One potential issue is where simultaneous edits by multiple clients could lead to inconsistent or states of the shared sketch. For instance, if two clients attempt to modify the same graphical element at the same time, there might be a conflict in the final state of that element, depending on which client's modification gets processed last. To address these issues, synchronized methods are used. These methods ensure that only one thread can execute a block of code that modifies shared data at a time, thus preventing concurrent access and modification of shared resources. However, while synchronized methods are effective in preventing these conditions and ensuring data consistency, they might not handle all issues related to multi-client interactions. For example, they do not inherently solve problems related to the ordering of operations or deadlocks, where two or more operations wait indefinitely for each other to release needed resources. While synchronized methods are crucial in managing access to shared resources in a multi-client environment, they are part of a broader strategy required to handle all the complexities of concurrent programming. It is important to consider other techniques and designs, such as more advanced control mechanisms, to ensure effectiveness in the multi-client system.