|  |  |  |
| --- | --- | --- |
|  | **Interface** | **Abstract Class** |
| Khái niệm | + Không phải là một class, giống như lớp trừu tượng , chứa các phương thức không có phần thân.  + Một interface không thể tự làm bất cứ điều gì. | + Là một lớp nhưng không thể tạo ra thực thể (Không thể tạo ra đối tượng)  + Có thể chứa các phương thức trừu tượng hoặc không chứa phương thức trừu tượng ,hoặc vừa chứa phương thức thực, vừa chứa phương thức trừu tượng.  + Chú ý rằng phương thức trừu tượng ở đây là chỉ có khai báo, không có phần thân, hay còn gọi là phần cài đặt. |
| Mục đích | + Là một chức năng có thể thêm vào bất kì  class nào.  + Interface có thể bao gồm nhiều hàm, phương thức và cùng phục vụ cho một chức năng. | + Là class cho cho tất cả các class có cùng bản chất (kiểu, loại, nhiệm vụ của class,…)  + Hai class cùng thực hiện một interface có thể khác nhau hoàn toàn về bản chất. |
| Đặc điểm về đa kế thừa | + Một class có thể hiện thực nhiều interface ( Hỗ trợ đa kế thừa ) | + Không hỗ trợ đa kế thừa. |
|  | + Không thể định nghĩa code xử lí, chỉ có thể khai báo. | + Có thể định nghĩa thân phương thức , thuộc tính. |
|  | + Mọi phương thức , property đều mặc định là public. | + Có thể xác định modifier. |
|  | + Từ khóa interface được sử dụng để khai báo giao diện.  + Một interface chỉ có thể implement interface khác. | + Từ khóa “abstract” được sử dụng để khai báo lớp trừu tượng.  + Một lớp trừu tượng có thể extends một lớp java khác và thực hiện nhiều interface |
|  | + Các lớp triển khai interface sử dụng từ khóa implements. | + Có thể kế thừa lớp trừu tượng bằng cách sử dụng từ khóa extends. |
| Công thức | + public interface TEN {  void InThongTin();  } | + public abstract class TEN{  public abstract void Ve();  } |
| Ví dụ cụ thể | Định nghĩa interface Vehicle  interface Vehicle {  void start(); // Phương thức bắt đầu  void stop(); // Phương thức dừng lại  void accelerate(); // Phương thức tăng tốc  void brake(); // Phương thức phanh  }  Triển khai interface Vehicle trong lớp Car  class Car implements Vehicle {  @Override  public void start() {  System.out.println("Car is starting.");  }  @Override  public void stop() {  System.out.println("Car is stopping.");  }  @Override  public void accelerate() {  System.out.println("Car is accelerating.");  }  @Override  public void brake() {  System.out.println("Car is braking.");  }  }  Triển khai interface Vehicle trong lớp Bicycle  class Bicycle implements Vehicle {  @Override  public void start() {  System.out.println("Bicycle is starting.");  }  @Override  public void stop() {  System.out.println("Bicycle is stopping.");  }  @Override  public void accelerate() {  System.out.println("Bicycle is speeding up.");  }  @Override  public void brake() {  System.out.println("Bicycle is applying brakes.");  }  }  Lớp chính sử dụng các đối tượng Car và Bicycle  public class Main {  public static void main(String[] args) {  // Sử dụng lớp Car  TA CÓ THỂ SỬ DỤNG INTERFACE KHỞI TẠO LỚP CON  Vehicle car = new Car();  car.start();  car.accelerate();  car.brake();  car.stop();  // Sử dụng lớp Bicycle  Vehicle bicycle = new Bicycle();  bicycle.start();  bicycle.accelerate();  bicycle.brake();  bicycle.stop();  }  } | Lớp trừu tượng đại diện cho hình học  abstract class HinhHoc {  // Phương thức trừu tượng tính diện tích  public abstract double tinhDienTich();  // Phương thức trừu tượng tính chu vi  public abstract double tinhChuVi();  }  Lớp con kế thừa từ lớp trừu tượng Hình Học - Hình Tròn  class HinhTron extends HinhHoc {  private double banKinh;  // Constructor của lớp con  public HinhTron(double banKinh) {  this.banKinh = banKinh;  }  // Implement phương thức tính diện tích cho Hình Tròn  public double tinhDienTich() {  return Math.PI \* banKinh \* banKinh;  }  // Implement phương thức tính chu vi cho Hình Tròn  public double tinhChuVi() {  return 2 \* Math.PI \* banKinh;  }  }  Lớp con khác kế thừa từ lớp trừu tượng Hình Học - Hình Chữ Nhật  class HinhChuNhat extends HinhHoc {  private double chieuDai;  private double chieuRong;  // Constructor của lớp con  public HinhChuNhat(double chieuDai, double chieuRong) {  this.chieuDai = chieuDai;  this.chieuRong = chieuRong;  }  // Implement phương thức tính diện tích cho Hình Chữ Nhật  public double tinhDienTich() {  return chieuDai \* chieuRong;  }  // Implement phương thức tính chu vi cho Hình Chữ Nhật  public double tinhChuVi() {  return 2 \* (chieuDai + chieuRong);  }  }  public class Main {  public static void main(String[] args) {  TA CÓ THỂ SỬ DỤNG ABSTRACT ĐỂ KHỞI TẠO ĐỐI TƯỢNG  // Tạo đối tượng Hình Tròn  HinhHoc hinhTron = new HinhTron(5.0);  // Tính diện tích và chu vi của Hình Tròn  System.out.println("Diện tích của Hình Tròn: " + hinhTron.tinhDienTich());  System.out.println("Chu vi của Hình Tròn: " + hinhTron.tinhChuVi());  // Tạo đối tượng Hình Chữ Nhật  HinhHoc hinhChuNhat = new HinhChuNhat(4.0, 6.0);  // Tính diện tích và chu vi của Hình Chữ Nhật  System.out.println("Diện tích của Hình Chữ Nhật: " + hinhChuNhat.tinhDienTich());  System.out.println("Chu vi của Hình Chữ Nhật: " + hinhChuNhat.tinhChuVi());  }  } |
| Cần triển khai | Khi một lớp con (class) triển khai (implements) một interface trong java, nó phải triển khai tất cả các phương thức trừu tượng được khai báo trong interface đó.  Trong java, mặc định các phương thức trừu tượng trong interface có phạm vi truy cập là  “public”. | Khi một lớp con (class) kế thừa một lớp trừu tượng trong JAVA, nó có thể triển khai hoặc không triển khai tất cả các phương thức trừu tượng được khi báo trong lớp trừu tượng đó.  Tuy nhiên nếu lớp con không triển khai tất cả các phương thức trừu tượng , thì lớp con đó cũng phải được đánh dấu là lớp trừu tượng. |

Khi một lớp con (class) kế thừa một lớp trừu tượng trong Java, nó có thể triển khai hoặc không triển khai tất cả các phương thức trừu tượng được khai báo trong lớp trừu tượng đó. Tuy nhiên, nếu lớp con không triển khai tất cả các phương thức trừu tượng, thì lớp con đó cũng phải được đánh dấu là trừu tượng.

* Khi một lớp con (subclass) kế thừa một lớp trừu tượng (abstract class) trong JAVA, nó phải cung cấp triển khai cho tất cả các phương thức trừu tượng (abstract class ) . Nếu lớp con đó không triển khai tất cả các phương thức trừu tượng, thì lớp con đó cũng phải được đánh dấu là trừu tượng. –

- Lớp trừu tượng bao gồm hai phương thức trừu tượng -

abstract class Shape {

abstract void draw();

abstract double calculateArea();

}

class Circle extends Shape {

private double radius;

@Override

void draw() {

System.out.println("Drawing a circle");

}

@Override

double calculateArea() {

return Math.PI \* radius \* radius;

}

}

// Lớp con khác kế thừa từ lớp trừu tượng nhưng không triển khai đủ phương thức

class Square extends Shape {

private double side;

// Phương thức triển khai draw cho Square

@Override

void draw() {

System.out.println("Drawing a square");

}

// Không triển khai calculateArea, do đó lớp này phải được đánh dấu là trừu tượng

// Lớp Square không triển khai đủ phương thức, nên nó cũng phải là trừu tượng

}

public class Main {

public static void main(String[] args) {

Circle circle = new Circle();

circle.draw();

System.out.println("Area: " + circle.calculateArea());

// Square square = new Square(); // Lỗi vì Square là lớp trừu tượng

}

}

* Tức là ta phải hiểu như này nếu một lớp kế thừa một lớp trừu tượng thì để cho cái lớp đó khởi tạo một cái đối tượng mà không bị lỗi gì thì nó phải triển khai tất cả các phương thức trừu tượng có trong lớp trừu tượng đó. Nếu nó không triển khai hết tất cả các phương thức trừu tượng thì đối tượng đó bị sai.
* Tôi muốn nhắc lại nguyên tắc : Khi một lớp kế thừa từ một lớp trừu tượng , nó phải triển khai (Hoặc để cho lớp con tiếp tục làm lớp trừu tượng – Tức là nó không triển khai đủ các phương thức trừu tượng) . Nếu lớp con không triển khai đầy đủ các phương thức trừu tượng , lớp con đó cũng phải được đánh dấu là lớp trừu tượng🡺 Tức là ta không thể khởi tạo được đối tượng từ cái lớp con được đánh dấu trừu tượng này.
* Ta muốn sử dụng interface thay cho đa kế thừa thì chúng ta sẽ phải biến mấy cái lớp mà một lớp con muốn kế thừa từ các lớp đó thành interface , rồi chúng ta sẽ dùng cái lớp đa kế thừa đó triển khai các interface , như thế nó mới có thể sử dụng đa kế thừa được.