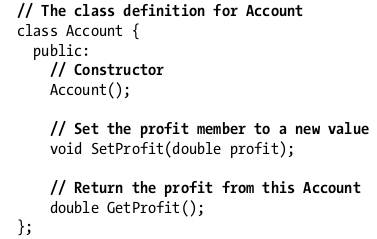
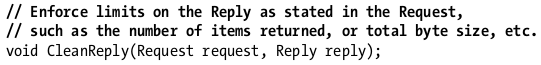
**Chapter 5 Knowing What to Comment**

* reading a comment makes time away from reading the actual code.
* example of unnecessary comment:



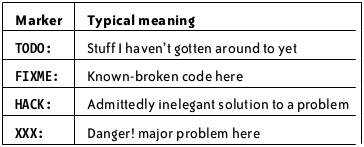
* these comments are worthless because they don’t provide any new information for the reader.
* reading the commented code is much faster than understanding the code without it.
* don’t comment just for the sake of commenting
* don’t comment bad names, fix the names instead.



better:



* better name is better than a good comment because it will be seen everywhere the function is used.
* Include comments to record valuable insights about the code.
* note when improvements should be made:
* //TODO: use faster algorithm or //TODO: handle other image formats besides JPEG
* Some Markers:



* comment on your constants; example:



* put yourself in the reader’s shoes
* anticipate likely question



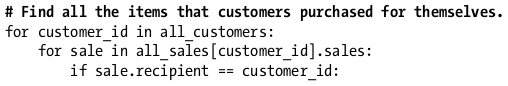
* anticipate the problems that people might run into when using your code.
* comment the implementation detail; example:



* high level comments:



* doesn’t mean you have to write formal documentation.
* summary comments - summarizes the low-level code below it.



* comment the why not the what and how.
* the best solution to writer’s block is to just start writing.
* Writing a comment:
  + Write down whatever comment is on your mind.
  + Read the comment, and see what needs to be improved.
  + Make improvements.