Constructor with No Arguments Used for Arrays

Two **public** constructors are **created by default**.

1. Constructor with **no arguments**
   - created **only if no constructors declared** in class definition
   - **used for** initializing **array elements**
   - arrays of a class' instances are not allowed if no constructor with no arguments exists

Operation of Default Constructor with No Arguments

What does the default constructor with no arguments do?

Initializes each field based on its type.

Fields that are **not instances** are **left as bits** (as with **C**).

Fields that are **instances** (of another class)
- are **initialized using the constructor with no arguments** for their class,
- which must exist and be accessible.

Copy Constructor Also Created by Default

2. Copy constructor
- also created by default (and made public)
- **takes** one argument: **a reference* to another instance** of the same class

```
MyClass m;
MyClass n = m; // used here
MyClass p (m); // also used here
p = n;         // not used here!
```

*Briefly for now: a reference looks like an instance syntactically, but is implemented as a pointer.

Operation of Default Constructor with No Arguments

What does the default copy constructor do?

Initializes each field based on its type.

Fields that are **not instances** are **copied as bits** (as with **C**).

Fields that are **instances** (of another class)
- are **copied using the copy constructor** for their class,
- which must exist* and be accessible.

*A default copy constructor is not available (said to be "deleted") if this condition is not met.

