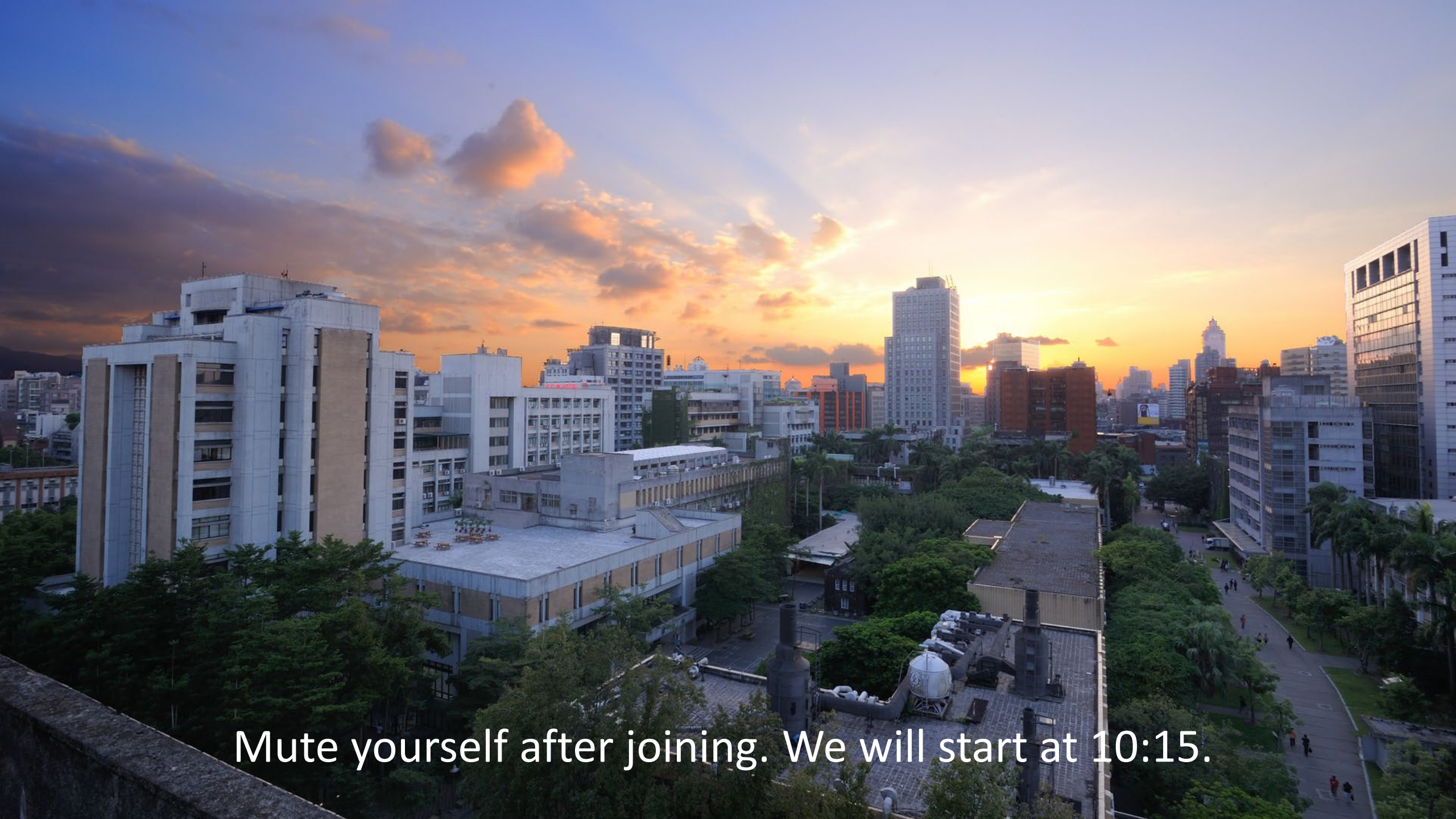

Mute yourself after joining. We will start at 10:15.

TAIPEI TECH 國立臺北科技大學 National Taipei University of Technology

Department of Computer Science and Information Engineering

Object Oriented Programming

Lecture 00: Course Information

Shuo-Han Chen (陳碩漢),
shchen@ntut.edu.tw

The Sixth Teaching Building 327
M 15:10 - 16:00 & F 10:10 - 12:00

Course Information

- Object Oriented Programming
- Course Time and Place
  - Lecture (3 Hours)
    - Monday 15:10 - 16:00
    - Friday 10:10 - 12:00
    - The Sixth Teaching Building 327
  - Midterm & Final will take place Online or in Computer Classroom
    - Check Announcement beforehand
- Course Website : https://css-gitlab.csie.ntut.edu.tw/109000000/oop2021f

Course Website

Course Instructor & Teaching Assistant

- Course Instructor
  - Dr. Shuo-Han Chen (陳碩漢)
    - Office: 宏裕科技大樓 1532
    - Office Hours: Monday 11:00 - 15:00
    - Email: shchen@ntut.edu.tw
- Teaching Assistants
  - 林立軒、陳廷豪
  - Office: 宏裕科技大樓 438
    - Office Hours: Wednesday 13:00 - 15:00、Thursday 13:00 - 17:00
    - Email: {t110598040, t110598065}@ntut.edu.tw

Admission to this Course

- My policy : First come first serve & Senior students first
- We are at our upper limit of accepting new students
  - Maximum : 95
  - Current : 99

學生來源	人數	等待加選
僑生	3	1
輔/外系	7	3
隨班附讀	6	
電資	2	1
資工碩	1	
資工二	60	
資工三	9	
資工四	5	1
加總	93	6

  - You may fail this course
  - Please reconsider why joining this course

Prerequisites

- Addend & Interact
- Familiar with C or Taken 程設 (一) & (二)
- Do not be afraid of English
- Read the text book

- And, of course, willing to learn C++

The book we are using

- We will assign reading chapter every week.
- It would be very helpful if you can get one

Absolute C++, 6/e (IE-Paperback)

Walter Savitch , Kenrick Mock

出版商:	Pearson FT Press
售價:	$1,390
貴賓價:	9.8 折 $1,362
語言:	英文
裝訂:	Paperback
ISBN:	1292098597
ISBN-13:	9781292098593
相關分類:	C++ 程式語言

立即出貨 讚 0 分享

加入購物車 加入追蹤清單

Course Grading Policy

- **Participance: 10 % -> BOPPPS & IRS**
- **Assignment: 30 %**
  - Weekly or Biweekly
    - HW that requires PDF report -> submit to i 學園
    - HW that requires code -> gitlab: https://css-gitlab.csie.ntut.edu.tw/users/sign_in
    - Accounts will be created by next Wednesday – If you have no student ID, we will assign one
- **Midterm Exam: 30 %**
- **Final Exams: 30 %**
  - 50 minutes hand-written exams on Monday
  - 3 hour Computer-based Exams on Friday
  - Do the survey and mark your calendar before 10/01 24:00 !
    - https://forms.gle/tWaX8RrX7t1gDKwU8
    - If you do, you will get +1 bonus on the final grade
    - If you do not, you will get -3 points bonus on the final grade

GitLab Login Page

Exam Time Survey

Course Schedule

Due to national holidays, there will be no class on Sept. 20, Oct. 11, and Dec. 31.

W	Date	Lecture	Method	Readings	Homework
1	Sept. 20, 24	Lec01: Moon Festival / Introduction & Environment Setup		Chapter 1	HW00
2	Sept. 27, Oct.1	Lec02: Flow of Control			
3	Oct. 4, 8	Lec03: Function Basics / Exception Handling			
4	Oct. 11, 15	Lec04: National Day / String	(15) BOPPPS & IRS		
5	Oct. 18, 22	Lec05: Pointer	(18) BOPPPS & IRS		
6	Oct. 25, 29	Lec06: Parameters and Overloading			
7	Nov. 1, 5	Lec07: Structures and Classes	(1) BOPPPS & IRS		
8	Nov. 8, 12	Lec08: Constructors and Other Tools			
9	Nov. 15, 19	Lec09: Operator Overloading, Friends, and References	(19) BOPPPS & IRS		
10	Nov. 22, 26	Midterm -> 50M Hand-written on Nov. 22 and 3H Computer-based on Nov. 26		No Class	
11	Nov. 29, Dec. 3	Lec10: Inheritance			
12	Dec. 6, 10	Lec11: Inheritance			
13	Dec. 13, 17	Lec12: Polymorphism and Virtual Functions	(12) BOPPPS & IRS		
14	Dec. 20, 24	Lec13: Polymorphism and Virtual Functions			
15	Dec. 27, 31	Lec14: Standard Template Library / New Year Holiday			
16	Jan. 3, 7	Lec15: Templates			
17	Jan. 10, 14	Lec16: Streams and File I/O / Namespace	(14) BOPPPS & IRS		
18	Jan. 17, 21	Final -> 50M Hand-written on Jan. 17 and 3H Computer-based on Jan. 21		No Class	

BOPPPS & IRS雙手聯彈

- It’s an in-school project for practicing new teaching method
- It’s composed of 6 steps
  - Bridge-in (導言)
  - Objective (學習目標)
  - Pre-assessment (前測) + Zuvio
  - Participatory Learning (參與式學習) -> Write code together + Cosmobuzz + Peardeck
  - Post-assessment (後測) + Zuvio
  - Summary (總結) + Peardeck
- What you should do? Don’t worry too much
  - Please read the textbook before class
  - Actively answer questions on Zuvio
  - It’s will be considered as part of participance

Object-Oriented Programming (OOP)

- OOP is a program design philosophy
  - Modularization
  - Reusability
- The key idea is
  - To describe the real world accurately as a collection of objects that interact
- Everything in OOP is self sustainable *objects*, which include
  - States (data)
  - Behaviors (methods)

Procedural Programming	Object-Oriented Programming
Functions	Objects
Top down Approach	Bottom up approach
C	C++, Java, Python, C#

If you're still unsure about the differences

- Example: We are writing a program to control toy airplane
- Procedural Programming – function-based
  - In this example, you need to control the airplane by functions

```c
#include <stdio.h>

struct Airplane
{
    double speed;
    double heading;
};

void controlSpeed(Airplane plane, double value) {
  plane.speed += value;
}

void setHeading(Airplane plane, double value) {
  plane.heading = value;
}

```

```c
int main(){
  Airplane myAirplane = {
    .speed = 0,
    .heading = 0
  };

  controlSpeed(myAirplane, 100);
  setHeading(myAirplane, 180);

  return 0;
}
```

If you're still unsure about the differences (Cont'd)

- But in object-oriented programming, the state and functions are encapsulated into an object (class)

```
1   #include <iostream>
2   using namespace std
3
4   class Airplane {
5
6   public:
7       Airplane(double speed, double heading)
8       void void controlSpeed(double value);
9       void oid setHeading(double value);
10
11  private:
12        double speed;
13        double heading;
14  };
```

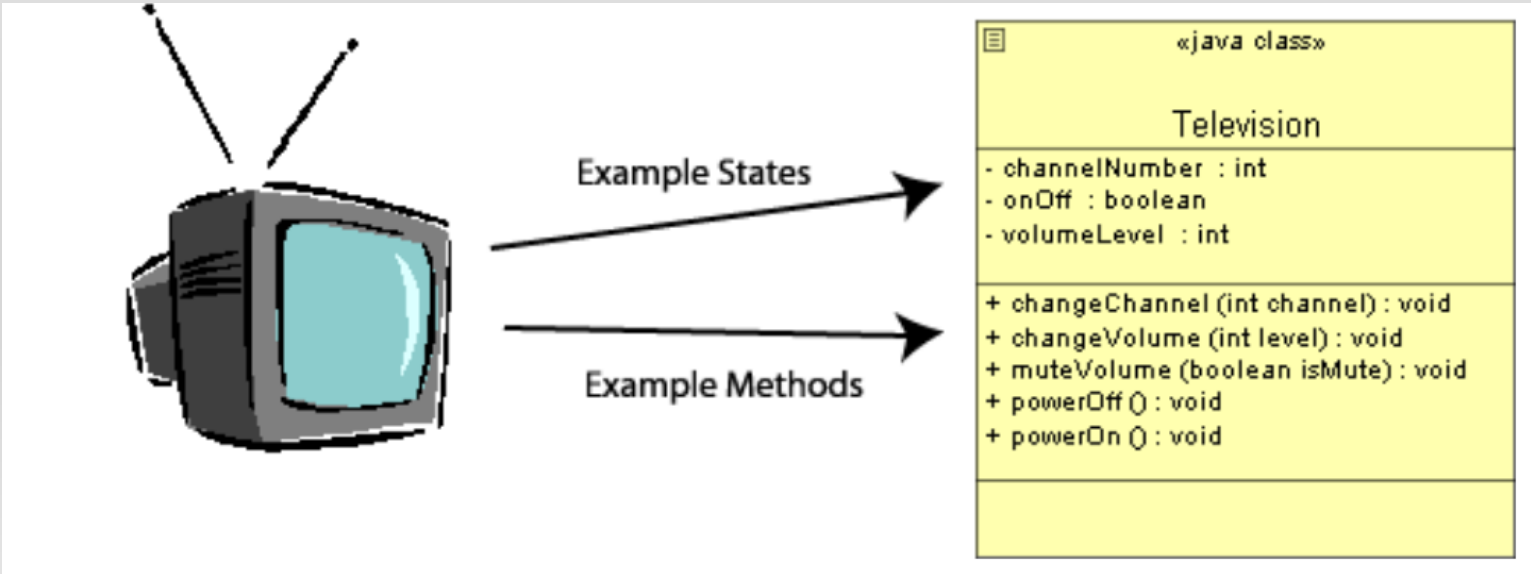
```
16  int main(){
17    Airplane myAirplane = new Airplane(0, 0);
18
19    myAirplane.controlSpeed(100);
20    myAirplane.setSpeed(180);
21
22    return 0;
23  }
24
25  void Airplane::controlSpeed(double value) {
26    plane.speed += value;
27  }
28
29  void Airplane::setHeading(double value) {
30    plane.heading = value;
31  }
```

Main Features of OOP

- OOP generally supports
  - Classes
  - Objects
  - Encapsulation
  - Inheritance
  - Polymorphism

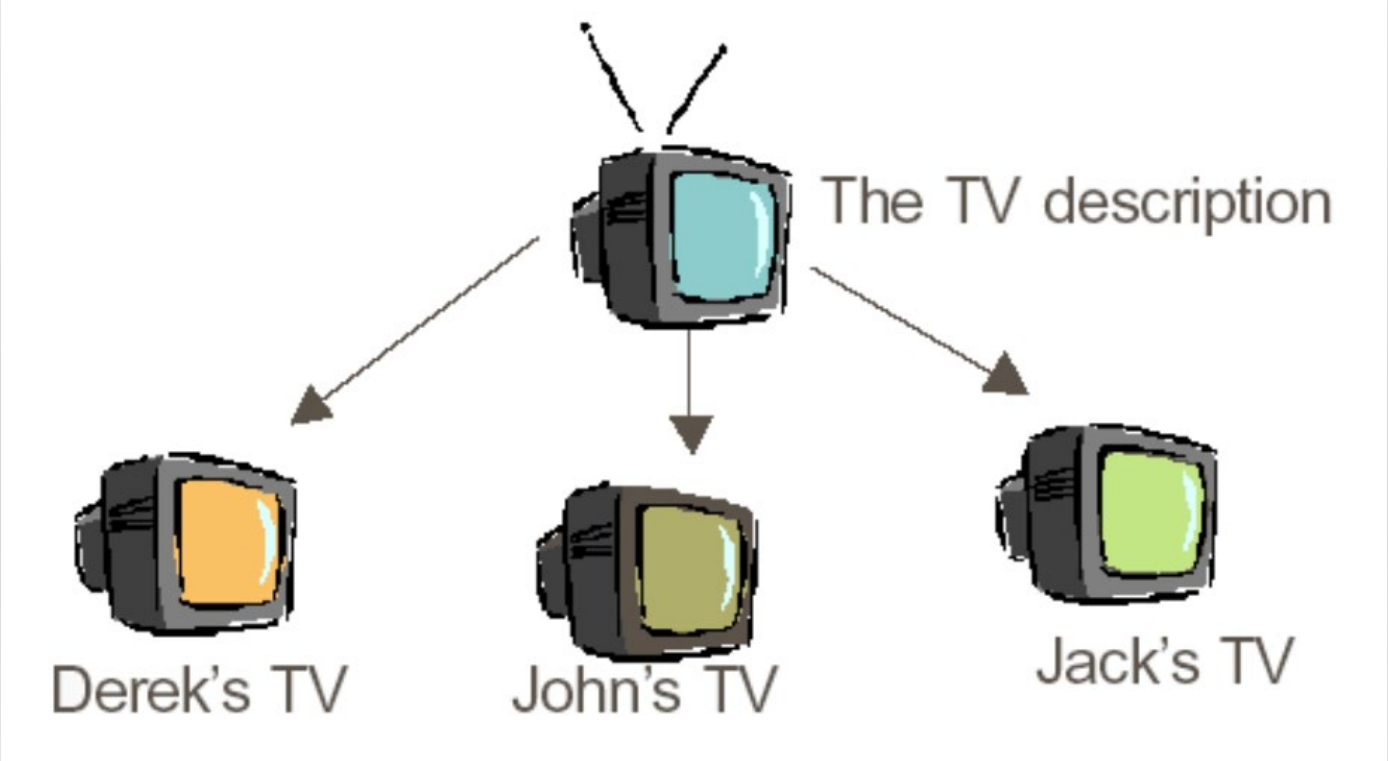
Classes

- Represent the structure of objects with
  - State (Data) are the states that the object has
  - Behavior (Methods) are the ways in which the object can be interacted with
- Class should
  - Represent a clear concept - Such as the concept of a television
  - Provide a well-defined interface - Such as the remote control of the television
  - Be complete and well-documented - TV should have a plug and a manual



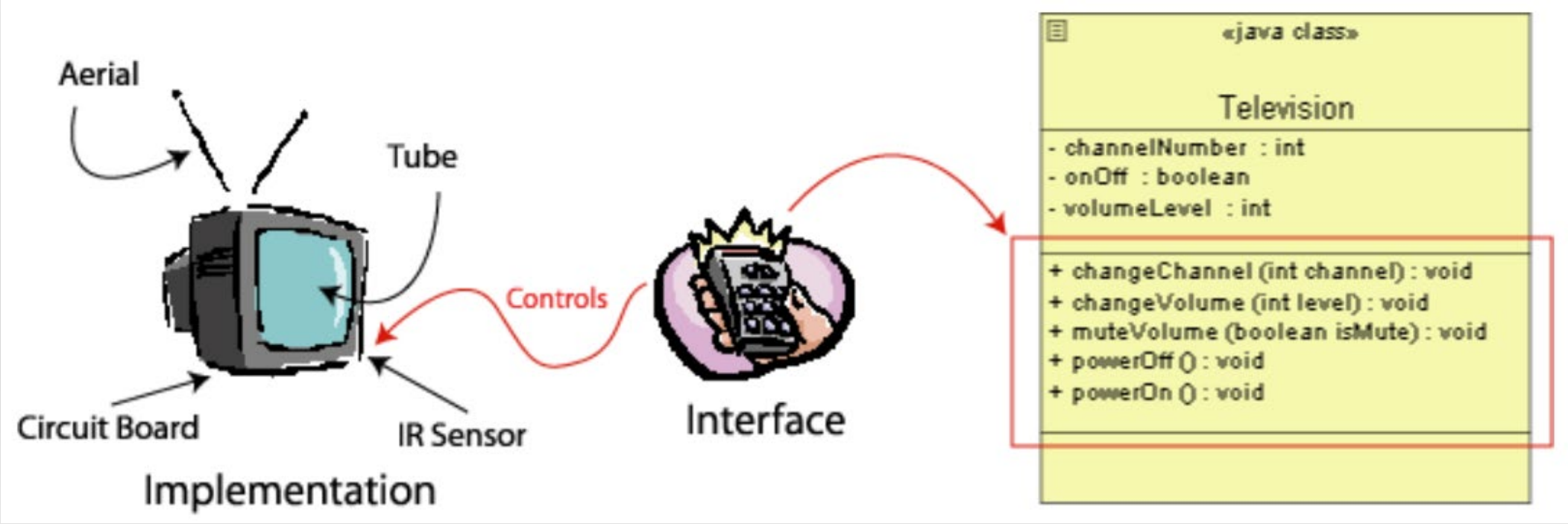
Objects

- An instance of a class
  - If a class is the description of a concept
  - An object is the realization of the description in calss
  - To create an independent distinguishable entities – TVs can be on/off independently
  - Each with its own individual identity – Each TV has different serial number



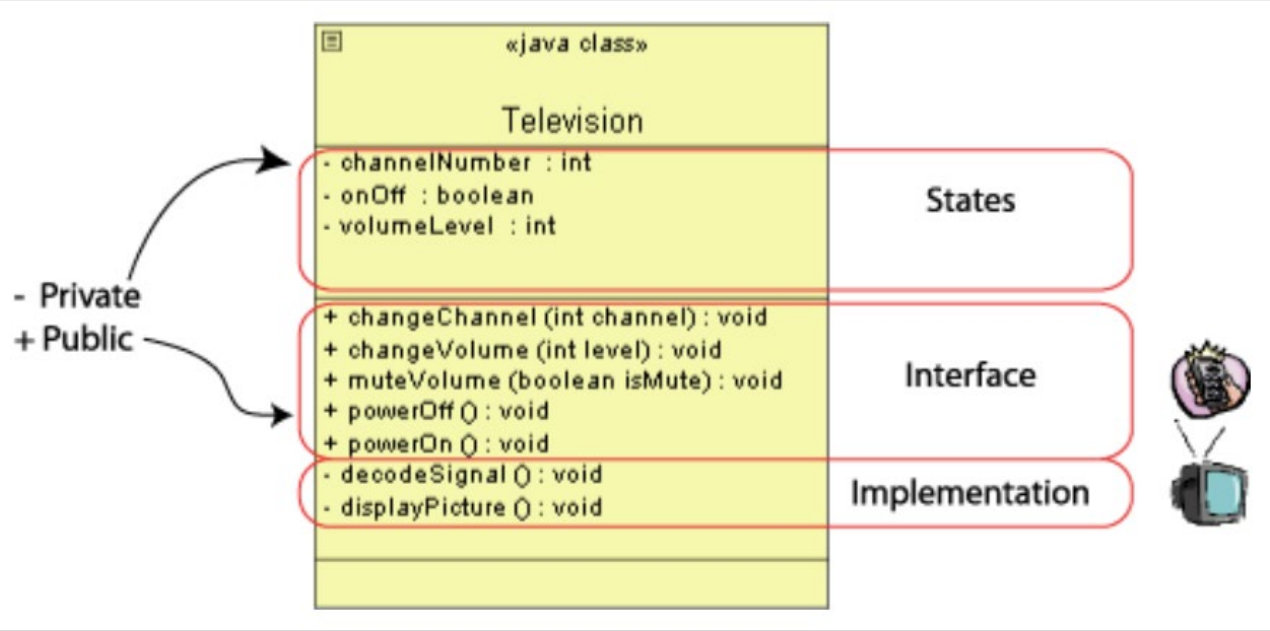
Encapsulation

- Use to hide the mechanics of the object
- Allow the actual implementation of the object to be hidden
- In plain
  - Users don't need to understand how the object works
  - All Users need to understand is the interface that is provided for us



Encapsulation (Cont'd)

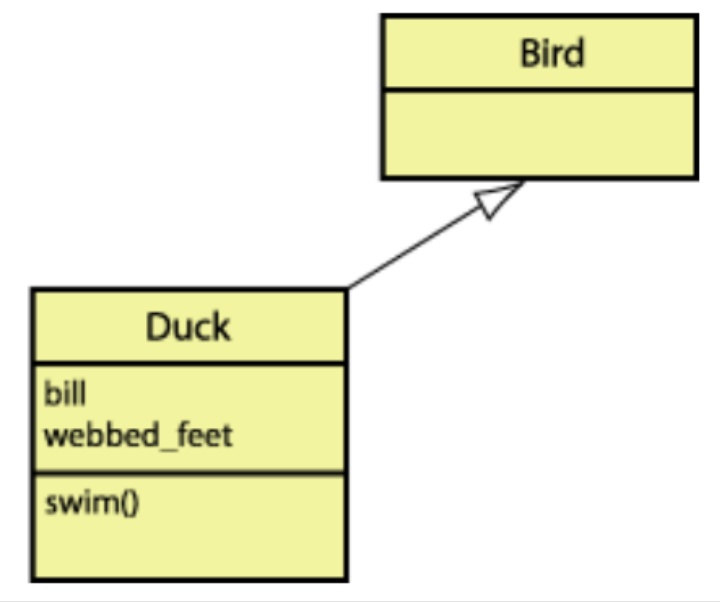
- The implementation of interface are implemented as methods
  - Public methods: Interface seen by users
  - Private methods: Implementation used by other methods in the same class



- Encapsulation can be thought as data-hiding
  - Users or other programmers only need to know the interface
  - Programmer can change the implementation, but need not notify the user

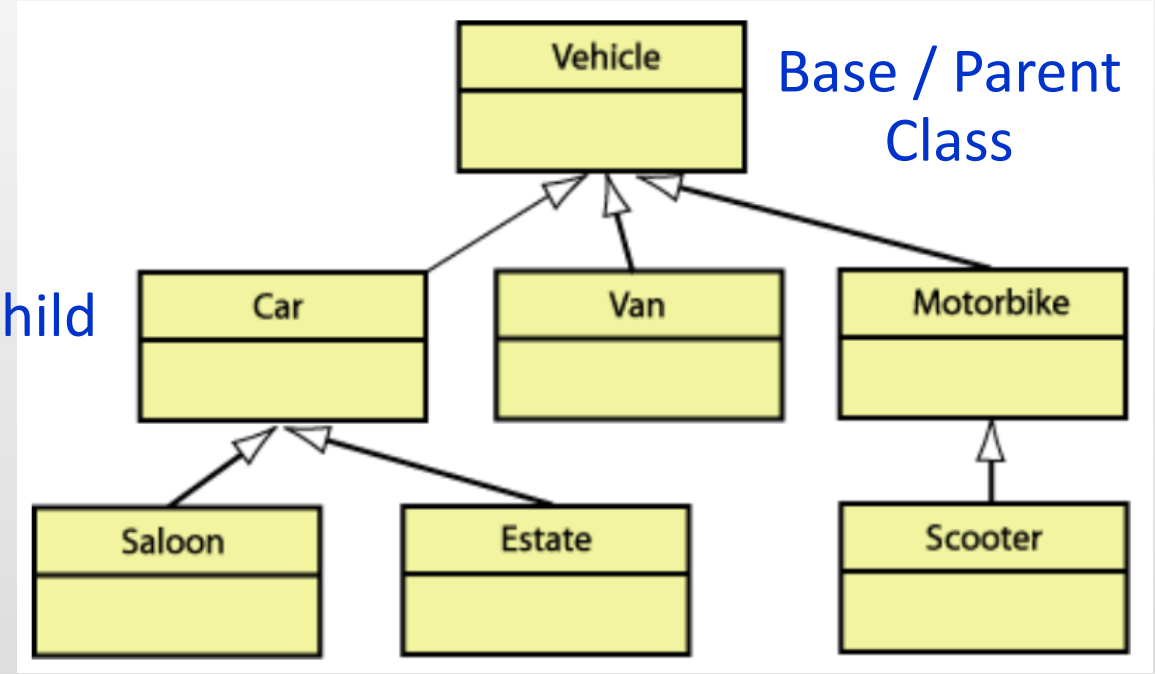
Inheritance

- Provide a compact representation of descriptions that share some commonalities
- OOP allows us to group the commonalities and create classes that can describe their differences from other classes.
- Ex. What is a duck?
  - A bird that swims, with webbed feet, and a yellow bill
  - A duck is a special type of bird



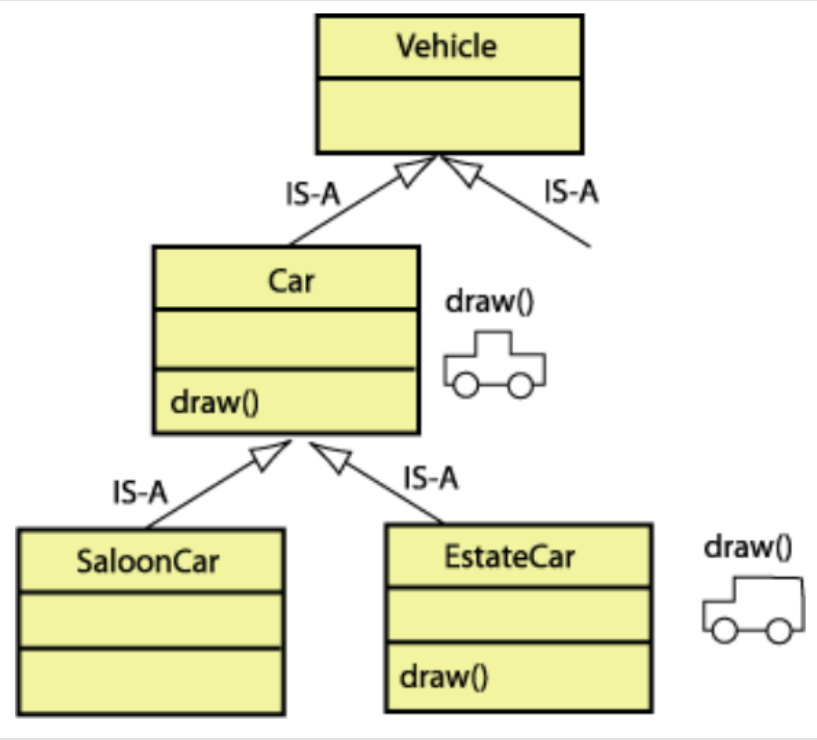
Inheritance (Cont'd)

- An example of organizing different vehicles via their differences
  - Car inherits from Vehicle
  - Saloon inherits from Car
  - …
- Child class inherits all the data and methods of the parent class
  - Car class inherits the methods of the vehicle class
    - engineStart(), gearChange(), lightsOn()
  - Car class inherits the data of the vehicle class
    - isEngineOn, isLightsOn, numberOfWheels



Polymorphism

- Polymorphism means "multiple forms“ of the same method
  - Exact same method name can behave differently in derived classes
  - Same method name can be used in the same class with slightly different parameters
- Two forms of polymorphism, **over-riding** and **over-loading**



Polymorphism (Cont'd)

- Difference between **over-riding** and **over-loading**

Overriding

```
class Dog{
    public void bark(){
        System.out.println("woof ");
    }
}
class Hound extends Dog{
    public void sniff(){
        System.out.println("sniff ");
    }

    public void bark(){
        System.out.println("bowl");
    }
}
```

Same Method Name, Same parameter

Overloading

```
class Dog{
    public void bark(){
        System.out.println("woof ");
    }

    //overloading method
    public void bark(int num){
        for(int i=0; i<num; i++)
            System.out.println("woof ");
    }
}
```

Same Method Name, Different Parameter

Q & A

Thank you for your attention.

HW 00 is released. Check Course website or I study.

Do the survey and mark your calendar to get 1 points bonus

Sing up here for the third phase of course registration