Classes

EECS 211

Winter 2018

A counter struct

```cpp
// A counter
struct Counter
{
    unsigned long value;
};
```

A counter struct

```cpp
// A counter
struct Counter
{
    unsigned long value;
};

// Increments the value of the given counter
void increment(Counter& counter);

// Returns the value of the given counter
unsigned long get(const Counter& counter);
```

Counter client code

```
Counter c{0};

increment(c);
increment(c);

CHECK( 2 == get(c) );
```

Bad counter client code

```
Counter c{0};

increment(c);
increment(c);

CHECK( 2 == get(c) );

c.value = 0;
```

Bad!

Encapsulation

Encapsulation restricts direct access to (some of) an object's components:

```cpp
// A counter
class Counter
{
    unsigned long value;     // private now!
};

// Increments the value of the given counter
void increment(Counter& counter);

// Returns the value of the given counter
unsigned long get(const Counter& counter);
```

Implementation of Counter class

```
void increment(Counter& counter)
{

    ++counter.value;    // error! member value is private
}
unsigned long get(const Counter& counter)
{
    return counter.value; // error! member value is private
}
```

This doesn't work anymore because making Counter a class
made member value private

Member functions

In Counter.h:

```cpp
class Counter
{
    unsigned long value_ = 0;
public:
    void increment();
    unsigned long get() const;
};
```

In Counter.cpp:

```cpp
void Counter::increment()
{ ++value_; }

unsigned long Counter::get() const
{ return value_; }
```

Counter class client code

```
Counter c;

c.increment();
c.increment();

CHECK( 2 == c.get() );
```

The rule

```
class Class_name
{
    // private stuff

public:
    // public stuff

private:
    // more private stuff
};
```

Private members can only be accessed by other members

The rule

```
class Class_name
{
    // private stuff

public:
    // public stuff

private:
    // more private stuff
};
```

Private members can only be accessed by other members

Thus, clients must interact via public members

Initialization

What if we want to initialize the counter to a number other than 0?

Counter c{10};

This won't work! Why?

Constructors

Constructors are special functions that are run each time a class object is created

A constructor is given an uninitialized object and must initialize it

The name of the constructor is the same as the name of the class, and it has no return type

Counter class with constructors

```cpp
class Counter
{
public:
    Counter();
    explicit Counter(unsigned long);

    void increment();
    unsigned long get() const;

private:
    unsigned long value_;
};
```

Counter class constructor implementation

```cpp
Counter::Counter() : value_(0)
{ }

Counter::Counter(unsigned long value) : value_(value)
{ }
```

Counter class client code

```
Counter c1;
Counter c2(10);

CHECK( 0 == c1.get() );
CHECK( 10 == c2.get() );
```

– To CLion! –