The compiler that produced this output disposed of the temporary object immediately, but it's possible a compiler might wait longer, in which case the destructor message would be displayed later.

The statement

\[
\text{stock2} = \text{stock1}; \quad // \text{object assignment}
\]

illustrates that you can assign one object to another of the same type. As with structure assignment, class object assignment, by default, copies the members of one object to the other. In this case, the original contents of \text{stock2} are overwritten.

\textbf{Remember}

When you assign one object to another of the same class, C++, by default, copies the contents of each data member of the source object to the corresponding data member of the target object.

You can use the constructor for more than initializing a new object. For example, the program has this statement in \texttt{main()}:

\[
\text{stock1} = \text{Stock("Nifty Foods", 10, 50.0)};
\]

The \texttt{stock1} object already exists. Thus, instead of initializing \texttt{stock1}, this statement assigns new values to the object. It does so by having the constructor create a new, temporary object and then copy the contents of the new object to \texttt{stock1}. Then the program disposes of the temporary object, invoking the destructor as it does so.

Using a constructor to reset an object

Constructor using Nifty Foods called \[\text{la}\text{]temporary object created\]
Bye, Nifty Foods! \[\text{la}\text{]temporary object destroyed\]
Revised stock1:
Company: Nifty Foods  Shares: 10  \[\text{la}\text{]data now copied to stock1\]
  Share Price: $50.00  Total Worth: $500.00