‘Make It Scale’ – An Introduction

ENCE464 Embedded Software and Advanced Computing

Course coordinator: Steve Weddell (steve.weddell@canterbury.ac.nz)

Lecturer: Le Yang (le.yang@canterbury.ac.nz)

Department of Electrical and Computer Engineering

Where we’re going today

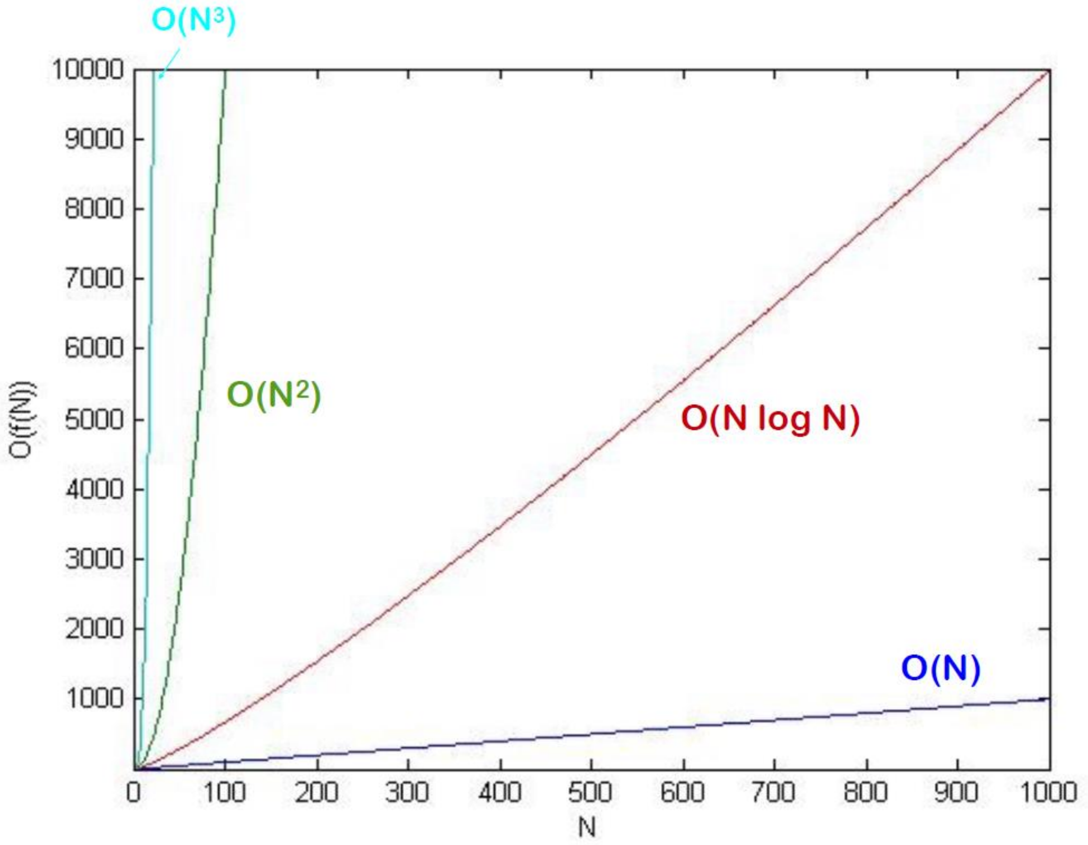
- **Tyranny of scale**

- Design for changes

- Manage dependencies

Tyranny of scale

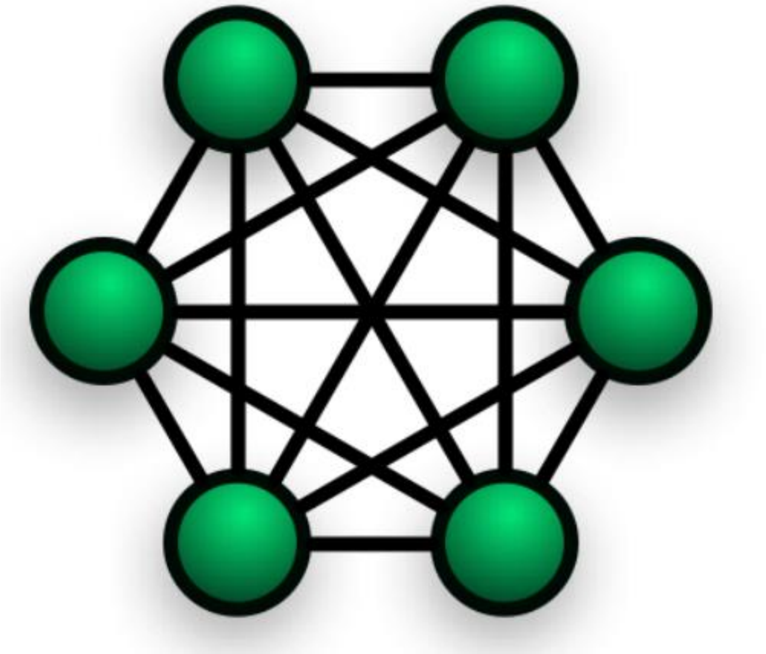
- Complexity increases in the order of O(N), $O(NlogN)$, $O(N^2)$ and $O(N^3)$



- Even worse situation: exponential increase in the complexity

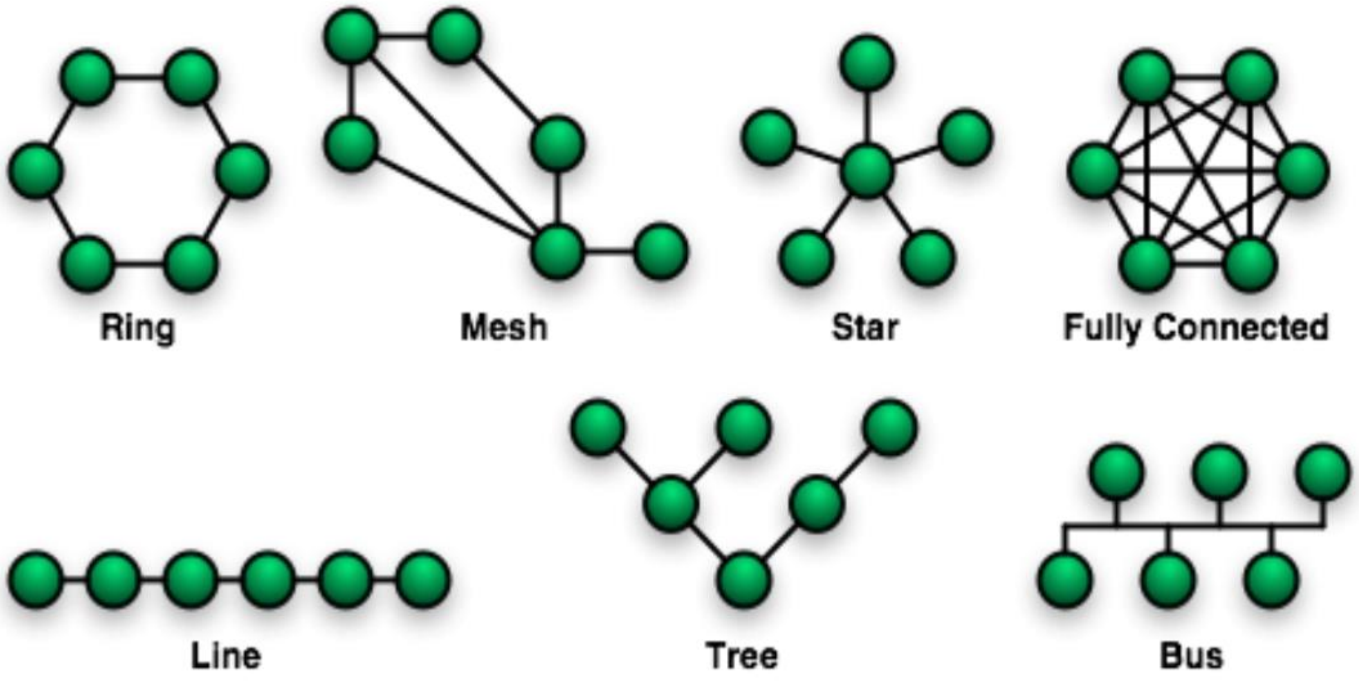
Telephone Networks

- Use a separate line to connect any user pair
- P users in total → $C_P^2 = \frac{P!}{(P-2)!2!} = \frac{P(P-1)}{2}$
  - This approach does not scale well



Computer Networks

- Other topologies proposed other than the fully connected topology



- Cost?
- Pros and cons?

Where we’re going today

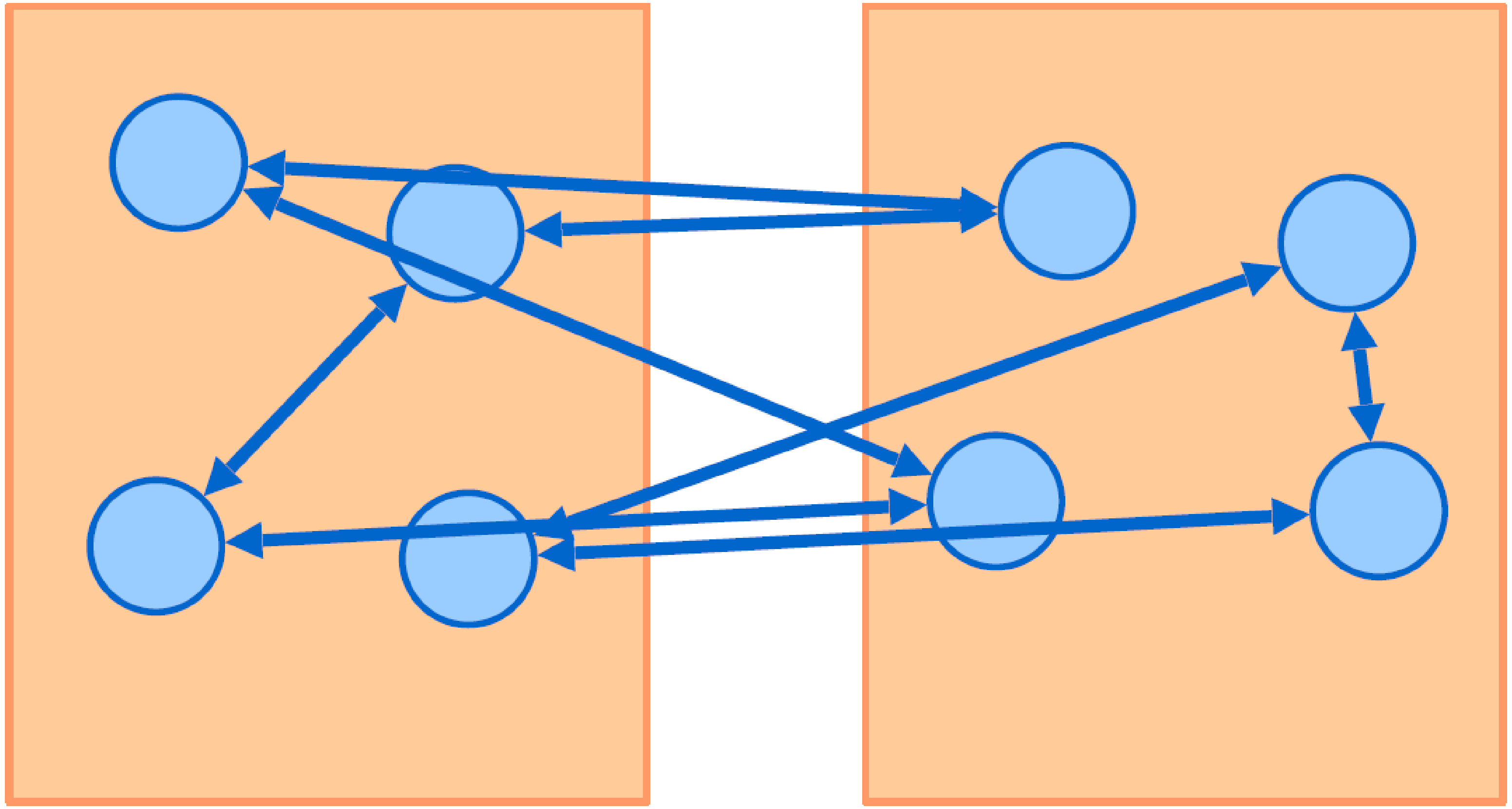
- Tyranny of scale
- **Design for changes**
- Manage dependencies

Approaches at a Glance

- Encapsulation
  - One of the fundamental concept in object-oriented programming (OOP)
  - Bundling data and methods that work on the data within one unit
    - Hide internal representation or state of an object from outside (information hiding at implementation level)
- Abstraction
  - One of the fundamental concept in object-oriented programming (OOP)
  - Hide unnecessary details (in design level) and allow users to realize more complex operations based on provided abstraction without knowing them
- Interfaces
  - Programming structure provided by an object that allows enforcing certain properties or operations

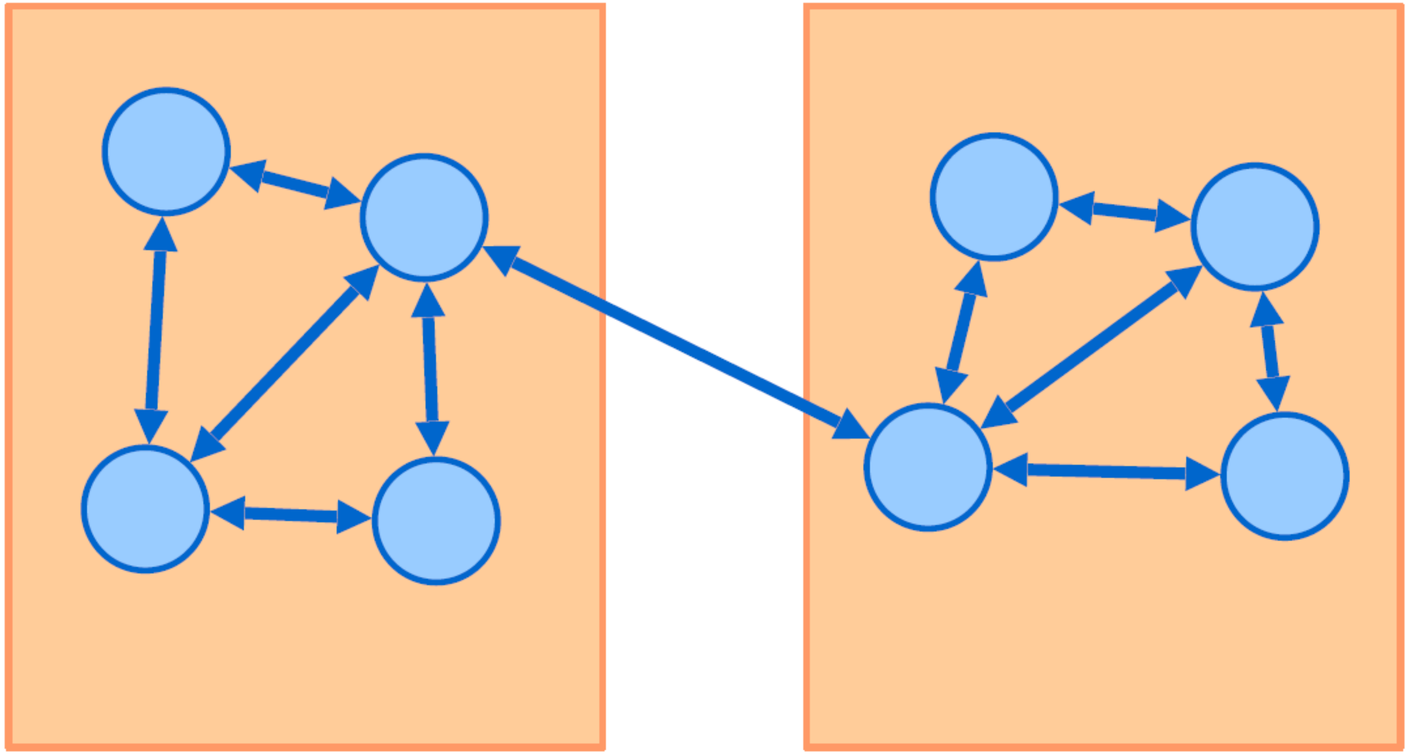
Modularization (1)

- Poor design example



Modularization (2)

- Improved design (example_



Where we’re going today

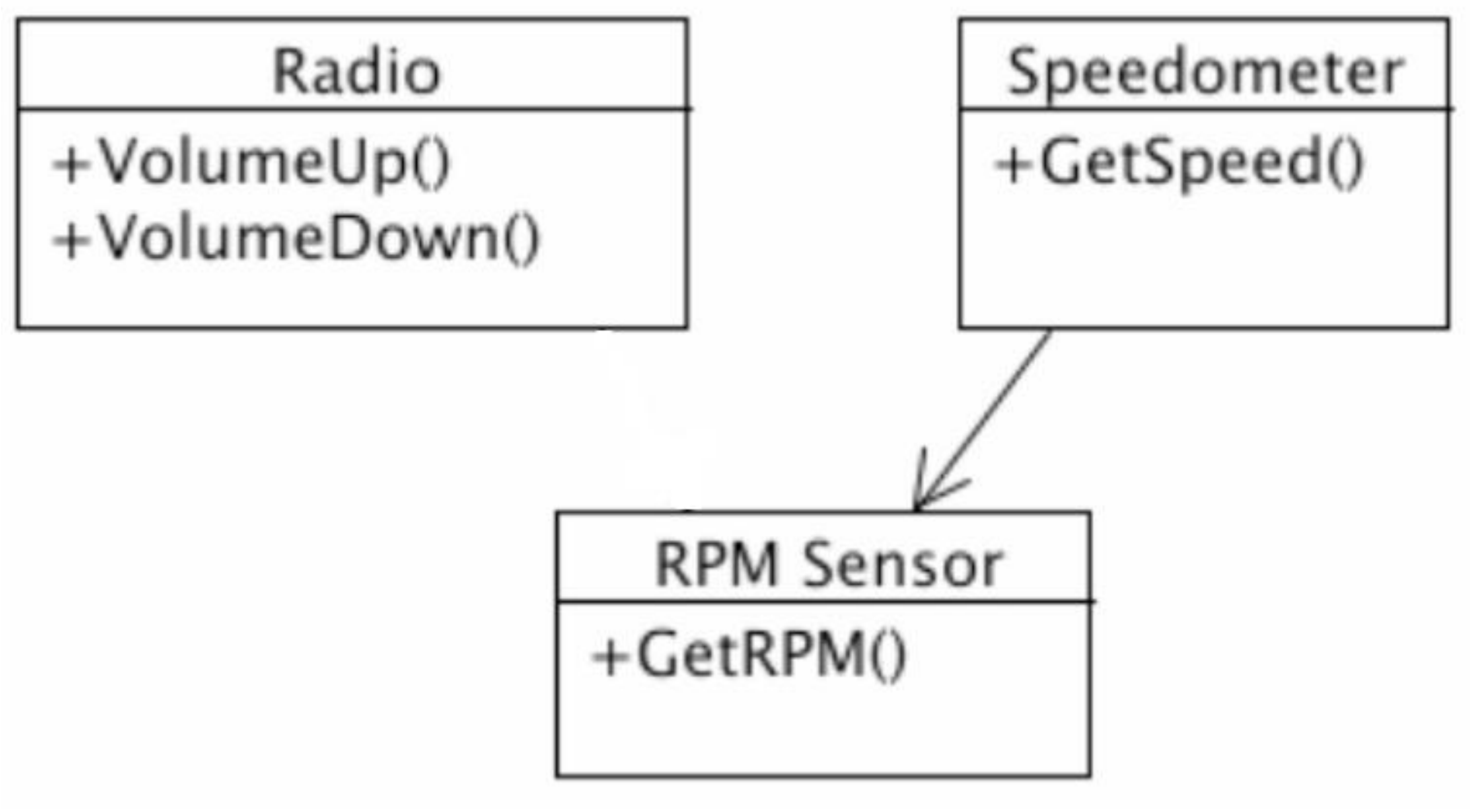
- Tyranny of scale
- Design for changes
- **Manage dependencies**

Illustrative Example (1)

- Change in requirements
  - Previous: in 2016 model, the radio is independent from the whole care
  - Now: in 2017 model, the radio volume can be automatically adjusted according to vehicle speed
    - Maybe for lowering the noise level at high speeds
  - Approaches
    - Let radio know the RPM of the wheels
    - Radio asks speedometer for current speed information
    - Speedometer sends volume adjustment request to radio
    - All these methods degrade the previously nice independent design

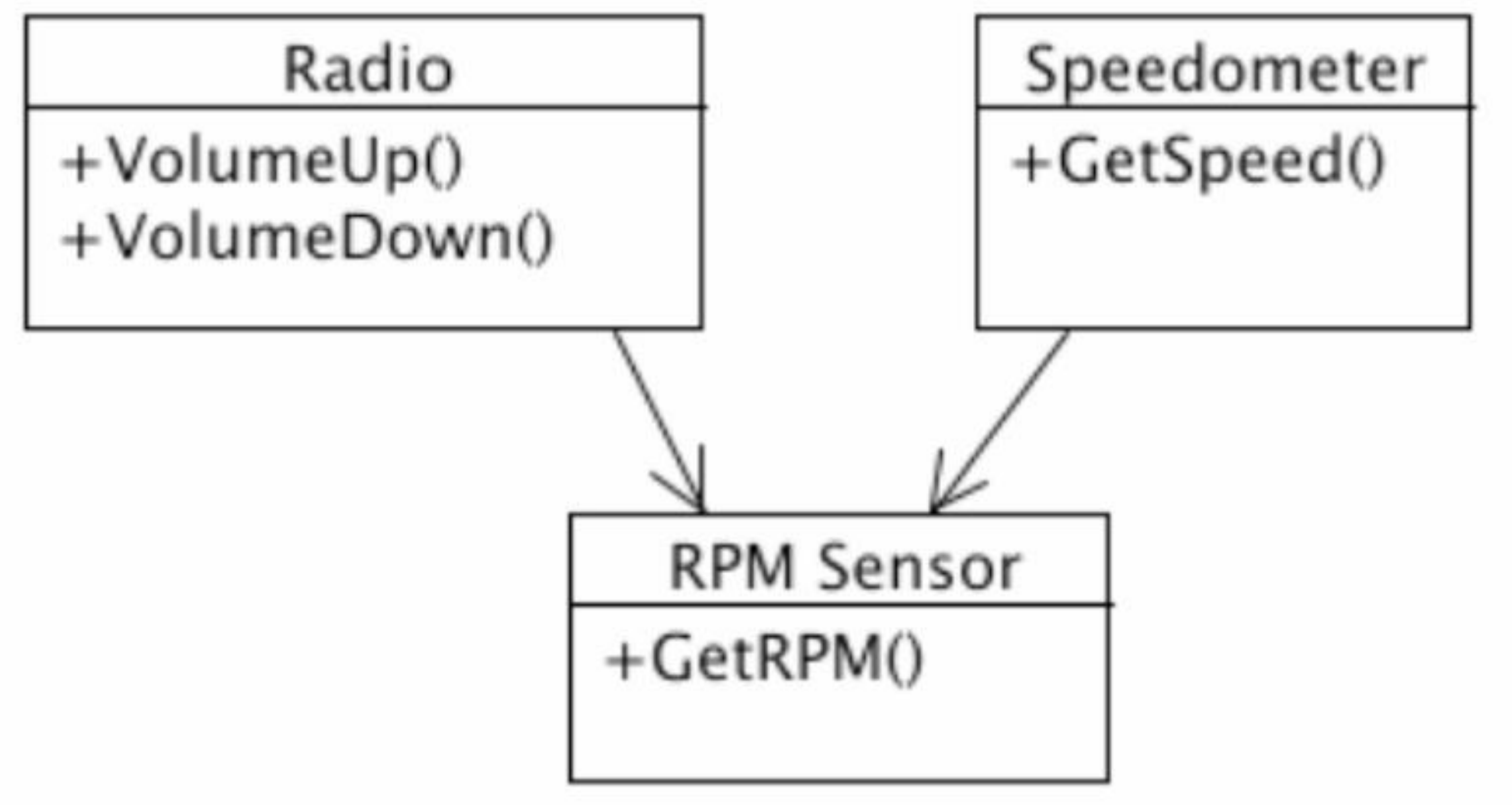
Illustrative Example (2)

- In 2006 model, radio was completely independent from the car, except for power requirement



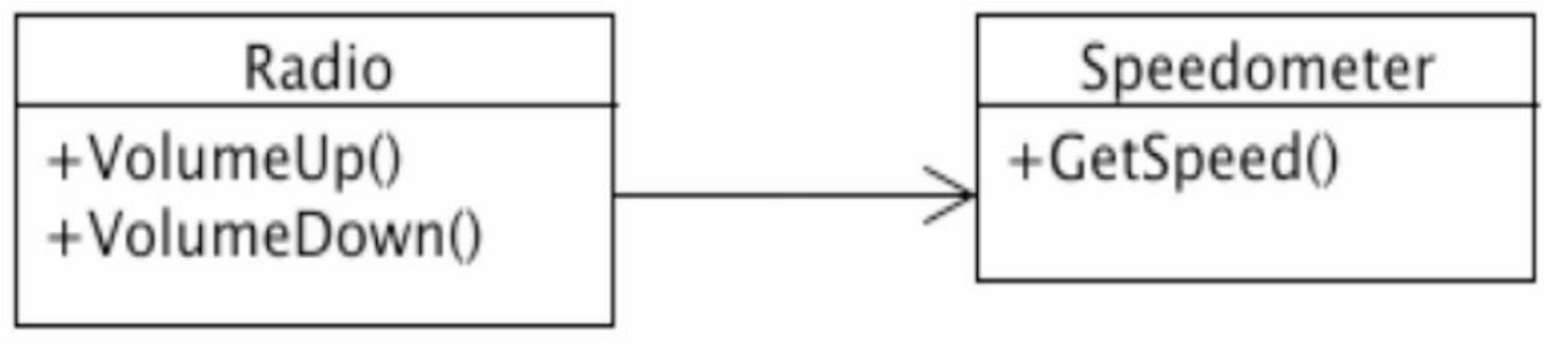
Illustrative Example (3)

- Approach 1: Radio and speedometer both know RPM sensor readings



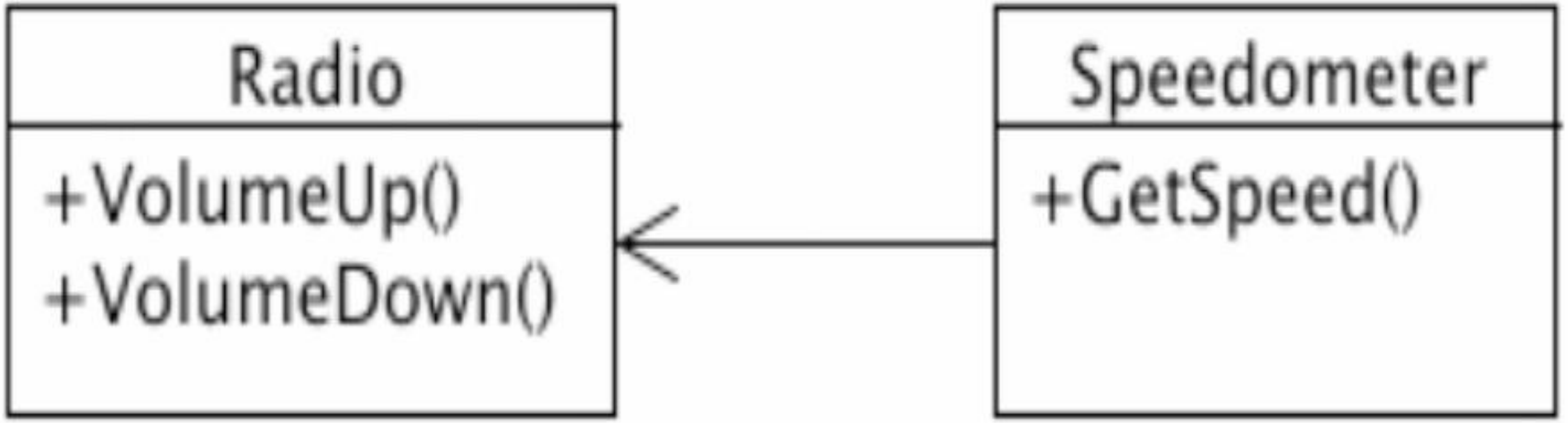
Illustrative Example (4)

- Radio asks speedometer for current speed information



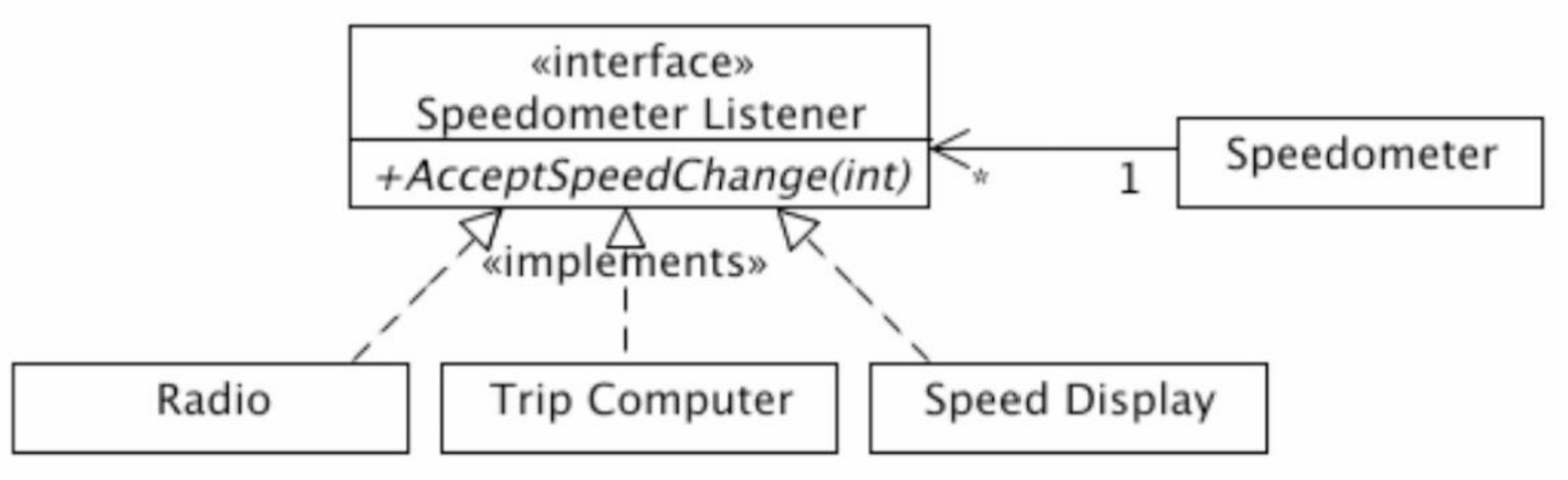
Illustrative Example (5)

- Speedometer sends volume adjustment request to radio



Illustrative Example (6)

- Design that adheres to the Open/Closed principle



- Bertrand Meyer wrote in 1988 in his book Object-oriented software construction
  - *Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification*
  - Allow adding new functionality without changing existing code
  - Prevent that a change to one of classes also requires adapting all depending classes

Illustrative Example (7)

```
#ifndef SPEEDOMETER_LISTENER_H_
#define SPEEDOMETER_LISTENER_H_
/* SpeedometerListener.h
  * Header for the Speedometer Listener module. Provides an interface
  * so that other modules can receive notification when the speed
  * changes. Maintains a dynamically allocated list of listener callback functions to call.  */

/* RegisterSpeedometerListener: Provide a pointer to a call back
  * routine that is to be called in the event of a speedometer change.
  * The callback function must accept one integer parameter
  * and have void return type. Returns TRUE for success, FALSE if
  * memory allocation fails. */
boolean RegisterSpeedometerListener (void (*AcceptSpeedChange) (int));

/* SignalSpeedChange: Function for the Speedometer to call ("Tell,
* don't ask" principle) when it detects a speed change. */
void SignalSpeedChange(int newSpeed);
#endif
```

Required Reading on LEARN

- James W. Grenning, “Object oriented design for embedded software engineers,” Embedded Systems Conference, San Jose, CA, 2007

- James W. Grenning, “SOLID design for embedded C,” Embedded Systems Conference, San Jose, CA 2011