Concurrency problems in our strategy game -catan app

We were able to find a few cases of concurrency in our game, most of them being based on race condition and memory consistency issues.

Case 1:

- player1 (thread 1) sends the command to get all shared available resources

-another player (thread 2) sends a command to create a house, substracting 3 cards from him and adding them to the shared cards.

Which one does it first?

Case 2:

- player1 (thread 1) sends the command to get all shared available resources

-another player (thread 2) sends a command to create a city, substracting 5 cards from him and adding them to the shared cards.

Which one does it first?

Case 3:

- player1 (thread 1) sends the command to get all shared available resources

-another player (thread 2) sends a command to get a new resource card, substracting 3 cards from him and adding them to the shared cards , also substracting the wanted card.

Which one does it first?

Case 4:

-thread 1 sends exchange-resource request to thread 2 for resource x

-thread 2 sends create house request to build house, using resource x

Solving concurrency issues

Given that we have shared resources, and trade requests for exchanging them, we will have a visibility problem and an access data problem. To make our code thread safe, we will use syncronized methods, ensuring that only a single thread can execute a block of code at the same time, seeing the effects of all previous modifications that were guarded by the same lock.

We will use semaphores for blocking a thread execution and unlocking it. But we need to be carefull not to get into a deadlock or a livelock. Also a monitor lock can be used, so that only a thread can enter into a method, being able to wait for the condition variable or to notify other threads that the condition is met.

For some parts of the code that require calculation a timeout will be set, and the current thread will be interrupted. So, if we will use blocking of threads, those blockers will be also interrupted.

Also, we can use imutability, for each resources.