A use of “this.”

The expression \texttt{this} in a method evaluates to a pointer (or the name of) the object in which it occurs. Therefore, the following expression refers to field \( f \) of the object:

\begin{verbatim}
   this.f
\end{verbatim}

Generally, we suggest using “\texttt{this}.” \textit{only} when it necessary. Don’t clutter up a program with useless stuff.

We now show a case where it is necessary. Consider the class and constructor shown below. The constructor is supposed to store its parameter \texttt{name} in field \texttt{name}. However, by the inside-out rule, both occurrences of \texttt{name} in the assignment statement refer to parameter \texttt{name}. This assignment statement simply copies the value in parameter \texttt{name} and stores it back in parameter \texttt{name}.

Here’s some terminology: Parameter \texttt{name} \textit{shadows} field \texttt{name}; it prevents referencing field \texttt{name} directly.

\begin{verbatim}
public class Person {
   String name;    // name of the person
   /** Constructor: an instance with name \texttt{name}. */
   public Person(String name) {
      name = name; // this does not work!
   }
}
\end{verbatim}

The class given below solves the problem, using “\texttt{this}.”.

\begin{verbatim}
public class Person {
   String name;    // name of the person
   /** Constructor: an instance with name \texttt{name}. */
   public Person(String name) {
      this.name = name; // this works!
   }
}
\end{verbatim}

Thus, use “\texttt{this.f}” when field \( f \) has been shadowed by a declaration of \( f \) in some method, either as a parameter or as a local variable.

Note that there is \textit{never} a need in Java to use “\texttt{this}.” in a method call, as in \begin{verbatim}
   this.m(5). // Please don’t do this!
\end{verbatim}
because method names cannot be shadowed —one cannot declare a method within a method in Java.