JAVA OOP NÂNG CAO

Abstract class - Interface

Lớp trừu tượng
Abstract Class

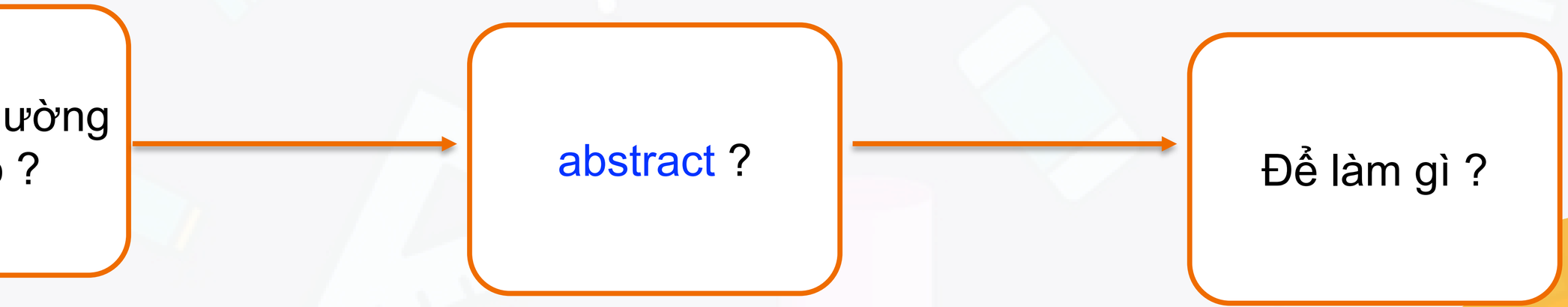
Interface
Kế thừa dạng Interface

Luyện tập dự án
Abstract, Interface, OOP, Đọc dữ liệu từ file

LỚP TRỪU TƯỢNG HAY ABSTRACT CLASS

Khai báo:

```
abstract  class  tên_lớp {
    các_thuộc_tính;
    các_phương_thức;
}
```



LỚP TRỪU TƯỢNG HAY ABSTRACT CLASS

- **Để đi phỏng vấn :) —> Hỏi Interface và abstract class khác thế nào**
- **Khai báo Lớp trừu tượng (Abstract) là trong lớp đó có ít nhất 1 phương thức trừu tượng (Abstract).**

Phương thức Trừu tượng là gì? Chính là phương thức có định nghĩa từ khóa ***abstract*** khi khai báo. Cú pháp khai báo một phương thức *Trừu tượng* cũng giống như khai báo một lớp *Trừu tượng* thôi.

```
[khả_năng_truy_cập]  abstract  kiểu_trả_về  tên_phương_thức  ([ các_tham_số_truyền_vào"]);
```

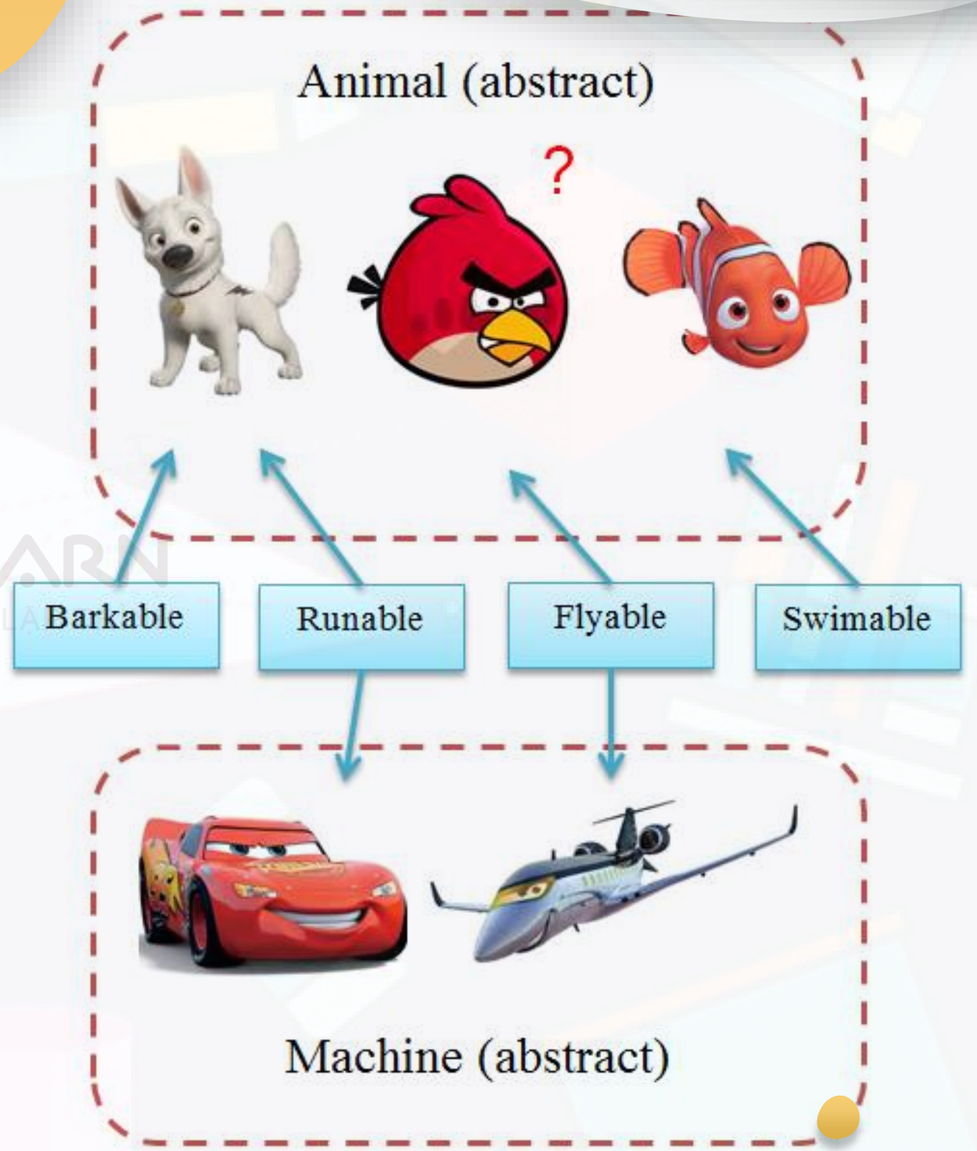
- **Chú ý:**
  - ★ **Phương thức trừu tượng**
    - Không có thân hàm
    - Các lớp con kế thừa buộc phải cài đặt hàm này
  - ★ **Lớp trừu tượng:**
    - Không thể tạo Thể hiện (Instance) của lớp này.

MỤC ĐÍCH

- Hãy suy nghĩ lại lớp không có Abstract và có Abstract về các phương thức liên quan đến Override ở lớp con
  - Phương thức chưa biết sẽ được cài đặt thế nào, chi tiết cài đặt sẽ nằm ở lớp con (Ví dụ tính diện tích hoặc chu vi cho các Class hình)
  - Nếu không có Abstract —> Lớp con không cài cũng ok —> Nếu là người Code lớp cha và người code lớp code là khác nhau —> Missed code
  - Nếu có Abstract —> Không thể nào bỏ sót, IDE tự báo là bắt buộc phải cài đặt các phương thức abstract này.

INTERFACE

- Không phải là class
- Quan hệ về mặt chức năng, gôm nhóm các phương thức hỗ trợ cho nhiều lớp có thể cài đặt.
- Một class có thể thực hiện nhiều interface (Đa kế thừa)
- Interface chỉ có thân hàm, không có cài đặt, việc cài đặt được thực hiện tại các lớp implement nó. Java 8 trở đi cho phép có thân hàm ở phương thức default và static.
- Các thuộc tính mặc định là static và public và final —> Không cần ghi khi khai báo, truy xuất như biến static (Dùng tên Interface truy xuất thẳng đến thuộc tính)
- Các phương thức mặc định là public và abstract
- Từ Java 8 trở đi có phương thức default và static



VÍ DỤ 1

```
public interface HanhDongDiChuyen {

    void diChuyen();

    void capNhapViTri();

}
```

Lớp: Xe

Lớp: Nhân Vật

SO SÁNH ABSTRACT CLASS VÀ INTERFACE TRONG JAVA

Abstract class	Interface
1) Abstract class có phương thức **abstract** (không có thân hàm) và phương thức **non-abstract** (có thân hàm).	Interface chỉ có phương thức **abstract**. Từ java 8, nó có thêm **các phương thức default và static**.
2) Abstract class **không hỗ trợ đa kế thừa**.	Interface **có hỗ trợ đa kế thừa**
3) Abstract class có các biến **final, non-final, static and non-static**.	Interface chỉ có các biến **static và final**.
4) Abstract class **có thể cung cấp nội dung cài đặt cho phương thức của interface**.	Interface **không thể cung cấp nội dung cài đặt cho phương thức của abstract class**.
5) Từ khóa **abstract** được sử dụng để khai báo abstract class.	Từ khóa **interface** được sử dụng để khai báo interface.
Class abstract đại diện cho mối quan hệ "IS - A"	Interface đại diện cho mối quan hệ "like - A"

DỰ ÁN NỘP - QUẢN LÝ TASK

Xây dựng ứng dụng quản lý công việc cho công ty CyberSoft. Công ty gồm Tên & Mã số thuế, được chia thành nhiều phòng ban, mỗi phòng ban gồm Tên phòng ban, mã phòng ban, Danh sách nhân viên và Trưởng phòng quản lý. Nhân viên bao gồm các thông tin : Mã, Tên, Năm sinh, Email, SĐT, Mã phòng ban (mặc định -1), Danh sách Task. Thông tin Task bao gồm: Mã, Tên, Thời gian thực hiện (theo giờ), Mã NV (mặc định -1). Mỗi phòng ban được quản lý bởi một Trưởng phòng. Mỗi Trưởng phòng có thêm thuộc tính mã phòng ban. Hãy tổ chức và xây dựng hướng đối tượng để thực hiện các yêu cầu sau:

1. Xây dựng các lớp đối tượng để phục vụ các yêu cầu trên.
2. Đọc file để tạo dữ liệu cho dự án
3. Phân bổ nhân viên về phòng ban. Xuất thông tin DS Nhân viên
4. Chỉ định Trưởng Phòng quản lý phòng ban
5. Phân bổ Task cho Nhân viên. Xuất thông tin DS NV
6. Nhân viên (Tên, mã) nào đang thực hiện nhiều Task nhất. (Tính theo số lượng các task)
7. Nhân viên (Tên, mã) nào có tuổi trẻ nhất đang quản lý nhiều Task nhất. (Tính theo số lượng các task)
8. Phòng ban nào có nhân viên trẻ nhất
9. Xóa một nhân viên.
10. Sắp xếp DS Nhân viên theo thứ tự Task nhiều nhất lên trên cùng như mẫu dùng QuickSort

STT | Mã NV | Tên | Năm sinh | Mã Phòng | DS Mã Task

11. Sắp xếp Tên nhân viên theo thứ tự ABC dùng InterchangeSort
12. Tính lương như sau cho nhân viên như sau

- Lương nhân viên được tính dựa vào:
  - Lương = Số ngay làm việc * 200/1ngay + 300 phụ cấp (nếu số giờ làm task >35h sẽ được phụ cấp thêm 1000)
  - Lương trưởng phòng = Số ngày làm việc * 300/1ngay + 200 * số nhân viên quản lý thuộc phòng ban đó

13. Tính lương cho toàn công ty.

ĐỌC FILE

```
private static void readFile() {
    try {
        FileReader reader = new FileReader("src/Task.txt");
        BufferedReader bufferedReader = new BufferedReader(reader);
        String line;
        while ((line = bufferedReader.readLine()) != null) {
            System.out.println(line);
            String[] listInfo = line.split(" # ");
            for (String a : listInfo)
                System.out.println(a);
        }
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```