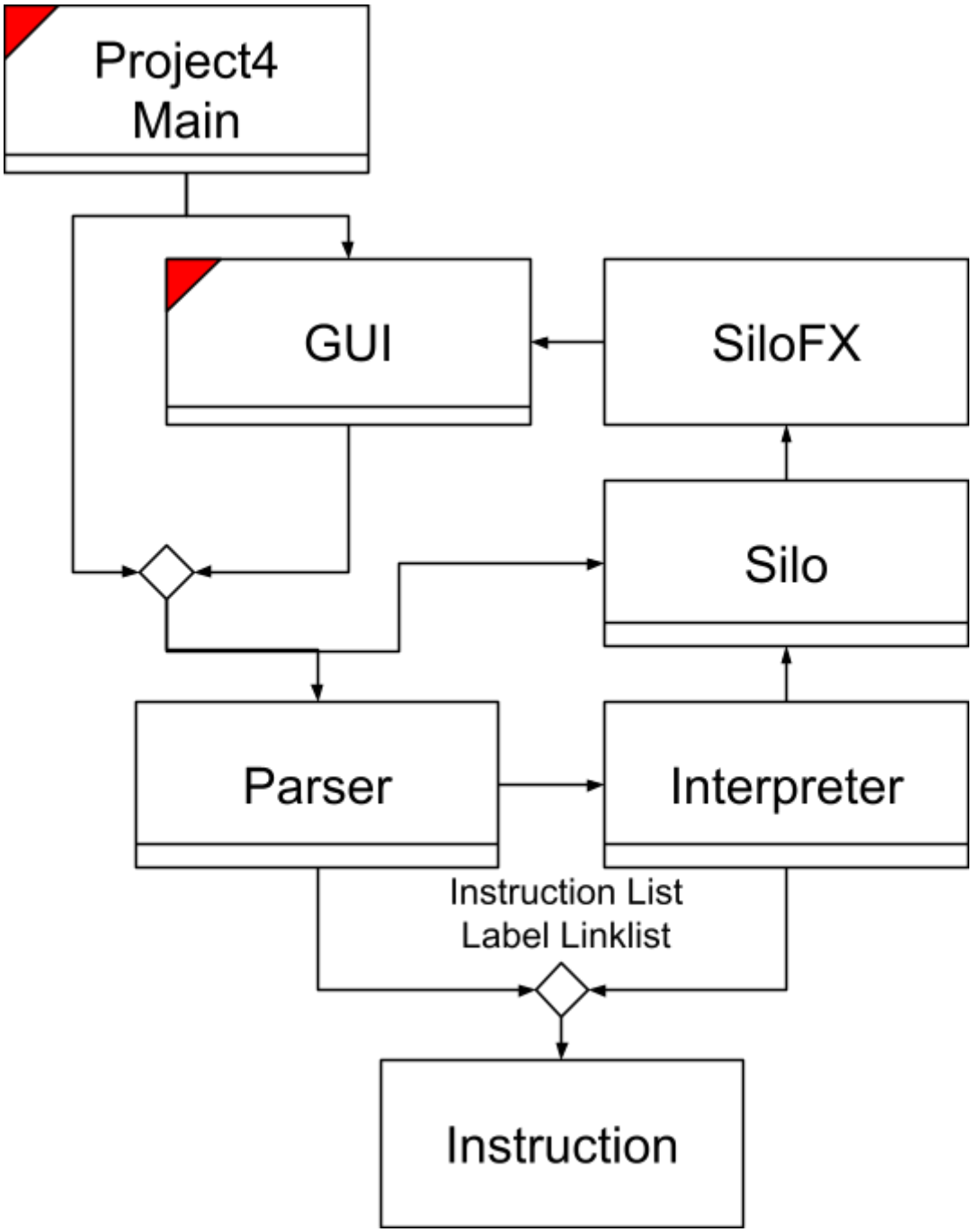
CS 351 Project 4 Group 1 Assembly Silos Design Doc

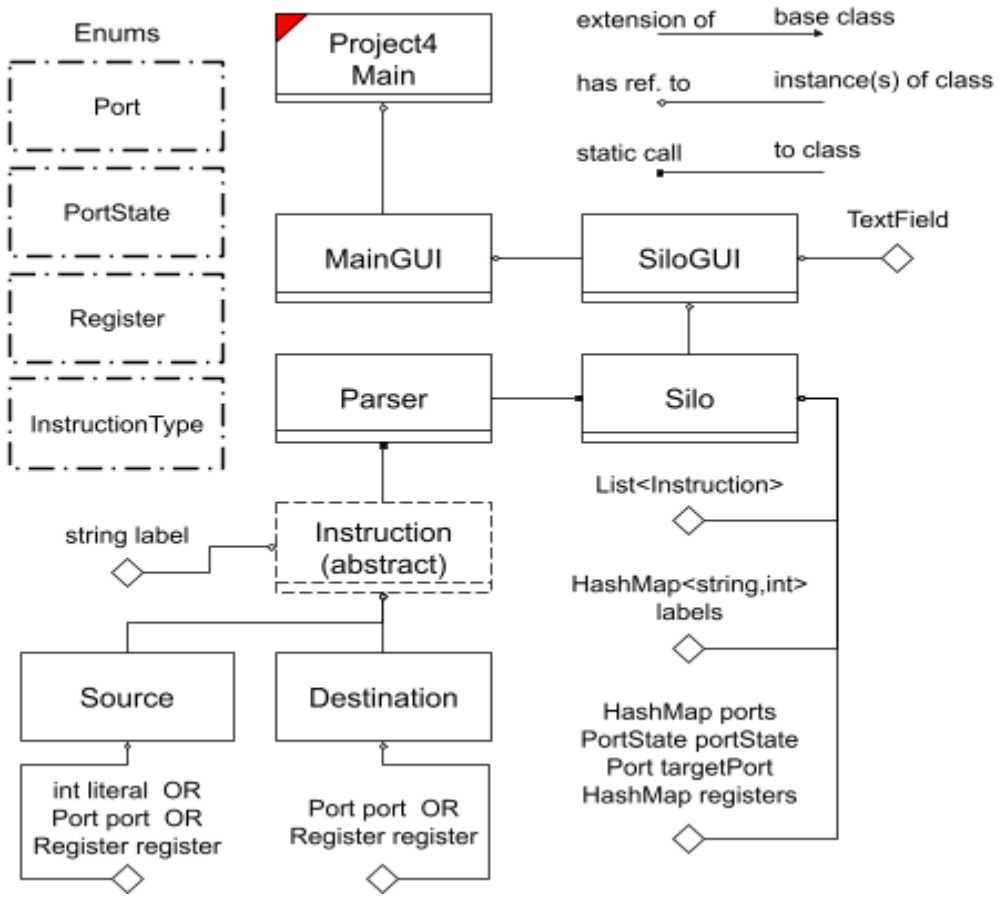
Carter Frost
Emely Seheon
Logan Sullivan

enum Ports{UP, RIGHT, DOWN, LEFT}
enum Registers{ACC, BAK, NIL}
enum Instructions{NOOP, MOVE, SWAP, SAVE, ADD, SUB, NEGATE, JUMP, JEZ, JNZ, JGZ, JLZ, JRO}

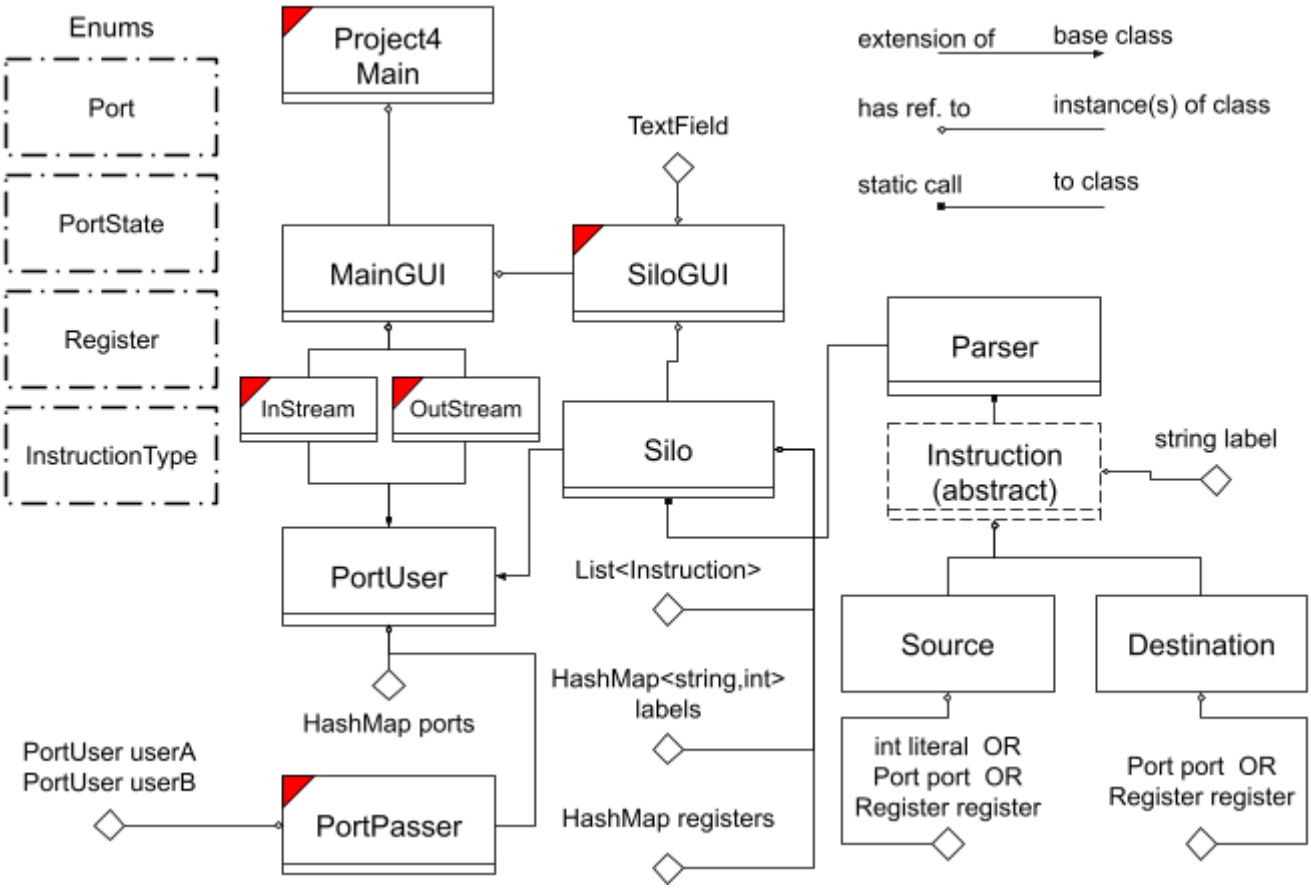
WIP design plan #1:



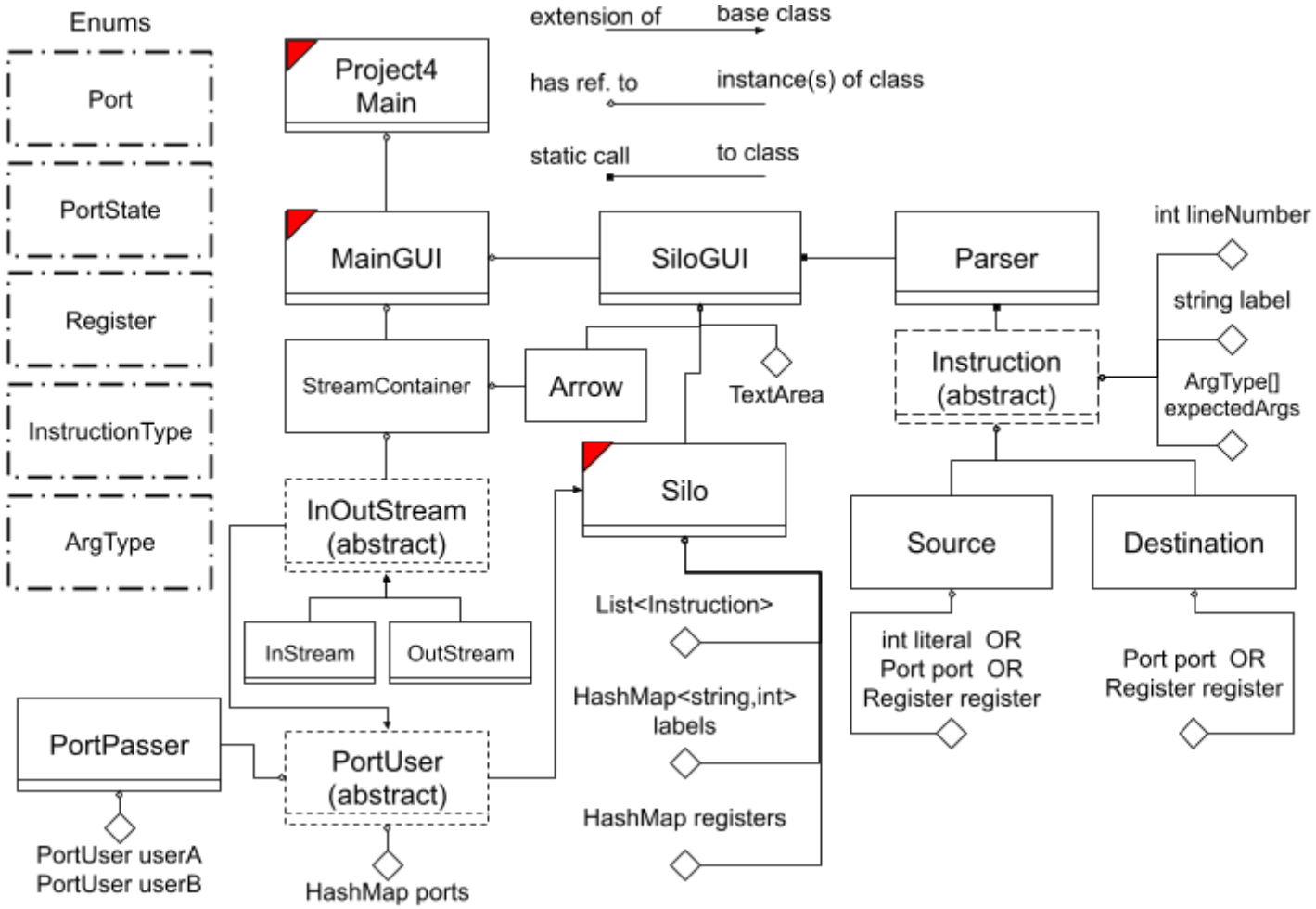
WIP design plan #2:



Chosen design plan (pre-implementation):



Final design plan (grows with implementation):



Extra notes:

- MainGUI is considered as active due to hosting the root Pane for the JavaFX scene, and managing the tick timer loop.
- Silo is where the separate threads are made, with each Silo having its own thread.
- PortPassers don't themselves contain a separate thread, but are thread safe in a way that allows for concurrency across all Silos.
    - Logical locks/state checks to ensure actions aren't marked as done until both sides are in cooperation (i.e. one is writing and the other is reading), and to prevent doubling up on actions.
    - Threading locks (ReentrantLock) to ensure thread safety.
    - A PortPasser represents one connection between two Silos.
    - Each 'transfer region' is concurrent because the read/write call comes from a Silo's thread, and it doesn't interrupt any unrelated Silos. The connected neighbor Silo is only interrupted if both try to use the PortPasser at the same time.
- PortUser ensures that it can only try to use (read/write) one port at a time out of its four.
- In/Out Streams don't need to have their own thread explicitly, just always read/write whenever their target port opens up while the game is playing. PortPasser will ensure thread safety and concurrency.
- Source and Destination are wrappers that handle getting/setting data across literal, port, and register values, and applying them to the context of a specific Silo.
    - Allows for Instructions to be easy and simple, and for success/fail checks to be automatically handled by the relevant context (e.g. registers always succeed while reading a port fails if the neighbor isn't writing)

- Destination will cache the raw int value it acquires from the given Source, and later remove said cache entry when appropriate. This prevents extra Source queries, such as what might occur when using MOVE PORT PORT.