Individual Pitch:

Team Member: Justin Viacrusis

One-liner: I will additionally be in charge of Collisions, but as the Lead Architect, I will mainly be in charge of making sure the overall structure of the code is polymorphic and well thought-out.

Outline: To bring together all the separate parts of the other members, I will be making the Game class, as well as the InputHandler class. I will also be implementing collision through the CollisionHandler class.

Technical Details: The Game class is a singleton that will aggregate many different GameElements and Game processes and call on them polymorphically. InputHandler will wrap the default inputs of PApplet, to give it more functionality and stability.
CollisionHandler will also aim to provide a polymorphic and low-coupling design for collision handling by taking all unique pairs and doing a 2 way collision where each object only knows the bare minimum (ball *only* knows what balls does when it hits a paddle, and paddle *only* knows what paddle does when it hits a ball.)

Impact: My contribution will give a strong foundation for the other members to latch on to. The SOLID principles applied will allow for a much easier time bringing together all of our code. Having the Game and the interfaces that Game uses, allows for each member to start virtually anywhere at any point and just start working, as soon as the bare bones interfaces have been made.

Timelines: The skeleton for Game and the interfaces shouldn't take more than a few days. This will be the latching-point for most other code in the codebase. Refining it and making sure it is bug free with every addition will take longer, but it will be more spread out throughout the project timeline.

The collision handling will be much more involved and will probably take closer to a few weeks worth of work to be able to complete, and will depend on the ball, slab, and other collidables to finish, before it is ready to be completed.

Risks and Mitigations: A large risk with this architecture is the startup cost. Laying a good foundation means more cost when it comes to just starting. To even get anything drawing, we would need several classes and several hundred lines.
To mitigate this overhead, Each team member will be set up in far corners of the code and test things in their own main() functions. After each element is confident in it's completeness, and the Game class skeleton is complete, we will be past the hurdle of startup costs.