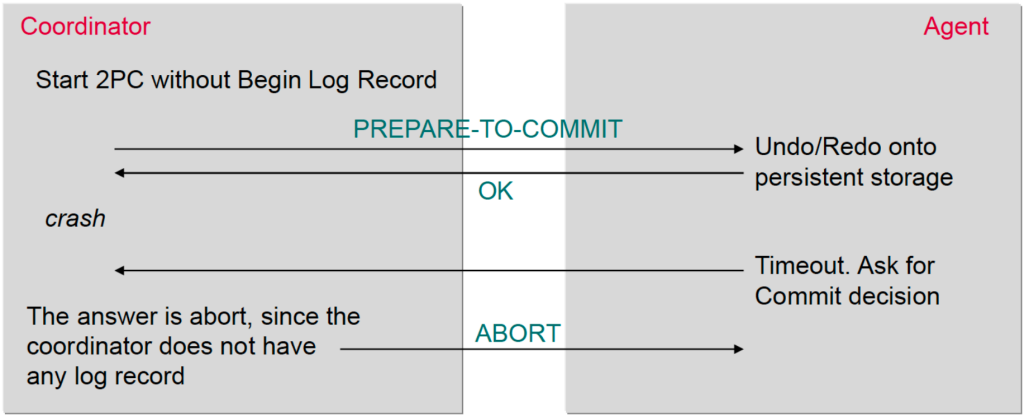
Arbër Hyseni
Parth Shandilya

1 Distributed Transactions with Java

(a) Attached to this document you will find the java files that implements the required functionalities.

(b) Presumed Abort 2PC: In the existing implementation it is not possible to implement this version of 2PC because it requires a different way how two banks communicate. It would be possible if the communication between banks is done asynchronously. Let's look at the illustration below



If the coordinator crashes, the agent should be capable of requesting the decision. However, in our current implementation, the agent lacks this capability due to the absence of a *run* function. Such a function could facilitate the use of queues to manage transactions and enable the agent to inquire about the appropriate course of action after a timeout.

Regarding Transfer of Coordination in the Two-Phase Commit (2PC) protocol, this mechanism could potentially be adapted to the current scenario. Transactions could be linked in a chain, and during the "prepare-to-commit" phase, the coordination token could also be transferred. This would allow the roles to switch dynamically: the coordinator would become an agent, and the agent would assume the role of the coordinator. In this setup, the final node acting as the coordinator would send both the commit message and the coordination token back in a chain-like manner.