Constant Declarations Encyclopedia Article

Constant Declarations

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.



BOOKRAGS

Contents

Constant Declarations Encyclopedia Article.....1

Contents.....2

Constant Declarations.....3

Constant Declarations

Constant **declarations** create named constants. A constant is the opposite of a **variable** and has a **value** that once set never changes during the life of the program.

There are two types of constant: literal constants and declared constants. A literal constant is a hard-coded number or **string** in the program **code**. For example, in the code below, the variable "x" is assigned the value "3," where the "3" is actually a literal constant:

- int x = 3;

A declaration gives an **object** a name, and in a typed language it will say what type it is. In contrast a definition gives the named symbol a value. Different languages have different **syntax**, but most allow the programmer to declare a constant and give it a name. Declaring a constant is usually very similar to declaring a variable in that it has a name and frequently a type and a value. The chief difference is that the constant has to be defined at the same time as it is declared, like this C/C++ constant, for example:

- const int theAnswer = 42; // **C** and **C++**

If "theAnswer" were a variable it could be declared and then defined elsewhere; but a constant cannot be treated in this way because it is constant and is not allowed to change. It is an error to do this:

- const int theAnswer = 42;
- theAnswer = 43;

C and C++ use the "const" keyword to indicate that a named symbol is a constant, as in the examples above. **Java** has a identical semantics but uses the keyword "final" instead:

- final int theAnswer = 42; // Java

Other languages like Ada and Smalltalk use different syntax altogether, but the principle remains: it is possible to declare named symbols that remain constant for the life of the program.