C++ Development

Lecture 2

Classes, objects, properties

Schedule

- Object oriented programming
- Key concepts of OOP
- Classes and objects
- Constructors
- Properties

Object oriented programming

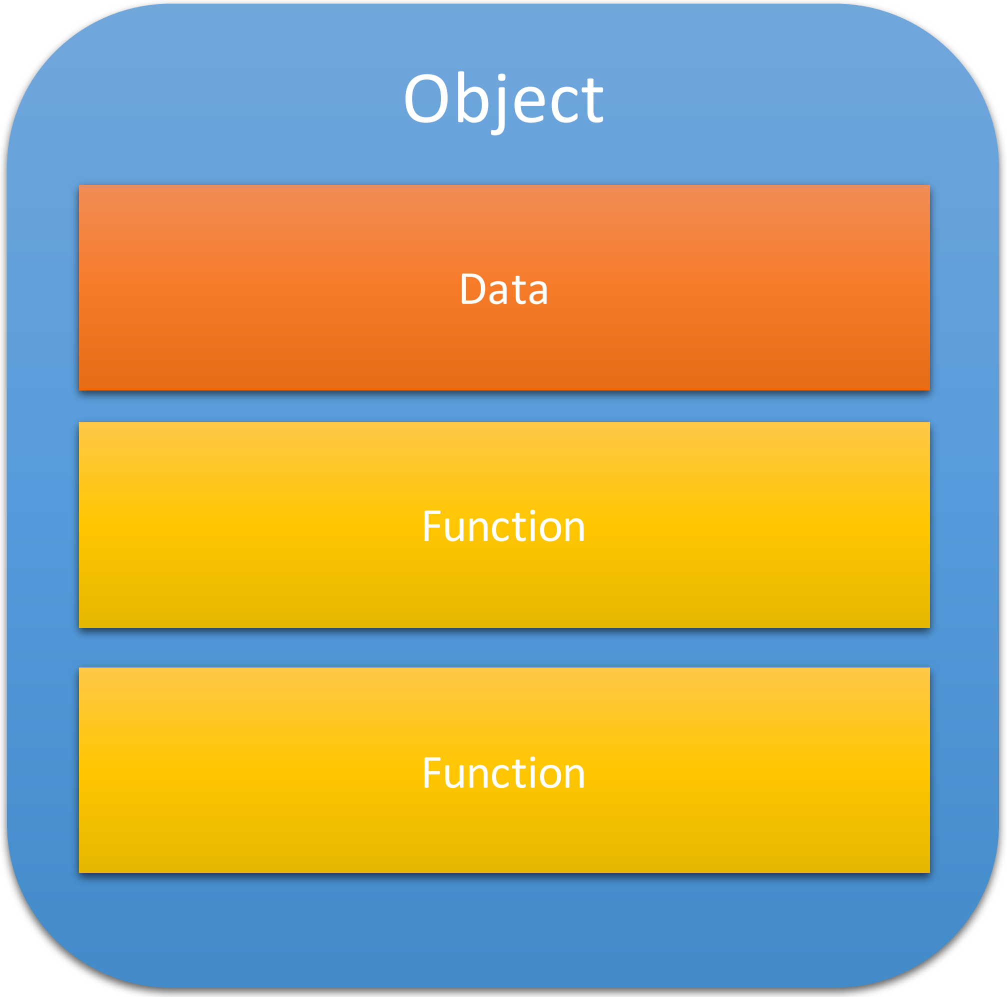
OOP

- Object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

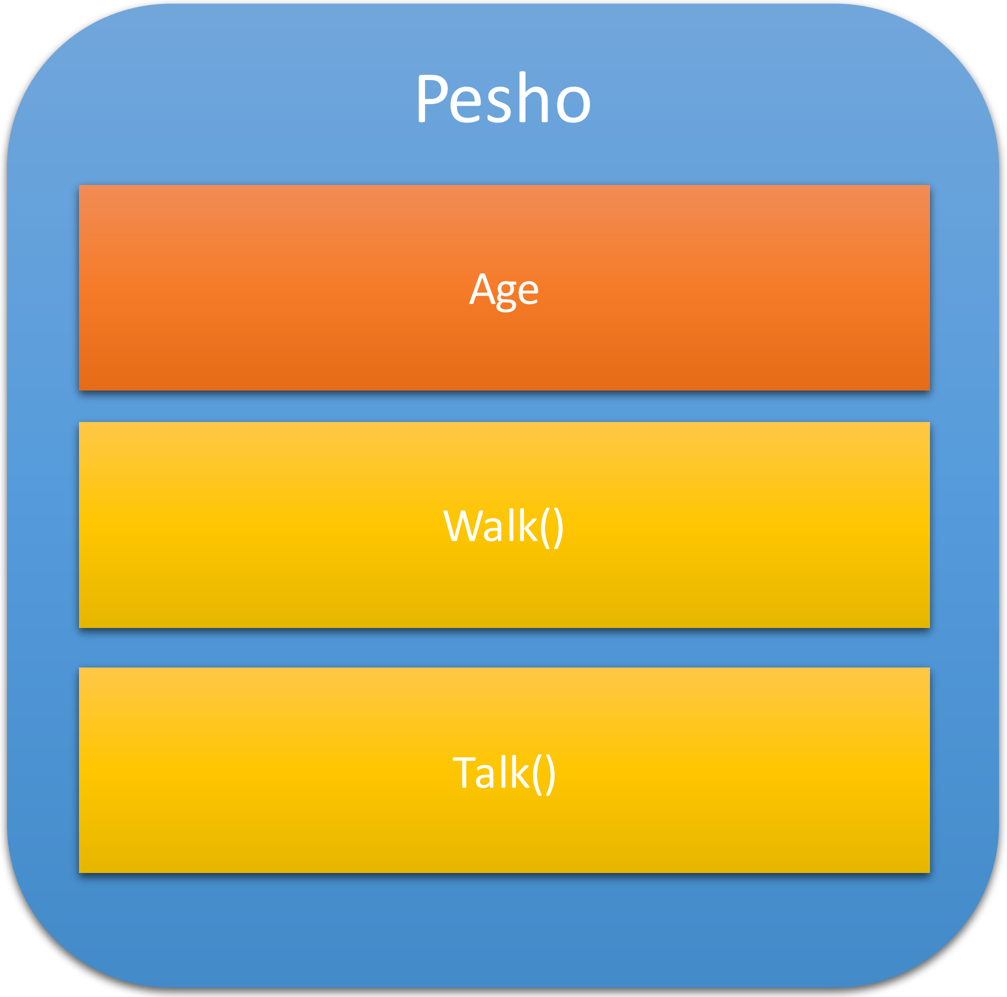
OOP Requirements

- It supports objects that are data abstractions with an interface of named operations and a hidden local state.
- Objects have an associated type [class].
- Types [classes] may inherit attributes from supertypes [superclasses].

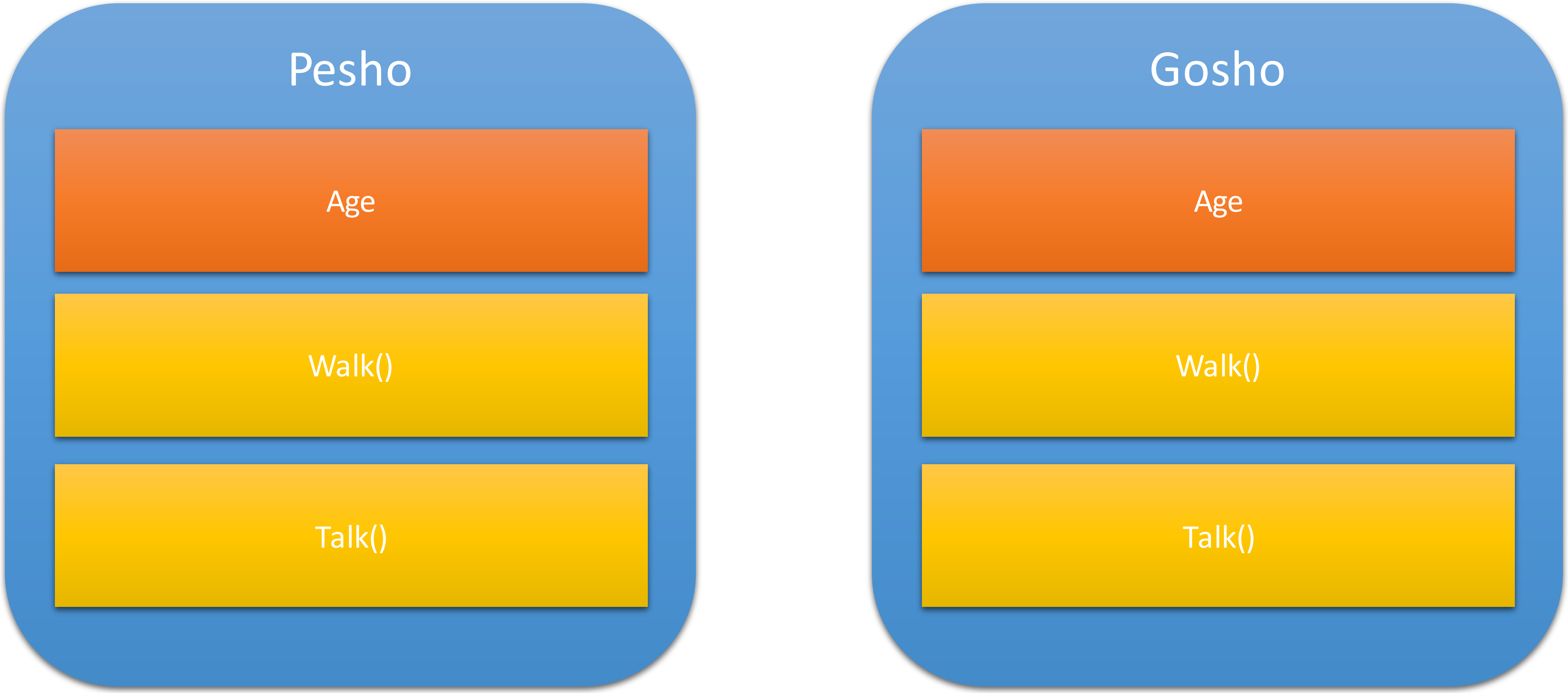
Object concept



Better example



Few objects



Differences and similarities

Similarities

- Structure
- Functions
- Data types

Differences

- Names
- Value of data

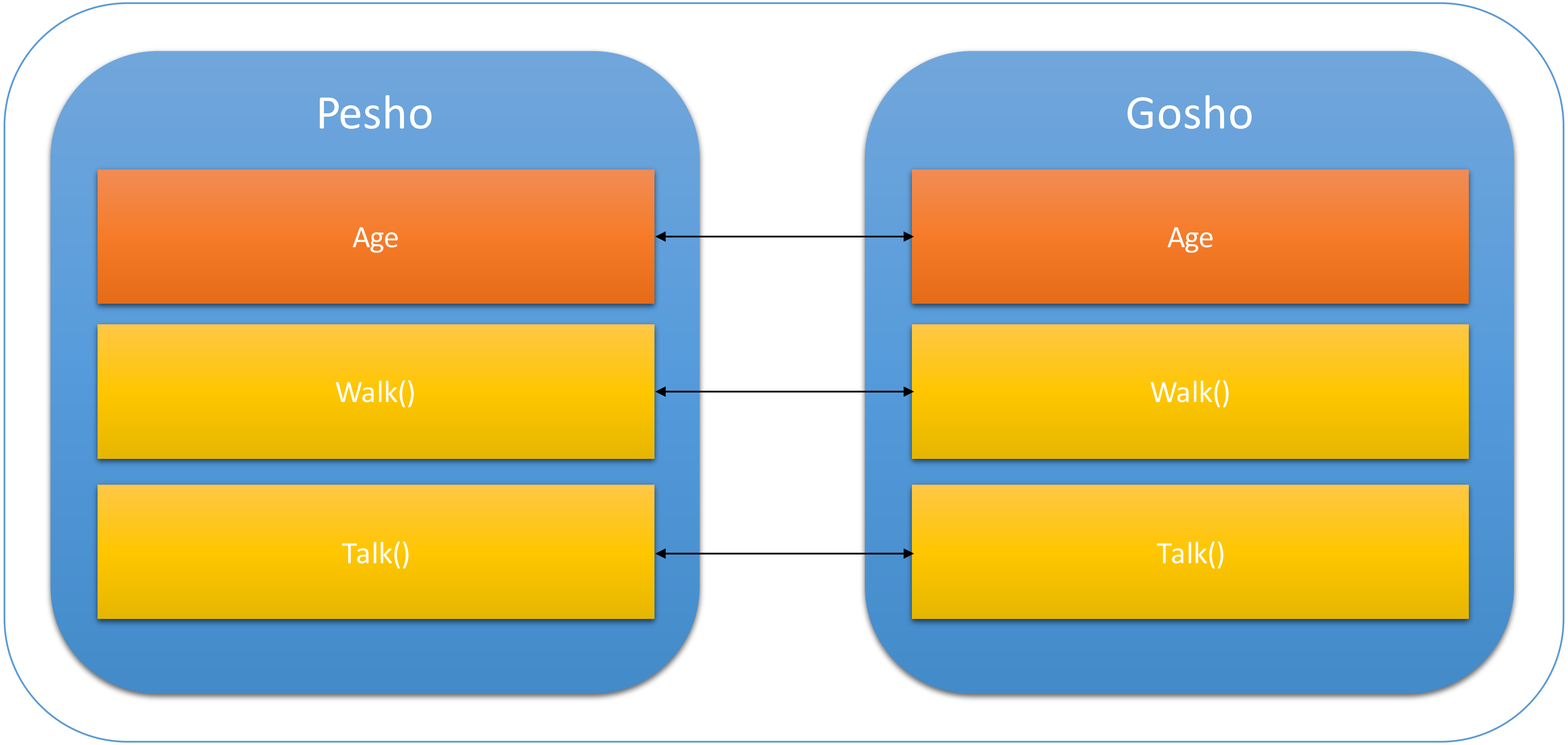
To conclude

- Both objects has same structure
- Both objects has different names

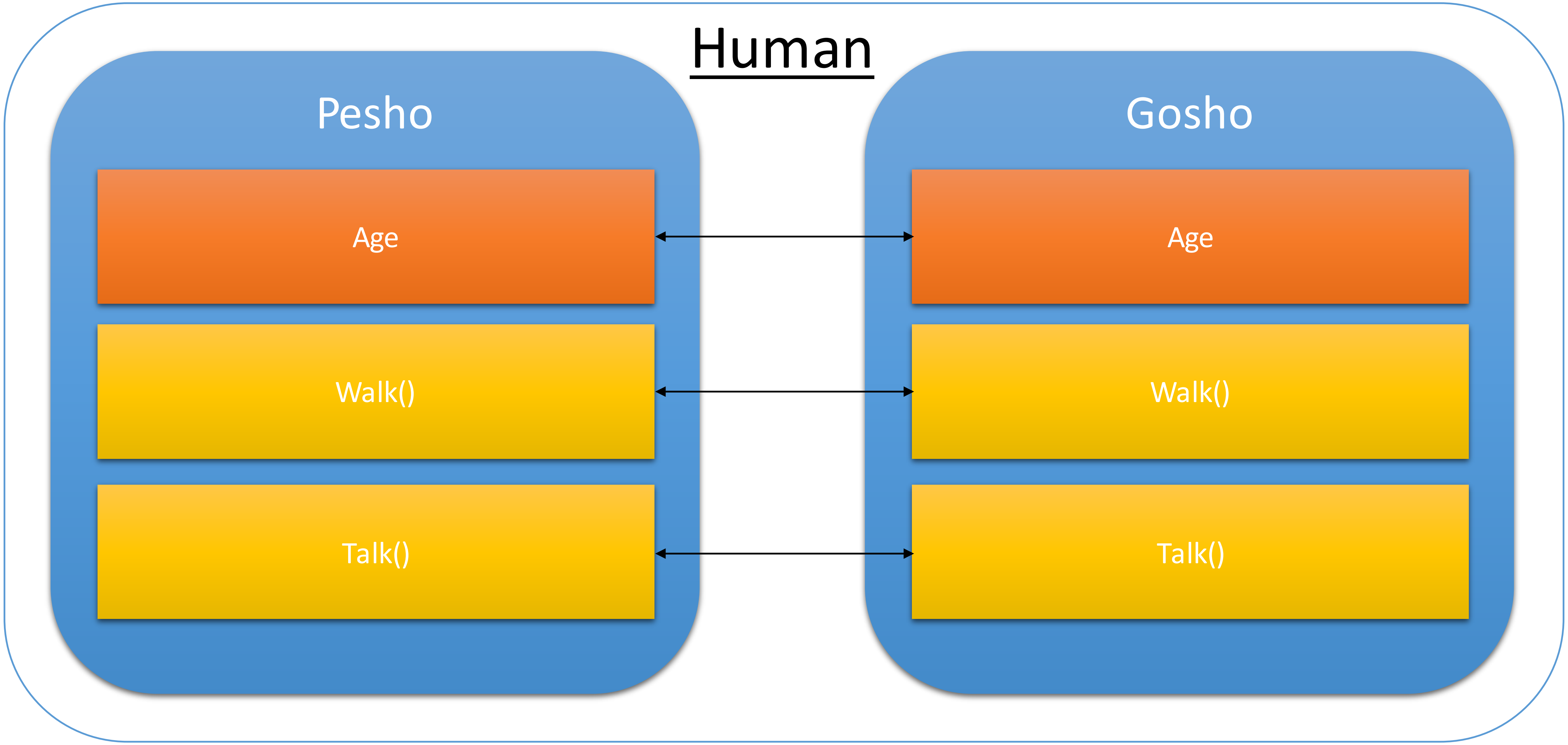
And now think abstractly

- Don’t look at the details, step back and look at the whole image

Abstraction



A Class



Class

- A class represents a template for several objects that have common properties.
- A class defines all the properties common to the object - attributes and methods.
- A class is sometimes called the object's type.

Some more classes

- Class Person:
    - Attributes – Name, Age, Hair color, Eye color
    - Operations – Speak(), Listen(), Walk()

- Class Vehicle:
    - Attributes – Name, Model, Color
    - Operation – StartEngine(), StopEngine(), Go(), Stop()

One example from you

- Class smartphone:
    - Attributes:
    - Operations:

Lets go back to objects

- Object is an instance of a class.
- An Object Oriented system is a collection of interacting Objects.

Types of objects

- User Interface objects
    - Objects that the user interacts directly with
- Operating environment objects
    - Provide services to other components
- Task Related objects
    - Documents, multimedia, problem domain

Questions ?

Our first OOP program

- Make a class car that describes:
  - Make
  - Mode
  - Horse power
  - 0 – 60 mph time
  - Fuel consumption
  - isRunning

- Methods:
  - Run()
  - Stop();

The objects

- First object :
  - Ford Mustang RTR 2016
- Second object
  - Ford Focus 2000

OOP terminology

- Objects contains data and instructions
- Class describe what is in common between similar objects
- Attribute contains data that is part of the object
- Data type describes the data type of an attribute
- Behavior describes what objects can do
- Method is a set of instructions
- Encapsulation expose only the data that is needed

Encapsulation

- In order to be “safe” an object has different access rights when it’s data is manipulated.
- Object support Information Hiding – Some attributes and methods can be hidden from the user.
- Live example: People has secrets.

How does the encapsulation work

- Public - the member/function can be accessed with no restriction
- Private – the member/function can be accessed only from the class
- Protected – the member/function can be accessed from the class and from the inheriting classes.

Example

- Class bank account with functions
  - Withdraw(),deposit()

  And members
  - Current balance

Abstraction

What is abstraction

- The technique of creating new data types that are well suited to an application.
- It allows the creation of user defined data types, having the properties of built in data types and more.

Task

- Define a class describing a geographic position(GPS POI) with name and position

Variant 1

- class GeographicPosition
- {
    - public:
    - std::string name;
    - double latitude;
    - double longitude;
- }

Variant 2

- class Coordinates
- {
  - public:
  - double latitude;
  - double longitude;
- }
- Class GeographicPosition
- {
  - Public:
  - std::string name;
  - Coordinates coordinates;
- }

Why is Variant 2 better

- Because we have better level of abstraction. If we have another place where we have to use latitude/longitude we can just use Coordinates class.

Task

- Write a class describing a rectangular in two dimentional system. It must have start position and size.

Properties

What is a class property

- The class properties are members of the class that describe it's data and have getter/setter

Property getter

- The property getter is a function which returns the value of a class member.

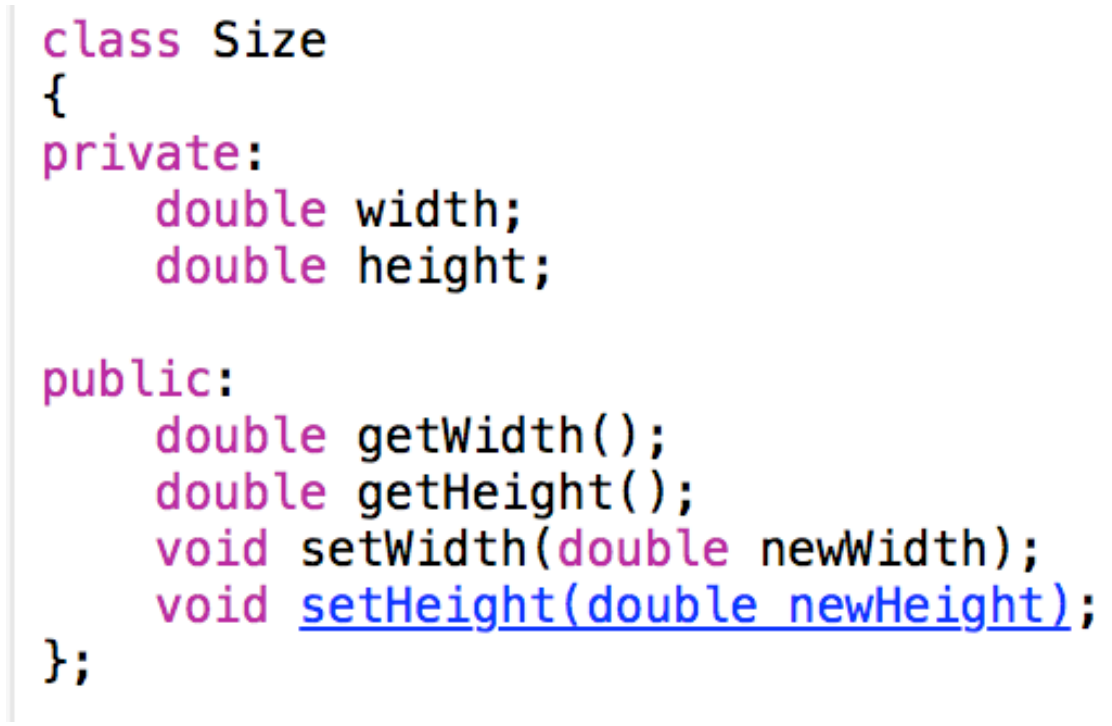
Property setter

- Typically void function that sets value to a member of the class.

A class WITHOUT properties

```
class Size
{
public:
    double width;
    double length;
};
```

A class with properties



```
class Size
{
private:
    double width;
    double height;

public:
    double getWidth();
    double getHeight();
    void setWidth(double newWidth);
    void setHeight(double newHeight);
};
```

Property getter and setter

```
double Size::getWidth()
{
    return width;
}

void Size::setWidth(double newWidth)
{
    width = newWidth;
}
```

Why we should write thaaaat much code

Because :

- 1.Better encapsulation
- 2. Better control;
- 3. You can execute other action in the getter and the setter

Constructors

What is a constructor

- A special public function used to initialize parameters of the class.
- Has the same name as the class
- Should be called on object initialization
- No return value

Syntax

```
class Size
{
private:
    double width;
    double height;

public:
    Size(double newWidth, double newHeight)
    {
        width = newWidth;
        height = newHeight;
    }
    double getWidth();
    double getHeight();
    void setWidth(double newWidth);
    void setHeight(double newHeight);
};
```

Destructor

- Same name as class
  - Preceded with tilde (~)
- No arguments
- Performs "termination housekeeping"
- No return

More about destructor

- In a class, we can have no more than 1 destructor.
- He seams like a function with ~
- He take no arguments and return noting
- He is automatically called for any stack or global object, when that object goes out of scope.

Aaaand more about destructor

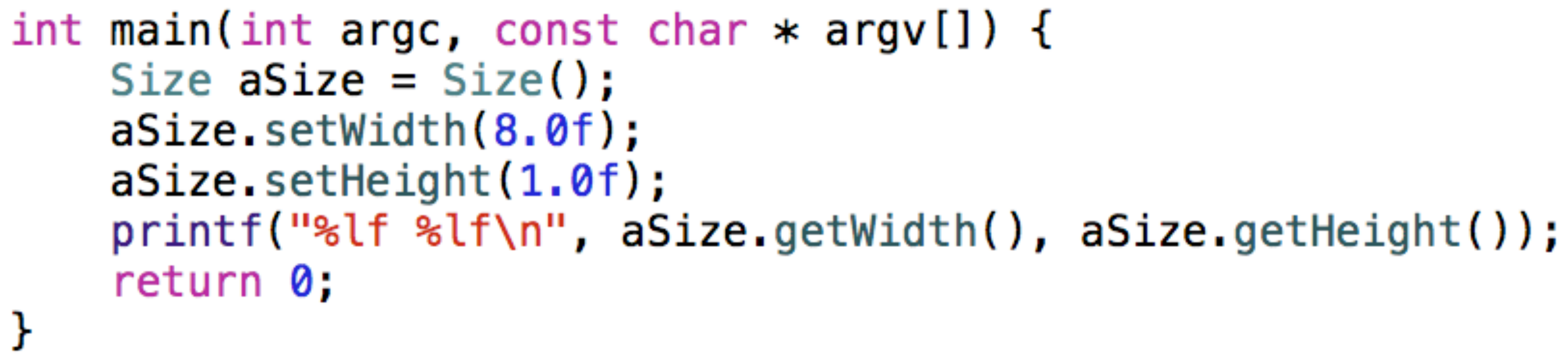
- If you don't write a destructor, the compiler generates a default for you.
- For data members, that are C++ objects, the default destructor calls those object's destructors.
- When destructing, the compiler releases the storage, occupied by that object.

Destructor implementation

```
class Size
{
private:
    double width;
    double height;

public:
    Size();
    ~Size();
    double getWidth();
    double getHeight();
    void setWidth(double newWidth);
    void setHeight(double newHeight);
};
```

Object flow implementation



```
int main(int argc, const char * argv[]) {
    Size aSize = Size();
    aSize.setWidth(8.0f);
    aSize.setHeight(1.0f);
    printf("%lf %lf\n", aSize.getWidth(), aSize.getHeight());
    return 0;
}
```