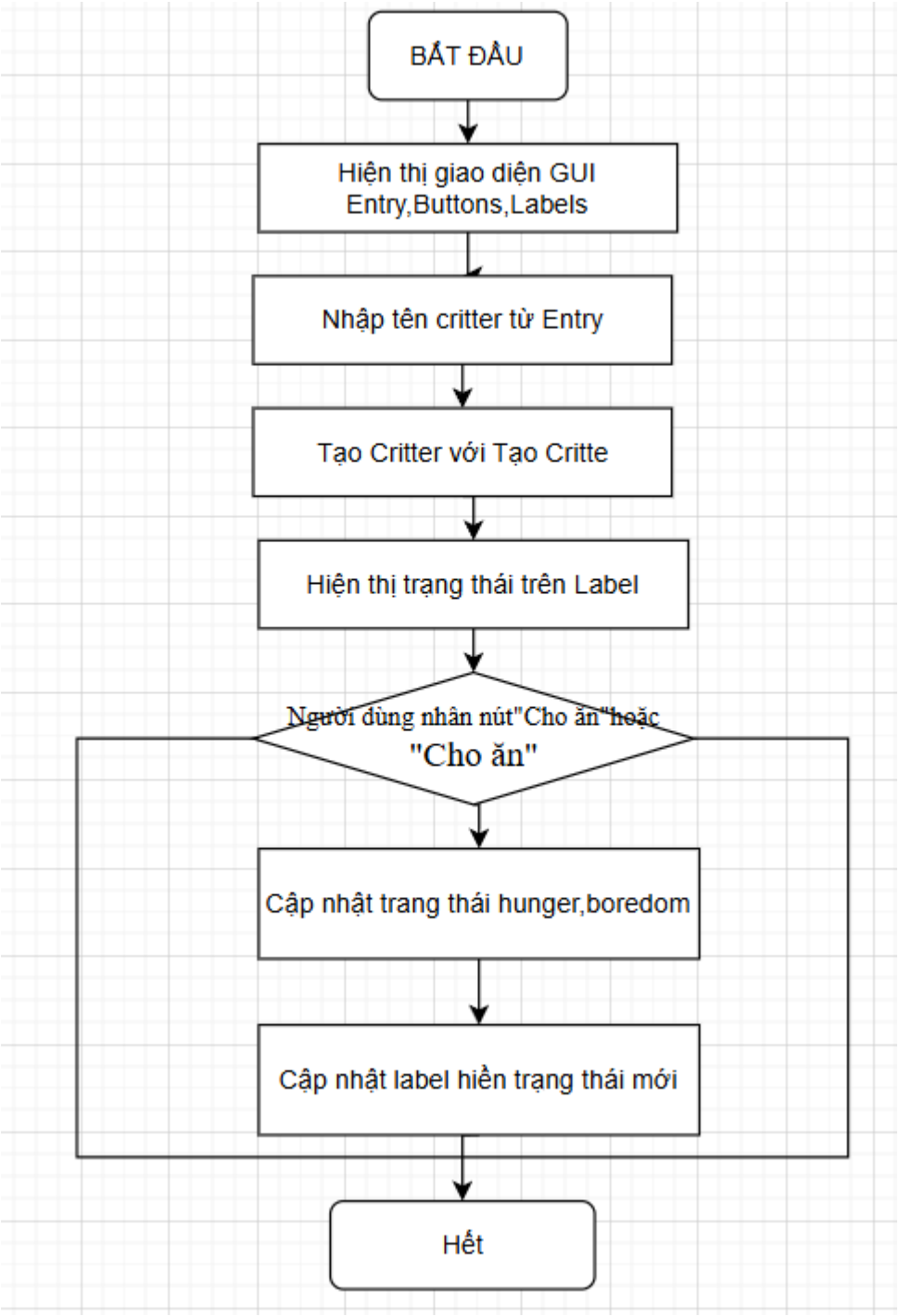
Trạng thái mới (hunger, fatigue, boredom) sẽ được hiển thị dưới dạng văn bản hoặc biểu đồ nhỏ trên giao diện.
Người dùng có thể nhìn thấy sự thay đổi sau mỗi lần tương tác, từ đó quyết định hành động tiếp theo.

3.3.7. Vòng lặp chăm sóc

Sau khi trạng thái được cập nhật, hệ thống quay trở lại bước chọn hành động.

Cho phép người dùng tiếp tục chăm sóc Critter cho đến khi đóng ứng dụng hoặc sinh vật đạt trạng thái tối ưu (hoặc bị bỏ đói nếu thiết kế có logic kiểm tra).

3.2. Sơ đồ thuật toán chính



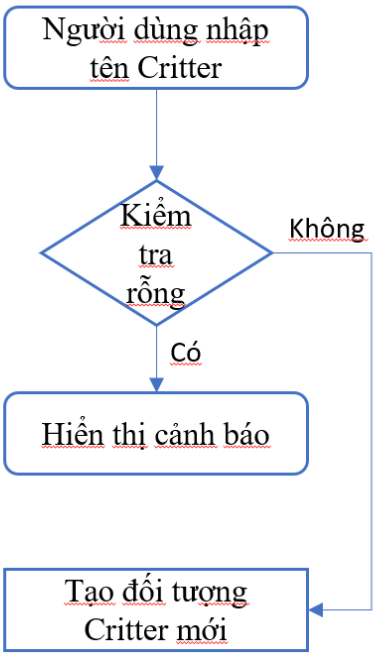
Hình 3.2 sơ đồ thuật toán

3.2.1.Thuật toán : Tạo Critter

Mục đích: Tạo đối tượng Critter mới từ tên do người dùng nhập.

Bước thực hiện:

- ❖ Người dùng nhập tên vào ô Entry.
- ❖ Kiểm tra tên có bị để trống không.
- ❖ Nếu trống → hiển thị cảnh báo.
- ❖ Nếu hợp lệ → tạo đối tượng Critter mới, thêm vào danh sách critters, và hiển thị trên Listbox.



Hình 3.2.1. Sơ đồ thuật toán tạo Critter

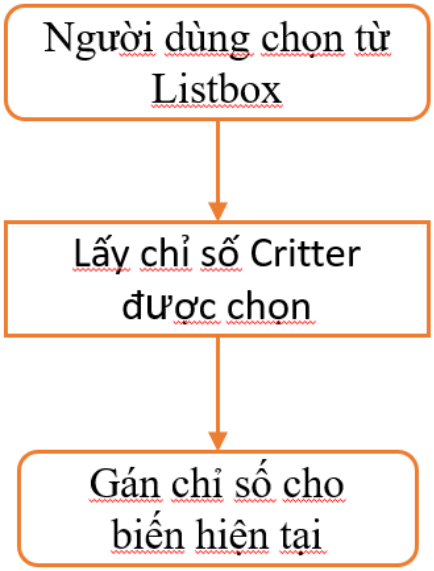
3.2.2.Thuật toán: Chọn Criitter

Mục đích: Ghi nhận Critter mà người dùng chọn để tương tác.

Bước thực hiện:

- ❖ Khi người dùng nhấp vào một dòng trong Listbox, hệ thống lấy chỉ số dòng đó.

- Lưu chỉ số này vào biến current_index để biết người dùng đang làm việc với Critter nào.



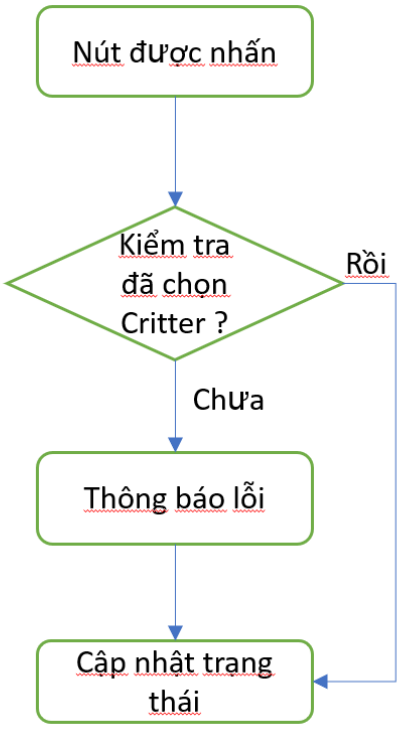
Hình 3.2.2. Sơ đồ thuật toán chọn Critter

3.2.3. Thuật toán :Cho Feed/Play/Sleep

Mục đích: Cập nhật trạng thái Critter khi người dùng nhấn nút tương ứng.

Bước thực hiện:

- Kiểm tra xem người dùng đã chọn Critter chưa (current_index khác None).
- Nếu chưa chọn → hiện thông báo lỗi.
- Nếu đã chọn:
- Gọi phương thức tương ứng (feed(), play(), sleep()) của Critter đang chọn.
- Cập nhật trạng thái trên giao diện bằng display_status().



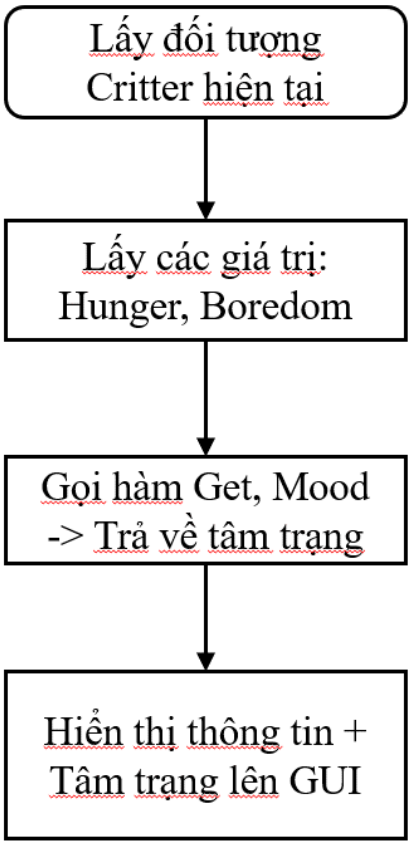
Hình 3.2.3. Sơ đồ thuật toán ăn/chơi/ngủ

3.2.4. Thuật toán :Hiển thị trạng thái Critter

Mục đích: Hiển thị thông tin chi tiết về Critter được chọn lên GUI.

Bước thực hiện:

- ❖ Lấy đối tượng Critter theo chỉ số current_index.
- ❖ Truy xuất thuộc tính hunger, boredom, gọi hàm get_mood() để lấy tâm trạng.
- ❖ Hiển thị đầy đủ thông tin lên hai nhãn (status_label và mood_label).



Hình 3.2.4. Sơ đồ thuật toán hiển thị trạng thái Critter

3.3.Cấu trúc dữ liệu

3.3.1. Bảng Critter

Tên trường	Kiểu dữ liệu
name	String
hunger	Integer
boredom	Integer

Bảng 3.3.1.Critter

3.3.2. Danh sách Critter

Tên biến: self.critters

Kiểu: List

Mỗi phần tử: một đối tượng Critter có đầy đủ thông tin như trên.

Vai trò: lưu tất cả các Critter được tạo bởi người dùng trong phiên làm việc hiện tại.

3.4. Các hàm chính trong chương trình,

3.4.1. Hàm khởi tạo lớp Critter

```
class Pet:
    def __init__(self, name):
        self.name = name
        self.hunger = 50
        self.energy = 50
        self.happiness = 50
```

Hình 3.4.1.hàm khởi lớp Critter

Dùng để khởi tạo một đối tượng Critter với tên, mức đói (***hunger***) và mức chán (***boredom***) ban đầu bằng 0

3.4.2. Hàm hành động của Critter

```
def feed(self):
    self.hunger = max(0, self.hunger - 20)

def sleep(self):
    self.energy = min(100, self.energy + 30)

def rest(self):
    self.energy = min(100, self.energy + 10)
    self.hunger = min(100, self.hunger + 5)

def play(self):
    if self.energy >= 10:
        self.happiness = min(100, self.happiness + 20)
        self.energy -= 10
        return True
    return False

def talk(self):
    phrases = [
        "Tôi yêu bạn!", "Chơi với tôi nha!", "Meo meo meo!",
        "Tôi đói rồi đấy...", "Tôi mệt lắm!"
    ]
    return random.choice(phrases)
```

Hình 3.4.2. .Hàm hành động của Critter

feed(): Giảm hunger đi 1 đơn vị (không giảm dưới 0).
play(): Tăng hunger lên 2, giảm boredom đi 1 (không giảm dưới 0).
sleep(): Tăng cả hunger và boredom thêm 1.

3.4.3.Hàm xác định tâm trạng

```
def talk(self):
    phrases = [
        "Tôi yêu bạn!", "Chơi với tôi nha!", "Meo meo meo!",
        "Tôi đói rồi đấy...", "Tôi mệt lắm!"
    ]
    return random.choice(phrases)

def get_status(self):
    return f"Đói: {self.hunger}/100 | Năng lượng: {self.energy}/100 | Vui vẻ: {self.happiness}/100"
```

Hình 3.4.3.Hình xác định tâm trạng

Trả về tâm trạng hiện tại của Critter dựa trên chỉ số **hunger** và **boredom**

3.4.4 Hàm tạo Critter

```
def get_status(self):
```

Hình 3.4.4.hàm tạo critter

Lấy tên người dùng nhập, tạo đối tượng Critter, thêm vào danh sách **self.critters**, và cập nhật **Listbox**.

3.4.5. Hàm chọn Critter trong danh sách

```
# Các nút chức năng
self.feed_btn = tk.Button(self.buttons_frame, text="🍗 Cho Ăn", width=15, command=self.feed, bg="#fcd5ce")
self.feed_btn.grid(row=0, column=0, padx=10, pady=5)

self.sleep_btn = tk.Button(self.buttons_frame, text="😴 Ngủ", width=15, command=self.sleep, bg="#d0f4de")
self.sleep_btn.grid(row=0, column=1, padx=10, pady=5)

self.rest_btn = tk.Button(self.buttons_frame, text="🛋️ Nghỉ Ngơi", width=15, command=self.rest, bg="#dbe7f0")
self.rest_btn.grid(row=1, column=0, padx=10, pady=5)

self.play_btn = tk.Button(self.buttons_frame, text="🏀 Chơi", width=15, command=self.play, bg="#ffcad4")
self.play_btn.grid(row=1, column=1, padx=10, pady=5)

self.talk_btn = tk.Button(root, text="💬 Nói chuyện", font=("Arial", 13), bg="#e0bbe4", command=self.talk)
self.talk_btn.pack(pady=20)
```

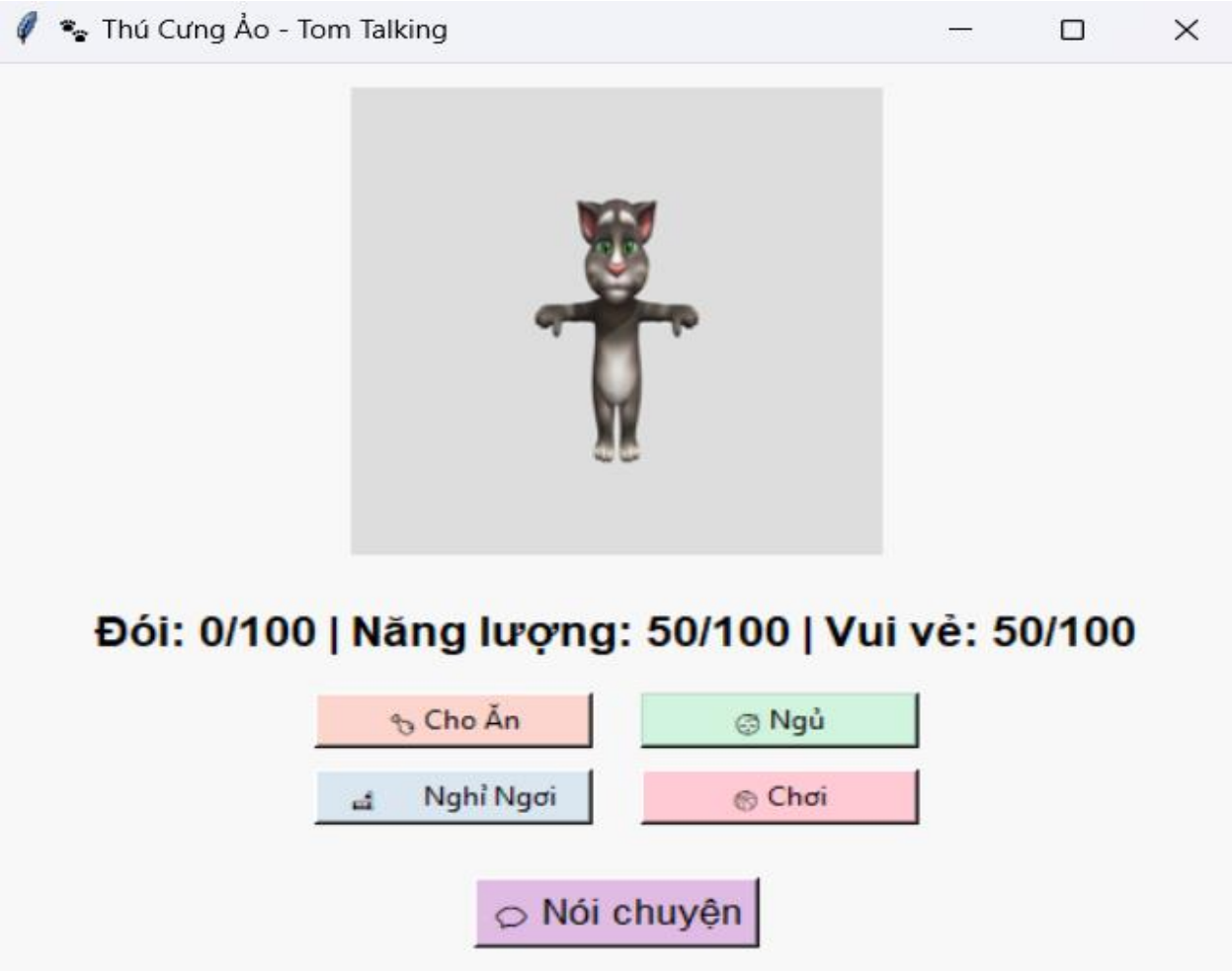
Hình 3.4.5. danh sách chức năng

Khi người dùng chọn một ***Critter*** trong ***Listbox***, hàm lấy chỉ số và cập nhật giao diện hiển thị thông tin.

CHƯƠNG 4. THỰC NGHIỆM VÀ KẾT LUẬN

4.1. Kết quả thực nghiệm

4.1.1. Chức năng tạo Critter và Lưu vào danh sách



Hình 4.1.1.Giao diện đề mô

```python
class PetApp:
    def __init__(self, root):
        self.pet = Pet("Tom")
        pygame.mixer.init()

        root.title("🐾 Thú Cưng Ảo - Tom Talking")
        root.geometry("520x650")
        root.configure(bg="#f8f8f8")
```

Hình 4.1.1. đặt tên và đổi tên thú cưng
ĐỀ MÔ

4.1.2. Chức năng hành động cho từng Critter và Tâm trạng của Critter

Hình 4.1.2. Chức năng hành động cho từng Critter và Tâm trạng của Critter
ĐỀ MÔ

```
# Các nút chức năng
self.feed_btn = tk.Button(self.buttons_frame, text="🍗 Cho Ăn", width=15, command=self.feed, bg="#fcd5ce")
self.feed_btn.grid(row=0, column=0, padx=10, pady=5)

self.sleep_btn = tk.Button(self.buttons_frame, text="😴 Ngủ", width=15, command=self.sleep, bg="#d0f4de")
self.sleep_btn.grid(row=0, column=1, padx=10, pady=5)

self.rest_btn = tk.Button(self.buttons_frame, text="🛋️ Nghỉ Ngơi", width=15, command=self.rest, bg="#dbe7f0")
self.rest_btn.grid(row=1, column=0, padx=10, pady=5)

self.play_btn = tk.Button(self.buttons_frame, text="🏀 Chơi", width=15, command=self.play, bg="#ffcad4")
self.play_btn.grid(row=1, column=1, padx=10, pady=5)

self.talk_btn = tk.Button(root, text="💬 Nói chuyện", font=("Arial", 13), bg="#e0bbe4", command=self.talk)
self.talk_btn.pack(pady=20)
```

Hình 4.1.2.ĐỀ MÔ CODE

4.2. Sản phẩm đã làm đã làm được những gì

4.2.1.Tạo mới thú ảo Critter thành công

Người dùng có thể dễ dàng **khởi tạo một Critter mới** bằng cách nhập tên qua ô nhập liệu. Hệ thống sẽ lưu tên này và khởi tạo các chỉ số trạng thái mặc định như:

- **Hunger (mức độ đói)**: bắt đầu từ 0.
- **Boredom(mức độ chán)**:bắt đầu từ 0.
  Điều này cho thấy lớp Critter đã được xây dựng đúng nguyên tắc đóng gói (encapsulation), khởi tạo đối tượng với thuộc tính riêng biệt.

4.2.2.Giao tiếp với Critter qua các nút chức năng

Ứng dụng cung cấp giao diện trực quan với 3 nút chính:

- **Tạo Critter**
- **Cho ăn Critter**
- **Cho ngủ Critter**
  Các nút này thực hiện lời gọi tới các hàm tương ứng của lớp Critter, giúp người dùng có thể tương tác với thú ảo một cách đơn giản và hiệu quả.

4.2.3.Cập nhật trạng thái theo thời gian

Mỗi hành động của người dùng (như cho ăn hoặc cho ngủ) không chỉ tác động đến trạng thái chính của Critter mà còn **kích hoạt hàm pass_time()**, tăng dần các chỉ số **hunger** và **boredom** theo thời gian.
Điều này giúp mô phỏng thực tế rằng một sinh vật sống sẽ ngày càng đói và buồn chán nếu không được chăm sóc.

4.2.4Hiển thị tâm trạng và các chỉ số rõ ràng

Thông qua hàm get_status() và get_mood(), chương trình có thể đánh giá tâm trạng của Critter và hiển thị cho người dùng:

- Mức độ đói
- Mức độ chán
- Tâm trạng tổng hợp: Rất vui , Bình thường , Khó chịu , Tức giận
  Giao diện hiển thị rõ ràng bằng các dòng văn bản trên Label, giúp người dùng hiểu được tình trạng hiện tại của thú ảo.

4.2.5.Giao diện đồ họa thân thiện, dễ thao tác

Ứng dụng sử dụng **thư viện Tkinter**, tạo ra cửa sổ ứng dụng với bố cục rõ ràng, thao tác đơn giản. GUI đáp ứng đầy đủ:

- Nhập liệu
- Nút bấm tương tác
- Hiển thị thông tin phản hồi
  Điều này hỗ trợ rất tốt cho người mới học lập trình và giúp trải nghiệm người dùng mượt mà.

4.3.Những điều học được

- Cách khai báo và sử dụng lớp (class) trong Python
- Biết cách định nghĩa một lớp để mô phỏng một đối tượng cụ thể (ở đây là một Critter).
- Hiểu được cấu trúc một lớp bao gồm: hàm khởi tạo (__init__) và các phương thức xử lý hành vi.
- Biết sử dụng từ khóa self để tham chiếu đến chính đối tượng đang được xử l
- Biết cách sử dụng các widget cơ bản như Entry, Button, Listbox, Label, Frame.
- Nắm được cách gắn sự kiện, xử lý hành vi người dùng và cập nhật giao diện động.
- Rèn luyện tư duy tổ chức mã, phân chia chức năng rõ ràng giữa logic và giao diện.

4.4. Hướng cải tiến trong tương lai

Trong tương lai, chương trình Critter Caretaker có thể được cải tiến toàn diện bằng cách áp dụng sâu hơn các tính năng nâng cao của Python.
Chẳng hạn như sử dụng cơ sở dữ liệu SQLite hoặc file JSON để lưu trữ và truy xuất trạng thái của các Critter qua các phiên làm việc khác nhau, từ đó tạo ra trải nghiệm liên tục cho người dùng
Tiếp theo là tăng cường khả năng mô phỏng thời gian thực bằng cách áp dụng đa luồng (multithreading hoặc after() trong Tkinter**)** để các chỉ số như đói và chán có thể tăng dần theo thời gian kể cả khi người dùng không thao tác, giúp thú ảo hoạt động giống thật hơn.
Chương trình cũng có thể mở rộng khả năng quản lý nhiều Critter cùng lúc bằng cách sử dụng danh sách các đối tượng (list of objects) và xây dựng giao diện lựa chọn, ví dụ như combobox hoặc danh sách động, từ đó cho phép nuôi và chăm sóc cả đàn thú với trạng thái riêng biệt; đồng thời có thể áp dụng tính kế thừa và đa hình trong OOP
Ví dụ tạo ra các lớp con như HappyCritter, LazyCritter, AngryCritter với hành vi và chỉ số đặc trưng, giúp chương trình trở nên sinh động hơn; bên cạnh đó, việc tách riêng phần xử lý logic (business logic) ra khỏi phần giao diện bằng mô hình MVC (Model-View-Controller) cũng là một bước quan trọng giúp mã nguồn dễ bảo trì.
Mở rộng và tích hợp về sau; ngoài ra, có thể cải tiến hiệu năng và độ ngắn gọn của mã bằng cách ứng dụng các cú pháp Python hiện đại như biểu thức lambda, hàm map/filter, hoặc decorator để tạo các hành vi linh hoạt hơn, từ đó nâng cao chất lượng phần mềm; cuối cùng.

KẾT LUẬN

Qua quá trình thiết kế và xây dựng chương trình Critter Caretaker với giao diện đồ họa sử dụng ngôn ngữ lập trình Python và thư viện Tkinter, em đã hoàn thành được một ứng dụng đơn giản nhưng giàu tính thực hành, mô phỏng quá trình chăm sóc một sinh vật ảo (Critter) theo thời gian.

Chương trình cho phép người dùng nhập tên Critter, cho ăn, cho chơi hoặc để Critter tự thay đổi trạng thái theo thời gian, đồng thời hiển thị các trạng thái hiện tại như mức độ đói, buồn chán và tâm trạng thông qua giao diện thân thiện và dễ sử dụng. Những tính năng này không chỉ đáp ứng đúng yêu cầu bài toán đặt ra trong chương trình học mà còn góp phần giúp người học hình dung rõ ràng hơn về cách xây dựng một ứng dụng tương tác, có tính phản hồi và có thể mở rộng sau này.

Việc hoàn thiện chương trình này giúp em củng cố rất nhiều kiến thức lý thuyết đã học trong môn học, đặc biệt là trong lập trình hướng đối tượng (OOP) với Python, bao gồm cách định nghĩa lớp, tạo đối tượng, khởi tạo thuộc tính và triển khai các phương thức thao tác với đối tượng.

Em cũng được thực hành và làm quen với xây dựng giao diện người dùng (GUI) bằng Tkinter – một thư viện phổ biến và trực quan cho các ứng dụng Python đơn giản, từ đó hiểu được mối liên hệ giữa giao diện và phần xử lý logic phía sau. Ngoài ra, trong quá trình phát triển, em cũng phải tự nghiên cứu, gỡ lỗi, chỉnh sửa giao diện sao cho hợp lý và tối ưu điều này giúp em rèn luyện kỹ năng tư duy giải quyết vấn đề, kỹ năng kiểm thử và cải tiến chương trình.

Tuy chương trình hiện tại mới chỉ đáp ứng ở mức cơ bản, song nó đã tạo nền tảng tốt để từ đây có thể tiếp tục mở rộng và phát triển thêm nhiều tính năng nâng cao như: lưu trữ trạng thái Critter bằng cơ sở dữ liệu, hỗ trợ nhiều Critter cùng lúc, xây dựng thêm nhiều hành vi mới cho Critter, tạo giao diện sinh động hơn, hoặc thậm chí chuyển thành ứng dụng web hay di động.

Những hướng phát triển này hoàn toàn khả thi nếu kết hợp kiến thức Python nâng cao với mô hình tổ chức mã nguồn rõ ràng, có thể bảo trì lâu dài.

TÀI LIỆU KHAM KHẢO

1. “Programming Python” – Mark Lutz
   Quyển sách chuyên sâu về Python, đặc biệt phần GUI với Tkinter rất hữu ích

2. “Python GUI Programming with Tkinter” – Alan D. Moore
   Cung cấp ví dụ thực hành, hướng dẫn chi tiết thiết kế giao diện.

3. “Python for Beginners: Learn Python in One Day and Learn It Well” – Jamie Chan
   Dành cho người mới, giúp nắm chắc các khái niệm hướng đối tượng (OOP) cần thiết.