Challenges and Lessons learned

* Challenges
  + Out of date objects
    - Due to the use of a serialized data any modifications to current objects would not be actively reflected in the database. This means that we have to explicitly update the stored objects.
    - Once code has been changed we also have to regenerate the database with the new objects. Many times we added serialization support to objects and forgot to update the database and wasted time searching for a bug that didn't exist.
    - Objects that were created with the wrong version of the codebase also caused issues if people were working in different branches to build the codebase.
  + Database communication
    - It took us a while to get a database working correctly since we had permission errors early on
    - Serialization also cause many problems since objects were being updated across different versions. We also had issues with VPN access and some odd connection issues when working on the CS lab machines.
  + AI training
    - Trying to balance out the AI in a way that is fun to play yet easy to train was a challenge since there are almost an infinite number of board,state combinations. This means that we cannot create an exhaustive list of moves for the AI client to make. This means that we have to have the AI appear smart even when it doesn't know what to do or hasn't experienced the current state. This took some tweaking and lots of training for us to reach a state we were happy with.
  + Serialization
    - We ran into problems with different environments across different computers that resulted in the objects not being compatible. To fix this we standardized all our working spaces which took a lot of time away from other tasks
  + Connecting front and back end
    - We had created our front in before our back end and this resulted in some issues connecting the two since they were mostly built in isolation. We also had some severe issues with CPU performance due to swing.
  + Scaling issues with Swing
    - When running in java 1.8 the JDK tells the OS that it is capable of handling high DPI scaling but in fact does not. This resulted in hi res screens showing the gui very small and lower res screens having elements off the edges. Eventually we found a fix but the better solution would have been to use java 1.9 for the project.
  + Achieving functionality across multiple environments
    - Due to having different JRE and JDK object serialization was problematic. Certain operating systems would also refuse to play well with one another which resulted in much frustration.
* Lessons
  + Standardize early
    - By listing out all message types and communications between classes and components we would have saved a lot of time since we wouldn't be debugging so often. It would also allow us to remove many casting calls and type conversions since both ends of the pipeline would take the same information.
  + Pair program more often
    - By pair programming we were able to think and code more efficiently since the partner would be able to think about all the issues that would arise from the current additions. We also paired front end and back end group members together later on so that we could be more well rounded and that greatly improved our overall knowledge of the code base
  + Learn the whole codebase sooner
    - We ended up having to ask specific group members to do certain tasks since often we were not familiar with all the code. Eventually through pair programming and lab sessions we reached fluency but if we had done it sooner we would have been much for efficient overall.
  + Documentation adds up
    - Often times we would leave the documation to the days before or just after the presentation which meant that it was often rushed or not as thought through as well as it should have been. We learned to start adding to the documents or atleast noting what would need to be added ahead of time so that when we did update it we could do it faster and more efficiently.
  + Communication is key
    - Our communication overall was pretty good but could have been improved especially when talking about how we planned on sending data between modules. This ties back to the standardize early and pair program often sections mentioned above. We had issues when asking one another about what was going on in the code and keeping one another up to date on our respective branches. This meant that some features were implemented differently or progress was made redundant.
  + There will always be more bugs
    - Project are never over and we learned that even when we thought everything was ready that something would break or be discovered. We had to be ready at all time to consult on how to fix the bug which took some time since we were all very busy this semester.