# Object Lifetime for Instances of Classes

```
Class barn {
public:









private:
    Cow * c *_;
    size_t numStalls_;
};
```

## Reminder: Allocation

# Reminder: Initialization

```
barn::barn()
    : c_{new Cow*[4]},
      numStalls_{4}
    {
      for (size_t i=0; i<numStalls_; ++i){
         c_[i] = new Cow{"bessie", 3};
      }

    }
```

# Default Constructors

header: barn();

implentation:

```
barn::barn()
  : c_{"bessie", 3}, numStalls_{4}
{
    ~~c_ = Cow{"bessie", 3};~~
    ~~numStalls_ = 4;~~
    //nothing (else) to do
}
```

member initializers

in the order of the .hpp

use! initialization already happened!

# Parameterized Constructors

```
header:  barn(string cowName, size_t cowSpots,
                   size_t numStalls);

impl:  barn::barn(string cowName, size_t cowSpots,
                   size_t numStalls)
               : c_{cowName, cowSpots},
               numStalls_{numStalls}
    {
         //nothing else
    }
```

# Copy Constructors

header: barn(const barn& other);

can't init parameter w/out a copy constructor

impl:
```
barn::barn(const barn& other)
    : c_{other.c_}
      numStalls_{other.numStalls_}
{
    // nothing to do
}
```

# Destruction: HOW (Destructors)

barn.hpp
header:

```
~barn();
```

impl:
barn.cpp

```
barn::~barn() {
    for (size_t i=0; i<numStalls_; ++i) {
        delete c_[i];
    }
    delete[] c_;
}
```

---

NEVER call it explicitly!

# Assignment Operator

```
barn b1;
barn b2;
~~barn~~ .
 b2 = b1;
```