CS1101S Discussion Group Week 11: *Object-oriented Programming*

Niu Yunpeng

niuyunpeng@u.nus.edu

October 31, 2017

Overview

From Last Week

Object

- Object is a collection of key-value pairs;
- Object is a generalization of "traditional" array;
- Key is string, value can be anything (function, data structure)

Object-oriented Programming

Object accessor

- Using object is really similar to using array.

```
var obj = {"aa": 4,
           "bb": true,
           "cc": function(x) { return x * x; } };

obj["aa"];
obj["bb"];
obj["cc"](5);    // returns 25
```

Dot operator

- Dot operator is a shortcut for object accessor.

```
var obj = {"aa": 4,
           "bb": true,
           "cc": function(x) { return x * x; } };

obj.aa;
obj.bb;
obj.cc(5);  // returns 25
```

Overview

Object-oriented Programming

Terminology

- Class
- Object
- Instance
- Field
- Attribute
- Method
- Constructor
- Inheritance
- Polymorphism
- Override
- ...

Object-oriented Programming

Class, object & instance

- Class: a blueprint/template of all the things of one type.
- Object: a particular thing of one type.
- Instance: a unique copy of information for an object in memory.

Relationship

- ⟨*class_name*⟩ **includes many** ⟨*object_name*⟩s.
- ⟨*object_name*⟩ **is a** ⟨*class_name*⟩.

Object-oriented Programming

Example

- Class: `Country`
- Objects: `Singapore`, `China`, `Russia`,...

Relationship

- `Country` includes `Singapore`, `China` and `Russia`.
- `Singapore` is a `Country`.
- `China` is a `Country`.
- `Russia` is a `Country`.

Object-oriented Programming

To describe an object

- Use adjectives: how large? how long? how old? ...
  or equivalent to:
  Use nouns: size, length, age, ...
- Use verbs: can jump? can swim? can speak?

Thus...

- Use *adjectives/nouns* to describe **states**;
- Use *verbs* to describe **behaviours**.

Object-oriented Programming

Property & method

- Property: variables that describes states of an object;
- Method: functions that operate on an object.

Relationship

- ⟨*class_name*⟩ or ⟨*obj_name*⟩ **has many** properties.
- Fields/attributes/methods **describes** ⟨*class_name*⟩ or ⟨*obj_name*⟩.

Object-oriented Programming

Example

- Class: `Student`
- Properties: `name`, `age`, `major`, ...
- Methods: `study`, `play`, ...

Relationship

- `name`, `age` and `major` describes a `Student`.
- A `Student` can `study` and `play`.

Object-oriented Programming

Constructor

- Constructor: to create a new instance of a class and perform related initialization actions.
  - Usually, the constructor will set the initial values of compulsory fields.

Relationship

- We **use** the constructor to **instantiate** a copy of ⟨*class_name*⟩ to get a new ⟨*obj_name*⟩.

Common patterns between different classes

- We know there are a lot of common patterns within a class.
- However, different classes may also have common patterns.

Problem...

- How can we share common patterns between different classes?

Object-oriented Programming

Inheritance

- Inheritance: abstract the common patterns into one superclass, and keep the specification within each subclass.

Polymorphism

- Polymorphism: the same method may behave in different ways due to different and potentially heterogeneous implementations.
- Polymorphism in OOP is usually achieved via method override.

Object-oriented Programming

Three terms

- Override
- Overwrite
- Overload

Your task today

- Find out the difference between these three terms.

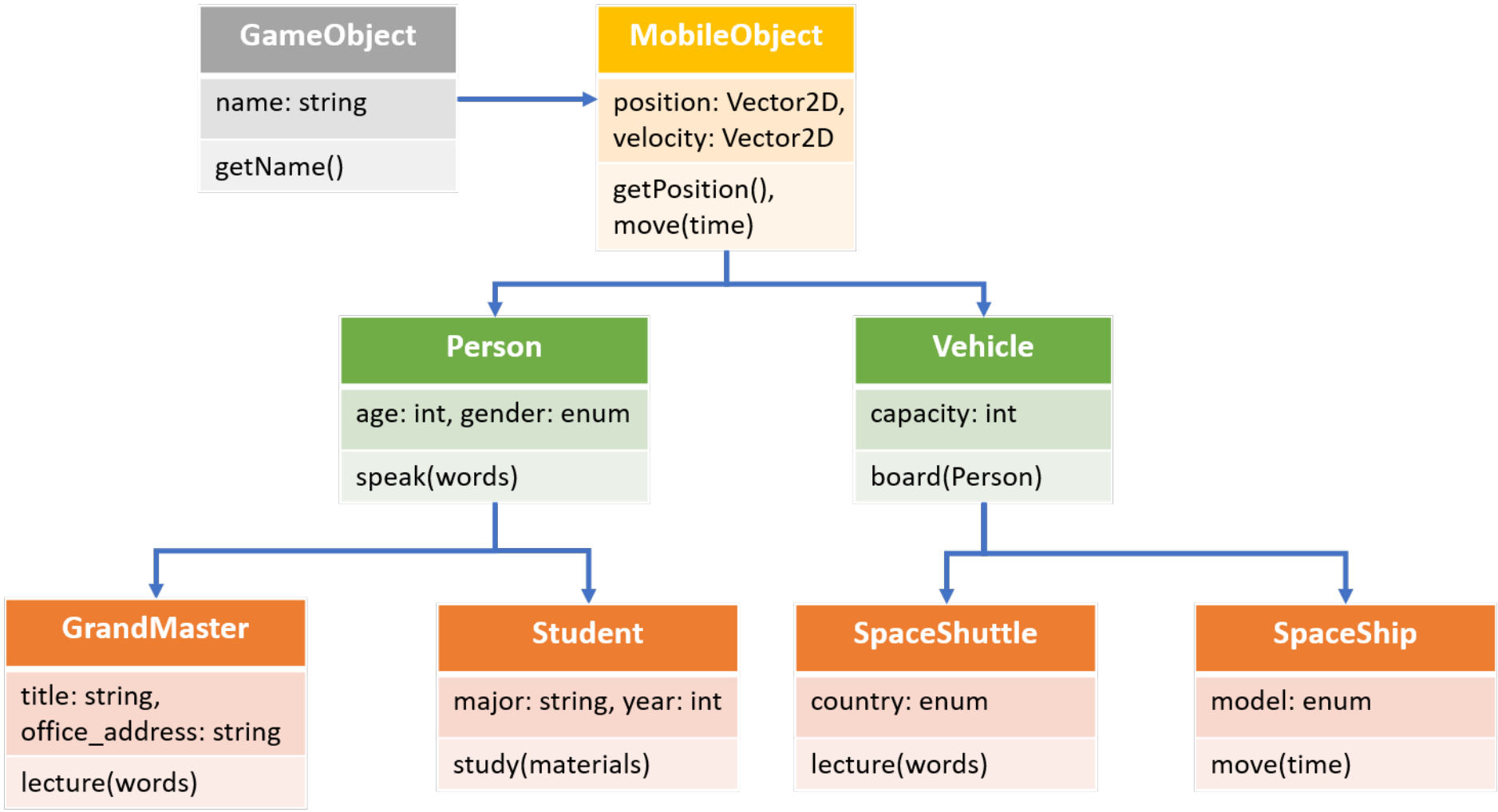
Relationship

- A ⟨*super_class_name*⟩ **has many** ⟨*sub_class_name*⟩s.
- A ⟨*sub_class_name*⟩ **belongs to** a ⟨*super_class_name*⟩.
- A ⟨*sub_class_name*⟩ **inherits from** its ⟨*super_class_name*⟩.

Diagram

- We can draw a diagram to visualize the hierarchy relationship between all the superclasses and subclasses.
- The diagram is going to be a ***tree***.

Object-oriented Programming



Overview

1. From last week
   - Object
2. Object-oriented concepts
   - Field, attribute, & method
   - Inheritance & polymorphism
3. Object-oriented programming in Source

Object-oriented Programming

A few keywords

- new
- this
- Inherits
- prototype
- ___proto___
- ...

Object-oriented Programming

Class & object

- The class name is the same as the constructor name.
- An object can be created by calling its constructor with `new`.

Naming convention

- By convention, the first letter of class name should be uppercase.

Object-oriented Programming

Example

```
function Person(name, age) {
    this.name = name;
    this.age = age;
}

var this_person = new Person("Smith", 35);
```

Think about it...

- What should we do in the constructor?

Field & method

- To declare/access a field, use `this` keyword.
- To declare a method, add it into the `prototype` object of the class.

To declare a method

- Functions are also variables, methods are also fields.
- Technically, you can declare a method in the same way as a normal field: *use this keyword and the dot operator*.
- However, you should never do so.

Object-oriented Programming

Why `prototype` object?

- If you use the `prototype` object, there will be only one copy for methods (because it does not belong to the instances).
- For normal fields, we advise to make one copy for every object.
- For methods, we advise to make only one copy for the class.

Think about it...

- Can you do OOP without `prototype` at all?

Object-oriented Programming

Example

```
function Person(name, age) {
    this.name = name;
    this.age = age;
}

Person.prototype.speak = function (words) {
    display(this.name + " says: " + words);
};

var this_person = new Person("Smith", 35);
this_person.speak("Hello, world!");
```

Inheritance

To inherit from the superclass:

- call the superclass constructor to initialize the parent object;
- use the `Inherits` keyword to inherit all methods.

Polymorphism

- to override a method, re-define it in the subclasses.

Object-oriented Programming

Example

```
function Student(name, age, year, major) {
    Person.call(this, name, age);
    this.year = year;
    this.major = major;
}

Student.Inherits(Person);

Student.prototype.speak = function (words, university) {
    display(this.name + " says: " + words);
    display("I am a student from " + university + ".");
};

var my_student = new Student("Smith", 35, 1, "CS");
```

Three ways to call a method

- simply use the function name;
- use $\langle function\_name \rangle$ *.call* so as to pass the instance fields;
- use $\langle class\_name \rangle$ *.prototype.* $\langle function\_name \rangle$ *.call* to call the method from a certain class

For the 3^{rd} way

- Especially useful due to polymorphism.
- In the prototype chain, the interpreter will find the nearest version of the method and call it.
- If you explicitly declare the class, it can find another version.

Object-oriented Programming

Example

```
Student.prototype.introduce = function() {
    Person.prototype.speak.call(this, "Hello, everyone!");
    this.speak("My name is " + this.name + ".", "NUS");
};

var my_student = new Student("Smith", 35, 1, "CS");
my_student.introduce();

// Smith says: Hello, everyone!
// Smith says: My name is Smith.
// I am a student from NUS.
```

Discussion Group Problems

Let's discuss them now.

The End