Lola Keychenko

Bradley Sides

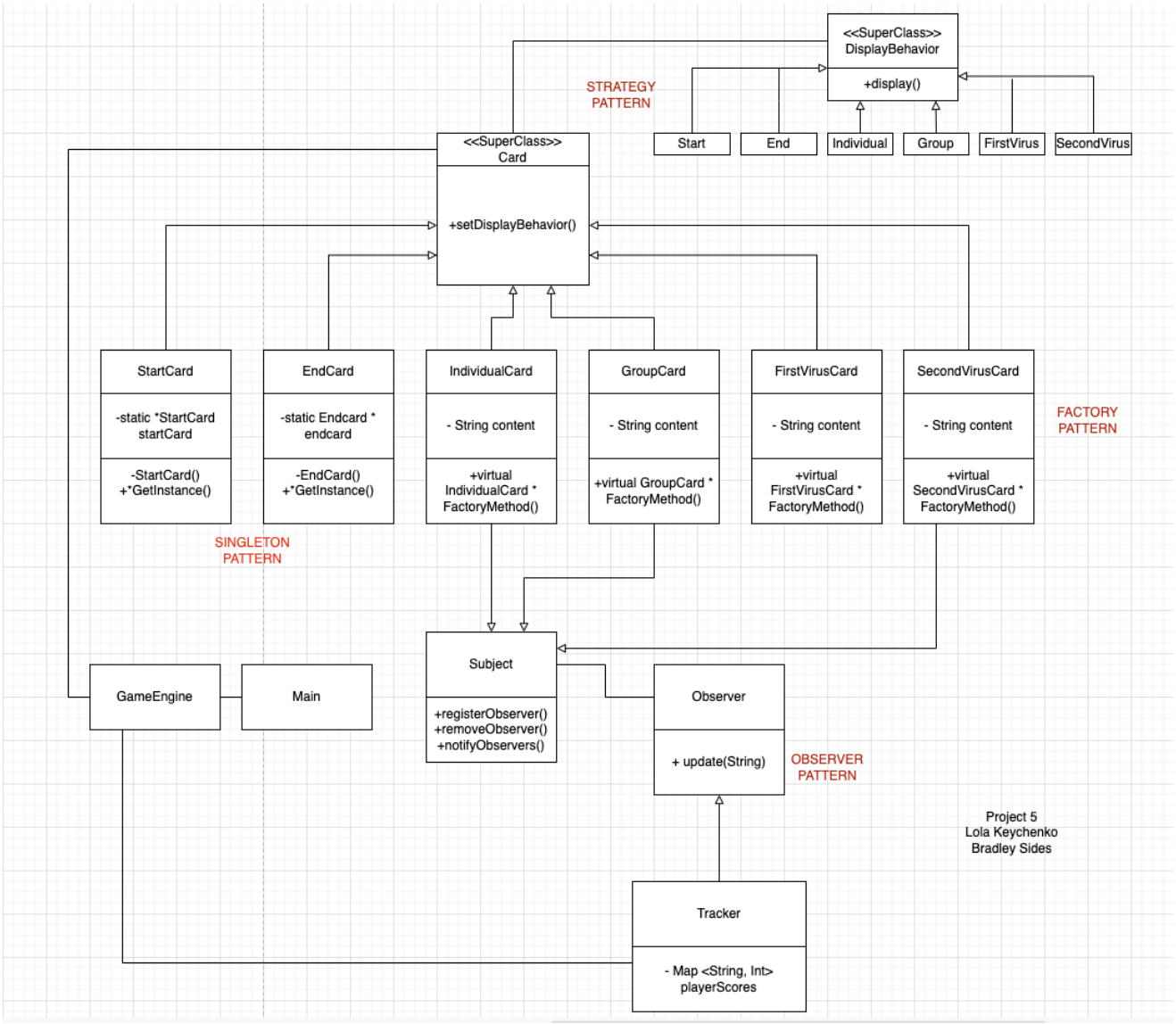
Project 7: Penalties

Final State of System Statement

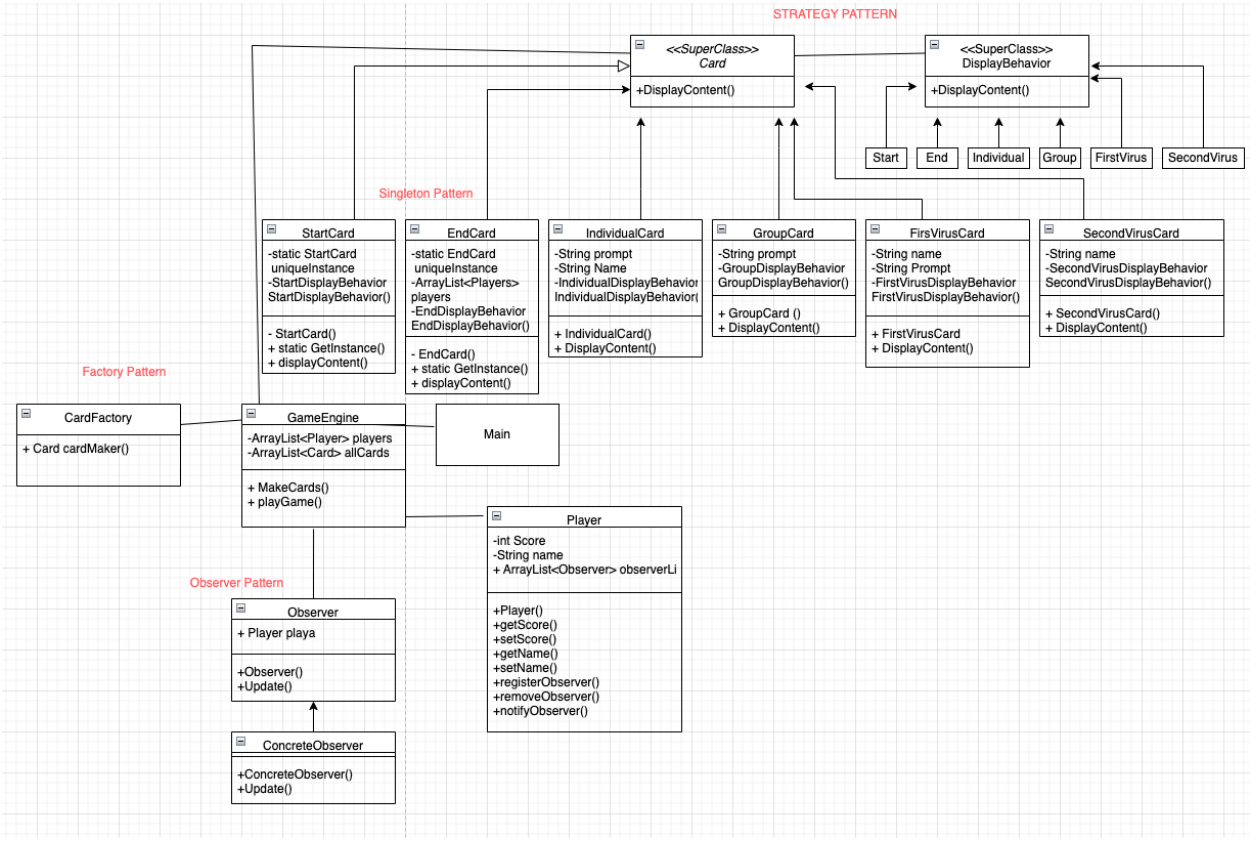
We implemented almost all of the features which we set out to. We have cards which are shown to the group that indicate the prompts which must be done. We have user input indicate when the prompt is completed successfully and gives a point to those that earn it. The cards come in a few categories: individual, group, and virus. We have a start card which collects the names of all players via command line and an end card which displays all of the players and the points that they have earned. We completed the UI aspect of the project, although differently than what we originally planned. We opted for a command line UI, instead of HTML wrapped in C++ since we were told this was a viable option during our project 6 demo. We also transferred all of our code from C++ to Java, since when we were putting together the project we were having so many issues with memory allocation and we did not need to use C++ for UI anymore.

Final Class Diagram and Comparison Statement

Project 5 Class Diagram:



Project 7 Class Diagram:



We did not have an abundant amount of changes in our UML diagrams. The largest change between the two was that we switched from C++ to Java, so we did not need to pass pointers and the way we implemented patterns was slightly different to fit the language. We also had changes in names of classes, in order to preserve clarity.

Third Party Code vs. Original Code Statement

We did not use any code from any third parties, however we did look at code examples for different patterns in C++ and online Java. Underneath are URLS for third party references:

https://refactoring.guru/design-patterns/factory-method/cpp/example

https://refactoring.guru/design-patterns/singleton/cpp/example#:~:text=Singleton%20in%20C%2B%2B,the%20modularity%20of%20your%20code.

Statement on the OOAD Process for Overall Semester Project

A key design process issue we experienced with this project was the use of C++ to create design patterns. Since we learned how to do many patterns in Java, we struggled with adjusting all patterns into C++ and understanding how to use references and pointers in an appropriate way to preserve the integrity of the pattern. A key element we did appreciate in the design process for this game was the long period of time to think about and redesign the flow and structure of our project. Having a large amount of time to write and rewrite UMLs and other structural charts allowed us to finalize a design we were confident in and comfortable creating. We believe we avoided wasting a lot of time by doing this initial process, but we still fumbled with development and gave into using Java anyways. One more element we struggled with was the UI component of this game. Originally, we had planned for a colorful UI via HTML, but as time went on we realized it would be difficult to implement and time consuming. After doing research about alternative UI's, like QTCreator, we eventually settled on a

command line UI once we found out it was an acceptable option. Had we had more time, a nicer UI would be the next step to advance this project.