Session 07:

JAVA Inheritance and polymorphism

# Ghi đè phương thức - Overriding trong lập trình Java

Overrding hay ghi đè phương thức, được sử dụng trong trường hợp lớp con kế thừa từ lớp cha và muốn định nghĩa lại một phương thức đã có mặt ở lớp cha. Lý do: Một lớp cha thường có thể có nhiều lớp con kế thừa, tuy nhiên phương thức ở lớp cha

Quy tắc:

Lớp con và lớp cha có mối quan hệ kế thừa.

Phương thức trong lớp con phải có tên giống với lớp cha.

Phương thức trong lớp con phải có tham số giống với lớp cha

public class Animal {

protected String name;

public Animal(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public void eat() {

System.out.println("Tôi đang ăn .... ");

}

public void makeSound() {

System.out.println("................");

}

public void sleep() {

System.out.println("Zzzzzzzzzzzzzzzz");

}

}

public class Dog extends Animal{

public Dog() {

super("Dog");

}

@Override

public void eat() {

System.out.println("Tôi ăn xương!");

}

@Override

public void makeSound() {

System.out.println("Gâu gâu");

}

}

public class Cat extends Animal{

public Cat() {

super("Cat");

}

@Override

public void eat() {

System.out.println("Tôi ăn cá!");

}

@Override

public void makeSound() {

System.out.println("Meo meo");

}

}

public class Bird extends Animal{

public Bird() {

super("Bird");

}

@Override

public void eat() {

System.out.println("Tôi ăn sâu!");

}

@Override

public void makeSound() {

System.out.println("Líu lo");

}

}

public class Test {

public static void main(String[] args) {

System.out.println("Test dog: ");

Dog d = new Dog();

d.eat();

d.makeSound();

d.sleep();

System.out.println("---- ");

System.out.println("Test cat: ");

Cat c= new Cat();

c.eat();

c.makeSound();

c.sleep();

System.out.println("---- ");

System.out.println("Test bird: ");

Bird b= new Bird();

b.eat();

b.makeSound();

b.sleep();

}

}

# Nạp chồng phương thức - Overloading trong Java

Khái niệm về Overloading

Overloading là việc tạo ra nhiều phương thức trong một lớp có chung tên nhưng khác tham số truyền vào Lý do: tăng tính sử dụng cho các phương thức bên trong một lớp.

public class MyMath {

public int timMin(int a, int b) {

if(a<b) {

return a;

}else {

return b;

}

}

*// public double timMin(int a, int b) {*

*// if(a<b) {*

*// return a;*

*// }else {*

*// return b;*

*// }*

*// }*

public double timMin(double a, double b) {

if(a<b) {

return a;

}else {

return b;

}

}

public double tinhTong(double a, double b) {

return a+b;

}

public double tinhTong(double[] arr) {

double tong = 0;

for (int i = 0; i < arr.length; i++) {

tong+=arr[i];

}

return tong;

}

}

public class Test {

public static void main(String[] args) {

MyMath mm = new MyMath();

System.out.println("min(5,5) = "+ mm.timMin(5, 5));

System.out.println("min(5.5, 6.0) = "+ mm.timMin(5.5, 6.0));

System.out.println("tinhTong(5,6)= "+mm.tinhTong(5.0, 6.0));

double arr[] = new double[] {1,2,3,4,5};

System.out.println("tinhTong(arr)= "+mm.tinhTong(arr));

}

}

# Lớp và phương thức trừu tượng Abstract trong lập trình Java

### Tính trừu tượng là gì:

#### Tính trừu tượng là một tiến trình ẩn các chi tiết trình triển khai và chỉ hiển thị tính năng tới người dùng. Lớp trừu tượng không thể được dùng để tạo ra các đối tượng như những lớp bình thường khác.

### Phương thức trừu tượng - abstract trong Java

#### Một phương thức được khai báo là abstract và không có trình triển khai thì đó là phương thức trừu tượng. Phương thức sẽ được triển khai trong các lớp con.

Text

Description automatically generated with medium confidence

public class ToaDo {

int x, y;

public ToaDo(int x, int y) {

this.x = x;

this.y = y;

}

public int getX() {

return x;

}

public void setX(int x) {

this.x = x;

}

public int getY() {

return y;

}

public void setY(int y) {

this.y = y;

}

}

public abstract class Hinh {

protected ToaDo toaDo;

public Hinh(ToaDo toaDo) {

this.toaDo = toaDo;

}

public ToaDo getToaDo() {

return toaDo;

}

public void setToaDo(ToaDo toaDo) {

this.toaDo = toaDo;

}

public abstract double tinhDienTich();

}

public class Diem extends Hinh{

public Diem(ToaDo toaDo) {

super(toaDo);

}

@Override

public double tinhDienTich() {

return 1;

}

}

public class HinhTron extends Hinh {

private double r;

public HinhTron(ToaDo toaDo, double r) {

super(toaDo);

this.r = r;

}

public double getR() {

return r;

}

public void setR(double r) {

this.r = r;

}

@Override

public double tinhDienTich() {

return Math.PI\*this.r\*this.r;

}

}

public class HinhChuNhat extends Hinh{

private double chieuRong, chieuCao;

public HinhChuNhat(ToaDo toaDo, double chieuRong, double chieuCao) {

super(toaDo);

this.chieuRong = chieuRong;

this.chieuCao = chieuCao;

}

@Override

public double tinhDienTich() {

return this.chieuRong\*this.chieuCao;

}

}

public class Test {

public static void main(String[] args) {

ToaDo td1 = new ToaDo(5, 5);

ToaDo td2 = new ToaDo(7, 9);

ToaDo td3 = new ToaDo(12, 1);

*// Hinh h = new Hinh(td1); => lỗi biên dịch*

Hinh h1 = new Diem(td1);

Hinh h2 = new HinhTron(td2, 10);

Hinh h3 = new HinhChuNhat(td3, 5, 10);

System.out.println("DT1: "+h1.tinhDienTich());

System.out.println("DT2: "+h2.tinhDienTich());

System.out.println("DT3: "+h3.tinhDienTich());

}

}

# Bài tập về kế thừa và abstract

## Quản lý phương tiện di chuyển

# Diagram Description automatically generated

Lớp trù tượng: Phương tiện di chuyển (PhuongTienDiChuyen)

Khái niệm cụ thể: Máy bay, Xe Ô Tô, Xe đạp thừa class PhuongTienDiChuyen

public class HangSanXuat {

private String tenHangSanXuat, tenQuocGia;

public HangSanXuat(String tenHangSanXuat, String tenQuocGia) {

this.tenHangSanXuat = tenHangSanXuat;

this.tenQuocGia = tenQuocGia;

}

public String getTenHangSanXuat() {

return tenHangSanXuat;

}

public void setTenHangSanXuat(String tenHangSanXuat) {

this.tenHangSanXuat = tenHangSanXuat;

}

public String getTenQuocGia() {

return tenQuocGia;

}

public void setTenQuocGia(String tenQuocGia) {

this.tenQuocGia = tenQuocGia;

}

}

public abstract class PhuongTienDiChuyen {

protected String tenLoaiPhuongTien;

protected HangSanXuat hangSanXuat;

public PhuongTienDiChuyen(String tenLoaiPhuongTien, HangSanXuat hangSanXuat) {

this.tenLoaiPhuongTien = tenLoaiPhuongTien;

this.hangSanXuat = hangSanXuat;

}

public String getTenLoaiPhuongTien() {

return tenLoaiPhuongTien;

}

public void setTenLoaiPhuongTien(String tenLoaiPhuongTien) {

this.tenLoaiPhuongTien = tenLoaiPhuongTien;

}

public String layTenHangSanXuat() {

return this.hangSanXuat.getTenHangSanXuat();

}

public void batDau() {

System.out.println("Bắt đầu ....");

}

public void tangToc() {

System.out.println("Tăng tốc ...");

}

public void dungLai() {

System.out.println("Dừng lại ...");

}

public abstract double layVanToc();

}

public class MayBay extends PhuongTienDiChuyen{

private String loaiNhienLieu;

public MayBay(HangSanXuat hangSanXuat, String loaiNhienLieu) {

super("Máy Bay", hangSanXuat);

this.loaiNhienLieu = loaiNhienLieu;

}

public String getLoaiNhienLieu() {

return loaiNhienLieu;

}

public void setLoaiNhienLieu(String loaiNhienLieu) {

this.loaiNhienLieu = loaiNhienLieu;

}

@Override

public double layVanToc() {

return 500;

}

public void catCanh() {

System.out.println("Cất cánh");

}

public void haCanh() {

System.out.println("Hạ cánh");

}

}

public class XeDap extends PhuongTienDiChuyen{

public XeDap(HangSanXuat hangSanXuat) {

super("Xe đạp", hangSanXuat);

}

@Override

public double layVanToc() {

return 20;

}

}

public class XeOTo extends PhuongTienDiChuyen{

private String loaiNhienLieu;

public XeOTo(HangSanXuat hangSanXuat , String loaiNhienLieu) {

super("Xe Ô tô", hangSanXuat);

this.loaiNhienLieu = loaiNhienLieu;

}

public String getLoaiNhienLieu() {

return loaiNhienLieu;

}

public void setLoaiNhienLieu(String loaiNhienLieu) {

this.loaiNhienLieu = loaiNhienLieu;

}

@Override

public double layVanToc() {

return 150;

}

}

public class Test {

public static void main(String[] args) {

HangSanXuat h1 = new HangSanXuat("Hãng 1", "Việt Nam");

HangSanXuat h2 = new HangSanXuat("Hãng 2", "USA");

HangSanXuat h3 = new HangSanXuat("Hãng 3", "Nhật Bản");

PhuongTienDiChuyen p1 = new XeOTo(h1, "Trường Hải");

MayBay p2 = new MayBay(h2, "Xăng");

PhuongTienDiChuyen p3 = new XeDap(h3);

System.out.println("Lấy hảng sản xuất");

System.out.println("p1: "+ p1.layTenHangSanXuat());

System.out.println("Bắt đầu: ");

p2.batDau();

System.out.println("Lấy vận tốc: ");

System.out.println("p1: "+ p1.layVanToc());

System.out.println("p2: "+ p2.layVanToc());

System.out.println("p3: "+ p3.layVanToc());

System.out.println("Cất cánh: ");

p2.catCanh();

}

}

# Hiểu rõ về INTERFACE trong lập trình Java

Interface là gì?

Một Interface trong Java là một bản thiết kế của một lớp. Nó chỉ có các phương thức trừu tượng và các hằng số. Tất cả các hằng số được mặc định là ở dạng public static final. Tất cả các phương thức là ở dạng public. Tương tự như lớp abstract, nó không thể được khởi tạo thành đối tượng.

Mục đích:

Để thu được tính trừu tượng hoàn toàn và đa kế thừa trong Java.

Lưu ý khi sử dụng: Một lớp có thể triển khai một hoặc nhiều interface tại một thời điểm. Một interface có thể kế thừa từ một interface khác, tương tự cách một lớp có thể kế thừa lớp khác.

Ví dụ: a, Tạo ra Interface có tên là MayTinhBoTuiInterface, có các phương thức mặc định là cộng, trừ, nhân, chia hai số thực.

Xây dựng các lớp hiện thực hóa Interface bên trên có tên là: MayTinhCasioFX500 và MayTinhVinaCal500MS

Diagram

Description automatically generated

public interface MayTinhBoTuiInterface {

public abstract double cong(double a, double b);

public double tru(double a, double b);

public double nhan(double a, double b);

public double chia(double a, double b);

}

public class MayTinhCasioFX500 implements MayTinhBoTuiInterface{

@Override

public double cong(double a, double b) {

return a+b;

}

@Override

public double tru(double a, double b) {

return a-b;

}

@Override

public double nhan(double a, double b) {

return a\*b;

}

@Override

public double chia(double a, double b) {

return a/b;

}

}

public class MayTinhVinacal500 implements MayTinhBoTuiInterface{

@Override

public double cong(double a, double b) {

return a-b;

}

@Override

public double tru(double a, double b) {

return a-b;

}

@Override

public double nhan(double a, double b) {

return a\*b;

}

@Override

public double chia(double a, double b) {

return a/b;

}

}

public interface SapXepInterface {

public void sapXepTang(double[] arr);

public void sapXepGiam(double[] arr);

}

public class SapXepChen implements SapXepInterface {

@Override

public void sapXepTang(double[] arr) {

int n = arr.length;

double key;

int i, j;

for (i = 1; i < n; i++) {

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

@Override

public void sapXepGiam(double[] arr) {

int n = arr.length;

double key;

int i, j;

for (i = 1; i < n; i++) {

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] < key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

}

public class SapXepChon implements SapXepInterface {

@Override

public void sapXepTang(double[] arr) {

int n = arr.length;

int i, j, min\_idx;

for (i = 0; i < n - 1; i++) {

min\_idx = i;

for (j = i + 1; j < n; j++)

if (arr[j] < arr[min\_idx])

min\_idx = j;

double temp = arr[min\_idx];

arr[min\_idx] = arr[i];

arr[i] = temp;

}

}

@Override

public void sapXepGiam(double[] arr) {

int n = arr.length;

int i, j, min\_idx;

for (i = 0; i < n - 1; i++) {

min\_idx = i;

for (j = i + 1; j < n; j++)

if (arr[j] > arr[min\_idx])

min\_idx = j;

double temp = arr[min\_idx];

arr[min\_idx] = arr[i];

arr[i] = temp;

}

}

}

public class PhanMemMayTinh implements MayTinhBoTuiInterface, SapXepInterface{

@Override

public double cong(double a, double b) {

return a+b;

}

@Override

public double tru(double a, double b) {

return a-b;

}

@Override

public double nhan(double a, double b) {

return a\*b;

}

@Override

public double chia(double a, double b) {

return a/b;

}

@Override

public void sapXepTang(double[] arr) {

int n = arr.length;

double key;

int i, j;

for (i = 1; i < n; i++) {

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

@Override

public void sapXepGiam(double[] arr) {

int n = arr.length;

double key;

int i, j;

for (i = 1; i < n; i++) {

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] < key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

}

import java.util.Iterator;

public class Test {

public static void main(String[] args) {

System.out.println("Test câu a: ");

MayTinhCasioFX500 mfx500 = new MayTinhCasioFX500();

MayTinhVinacal500 mvn500 = new MayTinhVinacal500();

System.out.println("5+3="+ mfx500.cong(5, 3));

System.out.println("4\*5="+ mvn500.nhan(4, 5));

System.out.println("4/0="+ mvn500.chia(4, 0));

System.out.println("Test câu b: ");

double[] arr = new double[] {5,1,3,4,5,8,0};

double[] arr2 = new double[] {6,2,7,9,2,4,5};

SapXepChen sxchen = new SapXepChen();

SapXepChon sxchon = new SapXepChon();

sxchen.sapXepTang(arr);

for (int i = 0; i < arr.length; i++) {

System.out.print(arr[i]+" ");

}

System.out.println();

sxchon.sapXepGiam(arr2);

for (int i = 0; i < arr2.length; i++) {

System.out.print(arr2[i]+" ");

}

System.out.println();

System.out.println("Test câu c:");

PhanMemMayTinh pmmt =new PhanMemMayTinh();

System.out.println("3+5=" + pmmt.cong(3, 5));

double[] arr3 = new double[] {6,2,7,9,2,4,5};

pmmt.sapXepTang(arr3);

for (int i = 0; i < arr3.length; i++) {

System.out.print(arr3[i]+" ");

}

}

}