**[Cơ bản 1] Tổng quan về tính trừu tượng**

**1. Mục tiêu:**

* Giúp sinh viên hiểu khái niệm **tính trừu tượng** (Abstraction) trong lập trình hướng đối tượng (OOP)
* Giúp sinh viên nắm được vai trò của tính trừu tượng trong việc ẩn giấu chi tiết triển khai và chỉ hiển thị chức năng cần thiết
* Phát triển khả năng áp dụng tính trừu tượng thông qua các câu hỏi lý thuyết dưới dạng điền vào ô trống

**2. Mô tả:**

Bài tập gồm các câu hỏi dạng lý thuyết liên quan đến tính trừu tượng. Sinh viên sẽ điền vào các ô trống để hoàn thiện định nghĩa, lợi ích và vai trò của tính trừu tượng trong lập trình. Các câu hỏi sẽ tập trung vào:

* Định nghĩa tính trừu tượng.
* Ví dụ thực tế minh họa tính trừu tượng.
* Vai trò của các lớp trừu tượng hoặc giao diện (interface) trong lập trình.

**Câu hỏi:**

1. Tính trừu tượng trong lập trình hướng đối tượng là quá trình \_\_\_\_\_\_\_\_\_\_ các chi tiết phức tạp và chỉ hiển thị các \_\_\_\_\_\_\_\_\_\_ cần thiết cho người dùng.
2. Trong Java, tính trừu tượng có thể được thực hiện bằng cách sử dụng \_\_\_\_\_\_\_\_\_\_ hoặc \_\_\_\_\_\_\_\_\_\_.
3. Một ví dụ thực tế về tính trừu tượng là \_\_\_\_\_\_\_\_\_\_, nơi người dùng chỉ cần biết cách sử dụng mà không cần hiểu chi tiết bên trong.

**[Cơ bản 2] Định nghĩa và vai trò của tính trừu tượng**

**1. Mục tiêu:**

* Giúp sinh viên hiểu rõ khái niệm **tính trừu tượng** trong lập trình hướng đối tượng (OOP)
* Xác định được vai trò của tính trừu tượng trong việc tổ chức và quản lý mã nguồn hiệu quả
* Phát triển tư duy phân tích và áp dụng tính trừu tượng thông qua lý thuyết và ví dụ minh họa

**2. Mô tả:**

Bài tập yêu cầu sinh viên định nghĩa tính trừu tượng và phân tích vai trò của nó trong lập trình Java. Sinh viên sẽ cần:

* Định nghĩa khái niệm tính trừu tượng trong OOP.
* Nêu ra các cách Java hỗ trợ tính trừu tượng (lớp trừu tượng, giao diện).
* Liên hệ thực tế để giải thích ý nghĩa của tính trừu tượng trong việc xây dựng phần mềm.
* Phân tích ngắn gọn ưu điểm và hạn chế khi sử dụng tính trừu tượng.

***Yêu cầu:***

**Phần định nghĩa**: Sinh viên phải viết một đoạn văn ngắn (khoảng 3-4 câu) để định nghĩa tính trừu tượng và vai trò của nó.

**Phần phân tích**: Trả lời các câu hỏi:

* Tại sao cần tính trừu tượng trong phần mềm phức tạp?
* Sự khác biệt giữa lớp trừu tượng và giao diện trong Java là gì?

**Liên hệ thực tế**: Lấy ví dụ minh họa từ đời sống (máy ATM, điều khiển TV, ô tô, v.v.) để giải thích tính trừu tượng.

**[Cơ bản 3] Triển khai lớp trừu tượng đơn giản**

**1. Mục tiêu:**

* Hiểu cách tạo và sử dụng **lớp trừu tượng** trong Java
* Biết cách triển khai phương thức trừu tượng và phương thức thông thường trong lớp kế thừa
* Phân biệt được sự khác nhau giữa lớp trừu tượng và lớp thường
* Phát triển kỹ năng tổ chức mã nguồn, sử dụng tính kế thừa và đa hình trong lập trình Java

**2. Mô tả:**

Sinh viên sẽ thực hiện bài tập triển khai một hệ thống quản lý phương tiện giao thông bằng cách sử dụng **lớp trừu tượng**. Qua bài tập này, sinh viên sẽ:

* Tạo một lớp trừu tượng chứa các thuộc tính và phương thức cơ bản.
* Tạo các lớp con kế thừa từ lớp trừu tượng và ghi đè các phương thức trừu tượng.
* Viết mã để khởi tạo và sử dụng các đối tượng từ lớp con để hiểu cách hoạt động của tính kế thừa và đa hình.

***Yêu cầu:***

* Phần cài đặt lớp trừu tượng:
  + Tạo một lớp trừu tượng **Vehicle** với:
    - Các thuộc tính: name (kiểu String), speed (kiểu int).
    - Một phương thức trừu tượng: void displayInfo().
    - Một phương thức thông thường: void start() in ra "Vehicle is starting...".
* Phần cài đặt lớp con:
  + Tạo hai lớp kế thừa Vehicle:
    - Car: Ghi đè phương thức displayInfo() để in thông tin của xe hơi.
    - Bike: Ghi đè phương thức displayInfo() để in thông tin của xe máy.
* Phần kiểm tra chương trình:
  + Tạo lớp Main để:
    - Khởi tạo đối tượng Car và Bike với các giá trị cho name và speed.
    - Gọi các phương thức start() và displayInfo() trên các đối tượng.
* Kết quả:
  + Khi chạy chương trình, hiển thị đúng thông tin của từng phương tiện theo cách riêng của nó.

**[Cơ bản 4] Sử dụng interface để quản lý thiết bị điện tử**

**1. Mục tiêu:**

* Hiểu cách khai báo và triển khai **interface** trong Java
* Biết cách sử dụng interface để định nghĩa các hành vi chung cho các đối tượng khác nhau
* Áp dụng tính **đa hình** để xử lý các đối tượng triển khai từ interface
* Phân biệt được sự khác nhau giữa **interface** và **lớp trừu tượng** trong Java

**2. Mô tả:**

Sinh viên sẽ thực hiện bài tập tạo một chương trình Java để quản lý các thiết bị điện tử bằng cách sử dụng interface. Chương trình yêu cầu sinh viên:

* Tạo một interface định nghĩa các hành vi chung cho các thiết bị điện tử.
* Triển khai interface trong các lớp đại diện cho các thiết bị cụ thể.
* Sử dụng tính đa hình để thao tác với các đối tượng triển khai interface.

Thông qua bài tập này, sinh viên sẽ nắm vững cách tổ chức mã nguồn bằng cách sử dụng interface để tăng tính linh hoạt và khả năng mở rộng của chương trình.

***Yêu cầu:***

* Phần khai báo interface:
  + Tạo một interface Electronic với các phương thức:
    - void turnOn() (bật thiết bị).
    - void turnOff() (tắt thiết bị).
* Phần triển khai interface:
  + Tạo hai lớp TV và Radio triển khai interface Electronic.
    - Trong lớp TV, phương thức turnOn() in ra: "Turning on TV..." và turnOff() in ra: "Turning off TV...".
    - Trong lớp Radio, phương thức turnOn() in ra: "Turning on Radio..." và turnOff() in ra: "Turning off Radio...".
* Phần kiểm tra chương trình:
  + Tạo lớp Main để:
    - Khởi tạo một danh sách các thiết bị (TV và Radio).
    - Duyệt qua danh sách và gọi các phương thức turnOn() và turnOff() cho từng thiết bị.
* Kết quả:
  + Chương trình sẽ hiển thị các thao tác bật/tắt thiết bị theo thứ tự trong danh sách.

**[Khá 5] Triển khai interface colorable cho các lớp hình học**

**1. Mục tiêu:**

* Hiểu và áp dụng **interface** trong Java
* Triển khai **interface** trong các lớp hình học để mở rộng tính năng

**2. Mô tả:**

Trong bài tập này, bạn sẽ triển khai một **interface** có tên là Colorable để cung cấp khả năng tô màu cho các đối tượng hình học. Các lớp hình học như Circle, Rectangle, và Square sẽ cài đặt interface này để có thể thay đổi màu sắc của chúng.

***Yêu cầu:***

* Tạo interface Colorable:
  + Định nghĩa một phương thức trừu tượng void setColor(String color) để thiết lập màu sắc cho các đối tượng.
* Tạo lớp Circle, Rectangle, và Square:
  + Mỗi lớp này đại diện cho một hình học cụ thể.
  + Mỗi lớp cần có các thuộc tính để lưu trữ kích thước của hình học (ví dụ, bán kính cho Circle, chiều dài và chiều rộng cho Rectangle và Square).
  + Các lớp này phải **triển khai** interface Colorable, ghi đè phương thức setColor() để thiết lập màu sắc của hình học.
* Tạo lớp Main để kiểm tra chương trình:
  + Tạo các đối tượng của các lớp Circle, Rectangle, và Square.
  + Thử thiết lập màu sắc cho các hình này và in ra màu sắc của chúng.

**[Khá 6] Quản lý động vật trong sở thú bằng lớp trừu tượng**

**1. Mục tiêu:**

* Hiểu cách sử dụng **lớp trừu tượng** trong Java để định nghĩa các hành vi chung cho các đối tượng
* Áp dụng tính kế thừa và đa hình trong việc xây dựng các lớp con từ lớp trừu tượng
* Phát triển kỹ năng tổ chức mã nguồn, sử dụng các phương thức trừu tượng và ghi đè phương thức trong các lớp con
* Cải thiện khả năng thiết kế phần mềm khi làm việc với các đối tượng có quan hệ kế thừa

**2. Mô tả:**

Trong bài tập này, sinh viên sẽ xây dựng một hệ thống quản lý động vật trong sở thú. Cụ thể, sinh viên sẽ:

* Tạo một lớp trừu tượng Animal với các thuộc tính và phương thức chung cho các loài động vật.
* Tạo các lớp con kế thừa từ Animal, mỗi lớp con sẽ đại diện cho một loài động vật cụ thể và thực hiện các hành vi đặc trưng của nó như phát ra âm thanh và di chuyển.
* Sử dụng lớp Zoo để quản lý các loài động vật trong sở thú, áp dụng phương thức trừu tượng để thực hiện hành vi cho từng loài động vật.

Bài tập này giúp sinh viên hiểu sâu hơn về cách sử dụng **lớp trừu tượng**, đồng thời luyện tập việc xây dựng và sử dụng các đối tượng kế thừa.

***Yêu cầu:***

* Tạo lớp trừu tượng Animal:
  + Các thuộc tính:
    - name (kiểu String) – Tên của động vật.
    - age (kiểu int) – Tuổi của động vật.
  + Các phương thức trừu tượng:
    - void sound() – Phương thức để phát ra âm thanh của động vật.
    - void move() – Phương thức để mô tả hành động di chuyển của động vật.
* Tạo các lớp con kế thừa từ Animal:
  + **Lion:**
    - Ghi đè phương thức sound() để in "Roar!" (âm thanh của sư tử).
    - Ghi đè phương thức move() để in "The lion is running." (hành động di chuyển của sư tử).
  + **Elephant:**
    - Ghi đè phương thức sound() để in "Trumpet!" (âm thanh của voi).
    - Ghi đè phương thức move() để in "The elephant is walking." (hành động di chuyển của voi).
* Tạo lớp Zoo để quản lý động vật:
  + Lớp Zoo sẽ chứa một danh sách các đối tượng Animal (bao gồm các động vật như Lion, Elephant).
  + Duyệt qua danh sách và gọi các phương thức sound() và move() cho từng đối tượng động vật.
* Phần kiểm tra chương trình:
  + Tạo một lớp Main để kiểm tra chương trình, khởi tạo các đối tượng động vật (Lion, Elephant), thêm chúng vào danh sách của Zoo và gọi các phương thức sound() và move().

**[Giỏi 7] Triển khai lớp IEmployee và các lớp Employee và FulltimeEmployee và ParttimeEmployee**

**1. Mục tiêu:**

* Luyện tập triển khai các phương thức interface và sử dụng kết thừa

**2. Mô tả:**

* IEmployee là interface định nghĩa các hành vi chung của nhân viên, gồm calculateSalary() để tính lương và getName() để lấy tên.
* Employee là lớp trừu tượng kế thừa từ IEmployee, lưu trữ thông tin chung như tên (name) và lương theo giờ (paymentPerHour), đồng thời cung cấp setter và getter.
* PartTimeEmployee kế thừa Employee, có số giờ làm việc (workingHours) và tính lương theo công thức workingHours \* paymentPerHour.
* FullTimeEmployee kế thừa Employee, lương được tính cố định theo công thức 8 \* paymentPerHour vì nhân viên full-time luôn làm việc 8 giờ/ngày

***Yêu cầu:***

* **Interface IEmployee:**
  + calculateSalary() là phương thức trừu tượng để tính lương nhân viên
  + getName() là phương thức lấy tên nhân viên
* **Lớp trừu tượng Employee (abstract class):**
  + Đây là lớp kế thừa từ **interface IEmployee**
  + Đây là lớp trừu tượng lưu thông tin chung về nhân viên
  + name,paymentPerHour lần lượt là các thuộc tính chỉ tên và số tiền mà nhân viên nhận được trên 1 giờ làm việc.
  + **Employee**(name: String, paymentPerHour: int) là constructor.
  + setName(),getName(),setPaymentPerHour(), getPaymentPerHour() là các setter và getter.
  + int calculateSalary(): Phương thức trừu tượng
* **Lớp PartTimeEmployee:**
  + Đây là lớp kế thừa từ lớp trừu tượng **Employee**
  + workingHours là thuộc tính chỉ số giờ làm việc trên 1 ngày của nhân viên.
  + calculateSalary() là phương thức ghi đè từ lớp trừu tượng Employee, phương thức này trả về tiền lương của nhân viên **part-time** theo công thức: **workingHours \* paymentPerHour.**
* **Lớp FullTimeEmployee:**
  + Đây là lớp kế thừa từ lớp trừu tượng **Employee**
  + calculateSalary() là phương thức ghi đè từ lớp trừu tượng Employee, phương thức này trả về tiền lương của nhân viên full-time theo công thức: 8 \* paymentPerHour (nhân viên full time luôn làm việc 8 tiếng một ngày nên lớp FullTimeEmployee không có thuộc tính workingHours)

**[Giỏi 8] Quản lý hệ thống xe ô tô với lớp trừu tượng và interface**

**1. Mục tiêu:**

* Hiểu và áp dụng **lớp trừu tượng** và **interface** trong Java
* Thực hành xây dựng hệ thống quản lý với các lớp trừu tượng để mô phỏng hành vi của các loại xe ô tô
* Sử dụng **đa hình** để quản lý các đối tượng xe ô tô khác nhau

**2. Mô tả:**

Trong bài tập này, bạn sẽ tạo một hệ thống quản lý các loại xe ô tô trong một gara ô tô. Cụ thể, bạn sẽ tạo các lớp trừu tượng để đại diện cho các hành vi chung của xe ô tô, và sử dụng interface để thêm các hành vi đặc thù như đỗ xe hoặc nạp nhiên liệu.

* Tạo một lớp trừu tượng Car với các thuộc tính cơ bản như model, year, price, và các phương thức trừu tượng start() và stop().
* Tạo các lớp con ElectricCar và GasCar kế thừa từ lớp Car và triển khai các phương thức của lớp trừu tượng.
* Tạo một interface Refuelable với phương thức refuel() để thực hiện hành vi nạp nhiên liệu cho xe.
* Lớp ElectricCar, GasCar implements interface Refuelable để triển khai phương thức refuel().

***Yêu cầu:***

* **Lớp Car**: Cần phải là lớp trừu tượng với các thuộc tính và phương thức trừu tượng.
* **Interface Refuelable**: Tạo interface này với phương thức refuel().
* **Lớp ElectricCar và GasCar**: Cả hai lớp này phải triển khai các phương thức của lớp Car và implement interface Refuelable với hành vi cụ thể.
  + VD: Car tesla = new ElectricCar("Tesla Model 3", 2023, 50000);
    - Phương thức **start()** trả về: “Xe **điện** **Tesla Model 3 (Năm SX: 2023, Giá: $50000.0)** đã khởi động không tiếng ồn”
    - Phương thức **stop()**trả về: Xe **điện Tesla Model 3 (Năm SX: 2023, Giá: $50000.0)** đã dừng và ngắt kết nối động cơ.
    - Phương thức **refuel()** trả về: Xe **điện Tesla Model 3 (Năm SX: 2023, Giá: $50000.0)** đang sạc điện…
  + VD: Car toyota = new GasCar("Toyota Camry", 2022, 35000);
    - Phương thức **start()** trả về: “Xe **chạy xăng** **Toyota Camry (Năm SX: 2022, Giá: $35000.0)** đã khởi động với tiếng động cơ.”
    - Phương thức **stop()**trả về: Xe **chạy xăng Toyota Camry (Năm SX: 2022, Giá: $35000.0)** đã dừng và động cơ tắt
    - Phương thức **refuel()** trả về: Xe **chạy xăng Toyota Camry (Năm SX: 2022, Giá: $35000.0)** đang được đổ xăng...
* **Lớp Garage**: Tạo lớp này để quản lý các xe trong gara và gọi các phương thức như start(), stop(), và refuel().

**[Xuất sắc 9] Quản lý các loại tài khoản ngân hàng bằng lớp trừu tượng**

**1. Mục tiêu:**

* Hiểu và áp dụng **tính trừu tượng** trong Java để định nghĩa các phương thức chung cho các loại tài khoản ngân hàng khác nhau
* Thực hành sử dụng **lớp trừu tượng** và **tính kế thừa** để tạo ra các lớp con có hành vi đặc trưng cho từng loại tài khoản
* Cải thiện khả năng tổ chức mã nguồn và thiết kế phần mềm với các đối tượng phức tạp như tài khoản ngân hàng

**2. Mô tả:**

Trong bài tập này, sinh viên sẽ xây dựng một hệ thống quản lý các loại tài khoản ngân hàng. Chương trình yêu cầu sinh viên:

* Tạo một lớp trừu tượng BankAccount để định nghĩa các hành vi chung cho các tài khoản ngân hàng, chẳng hạn như nạp tiền, rút tiền và kiểm tra số dư.
* Tạo các lớp con kế thừa từ BankAccount, mỗi lớp con sẽ đại diện cho một loại tài khoản ngân hàng cụ thể như tài khoản vãng lai và tài khoản tiết kiệm.
* Áp dụng các phương thức trừu tượng và tính đa hình để thực hiện các giao dịch ngân hàng và hiển thị kết quả.

Bài tập này giúp sinh viên không chỉ hiểu về **lớp trừu tượng**, mà còn nâng cao kỹ năng trong việc sử dụng **tính kế thừa** và **đa hình** khi thiết kế các đối tượng phức tạp.

***Yêu cầu:***

* Tạo lớp trừu tượng BankAccount:
  + Các thuộc tính:
    - accountNumber (kiểu String) – Số tài khoản của khách hàng.
    - balance (kiểu double) – Số dư tài khoản.
  + Các phương thức trừu tượng:
    - void withdraw(double amount) – Rút tiền từ tài khoản.
    - void deposit(double amount) – Nạp tiền vào tài khoản.
    - void displayBalance() – Hiển thị số dư tài khoản.
* Tạo các lớp con kế thừa từ **BankAccount**:
  + **SavingAccount:**
    - Ghi đè phương thức withdraw() để cho phép rút tiền từ tài khoản tiết kiệm (tài khoản tiết kiệm sẽ không cho phép rút vượt quá số dư).
    - Ghi đè phương thức deposit() để nạp tiền vào tài khoản tiết kiệm.
    - Ghi đè phương thức displayBalance() để hiển thị số dư tài khoản tiết kiệm.
  + **CheckingAccount:**
    - Ghi đè phương thức withdraw() để cho phép rút tiền vượt quá số dư tài khoản, tạo ra số dư âm (tài khoản vãng lai cho phép rút tiền quá số dư hiện tại).
    - Ghi đè phương thức deposit() để nạp tiền vào tài khoản vãng lai.
    - Ghi đè phương thức displayBalance() để hiển thị số dư tài khoản vãng lai, bao gồm số dư âm nếu có.
* Tạo lớp **Bank** để quản lý tài khoản ngân hàng(Chứa tối đa 10 tài khoản):
  + Lớp Bank sẽ chứa một danh sách các tài khoản ngân hàng (bao gồm cả SavingAccount và CheckingAccount).
  + Lớp Bank cung cấp các phương thức:
    - addAccount(BankAccount account) – Thêm một tài khoản vào ngân hàng.
    - displayBalance() – Duyệt qua danh sách các tài khoản và thực hiện hiển thị số dư của các tài khoản có trong ngân hàng.
* Phần kiểm tra chương trình:
  + Tạo lớp Main để kiểm tra chương trình.
  + Khởi tạo các tài khoản **SavingAccount** và **CheckingAccount**.
  + Thêm các tài khoản vào ngân hàng(**Bank**)
  + Thực hiện các giao dịch
  + In ra số dư của từng tài khoản bằng displayBalance().

**[Xuất sắc 10] Xây dựng hệ thống nhân viên với các lớp trừu tượng và interface**

**1. Mục tiêu:**

* Hiểu cách sử dụng **interface** và **lớp trừu tượng** để mô phỏng các hành vi của các loại nhân viên
* Tạo và triển khai các lớp kế thừa để tái sử dụng mã trong hệ thống

**2. Mô tả:**

Bạn cần xây dựng một hệ thống quản lý nhân viên trong một công ty. Cụ thể, bạn sẽ tạo các lớp trừu tượng để mô phỏng hành vi chung của các nhân viên, và sử dụng interface để thêm các hành vi đặc thù như tính lương hay tổ chức cuộc họp.

* Tạo một lớp trừu tượng Employee với các thuộc tính như name, salary, và các phương thức trừu tượng như work() và takeVacation().
* Tạo một interface AttendMeeting với phương thức attendMeeting() để các nhân viên tham gia cuộc họp.
* Tạo các lớp con Manager và Developer kế thừa từ lớp Employee và triển khai các phương thức work() và takeVacation().
* Lớp Manager phải triển khai phương thức attendMeeting().

***Yêu cầu:***

* **Lớp Employee**: Phải là lớp trừu tượng với các phương thức trừu tượng.
* **Interface AttendMeeting**: Tạo interface này với phương thức attendMeeting().
* **Lớp Manager và Developer**: Cả hai lớp này phải triển khai các phương thức work() và takeVacation(). Riêng **Manager** triển khai thêm attendMeeting()
  + VD: Manager manager = new Manager("Alice", 8000);
    - Phương thức work() trả về: “Quản lý Alice(Lương: $8000) đang giám sát dự án”
    - Phương thức takeVacation() trả về: “Quản lý Alice(Lương: $8000) đang đi nghỉ phép tại Pháp.”
    - Phương thức attendMeeting() trả về: “Quản lý Alice (Lương: $8000) đang tham gia cuộc họp”
  + VD: Developer developer = new Developer("Bob", 5000);
    - Phương thức work() trả về: “Lập trình viên Bob(Lương $5000) đang viết code cho hệ thống mới”
    - Phương thức takeVacation() trả về: “Lập trình viên Bob(Lương $5000) đang đi nghỉ phép ở Bali.”
* **Lớp Company**: Tạo lớp này để quản lý nhân viên và tổ chức các cuộc họp.