

SỬ BÀI TẬP NGÂN HÀNG

JAVA OOP

Kế Thừa - Đa hình



Tại sao cần kế thừa

Nhu cầu & giới thiệu

Làm việc với kế thừa & đa hình

Code trên dự án

Luyện tập các dự án nâng cao

Phân tích nghiệp vụ & code

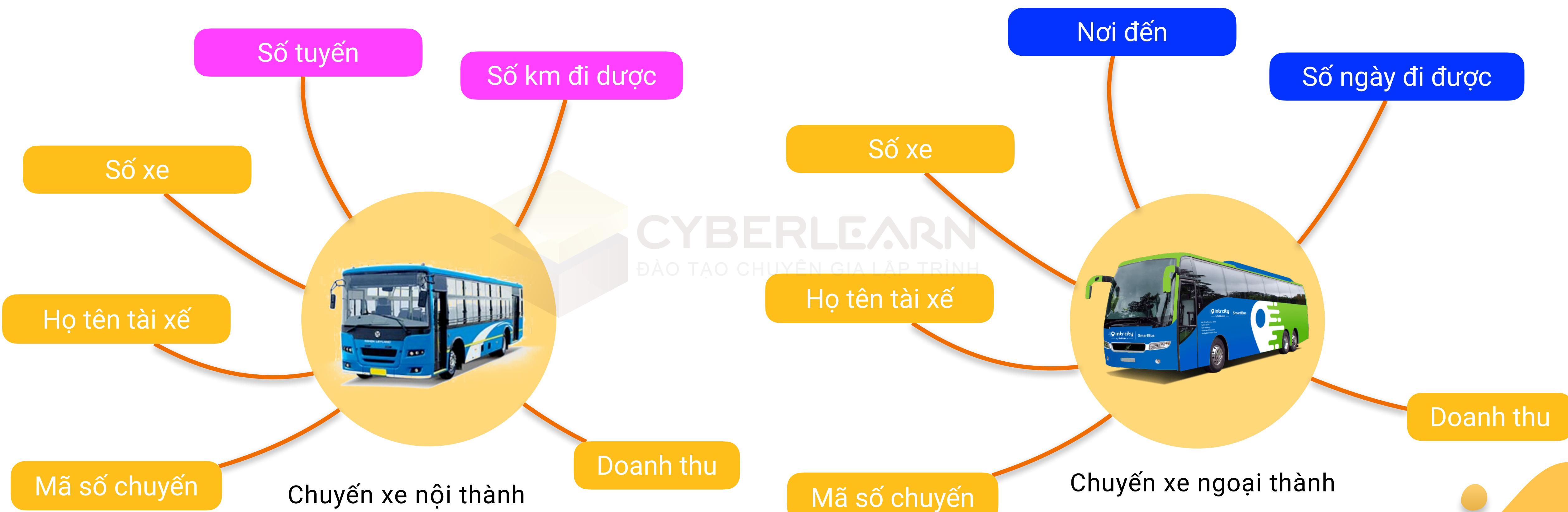
Inheritance

PHÂN TÍCH BÀI QUẢN LÝ CHUYẾN XE

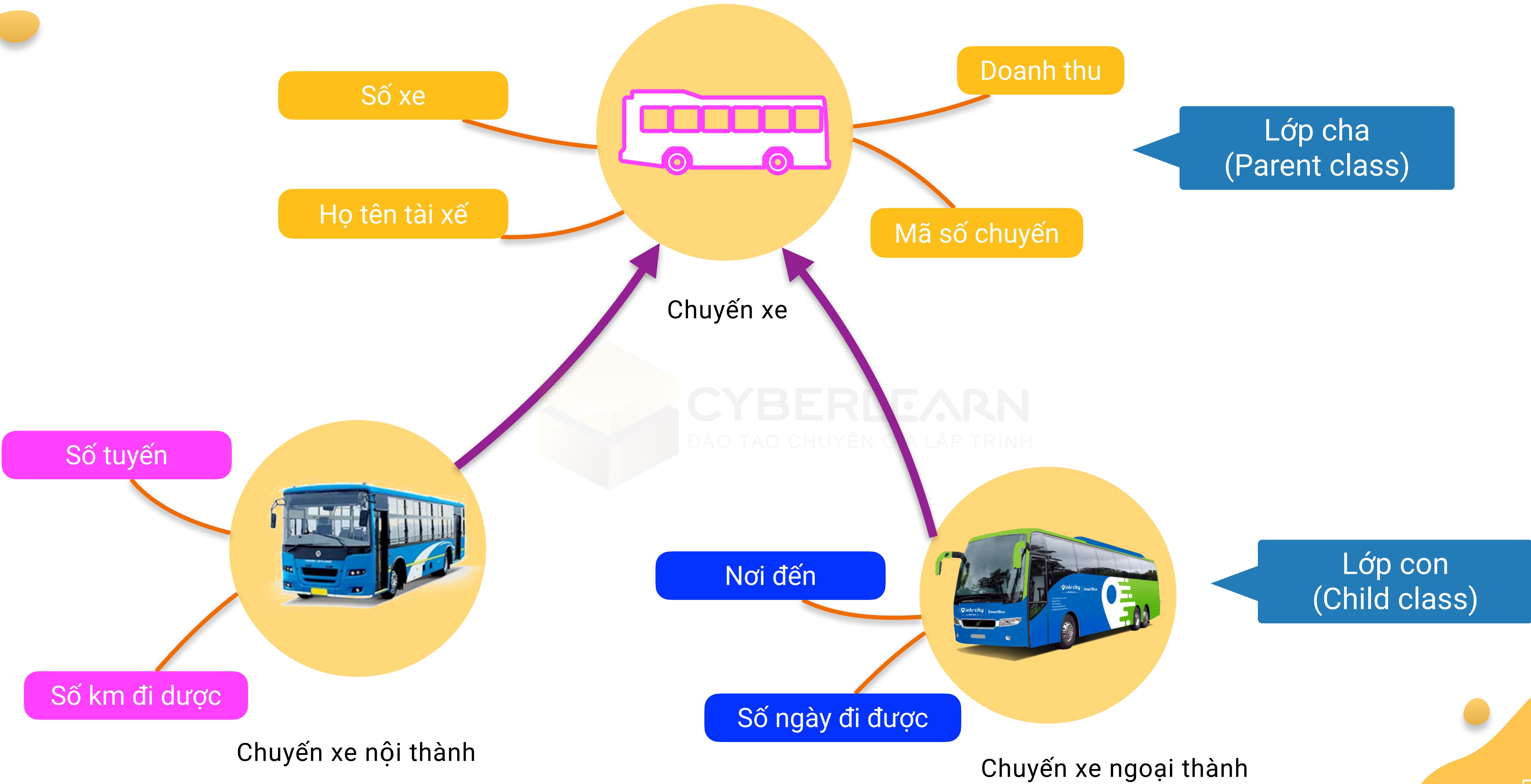
Công ty du lịch V quản lý thông tin là các chuyến xe. Thông tin của 2 loại chuyến xe:

* Chuyến xe nội thành: Mã số chuyến, Họ tên tài xế, số xe, số tuyến, số km đi được, doanh thu.

* Chuyến xe ngoại thành: Mã số chuyến, Họ tên tài xế, số xe, nơi đến, số ngày đi được, doanh thu. Xây dựng chương trình quản lý các chuyến xe, tính tổng doanh thu của công ty và doanh thu từng loại xe.



PHÂN TÍCH BÀI QUẢN LÝ CHUYẾN XE





INHERITANCE

- ✓ Inheritance là một khái niệm rất then chốt trong OOP.
- ✓ Inheritance cho phép tạo một class mới dựa trên một class có sẵn.
- ✓ Inheritance có thể làm đơn giản hóa thiết kế và cài đặt của ứng dụng.
- ✓ Class mới kế thừa từ class có sẵn được gọi là một derived class, child class hoặc subclass.
- ✓ Class có sẵn được gọi là một base class, parent class hoặc superclass.
- ✓ Subclass thừa hưởng được từ superclass các field và các method của superclass.
- ✓ Subclass có thể extend superclass bằng cách bổ sung thêm các field, các constructor và các method.
- ✓ Subclass cũng có khả năng override lại các method thừa kế của superclass để tạo thành version phù hợp.

@Override

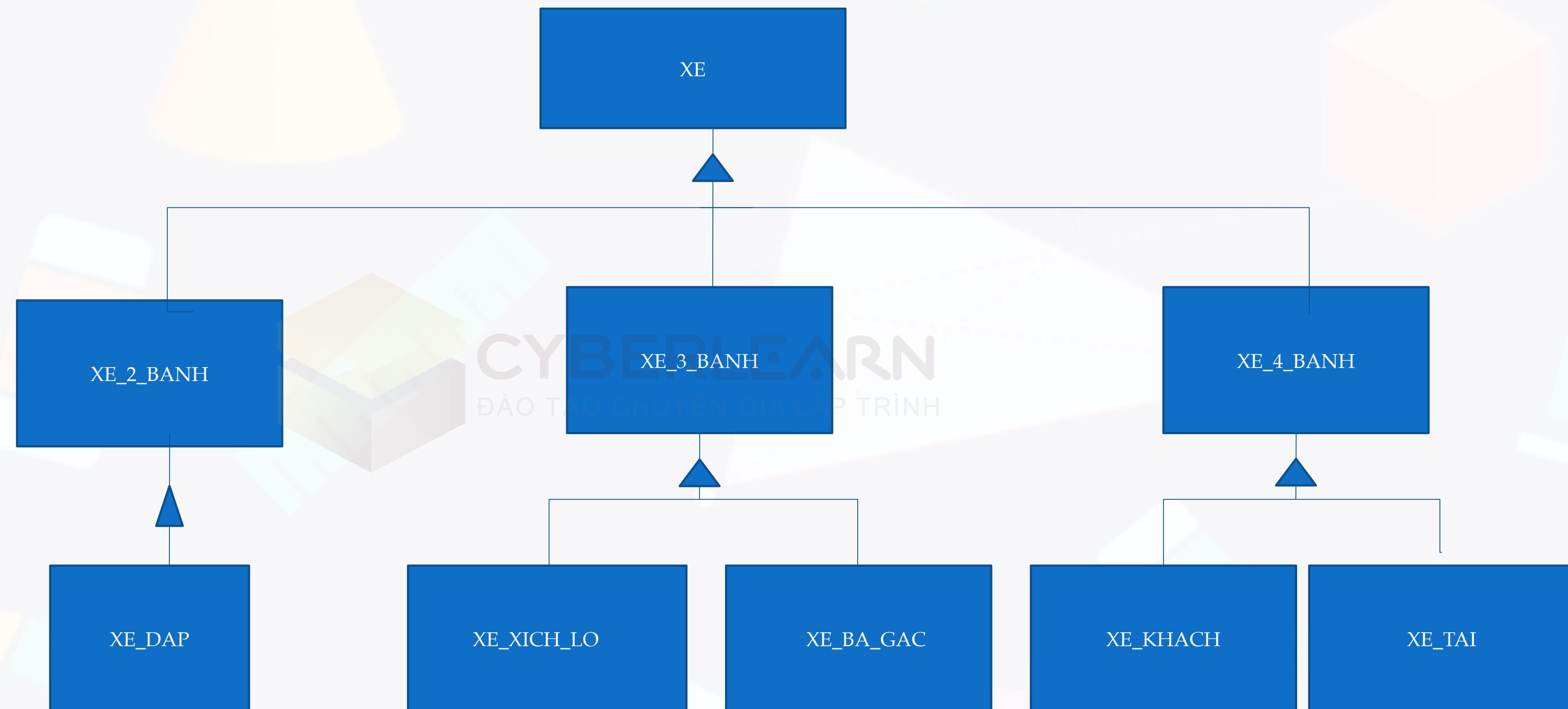


PHÂN TÍCH CÂY KẾ THỪA

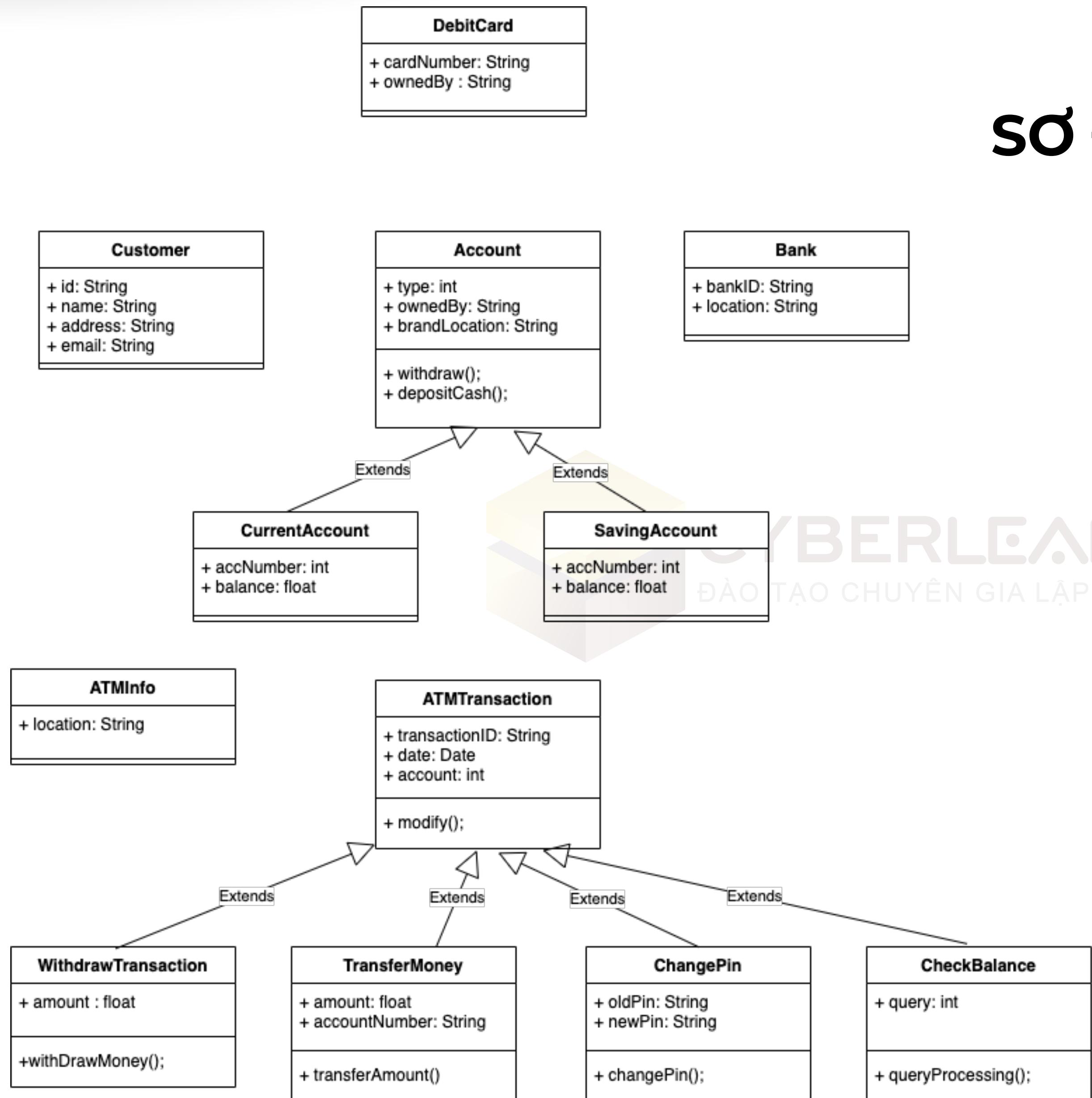
- Cây kế thừa là một cây đa nhánh thể hiện mối quan hệ giữa các lớp trong hệ thống, chương trình.
- Cây kế thừa cho các lớp đối tượng:
 - Lớp XE
 - Lớp XE_2_BANH
 - Lớp XE_3_BANH
 - Lớp XE_4_BANH
 - Lớp XE_XICH_LO
 - Lớp XE_BA_GAC
 - Lớp XE_KHACH
 - Lớp XE_TAI
 - Lớp XE_DAP



PHÂN TÍCH CÂY KẾ THỪA



SƠ ĐỒ LỚP TỔNG QUÁT



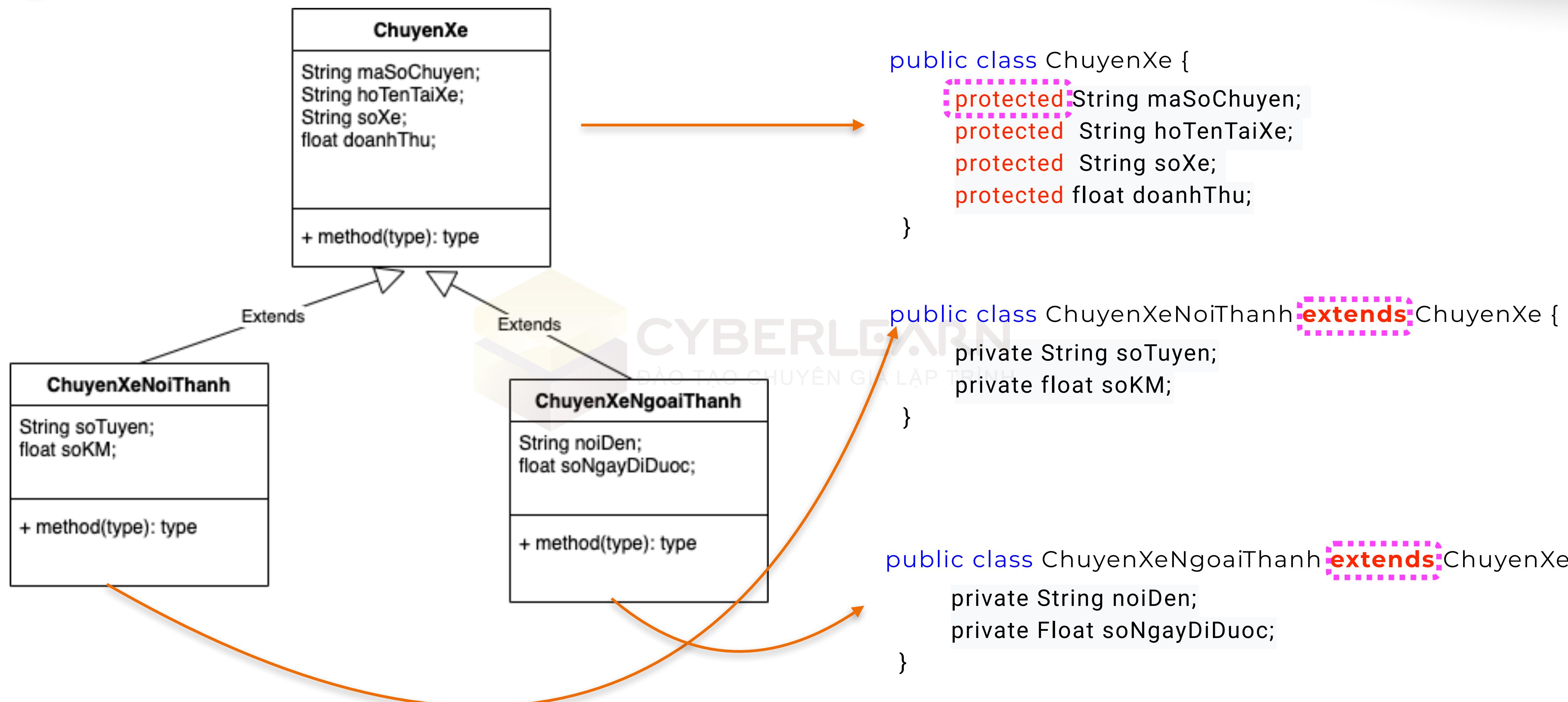
LẬP CÂY KẾ THỪA CHO CÁC YÊU CẦU



1. Lập cây kế thừa cho hệ thống quản lý nhân sự trong công ty. Ví dụ: Hội đồng quản trị, Nhân viên kinh doanh, Phòng ban, nhân viên bán thời gian,
2. Lập cây kế thừa cho hệ thống quản lý các phân khúc và loại xe (Xe ô tô và xe máy các loại)



CODE KẾ THỪA



DẪN XUẤT (ACCESS MODIFIER)

- * **private**

- Nhìn thấy trực tiếp trong class hiện hành.

- * **protected**

- Nhìn thấy trực tiếp trong các class cùng package và các subclass.

- * **public**

- Nhìn thấy trực tiếp trong các class tất cả các package.

- * **default (no keyword)**

- Nhìn thấy trực tiếp trong các class cùng package

CONSTRUCTOR

```
public class ChuyenXe {  
    protected String maSoChuyen;  
    protected String hoTenTaiXe;  
    protected String soXe;  
    protected float doanhThu;  
  
    public ChuyenXe(){  
    }  
  
    public ChuyenXe (String maSo, String hoTen, String soXe, float dThu) {  
        this.maSoChuyen = maSo;  
        this.hoTenTaiXe = hoTen;  
        this.soXe = soXe;  
        this.doanhThu = dThu;  
    }  
}
```

Chỉ định constructor của superclass bằng từ khóa **super**

– Vị trí chỉ định là **dòng đầu tiên** trong cài đặt của phương thức

subclass constructor.

– **super ()** : chỉ định default constructor.

– **super (...)** : chỉ định parameterized constructor.

```
public class ChuyenXeNoiThanh extends ChuyenXe {  
    private String soTuyen;  
    private float soKM;  
  
    public ChuyenXeNoiThanh(){  
        super (); // Chỉ định phương thức khởi tạo mặc định lớp cha  
    }  
  
    public ChuyenXeNoiThanh ( String maSo, String hoTen, String soXe, float dThu, String soTuyen, float soKM) {  
        super (maSo, hoTen, soXe, dThu); // Chỉ định phương thức khởi tạo có tham số lớp cha  
        this.soTuyen = soTuyen;  
        this.soKM = soKM;  
    }  
  
    public class ChuyenXeNgoaiThanh extends ChuyenXe {  
        private String noiDen;  
        private float soNgayDiDuoc;  
  
        public ChuyenXeNgoaiThanh(){  
            super (); // Chỉ định phương thức khởi tạo mặc định lớp cha  
        }  
  
        public ChuyenXeNgoaiThanh ( String maSo, String hoTen, String soXe, float dThu, String noiDen, float soNgayDi) {  
            super (maSo, hoTen, soXe, dThu); // Chỉ định phương thức khởi tạo có tham số lớp cha  
            this.soTuyen = noiDen;  
            this.soKM = soNgayDi;  
        }  
    }
```

JAVA OOP

Override



Lý do cần Override & cách thực hiện

Nhu cầu cần thực hiện Override & cách thực hiện

Một số ghi chú về Override

Nền tảng lý thuyết về Override

CyberSoft

ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Code trên dự án

Triển khai dự án

OVERRIDE

Xét bài toán quản lý nhân sự bao gồm Nhân viên thường, Trưởng phòng và giám đốc. Mỗi nhân sự có họ lương theo ngày và phụ cấp.
Xây dựng các lớp đối tượng và tính lương cho nhân viên

```
public class NhanSu {  
    protected String hoTen;  
    protected String maSo;  
    protected String soNgayLam;  
    protected float luong;  
  
    public NhanSu(){  
    }  
  
    public void tinhLuong () {  
        this.luong = 0;  
    }  
}
```

```
public class NhanVienThuong extends NhanSu {  
    final int PHU_CAP = 200;  
    final int LUONG_NGAY = 100;  
  
    public NhanVienThuong(){  
    }  
  
    @Override  
    public void tinhLuong () {  
        this.luong = this.soNgayLam * LUONG_NGAY + PHU_CAP;  
    }  
}
```

```
public class TruongPhong extends NhanSu {  
    final int PHU_CAP = 300;  
    final int LUONG_NGAY = 200;  
  
    public TruongPhong(){  
    }  
  
    @Override  
    public void tinhLuong () {  
        this.luong = this.soNgayLam * LUONG_NGAY + PHU_CAP;  
    }  
}
```

override

```
public class ChuyenXe {  
    protected String maSoChuyen;  
    protected String hoTenTaiXe;  
    protected String soXe;  
    protected float doanhThu;  
  
    public ChuyenXe(){  
    }  
  
    public void nhap (Scanner scan) {  
        System.out.println(" Nhập mã số:");  
        this.maSoChuyen = scan.nextLine();  
        System.out.println(" Nhập họ tên tài xế:");  
        this.hoTenTaiXe = scan.nextLine();  
        System.out.println(" Nhập số xe:");  
        this.soXe = scan.nextLine();  
        System.out.println(" Nhập doanh thu:");  
        this.doanhThu = Float.parseFloat(scan.nextLine());  
    }  
}
```

```
public class ChuyenXeNoiThanh extends ChuyenXe {  
    private String soTuyen;  
    private float soKM;  
  
    public ChuyenXeNoiThanh(){  
        super (); // Chỉ định phương thức khởi tạo mặc định lớp cha  
    }  
  
    @Override  
    public void nhap ( Scanner scan) {  
        super.nhap(); // Gọi phương thức nhập từ lớp cha  
        System.out.println(" Nhập số tuyến:");  
        this.soTuyen = scan.nextLine();  
        System.out.println(" Nhập số KM:");  
        this.soKM = Float.parseFloat(scan.nextLine());  
    }  
}  
  
public class ChuyenXeNgoaiThanh extends ChuyenXe {  
    private String noiDen;  
    private float soNgayDiDuoc;  
  
    public ChuyenXeNgoaiThanh(){  
        super (); // Chỉ định phương thức khởi tạo mặc định lớp cha  
    }  
  
    @Override  
    public void nhap ( Scanner scan) {  
        super.nhap(); // Gọi phương thức nhập từ lớp cha  
        System.out.println(" Nhập nơi đến:");  
        this.noiDen = scan.nextLine();  
        System.out.println(" Nhập số ngày đi được:");  
        this.soNgayDiDuoc = Float.parseFloat(scan.nextLine());  
    }  
}
```



output

```
> Nhập mã số: A12  
> Nhập họ tên tài xế: CyberLearn  
> Nhập số xe: 51G-67.789  
> Nhập doanh thu: 80000  
> Nhập số tuyến: Số 8  
> Nhập số KM: 80.7
```

override & dẫn xuất

- Subclass thừa hưởng được các method từ superclass
 - Tuy nhiên **một số method** thừa hưởng được **không còn phù hợp nữa** như là không phù hợp hoàn toàn hoặc cách cài đặt hoàn toàn khác.
 - Khi đó những method đó **cần cài đặt lại sao cho phù hợp với subclass**.
- Để thực hiện **Override** nên thêm **@Override** trước method của subclass

@Override

```
public void input(){
```

...

```
}
```

Khi method Override cần dùng lại method superclass có thể gọi bằng cách **super.methodName (...)** của superclass

@Override

```
public void input(){
```

```
    super.input();
```

```
}
```

Nếu method của superclass có phạm vi là **private** thì **không thể Override**.

Nếu method của superclass có phạm vi là **protected** hoặc **public** thì có thể **Override**.

Nếu method của superclass có phạm vi là **protected** thì khi **Override** ở subclass có thể **chuyển thành** phạm vi **public**. Ngược lại thì không được

ĐA HÌNH (POLYMORPHISM)

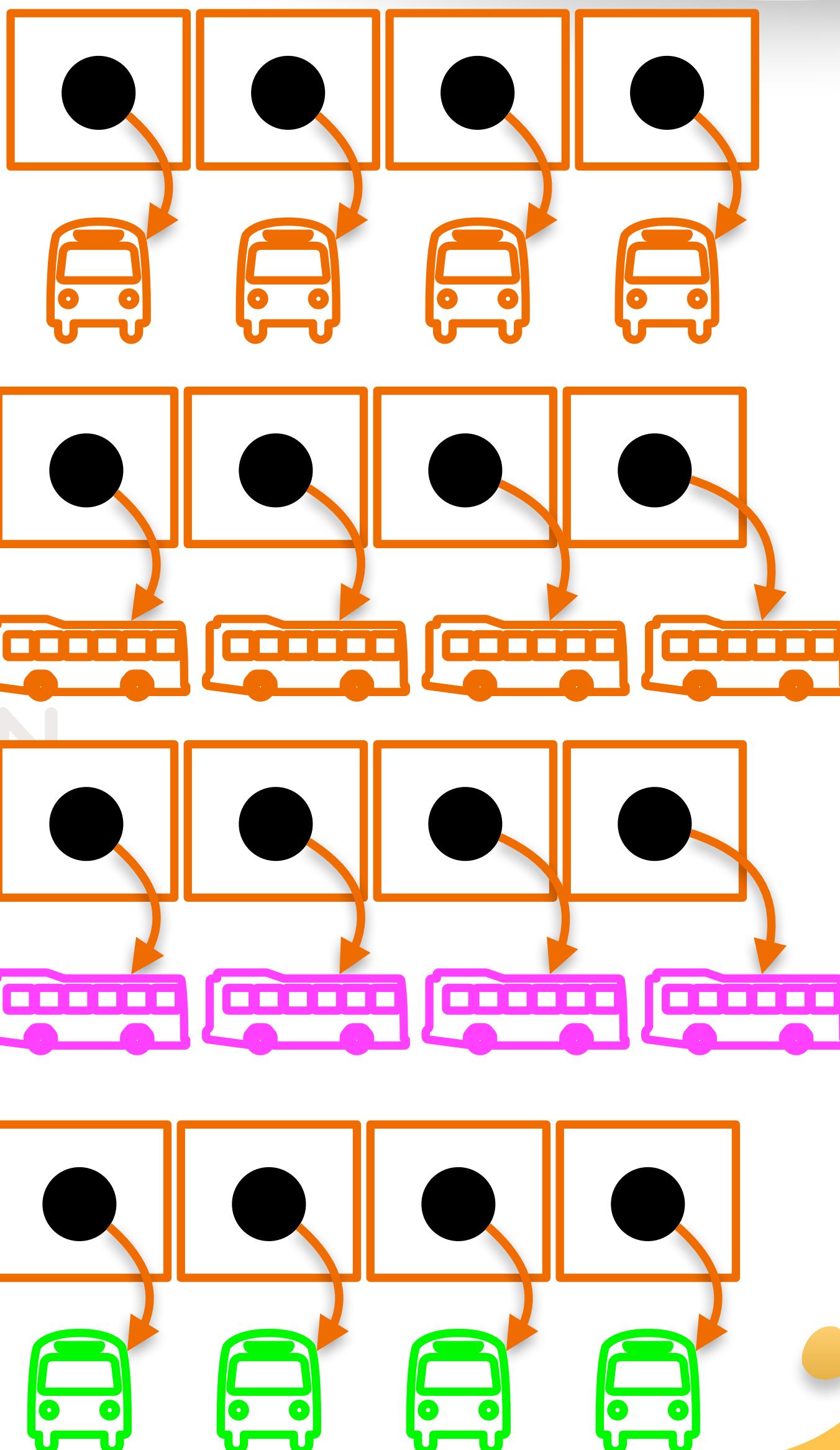
```
ArrayList<ChuyenXeNoiThanh> listNoiThanh = new ArrayList<ChuyenXeNoiThanh>();  
ChuyenXeNoiThanh cxnt = new ChuyenXeNoiThanh();  
listNoiThanh.add(cxnt);
```

```
ArrayList<ChuyenXeNgoaiThanh> listNgoaiThanh = new ArrayList<ChuyenXeNgoaiThanh> ();  
ChuyenXeNgoaiThanh cxngThanh = new ChuyenXeNgoaiThanh();  
listNgoaiThanh.add(cxngThanh);
```

```
ArrayList<ChuyenXeLao> listLao = new ArrayList<ChuyenXeLao> ();  
ChuyenXeLao cxLao = new ChuyenXeLao();  
listLao.add(cxngThanh);
```

```
ArrayList<ChuyenXeCampuchia> listCampuchia = new ArrayList<ChuyenXeCampuchia> ();  
ChuyenXeCampuchia cxCam = new ChuyenXeCampuchia();  
listCampuchia.add(cxCam);
```

TÍNH DOANH THU CHO TOÀN BỘ CÔNG TY ??



ĐA HÌNH (POLYMORPHISM)

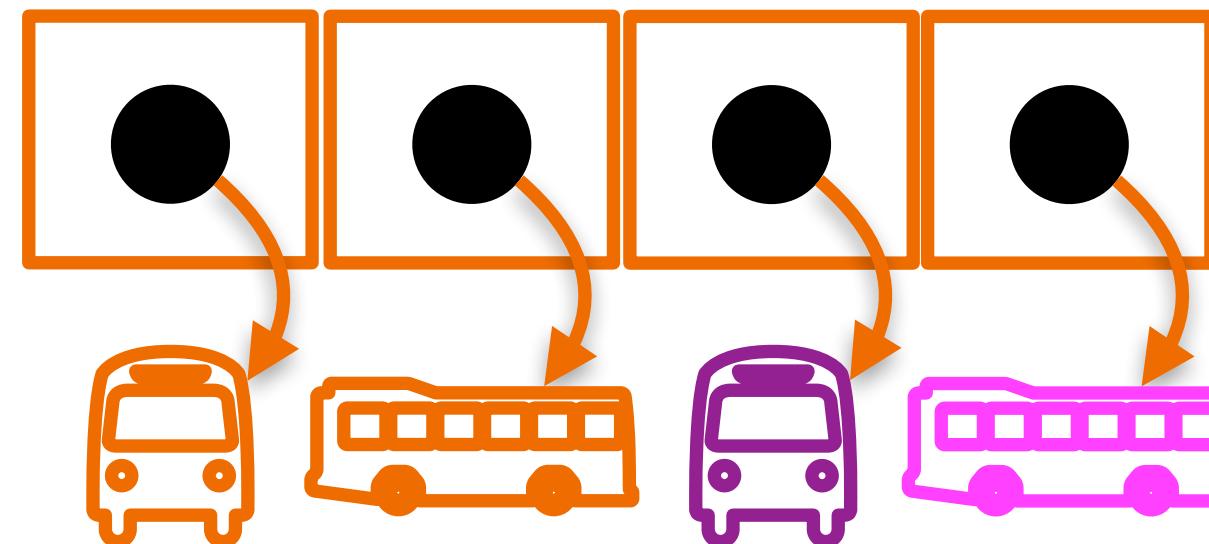
```
ArrayList<ChuyenXe> listChuyenXe = new ArrayList<ChuyenXe>();
```

```
ChuyenXe chuyenXe = new ChuyenXeNoiThanh();  
listChuyenXe.add(chuyenXe);
```

```
chuyenXe = new ChuyenXeNgoaiThanh();  
listChuyenXe.add(chuyenXe);
```

```
chuyenXe = new ChuyenXeLao();  
listChuyenXe.add(chuyenXe);
```

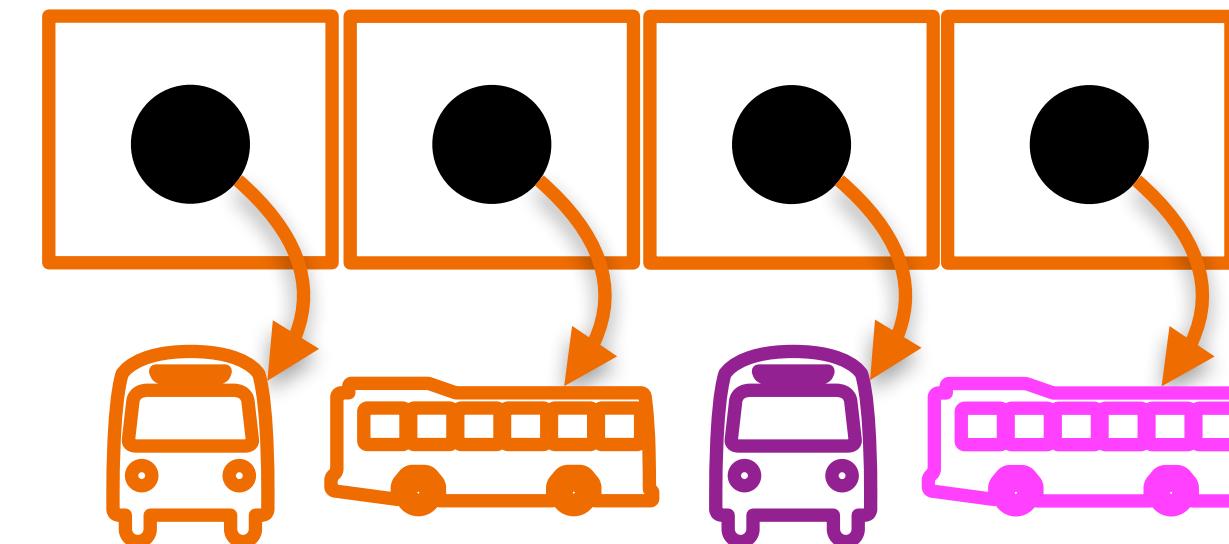
```
chuyenXe = new ChuyenXeCampuchia();  
listChuyenXe.add(cxCam);
```



Một object thuộc superclass có thể giữ reference đến object thuộc các subclass bất kỳ cấp nào.

ĐA HÌNH (POLYMORPHISM)

```
ArrayList<ChuyenXe> listChuyenXe = new ArrayList<ChuyenXe>();  
  
ChuyenXe chuyenXe = new ChuyenXeNoiThanh();  
listChuyenXe.add(chuyenXe);  
  
chuyenXe = new ChuyenXeNgoaiThanh();  
listChuyenXe.add(chuyenXe);  
  
chuyenXe = new ChuyenXeLao();  
listChuyenXe.add(chuyenXe);  
  
chuyenXe = new ChuyenXeCampuchia();  
listChuyenXe.add(cxCam);  
  
for (ChuyenXe cx: listChuyenXe){  
    this.tongDoanhThu += cx.tinhDoanhThu();  
}
```



Object thuộc superclass chỉ có thể **nhìn thấy trực tiếp** các method được cài đặt trong superclass nhưng **khi gọi** thì **thực sự gọi** các method của subclass nếu có **Override**

DEMO BÀI TOÁN XÀI TOÀN BỘ ĐÓNG GÓI + KẾ THỪA & ĐA HÌNH

Công ty du lịch V quản lý thông tin là các chuyến xe. Thông tin của 2 loại chuyến xe:

- * Chuyến xe nội thành: Mã số chuyến, Họ tên tài xế, số xe, số tuyến, số km đi được, doanh thu.
- * Chuyến xe ngoại thành: Mã số chuyến, Họ tên tài xế, số xe, nơi đến, số ngày đi được, doanh thu. Xây dựng chương trình quản lý các chuyến xe, tính tổng doanh thu của công ty và doanh thu từng loại xe

Tính doanh thu các cho công ty, cho ngoại thành và nội thành.

1. Đọc kỹ nghiệp vụ
2. Phân tích sơ đồ lớp, các cây kế thừa
3. Tổ chức lớp cha
4. Tổ chức lớp con
5. Xác định các phương thức Override
6. Xử lý các nghiệp vụ theo giải thuật

CYBERLEARN
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

DỰ ÁN CUỐI KHÓA



BÀI CUỐI KHÓA : QUẢN LÝ NHÂN SỰ

Xây dựng ứng dụng Quản lý nhân sự của 1 công ty bằng với các yêu cầu sau:

Công ty có tên công ty, mã số thuế, doanh thu tháng. Công ty có 3 loại nhân viên: giám đốc, trưởng phòng, nhân viên thường. Mỗi người trong công ty phải có các thông tin: mã số, họ tên, số điện thoại, số ngày làm việc, lương 1 ngày và cách tính lương. Ngoài các thông tin chung, mỗi chức vụ trong công ty còn có các thuộc tính riêng:

Nhân viên:

- Có thêm trưởng phòng quản lý. Nếu không có ai quản lý thì để null
- Công thức tính lương tháng : lương 1 ngày * số ngày làm việc .Lương 1 ngày của nhân viên: 100

Trưởng phòng:

- Có số lượng nhân viên dưới quyền. Thuộc tính này tăng lên khi có thêm 1 nhân viên thêm vào do trưởng phòng đó quản lý.
- Công thức tính lương tháng: lương 1 ngày * số ngày làm việc + 100 * số lượng nhân viên dưới quyền. Lương 1 ngày của trưởng phòng: 200

Giám đốc:

- Có thêm thuộc tính cổ phần trong công ty. Trị số này là số %, không được vượt quá 100%
- Công thức tính lương tháng : lương 1 ngày * số ngày làm việc. Lương 1 ngày của Giám đốc: 300

In ra menu cho chọn các chức năng sau: (Xuất thông tin theo biểu mẫu tablet có cột số thứ tự)

1. Nhập thông tin công ty
2. Phân bổ Nhân viên vào Trưởng phòng
3. Thêm, xóa thông tin một nhân sự (có thể là Nhân viên, trưởng phòng hoặc giám đốc). Lưu ý khi xóa trưởng phòng, phải ngắt liên kết với các nhân viên đang tham chiếu tới.
4. Xuất ra thông tin toàn bộ người trong công ty
5. Tính và xuất tổng lương cho toàn công ty
6. Tìm Nhân viên thường có lương cao nhất
7. Tìm Trưởng Phòng có số lượng nhân viên dưới quyền nhiều nhất
8. Sắp xếp nhân viên toàn công ty theo thứ tự abc
9. Sắp xếp nhân viên toàn công ty theo thứ tự lương giảm dần
10. Tìm Giám Đốc có số lượng cổ phần nhiều nhất
11. Tính và Xuất tổng THU NHẬP của từng Giám Đốc

Thu nhập = Lương tháng + số cổ phần * Lợi nhuận công ty

@Lợi nhuận công ty = Doanh thu tháng của công ty - tổng lương toàn công ty trong tháng

GIẢI THUẬT XÓA NHÂN SỰ

