Shallow vs. Deep Copy

If you do not write your own copy constructor or assignment operator, they are provided by the compiler.

The Bad News: The ones provided only do a shallow copy.

The Good News: In C++, usually a shallow copy is the same thing as a deep copy.
Shallow/Deep copies: Java vs. C++

class CatOwner {
    string name;
    Cat pet;
}

Java:
    Shallow copy and Deep copy are different

C++:
    Shallow copy and Deep copy are the same

© 2011 Fawzi Emad, Computer Science Department, UMCP
When Shallow is not Enough

class CBool
{
    string *name;
    Cat *pet;
}

When members are dynamically allocated, the built-in copy constructor and operator= are not going to be enough.
Example: BrokenClass

• Copy constructor is not working
• Operator= is not working

Remember that these two members are called implicitly by the system in many circumstances.

This is serious!

We must fix these two broken members.

• By the way, we also have memory leaks.
“The Big Three”

VERY IMPORTANT CONCEPT:

If your class has dynamically allocated members, you absolutely **MUST**:

1. Write a proper copy constructor
2. Overload the assignment operator

And **usually**, you will also need to:

3. Write a destructor that frees the memory.
Example: RepairedBookClass

• Destructor
  – Frees existing members

• Copy constructor
  – Copies members from another object

• Operator=
  – Frees existing members
  – Copies members from another object
Example: LinkedList

Everything but the kitchen sink.