Mục đích

- Giới thiệu các khái niệm cơ bản của lập trình hướng đối tượng.
- Trang bị các kỹ năng xây dựng lớp và các thành phần của lớp.
- Sau khi kết thúc chương có thể đặc tả và giải quyết các bài toán dựa trên hướng đối tượng.

Khái niệm

- Tính kế thừa cho phép định nghĩa một lớp mới dựa trên các lớp đã có.
- Một lớp kế thừa từ 1 lớp khác được gọi là lớp dẫn xuất (derived class, lớp con).
- Một lớp được lớp khác kế thừa gọi là lớp cơ sở (base class, lớp cha).
- Lớp dẫn xuất sẽ kế thừa các thành phần dữ liệu và hàm thành phần của lớp cơ sở đồng thời bổ sung thêm các thành phần mới.

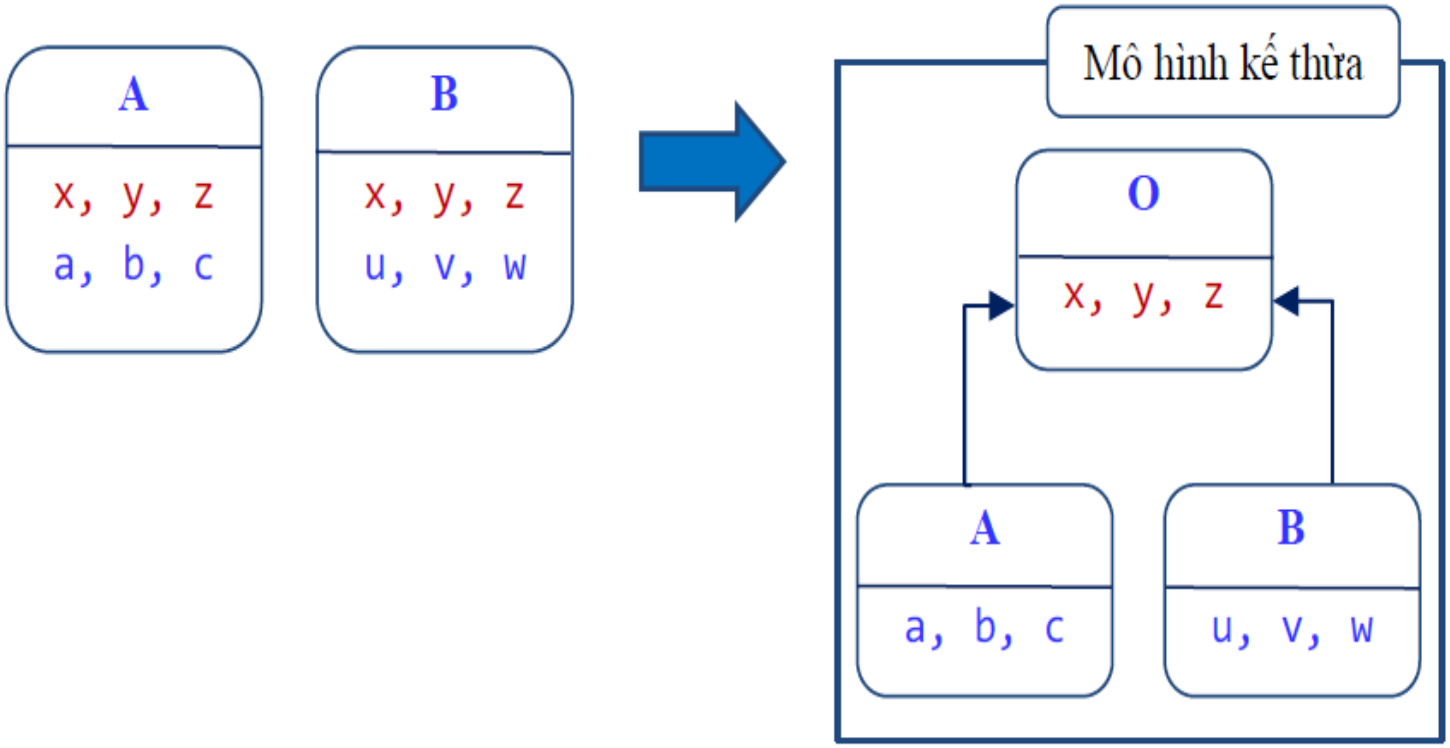
Khái niệm

- Java không có đa kế thừa. Mỗi đối tượng chỉ được phép kế thừa trực tiếp từ một đối tượng khác.
- Ta sử dụng từ khóa extends để biểu thị mối quan hệ kế thừa
  - Ví dụ: Hocsinh **extends** ConNguoi
  - ConNguoi: lớp cơ sở
  - HocSinh: lớp dẫn xuất

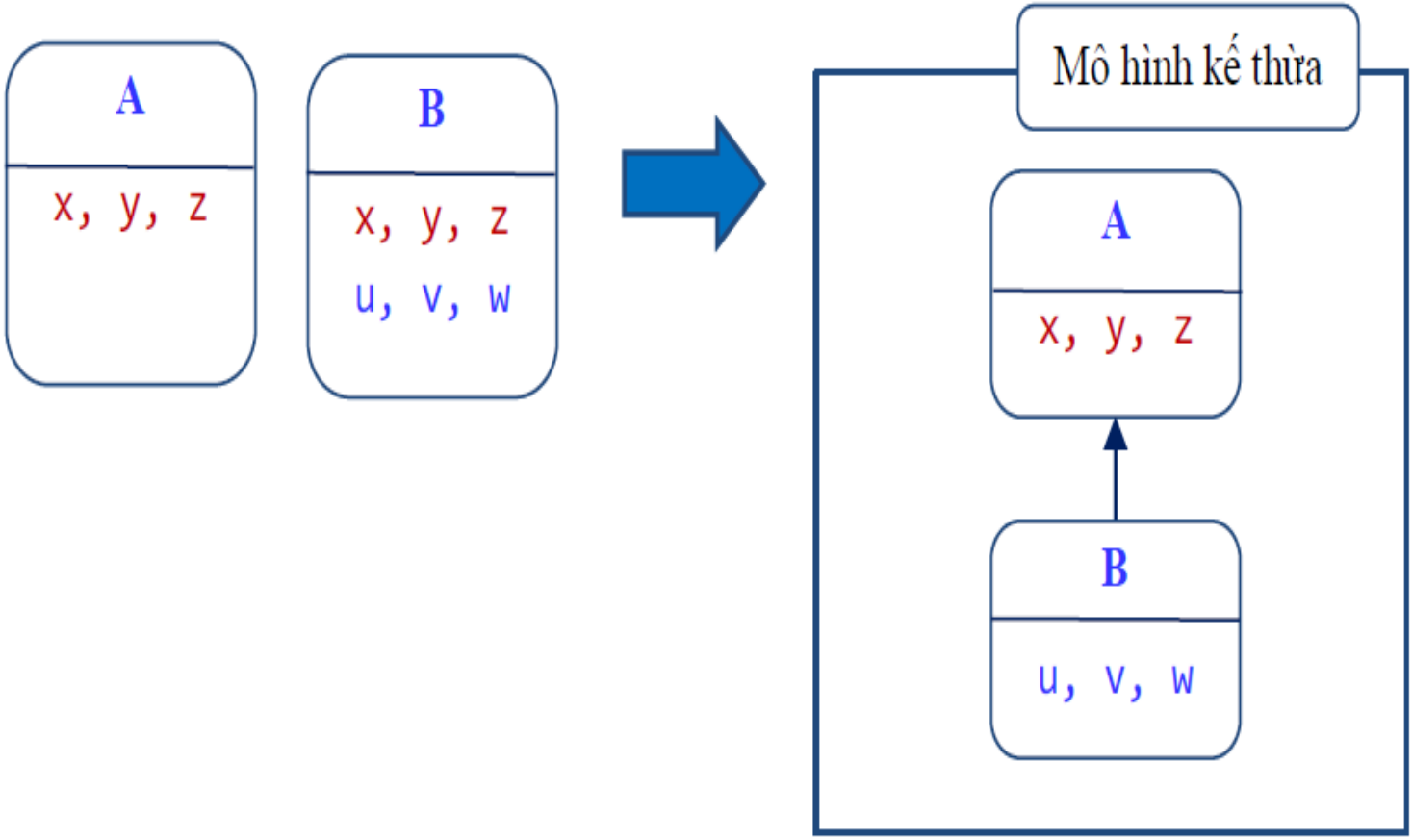
Ý nghĩa của tính kế thừa

- Nếu 2 đối tượng A và B có cùng bản chất, cùng chung 1 số đặc điểm và hành vi → Sẽ có đối tượng C gồm các đặc điểm chung của 2 đối tượng A và B. Hay ta nói A và B kế thừa từ đối tượng C.
- Khi ta dùng tính kế thừa trong phân tích sẽ tạo nên code tường minh, mạch lạc, giúp bảo trì, nâng cấp chương trình dễ dàng hơn.

Mô hình kế thừa



Mô hình kế thừa



Ví dụ 3.1

- Bài toán: viết chương trình **quản lý thông tin nhân viên** và **học sinh** của trường tiểu học.
- Phân tích bài toán:
  - Các đối tượng trong bài toán: nhân viên, học sinh, quản lý.
  - Đối tượng nhân viên
    - Gồm các thuộc tính: Mã HS, Họ tên, lớp, tuổi, Giới tính, Địa chỉ, Điểm toán, Điểm văn, điểm anh.
    - Các phương thức:
      - Ăn uống
      - Di chuyển
      - Tính điểm trung bình
      - Nhập thông tin
      - In thông tin
      - Học bài

Ví dụ 3.1

- Đối tượng nhân viên:
  - Thuộc tính: Mã NV, Họ tên, Chức vụ, Địa chỉ, Tuổi, Giới tính, Lương.
  - Phương thức: Nhập thông tin, In thông tin, Làm việc, Ăn uống, Di chuyển, Viết báo cáo.
- Quản lý:
  - Thuộc tính: Danh sách <nhân viên>, danh sách <Học sinh>
  - Phương thức: Thêm nhân viên, Thêm học sinh, Tìm kiếm nhân viên, Tìm kiếm học sinh, Xóa nhân viên, Xóa học sinh, Hiển thị danh sách nhân viên, Hiển thị danh sách học sinh.

Ví dụ 3.1

- Tìm kiếm mối quan hệ kế thừa:

Đối tượng	Thuộc tính chung	Phương thức chung
• Học sinh • Nhân viên	• Họ tên • Tuổi • Giới tính • Địa chỉ	• Nhập thông tin • Ăn uống • Di chuyển • In thông tin

- Hình thành nên đối tượng cha: Con Người gồm các đặc điểm chung và hành vi chung của Học sinh và Nhân viên. Vậy chúng ta có 4 đối tượng.

Ví dụ 3.1:

Con người	Học sinh
Thuộc tính: Họ tên, Tuổi, Giới tính, Địa chỉ **Phương thức**: Nhập thông tin, Ăn uống, Di chuyển, In thông tin	**Thuộc tính**: Mã HS, Lớp, Điểm toán, Điểm Văn, Điểm Anh **Phương thức**: Học bài, Tính điểm trung bình
Nhân viên	**Quản lý**
Thuộc tính: Mã NV, Chức vụ, Lương **Phương thức**: Làm việc, Viết báo cáo	**Thuộc tính**: Danh sách <Nhân viên>, danh sách <Học sinh> **Phương thức**: Thêm Nhân viên, Thêm học sinh, Tìm kiếm nhân viên, Tìm kiếm học sinh, Xóa nhân viên, Xóa học sinh, Hiển thị danh sách nhân viên, Hiển thị ds học sinh

Ví dụ 3.1

- Ta có thể viết dạng mô phỏng thô chương trình như sau:

```
class ConNguoi{
    String hoTen, gioiTinh, diaChi;
    int tuoi;
void nhapTT(String gTriHoTen, String sTriGioiTinh, String gTriDiaChi, int gTriTuoi) {
// Nhập thông tin cho các thuộc tính hoTen, gioiTinh,
//diaChi, tuoi
}
```

Ví dụ 3.1

```
void anUong(){
// Nội dung phương thức ăn uống
}
void diChuyen() {
// Nội dung phương thức di chuyển
}
void inTT() {
// In thông tin con người gồm hoTen, gioiTinh, diaChi, tuoi
}
```

Ví dụ 3.1

```
class HocSinh extends ConNguoi {
String maHS, lop;
int diemToan, diemVan, diemAnh;
void hocBai() {
// Nội dung phương thức học bài
}
float tinhDiemTB() {
// Nội dung phương thức tính điểm trung bình
return diemTB;
}
```

```
class NhanVien extends ConNguoi {
String maNV, chucVu;
int luong;
void lamViec() {
// Nội dung phương thức làm việc
}
void vietBaoCao() {
// Nội dung phương thức viết báo cáo
}
```

Ví dụ 3.1

```
Class QuanLy {
// Khai báo danh sách nhân viên
NhanVien [] dsNhanVien;
// Khai báo danh sách học sinh
HocSinh [] dsHocSinh;
void themNhanVien(NhanVien nhanvien) {
// Thêm nhân viên vào danh sách
}
void themHocSinh (HocSinh hocsinh) {
// Thêm học sinh vào danh sách
}
```

Ví dụ 3.1

```
NhanVien timKiemNhanVien(String maNV) {
// Tim fkiếm và trả về nhân viên theo mã nhân viên
return null; // không tìm thấy
}
HocSinh timKiemHocSinh(String maHS) {
// Tim fkiếm và trả về học sinh theo mã học sinh
return null; // không tìm thấy
}
void xoaNhanVien(String maNV){
// Phương thức xóa nhân viên có mã nhân viên =maNV
}
void xoaHocSinh(String maHS){
// Phương thức xóa học sinh có mã học sinh=maHS
}
```

Ví dụ 3.1

```
void hienThiHS() {
// Hiển thị danh sách thông tin từng học sinh
}
void hienThiNV () {
// Hiển thị danh sách thông tin từng nhân viên
}
```

Nạp chồng phương thức

- Xét ví dụ 3.1, ta có phươnng thức

nhapTT(String gTriHoTen, String sTriGioiTinh, String gTriDiaChi, int gTriTuoi)

Được xây dựng ở lớp ConNguoi và được sử dụng ở cả 2 lớp con HocSinh và NhanVien, tuy nhiên thông tin cần nhập cho các lớp con là khác so với lớp cha → Cần xây dựng lại phương thức nhapTT cho các lớp con

→ Sử dụng nạp chồng phương thức

Nạp chồng phương thức (tt)

```
class HocSinh extends ConNguoi {
String maHS, lop;
int diemToan, diemVan, diemAnh;
void nhapTT(String gTriHoTen, String sTriGioiTinh, String gTriDiaChi, int gTriTuoi, String maHS, String lop, int diemToan, int diemVan, int diemAnh) {
    // Goi đến phương thức nhập TT của đối tượng ConNgười
    nhapTT(nhapTT(gTriHoTen,sTriGioiTinh,gTriDiaChi,gTriTuoi);
    // NHẬP THÔNG TIN CHO CÁC THUỘC TÍNH maHS, lop, diemToan,
    //diemVan, diemAnh
}
void hocBai() {
// Nội dung phương thức học bài
}
float tinhDiemTB() {
// Nội dung phương thức tính điểm trung bình
return diemTB;
}
```

Nạp chồng phương thức (tt)

- Tương tự cho lớp NhanVien
- Tuy nhiên, phương thức inTT() lại không khả thi → Không có tham số truyền vào → Các phương thức trùng tên, trùng tham số → Không thể hiện được sự khác biệt khi ghi thông tin các lớp
- → Ghi đè phương thức (next slide)

Ghi đè phương thức – Override

- Là việc xây dựng lại nội dung của phương thức đã được định nghĩa hoặc xây dựng trong đối tượng cha cho phù hợp với các công việc cần xử lý của đối tượng con.
- Có 2 hình thức ghi đè:
  - Ghi đè hoàn toàn
  - Ghi đè không hoàn toàn

Ghi đè phương thức – Override

- Ghi đè hoàn toàn: là việc xóa bỏ và xây dựng lại hoàn toàn nội dung của phương thức cha trong lớp con.
- Ghi đè hoàn toàn được sử dụng khi:
  - Phương thức của đối tượng cha chỉ dừng ở mức diịnh nghĩa, không có nội dung.
  - Phương thức của đối tượng cha hoạt động kém hiệu quả, không tối ưu. Do đó ở lớp con cần xây dựng lại hoàn toàn.

Ghi đè phương thức – Override

- Ghi đè không hoàn toàn: là việc kế thừa và xây dựng them nội dung của phương thức cha trong lớp con.
- Ghi đè không hoàn toàn được sử dụng khi:
  - Phương thức của đối tượng cha chua thỏa mãn các công việc cần xử lý trong các lớp con, nên cần bổ sung trong lớp con.

Ghi đè phương thức – Override

- Cách thức thể hiện ghi đè phương thức:
  - Sử dụng ký hiệu **@Override** trước phương thức cần ghi đè trong đối tượng con
  - Sử dụng từ khóa **super.Tên_Phương_Thức**: để gọi đến phương thức của đối tượng cha để kế thừa lại các công việc đã được thực hiện ở đối tượng cha.

Ghi đè phương thức – Override

- Ví dụ:

```
class HocSinh extends ConNguoi {
String maHS, lop;
int diemToan, diemVan, diemAnh;
@Override
void inTT(){
    super.inTT(); // Gọi đến phương thức inTT của lớp cha
    // In them thông tin cho các  thuộc tính của maHS, lop, diemToan,
    //diemVan, diemAnh
}
…….
}
```

Đối tượng (lớp) trừu tượng

- Một lớp trở thành trừu tượng khi:
  - Bản thân lớp đó chứa các phương thức trừu tượng, lớp bắt buộc phải trở thành trừu tượng.
  - Khi đối tượng không trực tiếp tham gia vào bài toán, những đối tượng như vậy được định nghĩa là lớp trừu tượng.
  - Từ khóa abstract được sử dụng để định nghĩa 1 lớp trừu tượng.

Phương thức trừu tượng

- Là các phương thức được định nghĩa nhưng không có nội dung và nằm trong lớp cha.
- Một phương thức trở thành trừu tượng khi:
  - Không thể xây dựng nội dung hoạc rất khó để xây dựng nội dung cho phương thức trong lớp cha. Các lớp con sẽ thực hiện xây dựng nội dung cho các phương thức này.
  - Phương thức này bị ghi đè hoàn toàn trong tất cả các lớp con.
- Từ khóa abstract dung để khai báo các phương thức trừu tượng.

Ví dụ 3.1 (tiếp)

- Trong 2 đối tượng HocSinh và NhanVien đều có những phương thức anUong và diChuyen khác nhau. Khi đó ta có 2 phương thức này trừu tượng → ConNguoi là lớp trừu tượng

Ví dụ 3.1 (tt)

```
abstract class ConNguoi{
    String hoTen, gioiTinh, diaChi;
    int tuoi;
void nhapTT(String gTriHoTen, String sTriGioiTinh, String gTriDiaChi, int gTriTuoi)
{
// Nhập thông tin cho các thuộc tính hoTen, gioiTinh, //diaChi, tuoi
}
abstract void anUong(); // định nghĩa phương thức trừu tượng
abstract void diChuyen();// định nghĩa PT trừu tượng
void inTT() {
// In thông tin con người gồm hoTen, gioiTinh, diaChi, tuoi
}
```

Chương 3. Tính kế thừa

```
class HocSinh extends ConNguoi {
String maHS, lop; int diemToan, diemVan, diemAnh;
void nhapTT(String gTriHoTen, String sTriGioiTinh, String gTriDiaChi, int gTriTuoi
String maHS, String lop, int diemToan, int diemVan, int diemAnh) {
    nhapTT(nhapTT(gTriHoTen,sTriGioiTinh,gTriDiaChi,gTriTuoi);
    // NHẬP THÔNG TIN CHO CÁC THUỘC TÍNH maHS, lop, diemToan
    //diemVan, diemAnh
}
void hocBai() { ……}
float tinhDiemTB() { ……}
@Override
void anUong(){
// Xây dựng hành vi ăn uống của sinh viên, // ghi đè hoàn toàn nên ta không sử
//dụng super
}
```

```
@Override
void diChuyen(){
// Xây dựng hành vi di chuyển của sinh viên
// GHi đè hoàn toàn nên ta không sử dụng super
}
@Override
void inTT() {
// Gọi đến phương thức inTT của đối tượng cha
      super.inTT();
//In them thông tin cho các thuộc tính maHS, lop, diemToan, diemVan,
//diemAnh
}
```

Bài tập vận dụng

- Hãy in thông tin chu vi, diện tích, độ dài cạnh, chiều dài, chiều rộng của hình vuông và hình chữ nhật nếu biết thông tin 4 đỉnh hình vuông, hình chữ nhật đó.