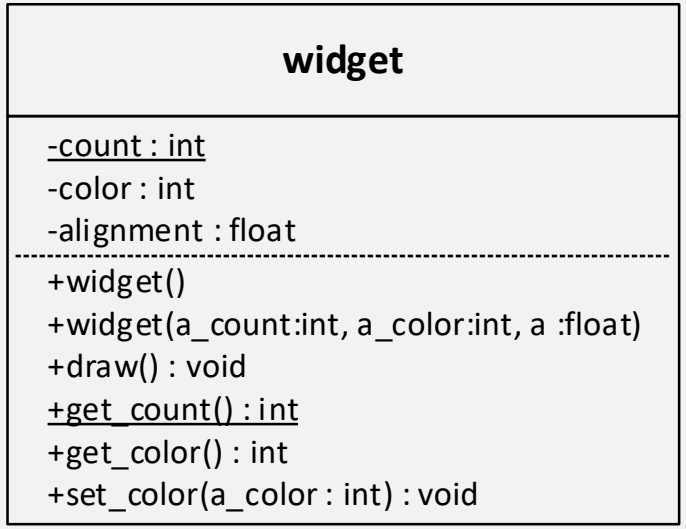
static DATA AND FUNCTIONS

Class Ownership vs. Instance Ownership

Delroy A. Brinkerhoff

static AND UML DIAGRAMS

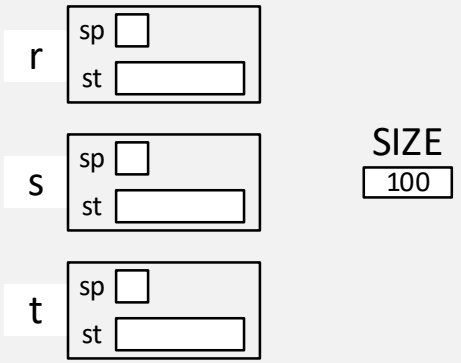
- Static features are denoted by underlining their name in UML class diagrams
  - Attributes or variables
  - Operations or functions
- C++ programmers translate the underline into the "static" keyword



static DATA

```
class stack
{
    private:
       static const int SIZE = 100;
       char st[SIZE];
       int sp;
};

stack  r;
stack  s;
stack  t;
```




```
r.SIZE
stack::SIZE
```

static FUNCTIONS

- static functions belong to the class and not to an instance of the class
  - Are not bound to an object
  - Do not have a “this” pointer
  - Cannot access non-static data
  - Cannot call non-static functions
- static functions are still bound to the class scope
  - `class_name::function_name();`

static EXAMPLE: PART I

foo.h

```
class foo
{
    private:
       static int counter;

    public:
       foo();
       static int get_counter();
};
```

foo.cpp

```
#include "foo.h"

int foo::counter = 0;

foo::foo()
{
       counter++;
}

int foo::get_counter()
{
       return counter;
}
```

static EXAMPLE PART 2

```
#include <iostream>
#include "foo.h"
using namespace std;

int main()
{
    foo   f0;
    foo   f1;
    foo   f2;

    cout << f0.get_counter() << endl;
    cout << foo::get_counter() << endl;

    //cout << counter << endl;        // error
    //cout << foo::counter << endl;   // error

    return 0;
}
```