Comments
The passage talks about comments in coding, saying they can sometimes help but often cause more harm than good. It suggests that relying too much on comments means you haven't written your code clearly enough. Comments can become outdated and wrong over time because they don't get updated along with the code. An example is given to show how comments can end up not matching the code anymore. It suggests that while it's important to keep comments organized, the main focus should be on writing clear and expressive code that doesn't need many comments. The author believes that the truth of the code is more important than comments, although sometimes comments are still needed.

Comments Do Not Make Up for Bad Code
The passage says it's crucial to make your code clear and easy to understand instead of relying on comments to explain bad code. It advises against just adding comments to messy code and suggests that the best way to fix it is by cleaning up the code itself. The main idea is that writing clear code is better than cluttered code with lots of comments. By spending time making code well-organized, developers can make software that's easier to work with, fix, and understand, which improves the quality of the code and how fast they can work.

Explain Yourself in Code
The paragraph talks about how some programmers think that writing explanations in code isn't helpful. It says it's important to keep code clear and simple. It argues against using long explanations in comments by showing a comparison between a long comment and clear code. It says that good code can explain itself without needing lots of comments. It suggests that programmers should focus on writing clear and expressive code instead of long comments. Overall, it says that well-designed code can communicate its purpose well and be easier to understand.

Good Comments
The passage talks about different types of comments in code and explains why they're important, but also warns about potential problems. It starts by mentioning that some comments are legally required, like copyright statements, but advises against making comments overly legalistic. It then discusses different kinds of helpful comments: informative comments give extra information, but might mean the code needs better names or organization; explanation of intent comments explain why the code was written a certain way, which can help but might be misunderstood; clarification comments make unclear code easier to understand, but need to be very accurate; warning comments point out potential issues; TODO comments highlight areas for improvement; and amplification comments stress important parts of the code. It also stresses the importance of clear and honest documentation for public APIs, but acknowledges that javadocs (a type of documentation) can have flaws. Overall, it says comments in code should be clear, accurate, and relevant, but warns against relying on them too much or using them incorrectly.

Bad Comments
It talks about why comments are important in coding and gives advice on how to make them helpful. It says that bad comments are often used to cover up messy code and that good comments should explain what the code does in a clear way. It also talks about different types of bad comments like ones that don't make sense, repeat what the code already says, or are just there because someone said they should be. The author suggests keeping comments short and to the point, and getting rid of code that's commented out instead of leaving it there.

It gives an example of how to rewrite code so it's easier to understand without needing lots of comments. It says that good comments should add to the understanding of the code, not make it harder to read. Overall, it's a reminder that writing good comments is an important part of making code that's easy to work with and understand.