

Copy Constructor Encyclopedia Article

Copy Constructor

The following sections of this BookRags Literature Study Guide is offprint from Gale's For Students Series: Presenting Analysis, Context, and Criticism on Commonly Studied Works: Introduction, Author Biography, Plot Summary, Characters, Themes, Style, Historical Context, Critical Overview, Criticism and Critical Essays, Media Adaptations, Topics for Further Study, Compare & Contrast, What Do I Read Next?, For Further Study, and Sources.

(c)1998-2002; (c)2002 by Gale. Gale is an imprint of The Gale Group, Inc., a division of Thomson Learning, Inc. Gale and Design and Thomson Learning are trademarks used herein under license.

The following sections, if they exist, are offprint from Beacham's Encyclopedia of Popular Fiction: "Social Concerns", "Thematic Overview", "Techniques", "Literary Precedents", "Key Questions", "Related Titles", "Adaptations", "Related Web Sites". (c)1994-2005, by Walton Beacham.

The following sections, if they exist, are offprint from Beacham's Guide to Literature for Young Adults: "About the Author", "Overview", "Setting", "Literary Qualities", "Social Sensitivity", "Topics for Discussion", "Ideas for Reports and Papers". (c)1994-2005, by Walton Beacham.

All other sections in this Literature Study Guide are owned and copyrighted by BookRags, Inc.

Contents

Copy Constructor Encyclopedia Article.....	1
Contents.....	2
Copy Constructor.....	3

Copy Constructor

A **constructor** is a special **function** belonging to a **class** that builds objects of that class from nothing. A "copy constructor" is a special kind of constructor that makes an **object** that is an identical copy of another. It builds the second object by copying the state of the first object into the second.

The copy constructor usually takes as its sole input **parameter** a reference to an object of its own type. In other words, a copy constructor for the `lunchBox` class would take a reference to an already-existing `lunchBox` as its input and create an identical copy of it.

In **programming** parlance this is sometimes referred to as the "X X ref" constructor, where "ref" is short for "reference," where the "reference" is a special kind of **identifier** that "refers" to an object and allows the program to **access** the object, but is not the object itself. In **C++** the reference is denoted by the ampersand symbol, `&`, and the declaration for a copy constructor for the `lunchBox` class might look like this:

- `lunchBox(const lunchBox& theBox);`

Programmers often fail to realize how important the copy constructor is, and, moreover, how important it is to **code** it correctly. For example, a C++ object that contains a `char*`, or a pointer to a **string**, cannot safely copy the **value** of the pointer because there would then be two objects with apparently different **data** members that are actually the same. When one or other of the objects changes the string, or worse, destroys it, the other object will then contain unexpectedly changed data. The best the programmer can hope for in these circumstances is a catastrophic program failure; other consequences might be much worse.

For this reason it is good programming practice to write a copy constructor for every class in the program and allow contained objects to copy themselves rather than have the containing object assume this responsibility. In the above example, the `char*` would be replaced by a properly implemented `String` class that would "know" how to make a copy of another `String` object.