**Bài tập 1: Áp dụng tính kế thừa, trừu tượng, và đa hình**

Đề bài:

Phát triển chương trình quản lý động vật trong sở thú từ bài tập trong buổi học:

**1. Thêm một phương thức action() trong lớp trừu tượng Animal.**

• Phương thức này sẽ được ghi đè ở từng lớp con để mô tả hành động đặc trưng của mỗi loại động vật:

* Lion: in ra “The lion is hunting.”
* Elephant: in ra “The elephant is spraying water.”
* Monkey: in ra “The monkey is climbing trees.”

**2. Thêm phương thức showInfo() để hiện thị thông tin chi tiết của mỗi con vật gồm tên, tuổi**

**3. Tạo class ManagerAnimal cho phép thực hiện các yêu cầu:**

1. có 1 danh sách các động vậy Lion,Elephant,Monkey, cho phép người dùng nhập vào n, chính là
2. tổng số lượng các động vật mà sở thú quản lý
3. nhập vào danh sách các động vật củ sở thú và cho phép người dùng lựa chọn
   1. nhập vào Lion
   2. nhập vào Elephant
   3. nhập vào Monkey
   4. Dừng nhập vào
4. chú ý: nhập vào là bao gồm cả các thông tin tên, tuổi
5. Hiển thị danh sách tất cả các động vật trong sở thú
6. Thêm 1 con vật nếu danh sách đầy, phải mở rộng danh sách, mặc định mở rộng 10 phần tử
7. Cho phép nhập tên con vật, hiển thị thông tin con vật nếu có, không phân biệt hoa thường
8. Cho phép xóa con vật theo tên

**Bài 2: Thiết kế lớp Account Đề bài:**

Thiết kế một lớp Account quản lý thông tin tài khoản ngân hàng với các yêu cầu sau:

Các thuộc tính:

* int id: ID tài khoản (mặc định là 0).
* double balance: Số dư tài khoản (mặc định là 0).
* double annualInterestRate: Lãi suất hàng năm (mặc định là 0, tính theo phần trăm, ví dụ: 4.5%).
  + Lưu ý: Lãi suất hàng năm là tỷ lệ phần trăm, bạn cần chia nó cho 100 để tính toán.
* Date dateCreated: Ngày tạo tài khoản (mặc định là ngày hiện tại).

Yêu cầu:

1. Viết hàm khởi tạo:
   1. Hàm không tham số (khởi tạo mặc định).
   2. Hàm có tham số để khởi tạo đầy đủ thông tin (id, balance, annualInterestRate).
2. Viết getter và setter cho tất cả các thuộc tính.
3. Thêm các phương thức:
   1. double getMonthlyInterestRate(): Trả về lãi suất hàng tháng (lãi suất hàng năm chia cho 12).
   2. double getMonthlyInterest(): Trả về tiền lãi hàng tháng (số dư tài khoản nhân với lãi suất hàng tháng).
   3. void withdraw(double amount): Rút một số tiền nhất định từ tài khoản.
   4. void deposit(double amount): Gửi một số tiền nhất định vào tài khoản.
4. Viết chương trình test:

• Tạo một đối tượng Account với:

• ID: 1122

• Số dư: 20,000 USD

• Lãi suất hàng năm: 4.5%

• Thực hiện:

• Rút 2,500 USD.

• Gửi 3,000 USD.

• In ra:

• Số dư hiện tại.

• Tiền lãi hàng tháng.

• Ngày tài khoản được tạo.

**Bài 3: Mô phỏng máy ATM Đề bài: Xây dựng một chương trình mô phỏng máy ATM bằng cách sử dụng lớp Account đã được thiết kế trong bài 2.**

Yêu cầu:

1. Khởi tạo hệ thống:

• Tạo một mảng gồm 10 tài khoản với:

• ID lần lượt từ 0 đến 9.

• Số dư ban đầu là $100 cho tất cả các tài khoản.

2. Chương trình mô phỏng:

• Khi hệ thống khởi chạy:

• Yêu cầu người dùng nhập ID tài khoản.

• Nếu nhập sai ID (không nằm trong khoảng từ 0 đến 9), yêu cầu nhập lại.

• Khi ID hợp lệ, hiển thị menu chính với các lựa chọn sau:

1. Xem số dư hiện tại: In ra số dư của tài khoản.
2. Rút tiền: Nhập số tiền cần rút (kiểm tra nếu số tiền rút lớn hơn số dư thì báo lỗi).
3. Gửi tiền: Nhập số tiền cần gửi.
4. Thoát: Quay lại màn hình nhập ID.

3. Quy trình thực hiện:

• Sau khi người dùng thoát khỏi menu chính (lựa chọn 4), hệ thống sẽ quay lại yêu cầu nhập ID tài khoản.

• Hệ thống sẽ chỉ kết thúc khi người dùng tắt chương trình.

Ví dụ giao diện:

Nhập ID tài khoản (0-9): 3

===== MENU CHÍNH =====

1: Xem số dư

2: Rút tiền

3: Gửi tiền

4: Thoát Lựa chọn của bạn: 1

Số dư hiện tại: $100

Lựa chọn của bạn:

2 Nhập số tiền cần rút: 50

Rút tiền thành công.

Số dư mới: $50

Lựa chọn của bạn: 4

Nhập ID tài khoản (0-9): 5

===== MENU CHÍNH ===== ...

Lưu ý:

• Kiểm tra kỹ các trường hợp:

• Người dùng nhập sai ID.

• Rút tiền vượt quá số dư.

• Nhập số tiền âm (báo lỗi).