Catan – CEBP Project

Project Team:

- Andăr Răzvan
- Buș Raul
- Chileban Dragoș
- Crișan Alexandru

GitHub Repo: https://github.com/crisanalex08/CatanCEBP

Description

Our goal is to create a strategy game inspired by the board game Catan. We want to build a turn-based game in which each player can build different types of buildings and can trade with other players for needed resources to win the game.

In our development journey we found multiple possible concurrency problems and for this milestone we resolved a few of them, among them is trying to trade with another player and trying to build at the same time. The solution we found is to use Concurrent Hash Map which is designed to handle concurrent access by multiple threads. So, in our Player Class we use a Concurrent Hash Map to hold the resources for our player and methods that modify the Resources map are synchronized allowing just one thread to run that function at a time. And for the building concern we use a Reentrant Lock letting just a single thread to build at a time.

Regarding trading, another concurrent problem is that multiple threads could interfere with trade. We use Countdown Latch for trade synchronization. Timeout is used for preventing deadlocks

We use Fair Reentrant Lock to ensure that threads acquire the lock in the order they requested it. Also, we use a Condition for the turn lock that the current player which is an AtomicInteger which is thread-safe and ensures that just one thread may be trying to update the current player index simultaneously.

For checking the win condition, we have a volatile Boolean gameEnded which ensures visibility across multiple threads. This allows all threads to see the most recent value of the flag.

Another concurrency problem we found is that when a player rolls the dice and the gameState offers resources based on the dice number, multiple threads may attempt to access and modify their resources simultaneously which can lead to an inconsistency state for the game. To address this issue, we use the same resourceLock , letting just one player at a time to get its resources.