Comments on Design Spec for CS22120 Group 11
Note that I won't comment on the repo again here. Make sure that you address the comments from previous feedback before the final hand-in.

That said, section 1 is incomplete. Cite requirements spec etc.

Architecture
2.1.1 is fine. You should also say that it is run as a JVM based process.

Have a separate entry for each class rather than listing them. E.g. state what tile subclasses do. Attack is not a great name for a class since it feels procedural. Normally, OO classes are noun based with verb-based name associated with methods on classes, e.g. Ship class with attack method that takes another ship as a parameter. What about the Card classes? Okay you have some in the table. I would include all classes that you can think of. Also, I don't see UI controller classes.

The requirements table looks good, although I haven't checked the mapping. Okay I can see one controller class. Be careful that you don't end up with very big classes (god classes) that do most of the work.

Dependencies
It's useful to show packages but you need much more, and this isn't a component diagram. You can make the classes identified earlier as components with key methods highlighted and important links between them. See the slides to see what a component diagram looks like. I know this is a small group, but don't sacrifice quality in order to try and implement all the functional requirements. I would prefer to see very well engineered system consisting of great docs and code rather than poor docs and code.

Interface
Don't use screenshots of bits of code. Use the format discussed in SE.QA.05. Otherwise this will become too implementation focused. Why are the fields public? That breaks encapsulation.

Be careful that when returning arrays or arraylists that you make copies rather than returning the original. Otherwise, you break encapsulation, i.e. the caller can change the contents of the data structure outside of the class within which it is supposed to be managed.

I don't see methods such as move, trade, attack. I mainly see getters and setters. Also ChanceCard subclasses to handle chance card behaviour, e.g. via overriding and polymorphism?

Detailed design
Good to see sequence diagrams linked to use cases. Add the use case title too, to make it easier to understand. Always have a caption on all figures.

Many of the messages to model objects don't really say what is happening, or just say pass the data. Try to think of the methods that would be called. I think more detail is required. Also where is alternative behaviour shown, e.g. move choices?

I can see from 5.2 that you will be using a large set of if else if statements (or switch) to handle chance kinds. This will work but is not very OO. Normally, I would expect to see grouping of behaviour with inheritance, overriding and polymorphism. That said, you may well be pushed for time, so your approach, although harder to maintain, should work.

There are no class diagrams nor object diagrams to should parts of the design in more detail. Good to see discussion of persistence. What about internal data structures for storing other kinds of data, e.g. card decks?

No statechart diagram. How will game states be handled?