

Introduction to Object Oriented Programming

Kiwi Wang

DISP

March 21, 2014

Introduction to
Object Oriented
Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Outline

- 1 History
- 2 Basic Concepts
- 3 Code Pieces
- 4 Conclusion

Introduction to
Object Oriented
Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Outline

- 1 History
- 2 Basic Concepts
- 3 Code Pieces
- 4 Conclusion

Introduction to
Object Oriented
Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Simula 67

- Working on simulations that deal with exploding ships.
- They grouped the ships into different categories.
- Each ship has its own class.
- Each class would generate its unique behaviors and data.

Smalltalk

- First use the term "object oriented programming(OOP)"
- First language to introduce the inheritance concept

Introduction to Object Oriented Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

after 1980...

- Object Oriented Programming has become prominent.
- The most importance factor is C++.
- Also play a importance role in the development of event-driven programming(ex. GUI)
- However, nowadays, when it comes to OOP, we mean OOP concepts combined with procedures that are useful to programmers.
- Explode out concepts of design patterns, data abstraction...

So, what's OOP?

Introduction to Object Oriented Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Outline

- 1 History
- 2 Basic Concepts
- 3 Code Pieces
- 4 Conclusion

Introduction to
Object Oriented
Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

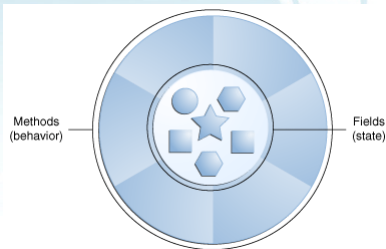
What's OOP languages

C

Conclusion

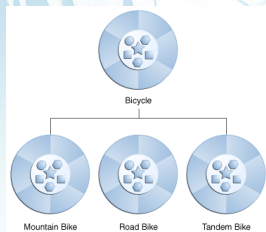
Everything is an object...

- Real-World: dogs, cats, computers...
 - They all have states and behaviors.
- Software Object: Mimic to Real-World objects
 - They also have states and behaviors.



Three Basic Properties of OOP

- Encapsulation
 - Classes -> Interfaces -> design patterns...
- Inheritance



- Polymorphism
 - Mysterious one! Go details next page.

Polymorphism

- The ability to appear in many forms.
- The ability to process objects differently depending on their data type or class.
- More specifically, it is the ability to redefine methods for derived classes.
- Polymorphism is considered to be a requirement of any true OOP.
- 多種不同的子物件繼承同一種上層物件時，我們可以用上層物件容納之，在呼叫時仍然會根據真實物件型態呼叫對應的子物件函數。

Polymorphism – examples

- Operator, such as + acts differently on integers, Strings, bits...
- Two basic ideas
 - Method Overriding: subclass, identical method signature.
 - Dynamic binding: during run-time, actual method executed is determined by type of objects, not type of reference.
- Allows superclass and subclass objects to be accessed in a regular, consistent way.(ex. draw different graphical objects in an array using the same **draw** function.)

Outline

- 1 History
- 2 Basic Concepts
- 3 Code Pieces**
- 4 Conclusion

Introduction to
Object Oriented
Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Encapsulation

```
package com.disp.oral;

public class Gorilla {

    private String name;
    private int age;
    private String gender;

    public Gorilla(String name, int age, String gender) {
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    protected void introduce(){
        System.out.printf("I'm a %s named \"%s\" with age %d, "
            + "gender %s.\n",this.getClass().getSimpleName(),
            this.getname(), this.getage(), this.getgender());
    }

    protected void walk() {
        System.out.printf("%s is walking with 4 feet.\n", this.getname());
    }

    protected void shape() {
        System.out.printf("I doesn't look like a pig!\n");
    }
}
```

Introduction to
Object Oriented
Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Inheritance

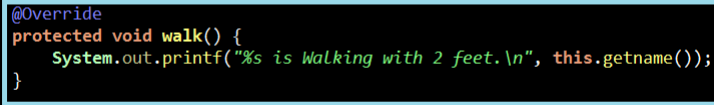
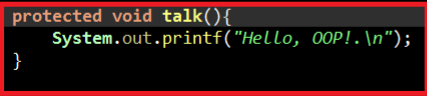

```
package com.disp.oral;

public class Human extends Gorilla {

    public Human(String name, int age, String gender) {
        super(name, age, gender);
        // TODO Auto-generated constructor stub
    }

    @Override
    protected void walk() {
        System.out.printf("%s is Walking with 2 feet.\n", this.getname());
    }

    protected void talk(){
        System.out.printf("Hello, OOP!\n");
    }
}
```



Polymorphism

```

package com.disp.oral;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        System.out.printf("*****\n");
        System.out.printf("Show class\n");
        System.out.printf("*****\n");
        Gorilla happy = new Gorilla("happy", 10, "Male");
        happy.introduce();
        happy.walk();
        happy.shape();

        System.out.printf("*****\n");
        System.out.printf("Show extended class\n");
        System.out.printf("*****\n");
        Human misfortune = new Human("misfortune", 99, "Female");
        misfortune.introduce();
        misfortune.walk();
        misfortune.shape();
        misfortune.talk();

        System.out.printf("*****\n");
        System.out.printf("Show Polymorphism\n");
        System.out.printf("*****\n");
        Gorilla children_of_happy[] = {new Gorilla("unhappy", 2, "Female"),
            new Human("just look like", 1, "Male")};

        children_of_happy[0].introduce();
        children_of_happy[1].introduce();
        children_of_happy[0].walk();
        children_of_happy[1].walk();
        children_of_happy[0].shape();
        children_of_happy[1].shape();
        //children_of_happy[1].talk(); wrong, because interface Gorilla doesn't have talk function
    }
}

```

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Result

```
*****
Show class
*****
I'm a Gorilla named "happy" with age 10, gender Male.
happy is walking with 4 feet.
I doesn't look like a pig!
*****
Show extended class
*****
I'm a Human named "misfortune" with age 99, gender Female.
misfortune is Walking with 2 feet.
I doesn't look like a pig!
Hello, OOP!.
*****
Show Polymorphism
*****
I'm a Gorilla named "unhappy" with age 2, gender Female.
I'm a Human named "just look like" with age 1, gender Male.
unhappy is walking with 4 feet.
just look like is Walking with 2 feet.
I doesn't look like a pig!
I doesn't look like a pig!
```

Matlab supports OOP, too.

```

classdef Human < Gorilla
%HUMAN Summary of this class goes here
% Detailed explanation goes here

methods

function obj = Human(name, age, gender)
    obj@Gorilla(name, age, gender);
end

function walk(obj)
    fprintf('%s is walking with 2 feet.\n', obj.name);
end

methods (Static)
function talk()
    fprintf('Hello OOP in Matlab!\n');
end
end

classdef Gorilla
%GORILLA Summary of this class goes here
% Detailed explanation goes here

properties (SetAccess = private)
    name;
    age;
    gender;
end

methods

function obj = Gorilla(name, age, gender)
    obj.name = name;
    obj.age = age;
    obj.gender = gender;
end

function introduce(obj)
    fprintf('I'm a %s named "%s" with age %d, gender %s\n', ...
        class(obj), obj.name, obj.age, obj.gender);
end

function walk(obj)
    fprintf('%s is walking with 4 feet.\n', obj.name);
end

methods (Static)
function shape
    fprintf('I doesn''t look like a pig!\n');
end
end
end

```

Matlab – main.m

```

fprintf('*****\n');
fprintf('Show class\n');
fprintf('*****\n');
happy = Gorilla('happy', 10, 'Male');
happy.introduce();
happy.walk();
happy.shape();

fprintf('*****\n');
fprintf('Show extended class\n');
fprintf('*****\n');
misfortune = Human('misfortune', 99, 'Female');
misfortune.introduce();
misfortune.walk();
misfortune.shape();
misfortune.talk();

fprintf('*****\n');
fprintf('Show Polymorphism\n');
fprintf('*****\n');
obj_array = {Gorilla('unhappy', 2, 'Female'), Human('just look like', 1, 'Male')};
obj_array{1}.introduce();
obj_array{2}.introduce();
obj_array{1}.walk();
obj_array{2}.walk();
obj_array{1}.shape();
obj_array{2}.shape();

```

Matlab – result

Command Window

```

*****
Show class
*****
I'm a Gorilla named "happy" with age 10, gender Male
happy is walking with 4 feet.
I doesn't look like a pig!
*****
Show extended class
*****
I'm a Human named "misfortune" with age 99, gender Female
misfortune is walking with 2 feet.
I doesn't look like a pig!
Hello OOP in Matlab!
*****
Show Polymorphism
*****
I'm a Gorilla named "unhappy" with age 2, gender Female
I'm a Human named "just look like" with age 1, gender Male
unhappy is walking with 4 feet.
just look like is walking with 2 feet.
I doesn't look like a pig!
I doesn't look like a pig!

```

fx >>

However...

- By the term "OOP-support language", we mean that it's easier and straightforward to develop programs under OOP concepts on that language.
- It's possible to develop an OOP-based program on any programming language.
- Actually, someone says that it's really how OOP is practiced in early time...

Use C to demo three basic features of OOP

```

main.c  Human.c  Gorilla.c  Makefile
18 int main(){
19
20     printstars("Show class");
21     Gorilla happy = {
22         .new=Gorilla_new
23     }; happy.new(&happy, "happy", 10, "Male");
24
25     happy.introduce(&happy);
26     happy.walk(&happy);
27     happy.shape(&happy);
28
29
30     printstars("Show extended class");
31     Human misfortune = {
32         .new = Human_new
33     }; misfortune.new(&misfortune, "misfortune", 99, "Female");
34
35     misfortune.introduce((Gorilla *)&misfortune);
36     misfortune.walk(&misfortune);
37     misfortune.shape((Gorilla *)&misfortune);
38     misfortune.talk(&misfortune);
39
40     printstars("Show Polymorphism");
41     Gorilla c0 = {
42         .new=Gorilla_new
43     }; c0.new(&c0, "unhappy", 2, "Female");
44     Human c1 = {
45         .new=Human_new
46     }; c1.new(&c1, "just look like", 1, "Male");
47
48     void * children_of_happy[] = {&c0, &c1};
49
50     ((Gorilla *)children_of_happy[0])->introduce((Gorilla *)children_of_happy[0]);
51     ((Human *)children_of_happy[1])->introduce((Gorilla *)children_of_happy[1]);
52     ((Gorilla *)children_of_happy[0])->walk((Gorilla *)children_of_happy[0]);
53     ((Human *)children_of_happy[1])->walk((Human *)children_of_happy[1]);
54     ((Gorilla *)children_of_happy[0])->shape((Gorilla *)children_of_happy[0]);
55     ((Human *)children_of_happy[1])->shape((Gorilla *)children_of_happy[1]);

```

Use C to demo three basic features of OOP – result

```
Console Problems Tasks Properties
<terminated> main.exe [C/C++ Application] D:\_CloudSpace\Dropbox\Eclipse_WorkSpace\workspace_cp
*****
Show class
*****
I'm a Gorilla named "happy" with age 10, gender Male.
happy is walking with 4 feet.
I doesn't look like a pig!
*****
Show extended class
*****
I'm a Human named "misfortune" with age 99, gender Female.
misfortune is walking with 2 feet.
I doesn't look like a pig!
Hello, OOP in C!.
*****
Show Polymorphism
*****
I'm a Gorilla named "unhappy" with age 2, gender Female.
I'm a Human named "just look like" with age 1, gender Male.
unhappy is walking with 4 feet.
just look like is walking with 2 feet.
I doesn't look like a pig!
I doesn't look like a pig!
```

Encapsulation

```
main.c Human.c Gorilla.c Makefile
gorilla.c
2
3
4
5
6
7
8 #ifndef G_H
9 #define G_H
10
11 #include<stdio.h>
12
13 typedef struct Gorilla {
14
15     void (*new) (struct Gorilla * obj, char * name, int age, char * gender);
16
17     char * class_name;
18
19     char * name;
20     int age;
21     char * gender;
22
23     void (*introduce)(struct Gorilla * obj);
24     void (*walk)(struct Gorilla * obj);
25     void (*shape)(struct Gorilla * obj);
26
27 } Gorilla;
28
29
30 void introduce(Gorilla * obj){
31     printf("I'm a %s named \"%s\" with age %d, "
32           "gender %s.\n", obj->class_name,
33           obj->name, obj->age, obj->gender);
34 }
35
36 void walk(Gorilla * obj){
37     printf("%s is walking with 4 feet.\n", obj->name);
38 }
39
40 void shape(Gorilla * obj){
41     printf("I doesn't look like a pig!\n");
42 }
```


Encapsulation – constructor

```
void Gorilla_new (Gorilla * obj, char * name, int age, char * gender){  
    obj->class_name = "Gorilla";  
    obj->name = name;  
    obj->age = age; //Really painful to write a constructor  
    obj->gender = gender;  
  
    obj->introduce = introduce;  
    obj->walk = walk;  
    obj->shape = shape;  
}
```

Introduction to Object Oriented Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Inheritance

```
main.c  Human.c  Gorilla.c  Makefile
9 #define H_H
10
11 #include "Gorilla.c"
12 #include <stdio.h>
13
14 typedef struct Human {
15     Gorilla super;
16
17     char * class_name;
18
19     void (*new) (struct Human * obj, char * name, int age, char * gender);
20
21     //pointer to super class fields
22
23     char * name;
24     int * age;
25     char * gender;
26
27
28     void (*introduce)(Gorilla * obj);
29     void (*shape)(Gorilla * obj);
30
31     //Override
32     void (*walk)(struct Human * obj);
33
34     //func of subclass
35     void (*talk)(struct Human * obj);
36
37
38 } Human;
39
40
41 void human_walk(Human * obj){
42     printf("%s is walking with 2 feet.\n", obj->name);
43 }
44
45 void talk(Human * obj){
46     printf("Hello, OOP in C!\n");
```

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Inheritance – constructor

```
void Human_new(Human * obj, char * name, int age, char * gender){  
    // corresponding java super()  
    obj->super.new = Gorilla_new;  
    Gorilla * super = &(obj->super);  
    super->new(super, name, age, gender);  
    obj->class_name = super->class_name = "Human";  
  
    // pointers to super class  
    obj->name = super->name; //Super painful  
    obj->age = &(super->age);  
    obj->gender = super->gender;  
    obj->introduce = super->introduce;  
    obj->shape = super->shape;  
  
    //override  
    obj->walk = human_walk;  
  
    //func of subclass  
    obj->talk = talk;  
}
```

Introduction to
Object Oriented
Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Polymorphism – Use generic pointers but hard to remember all pointer types that each function needs...

```

printstars("Show Polymorphism");
Gorilla c0 = {
    .new=Gorilla_new
}; c0.new(&c0, "unhappy", 2, "Female");
Human c1 = {
    .new=Human_new
}; c1.new(&c1, "just look like", 1, "Male");

void * children_of_happy[] = {&c0, &c1}; //What the hell here

((Gorilla *)children_of_happy[0])->introduce((Gorilla *)children_of_happy[0]);
((Human *)children_of_happy[1])->introduce((Gorilla *)children_of_happy[1]);
((Gorilla *)children_of_happy[0])->walk((Gorilla *)children_of_happy[0]);
((Human *)children_of_happy[1])->walk((Human *)children_of_happy[1]);
((Gorilla *)children_of_happy[0])->shape((Gorilla *)children_of_happy[0]);
((Human *)children_of_happy[1])->shape((Gorilla *)children_of_happy[1]);

```

Outline

- 1 History
- 2 Basic Concepts
- 3 Code Pieces
- 4 Conclusion

Introduction to
Object Oriented
Programming

Kiwi Wang

History

Early Development

From then to now

Basic Concepts

Object

Properties of OOP

Code Pieces

Java

Matlab

What's OOP languages

C

Conclusion

Usefulness and Debates

- GUI, Android SDK, LLVM, gcc...
- Important benefits come with OOP:
 - maintainments, readability, productivity, and the reuse of codes.
- However, there is no free lunch:
 - There are debates about OOP, such as type safety, coding structures...
- As a solid power to advance developments of software, it's still worth to know about OOP.