Program Design
-- Classes

Junjie Cao @ DLUT

Summer 2022

https://github.com/jjcao-school/c

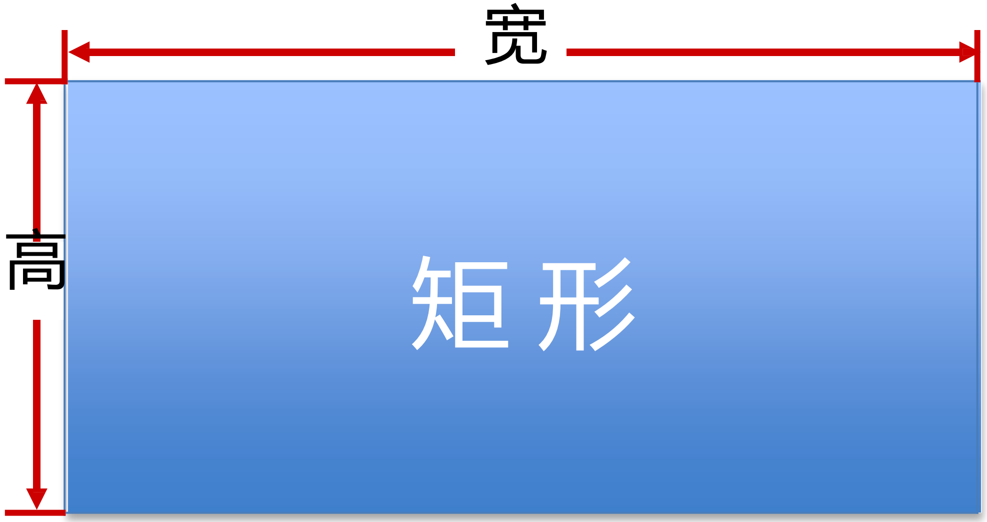
Classes and class members
1st Example

从客观事物抽象出类的例子

From 程序设计实习 by 郭炜 + 刘家瑛 @北京大学

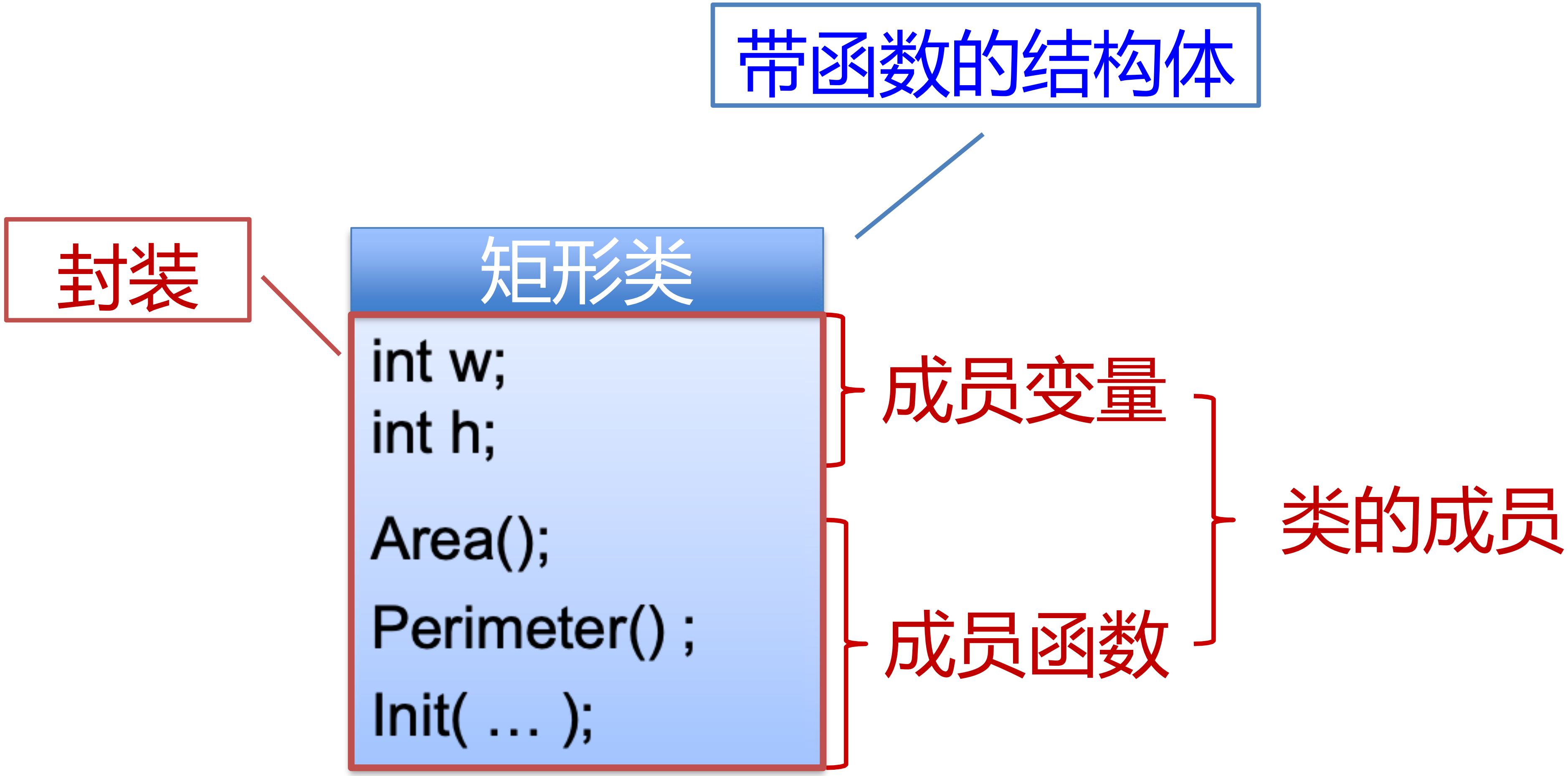
例: 客观事物→类

- 写一个程序, 输入矩形的宽和高, 输出面积和周长



- 矩形的属性 – 宽和高
  - 两个变量, 分别代表宽和高
- 对矩形的操作
  - 设置宽和高
  - 计算面积
  - 计算周长

例: 客观事物→类



带函数的结构体

封装

矩形类

```
int w;
int h;
Area();
Perimeter() ;
Init( … );
```

成员变量

成员函数

类的成员

```
class CRectangle {
   public:
              int w, h;
    void Init( int w_, int h_ ) {
        w = w_;  h = h_;
    }
    int Area() {
        return w * h;
    }
    int Perimeter() {
        return 2 * ( w + h );
    }
}; //必须有分号
```

```
int main() {
    int w, h;
    CRectangle r;  //r是一个对象
    cin >> w >> h;
    r.Init(w, h);
    cout << r.Area() << endl << r. Perimeter();
    return 0;
}
```

类定义的变量 → 类的实例 instance → “对象” object

```
class Employee{
public:
    std::string m_name;
    int m_id;
    double m_wage;


    void print(){
     std::cout << "Name: " << m_name <<
          "  Id: " << m_id <<
          "  Wage: $" << m_wage << '\n';
    }
};
```

```
int main()
{
    // Declare two employees
    Employee alex { "Alex", 1, 25.00 };
    Employee joe { "Joe", 2, 22.25 };

    // What will be printed?
    alex.print();
    joe.print();

    return 0;
}
```

对象的内存分配

- 对象的内存空间
  - 对象的大小 **=** 所有成员变量的大小之和
  - *E.g.* CRectangle类的对象, sizeof(CRectangle) = 8
- 每个对象各有自己的存储空间
  - 一个对象的某个成员变量被改变, 不会影响到其他的对象

对象间的运算

- 对象之间可以用 ‘=’ 进行赋值
- 不能用 ‘==’, ‘!=’, ‘>’, ‘<’, ‘>=’, ‘<=’进行比较
  - 除非这些运算符经过了 “重载”

访问类的成员变量和成员函数

- 用法1: 对象名.成员名 the member selector operator (.)

```
CRectangle r1, r2;  r1.w = 5;
  r2.Init(3,4);
```

访问类的成员变量和成员函数

- **用法2: 指针->成员名**

```
CRectangle  r1, r2;
CRectangle * p1 = & r1;
CRectangle * p2 = & r2;
p1->w = 5;
p2->Init(3,4);   //Init作用在p2指向的对象上
```

访问类的成员变量和成员函数

- 用法3: 引用名.成员名

```
CRectangle r2;
CRectangle & rr = r2;
rr.w = 5;
rr.Init(3,4);     //rr的值变了，r2的值也变
```

- 另一种输出结果的方式

```
void PrintRectangle(CRectangle & r) {
        cout << r.Area() << ","<<   r.Perimeter();
 }
 CRectangle r3;
 r3.Init(3,4);
 PrintRectangle(r3);
```

类的成员函数的另一种写法

- 成员函数体和类的定义分开写

```
class CRectangle
{
    public:
        int w, h;
        int Area();     //成员函数仅在此处声明
        int Perimeter() ;
        void Init( int w_, int h_ );
};
```

类的成员函数的另一种写法

```
int CRectangle::Area()  {
        return w * h;
}
int CRectangle::Perimeter() {
        return 2 * ( w + h );
}
void CRectangle::Init( int w_, int h_ ) {
        w = w_;  h = h_;
}
```

- 调用通过: 对象 / 对象的指针 / 对象的引用

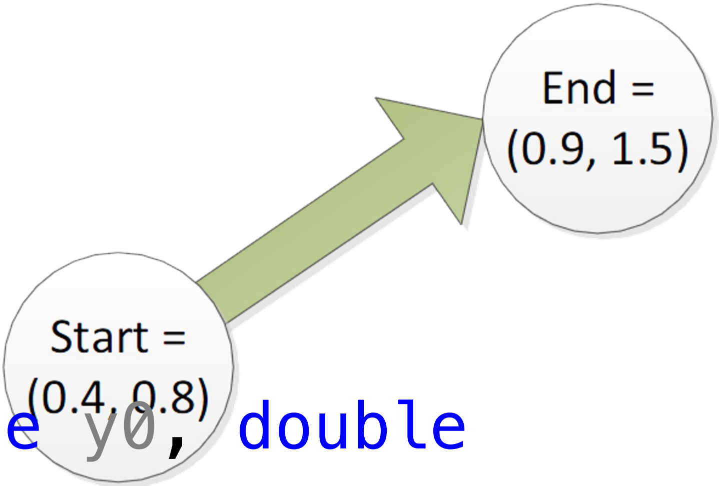
Classes and class members
1st Example

From introduction to c++ @MIT

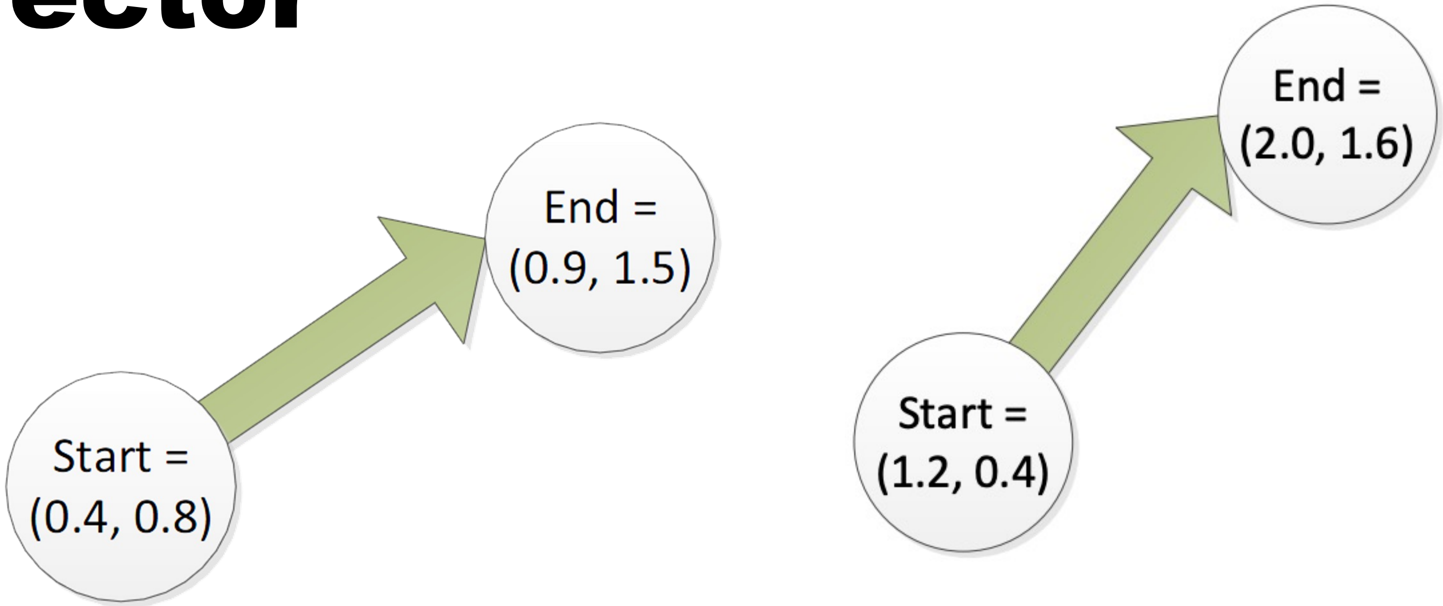
Representing a Vector

- Vector: 2 points (a start & a finish)
- Each point has an x and y coordinate

```
void printVector(double x0, double x1, double y0, double
y1) {
cout << "(" << x0 << "," << y0 << ") -> ("
<< x1 << "," << y1 << ")" << endl;
}
int main() {
double xStart = 1.2;
double xEnd = 2.0;
double yStart = 0.4;
double yEnd = 1.6;
printVector(xStart, xEnd, yStart, yEnd);// (1.2,2.0) -> (0.4,1.6)
}
```



offsetVector



```
int main() {
double xStart = 1.2;
double xEnd = 2.0;
double yStart = 0.4;
double yEnd = 1.6;
offsetVector(xStart, xEnd, yStart, yEnd, 1.0, 1.5);
printVector(xStart, xEnd, yStart, yEnd);// (2.2,1.9) ->
(3.8,4.3)
}
```

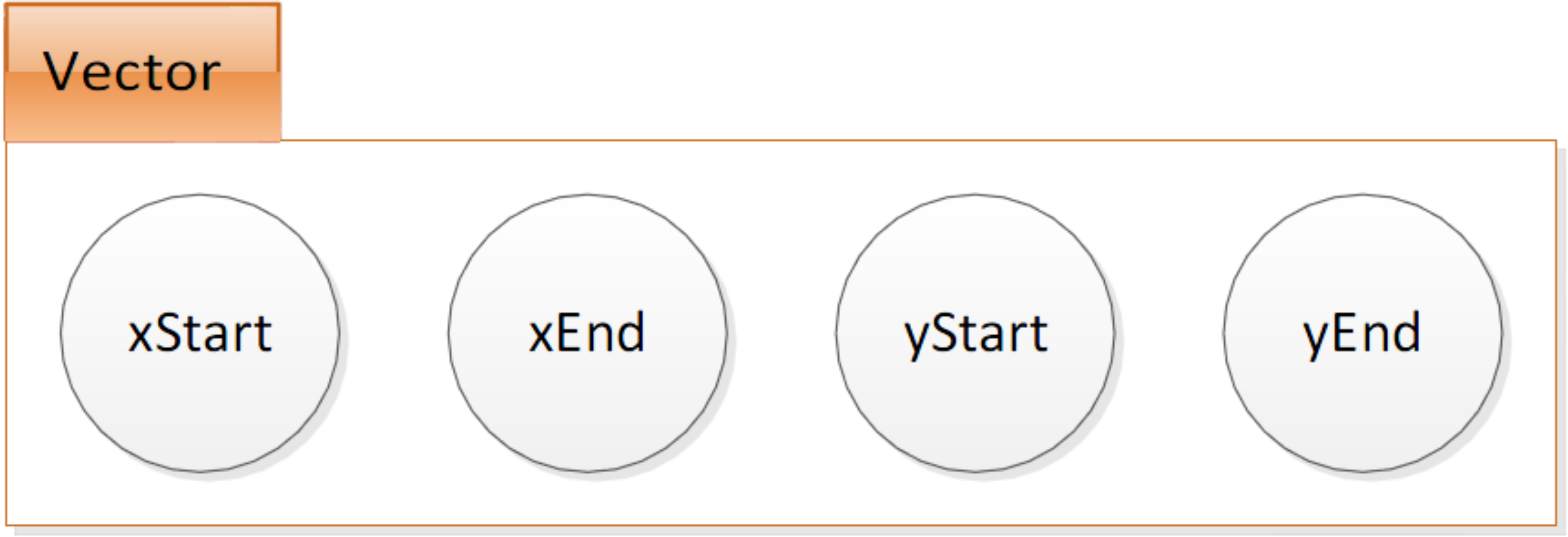
offsetVector

```
void offsetVector(double &x0, double &x1, double &y0, double
&y1, double offsetX, double offsetY) {
x0 += offsetX;
x1 += offsetX;
y0 += offsetY;
y1 += offsetY;
}
int main() {
double xStart = 1.2;
double xEnd = 2.0;
double yStart = 0.4;
double yEnd = 1.6;
offsetVector(xStart, xEnd, yStart, yEnd, 1.0, 1.5);
printVector(xStart, xEnd, yStart, yEnd);// (2.2,1.9) ->
(3.8,4.3)
}
```

Many variables being passed to functions

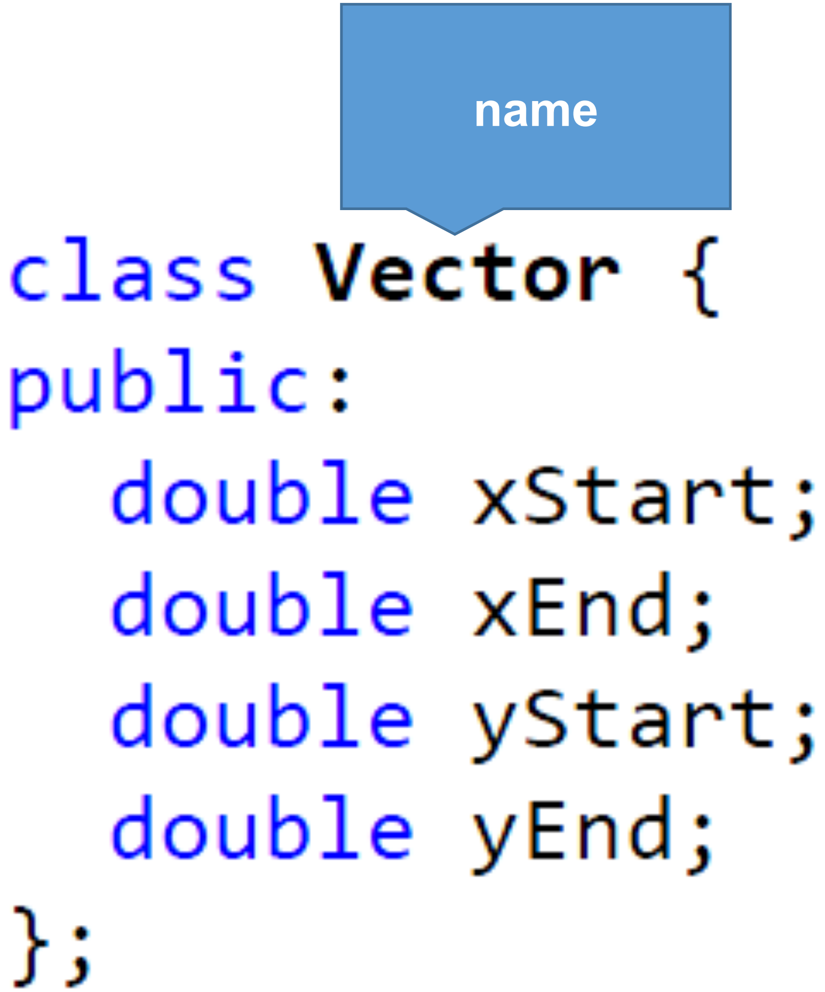
class

- A user-defined datatype which groups together related pieces of info.



class

- Name: indicates the new datatype defined is called Vector

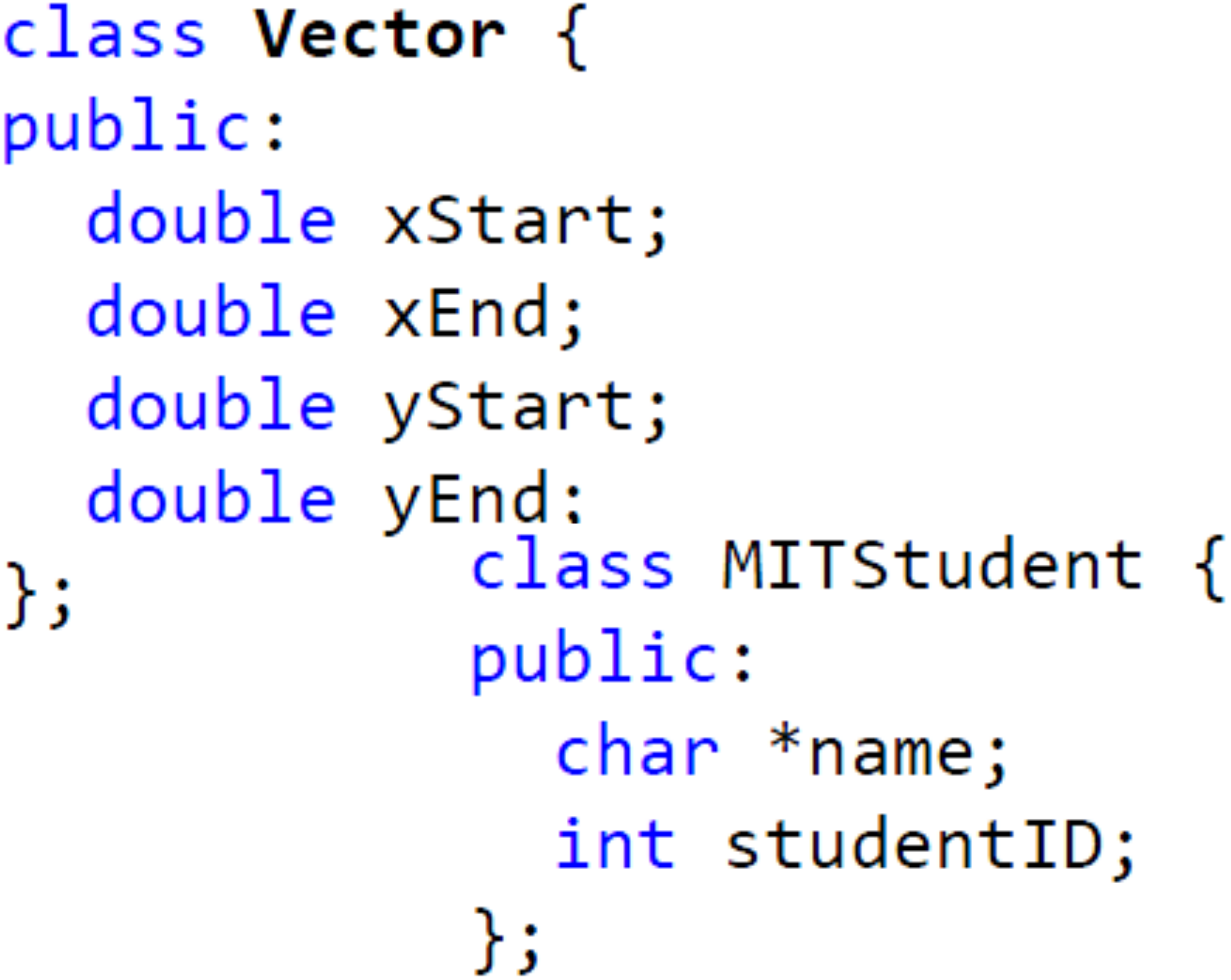


```
class Vector {
public:
  double xStart;
  double xEnd;
  double yStart;
  double yEnd;
};
```

class

- Fields (members) indicate what related pieces of info the class consists of
- Fields can have diff types

fields

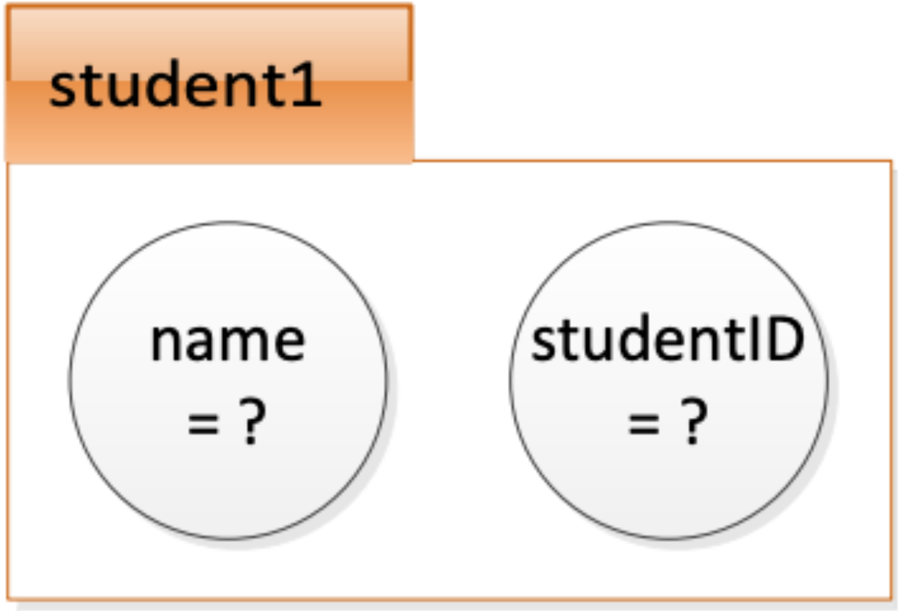


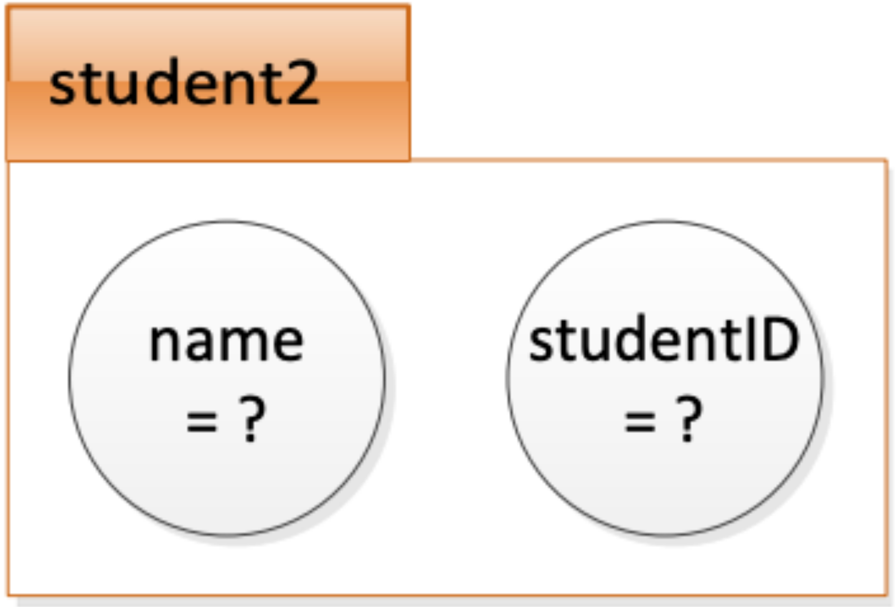
```
class Vector {
public:
  double xStart;
  double xEnd;
  double yStart;
  double yEnd:
};
```

```
class MITStudent {
public:
  char *name;
  int studentID;
};
```

Class & instance (实例)

- An instance is an occurrence of a class
  - MITStudent st1;
  - MITStudent st2;
- St1 & st2 are 2instances of the same class MITStudent



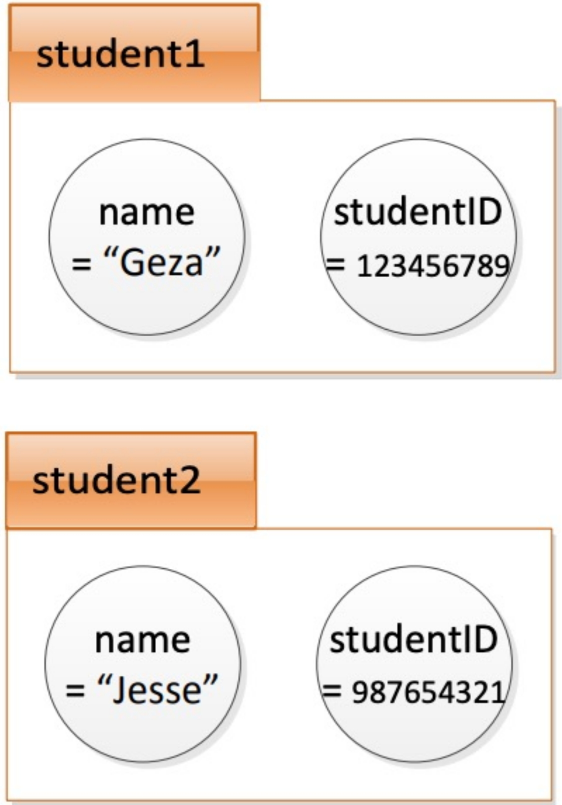


Accessing fields

```
class MITStudent {
public:
  char *name;
  int studentID;
};
```

```
cout << "student1 name is" << student1.name << endl;
cout << "student1 id is" << student1.studentID << endl;
cout << "student2 name is" << student2.name << endl;
cout << "student2 id is" << student2.studentID << endl;
```

```
int main() {
  MITStudent student1;
  MITStudent student2;
  student1.name = "Geza";
  student1.studentID = 123456789;
  student2.name = "Jesse";
  student2.studentID = 987654321;
}
```



student1

name = “Geza”

studentID = 123456789

student2

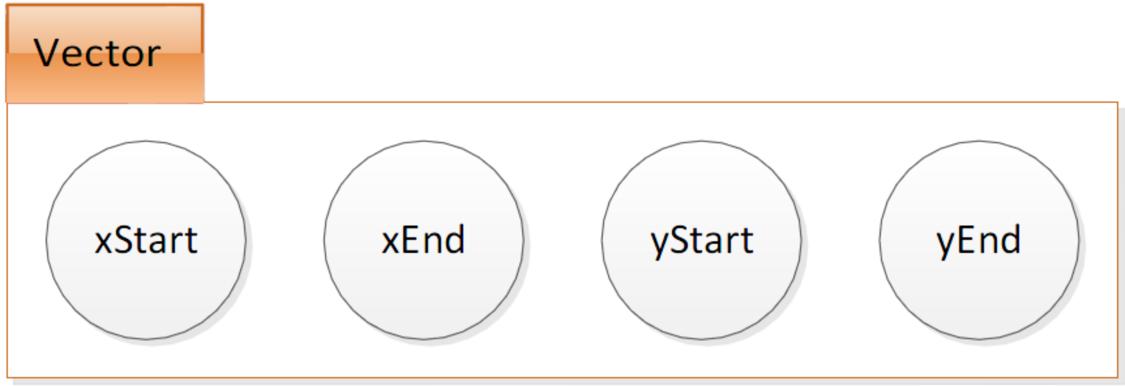
name = “Jesse”

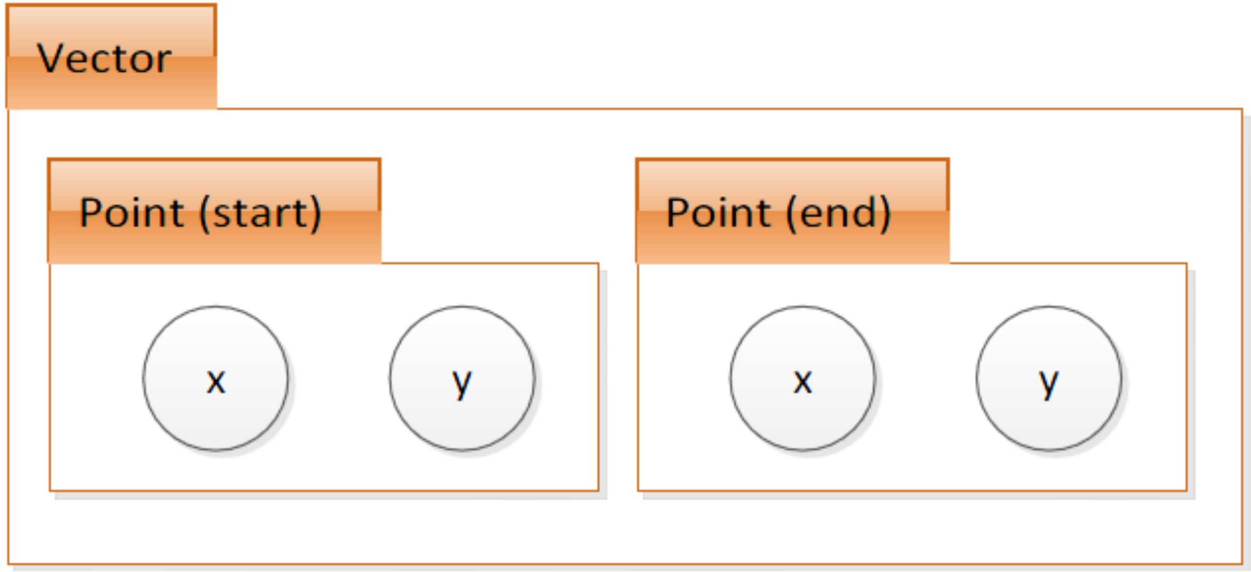
studentID = 987654321

```
class Vector {
public:
  double xStart;
  double xEnd;
  double yStart;
  double yEnd;
};
```

```
class Point {
public:
  double x;
  double y;
};
```

```
class Vector {
public:
  Point start;
  Point end;
};
```

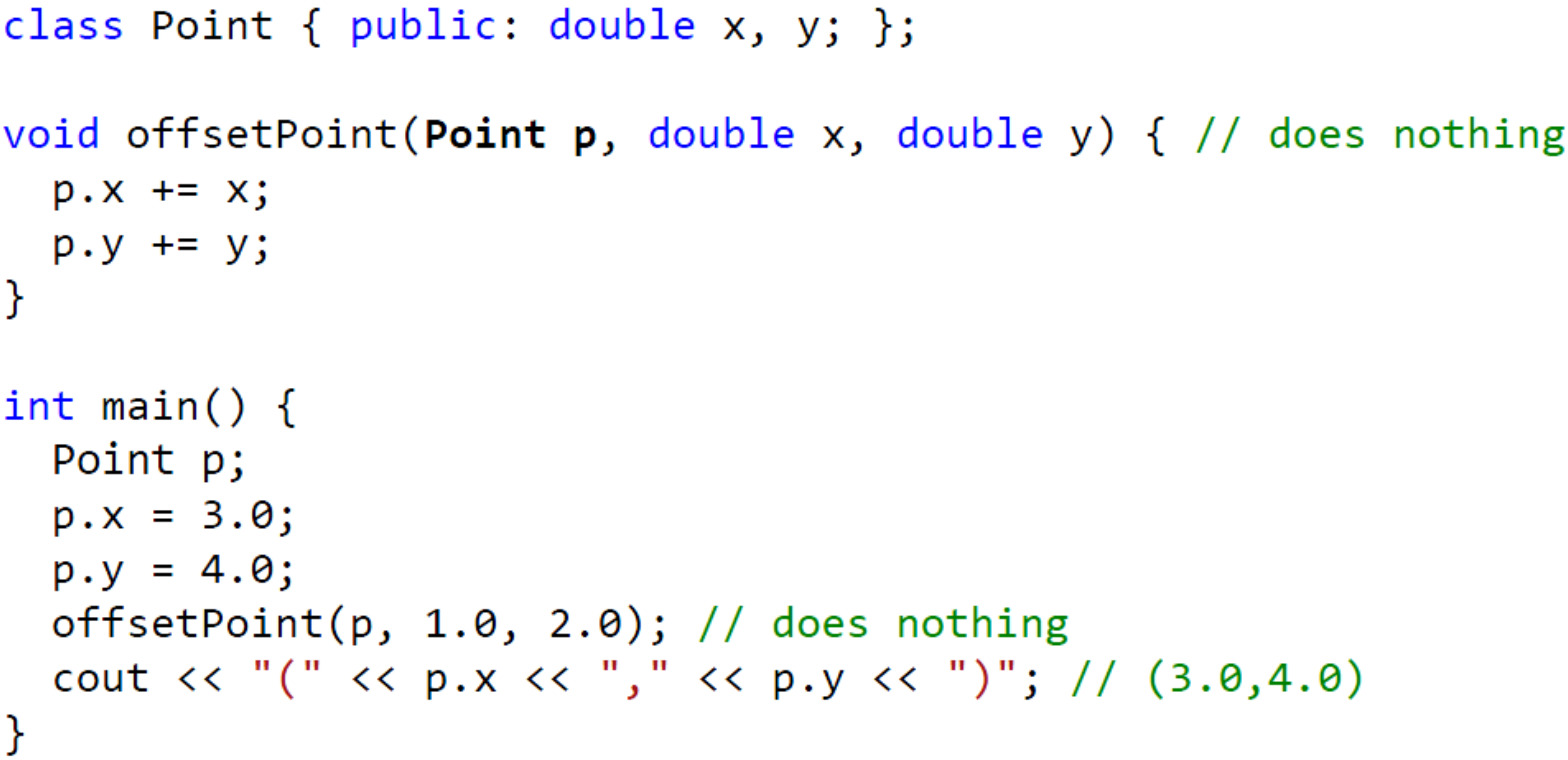




```
class Point {
public:
  double x, y;
};

class Vector {
public:
  Point start, end;
};
```

```
int main() {
  Vector vec1;
  vec1.start.x = 3.0;
  vec1.start.y = 4.0;
  vec1.end.x = 5.0;
  vec1.end.y = 6.0;
  Vector vec2;
  vec2.start = vec1.start;
}
```

Passing classes to functions as values



```
class Point { public: double x, y; };

void offsetPoint(Point p, double x, double y) { // does nothing
  p.x += x;
  p.y += y;
}

int main() {
  Point p;
  p.x = 3.0;
  p.y = 4.0;
  offsetPoint(p, 1.0, 2.0); // does nothing
  cout << "(" << p.x << "," << p.y << ")"; // (3.0,4.0)
}
```

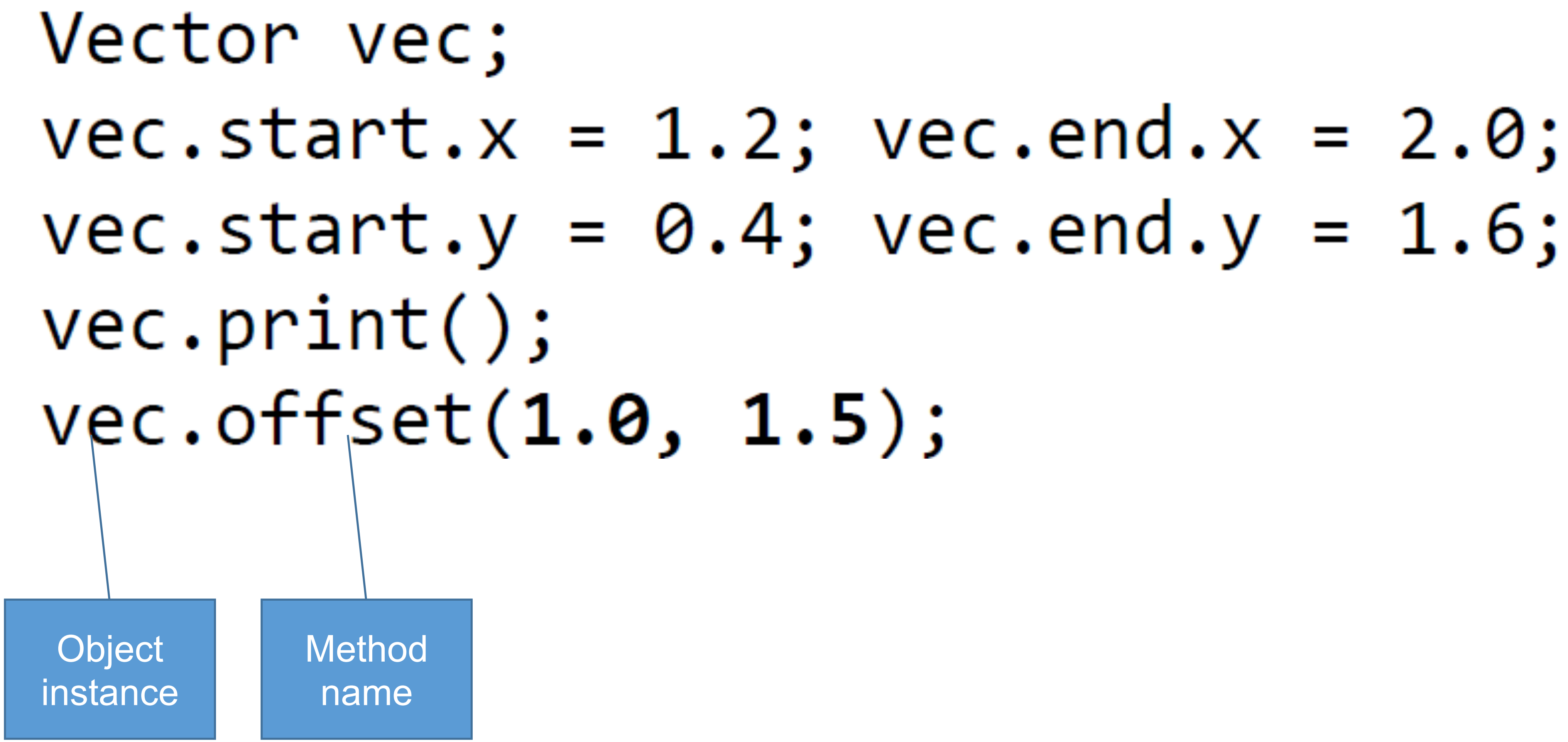
Passing classes to functions as references

```
class Point { public: double x, y; };

void offsetPoint(Point &p, double x, double y) { // works
  p.x += x;
  p.y += y;
}

int main() {
  Point p;
  p.x = 3.0;
  p.y = 4.0;
  offsetPoint(p, 1.0, 2.0); // works
  cout << "(" << p.x << "," << p.y << ")"; // (4.0,6.0)
}
```

Passed by reference

Methods

- Functions which are part of a class



```
Vector vec;
vec.start.x = 1.2; vec.end.x = 2.0;
vec.start.y = 0.4; vec.end.y = 1.6;
vec.print();
vec.offset(1.0, 1.5);
```

```
class Vector {
public:
  Point start;
  Point end;

  void offset(double offsetX, double offsetY) {
    start.x += offsetX;
    end.x += offsetX;
    start.y += offsetY;
    end.y += offsetY;
  }
  void print() {
    cout << "(" << start.x << "," << start.y << ") -> (" << end.x <<
"," << end.y << ")" << endl;
  }
};
```

Fields can be accessed in a method

methods

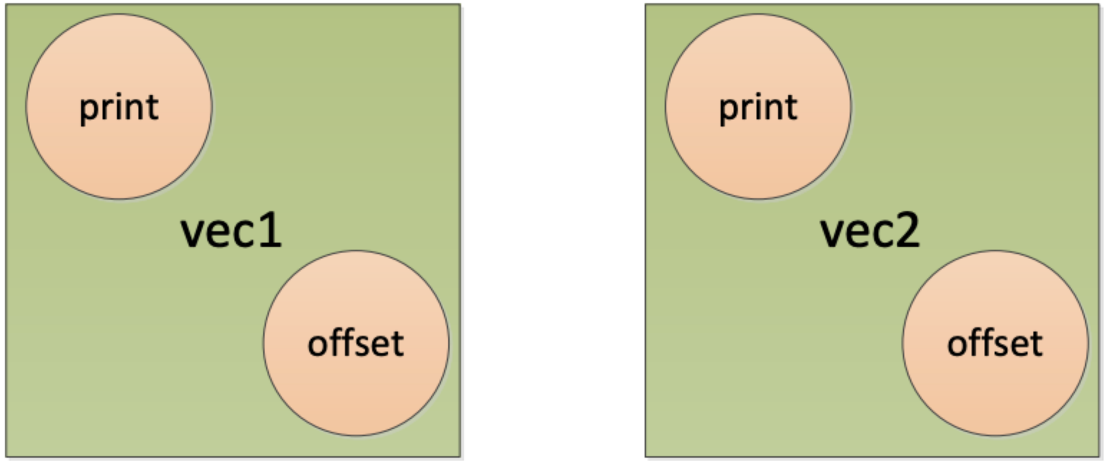
```
class Point {
public:
  double x, y;
  void offset(double offsetX, double offsetY) {
    x += offsetX; y += offsetY;
  }
  void print() {
    cout << "(" << x << "," << y << ")";
  }
};
```

```
class Vector {
public:
  Point start, end;

  void offset(double offsetX, double offsetY) {
    start.offset(offsetX, offsetY);
    end.offset(offsetX, offsetY);
  }
  void print() {
    start.print();
    cout << " -> ";
    end.print();
    cout << endl;
  }
};
```

methods of fields can be called

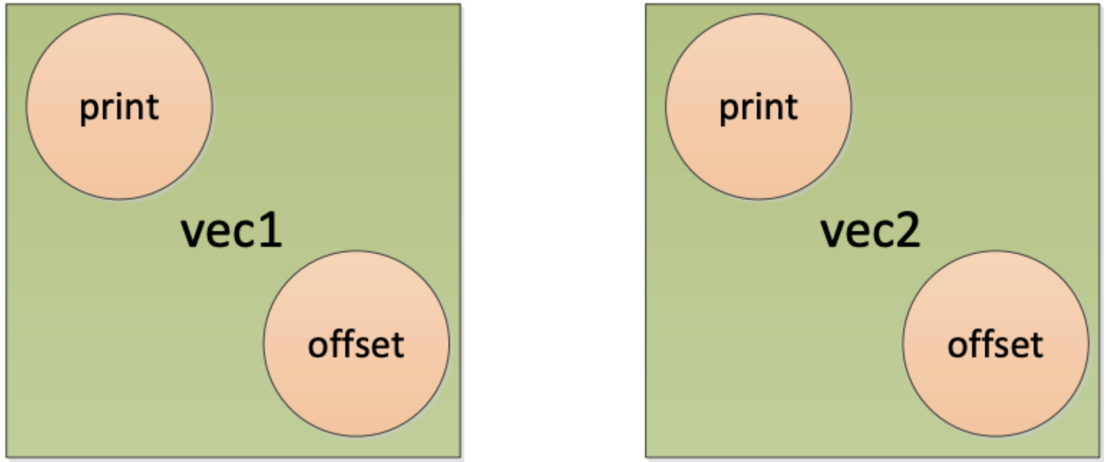
Methods & Instances

- Analogy: Methods are “buttons” on each box (instance), which do things when pressed



```
Vector vec1;
Vector vec2;
// initialize vec1 and vec2
vec1.print();
```

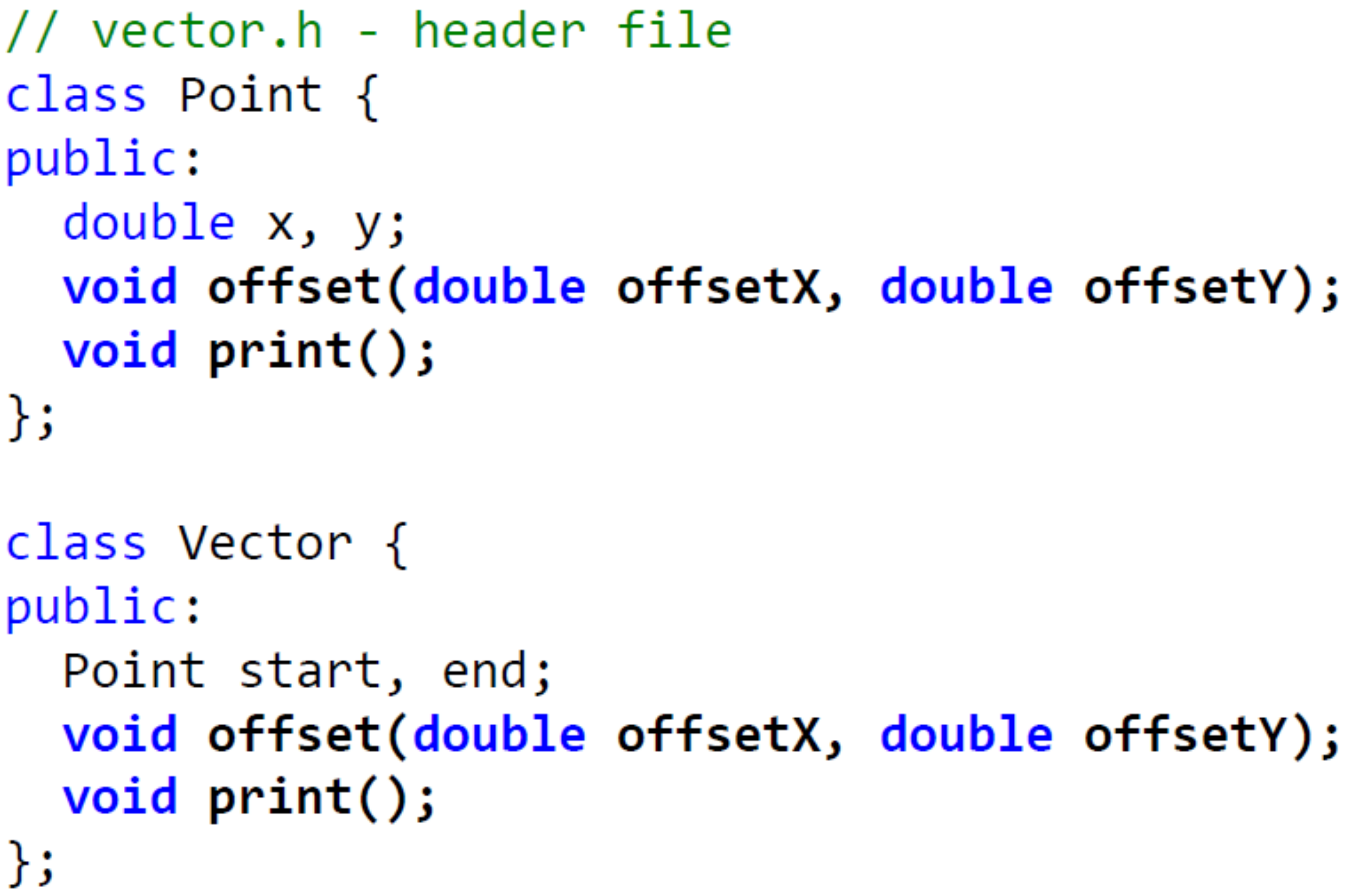
Which box’s button was pressed?



```
Vector vec1;
Vector vec2;
// initialize vec1 and vec2
vec1.print();
```

Which button was pressed?

Implementing Methods Separately



```
// vector.h - header file
class Point {
public:
  double x, y;
  void offset(double offsetX, double offsetY);
  void print();
};

class Vector {
public:
  Point start, end;
  void offset(double offsetX, double offsetY);
  void print();
};
```

```cpp
#include "vector.h"
// vector.cpp - method implementation
void Point::offset(double offsetX, double offsetY) {
  x += offsetX; y += offsetY;
}
void Point::print() {
  cout << "(" << x << "," << y << ")";
}
void Vector::offset(double offsetX, double offsetY) {
  start.offset(offsetX, offsetY);
  end.offset(offsetX, offsetY);
}
void Vector::print() {
  start.print();
  cout << " -> ";
  end.print();
  cout << endl;
}
```

:: indicates which class’ method is being implemented

Python vs C++

```
class Cat
    def getHumanAge(self):
        return self._age

    def setHumanAge(self, value):
        self._age = value

    def getAge(self):
        return self._age * 7
```

```
c = Cat()
c.setHumanAge(5)
print(c.getAge())
```

```
class Cat{
    float _age;
public:
    float getHumanAge(){
          return _age;}
    void setHumanAge(float age){
         _age=age;}
    float getAge(){
         return _age * 7;}
};
```

```
Cat cat;
cat.setHumanAge(5);
cout << cat.getAge();
```

More formal

```python
class Dog(object):
    def __init__(self, age=0):
        self.humanAge = age

    @property
    def humanAge(self):
        return self._age

    @humanAge.setter
    def humanAge(self, value):
        self._age = value

    @property
    def dogAge(self):
        return self._age * 7
```

```cpp
Cat cat;
cat.setHumanAge(5);
cout << cat.getAge();
```

```python
d = Dog(age=4)
print(d.humanAge)
print(d.dogAge)
```

Quiz time

- 2) Write a simple class named Point3d. The class should contain:
  * Three private member variables of type double named m_x, m_y, and m_z;
  * A public member function named setValues() that allows you to set values for m_x, m_y, and m_z.
  * A public member function named print() that prints the Point in the following format: <m_x, m_y, m_z>
- Make sure the following program executes correctly:

```
int main(){
  Point3d point;
  point.setValues(1.0, 2.0, 3.0);
  point.print();

  return 0;}
```

Assignment

- write a class that implements a simple stack.

```
int main(){
Stack stack;
stack.reset();
stack.print();

stack.push(5);stack.push(3);stack.push(8);stack.print();

stack.pop();stack.print();

stack.pop();stack.pop();stack.print();

return 0;}
```