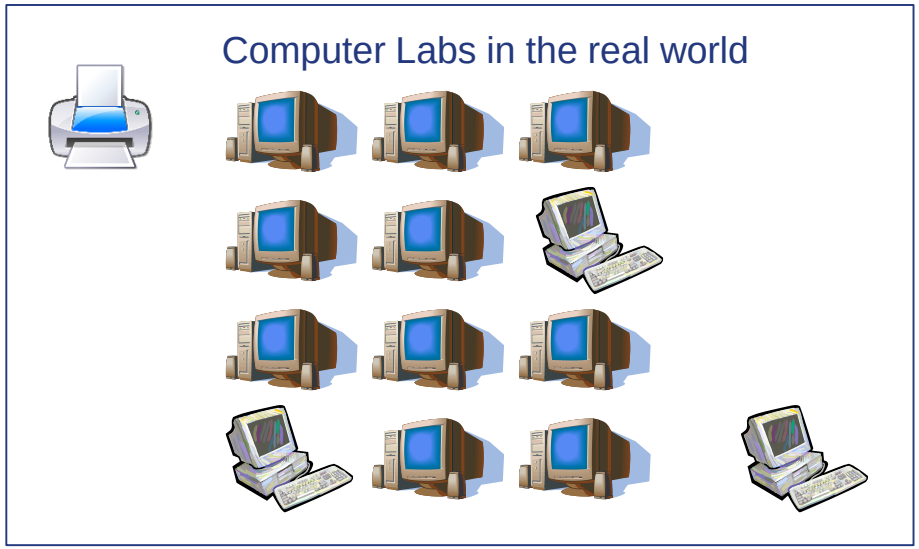
# Topic 5
# Classes

Definition of Classes
Data member and Member function
private and public members
friend function and friend class
static data member and static member function
const member function
Self reference: this
Nested classes

# Computer Labs Management System

- Objects sometimes correspond to things found in the real world
- Object-oriented programming is a whole programming paradigm based around objects (data structures) that contain data fields and methods




<Objects>

<Abstraction>

Computer

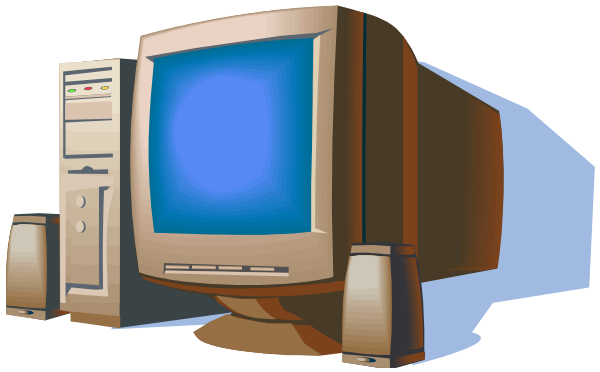
Monitor

Printer

<Class>

https://en.wikipedia.org/wiki/Object-oriented_programming

# Object: Examples



ID: IP-150

*Property*

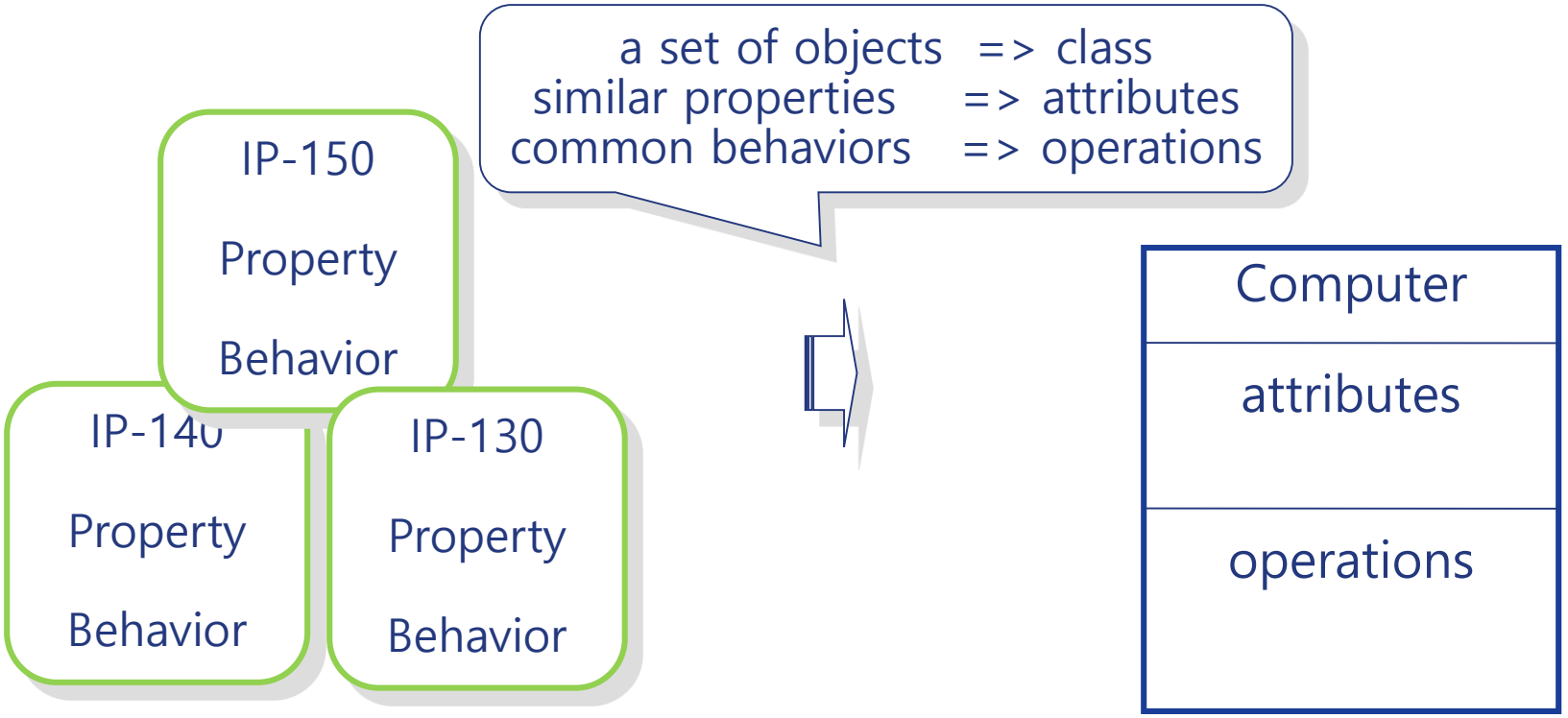
IP-150
Samsung Electronics
Monitor
3,000,000

*Behavior*

run
power-off

# Class

- A class defines a set of objects with similar properties and common behavior



# Object-oriented programming in C++

- 추상화 (Abstraction): 추상화는 어떤 특정 정보를 표시해야 하고 어떤 정보를 숨겨야 하는지 식별하는 데 도움이 되는 기술 ("핵심이 뭐야?", "어느 레벨에서 설명 드릴까요?")
  - 절차- 세부 사항을 무시하면서 필수 정보를 추출,
  - 실체(entity) – 실제에 대한 모델(model), 뷰(view), 표현(representation)
- 캡슐화 (Encapsulation): 숨겨야 할 내용을 숨기고 표시할 내용을 표시하는 방식으로 정보를 패키징하는 기술 (예 - 함수, 클래스, 모듈 등, "캡슐화된 모든 것이 숨겨져 있는가?")
  - 절차 – 물리적 논리적으로 둘러싸는 행위
  - 엔티티 – 패키지 혹은 인클로저

  ★ 정보 은닉(information hiding): 기능을 수행하는데 필요한 내부 구현이나, 변경될 가능성이 있는 의사결정의 내부 사항을 숨김

# Object-oriented programming in C++

- 상속 (Inheritance): 파생 클래스(derived class, 상속 받는 클래스)가 기본 클래스(base class)의 모든 멤버 변수와 멤버 함수를 포함함
- 다형성 (Polymorphism): 서로 다른 유형의 엔티티에 대한 단일 인터페이스를 제공하거나, 단일 기호를 사용하여 여러 유형을 나타내는 것 (동일한 함수 이름이 다르게 동작할 수 있음 - 함수 오버로딩, 클래스 멤버 함수 오버라이딩 등)

| 상속 | 다형성 |
|---|---|
| 기본적으로 클래스에 적용됨 | 기본적으로 함수나 클래스의 멤버 함수에 적용됨 |
| 코드의 재사용성을 높이고 코드 길이를 줄임 | 객체는 컴파일 타임이나 런타임에 어떤 형태의 함수를 실행할지 결정할 수 있음 |
| single, hybrid, multiple, hierarchical and multilevel inheritance | 컴파일 타임 다형성 (오버로드)과 런타임 다형성 (오버라이드)이 될 수 있음 |

https://www.geeksforgeeks.org/difference-between-inheritance-and-polymorphism/

# Role, Responsibility, and Collaboration

- **요청(request)**과 **응답(response)**을 통해 다른 **객체(object)**와 **협력(collaboration)**


- 객체간 **협력**하는 과정 속에서 특정한 **역할(role)**을 부여받음
- **역할**은 어떤 **협력**에 참여하는 특정한 객체가 **협력** 안에서 차지하는 **책임(responsibility)**이나 임무

# 객체 지향(object-oriented)

- 상태(state)와 행동(behavior)을 함께 지닌 자율적인 객체
- 적절한 책임(responsibility)을 수행하는 역할(role) 간의 유연하고 견고한 협력(collaboration) 관계를 구축
- 훌륭한 객체지향(object-oriented) 설계자가 되기 위한 도전
  - 코드를 담는 클래스의 관점에서 **메시지를 주고 받는 객체의 관점으로 사고의 중심을 전환**하는 것
  - 어떤 클래스가 필요한가가 아니라 **어떤 객체들이 어떤 메시지를 주고 받으며 협력하는가에 집중**
- 객체지향은 객체를 지향하는 것이지 클래스를 지향하는 것이 아님

| 객체 | 클래스 |
|---|---|
| 타입(type)은 객체를 분류하는 기준<br>예) 프린터, 컴퓨터, 모니터 | 타입(type)을 구현할 수 있는 여러가지 구현 방법 중 하나<br>(자바 스크립트는 ES5까지 객체를 prototype 이용해서 구현) |
| 객체가 어떤 타입에 속하는지를 결정하는 것은<br>객체가 수행하는 행동<br>예) 출력하는 행동을 하는 객체는 프린터 타입으로 분류 | - |
| 프로그램이 실행되어 객체의 상태 변경을 디버깅하는 시점<br>(동적인 모델) | 프로그래밍 언어를 이용해 클래스를 작성하는 시점<br>(정적인 모델) |

조영호, 객체지향의 사실과 오해, 위키북스, 2015