# Object Lifetime for Instances of Classes

```
Class barn {
public:




private:
    Cow** c_;
    size_t numStalls_;
};
```

# Reminder: Allocation

# Reminder: Initialization

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

# Default Constructors

header: barn();

implementation:

```
barn::barn()
 : c_{"bessie", 33}, numStalls_{43}
{
  // c_ = Cow{"bessie", 33};
  // numStalls_ = 4;
  // nothing (else) to do.
}
```

member initializers

use!

if we don't write, synthesized by compiler.

→ default constructor for all of the data members

# Parameterized Constructors

header

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

implementation:

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

same order as in the header

# Copy Constructors

header: barn(const barn& other);

implementation:

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

# Destruction: HOW (Destructors)

header: ~barn();

implementation:

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

# Assignment Operator

barn b;

barn c;

c = b; ← use "make c look like b"