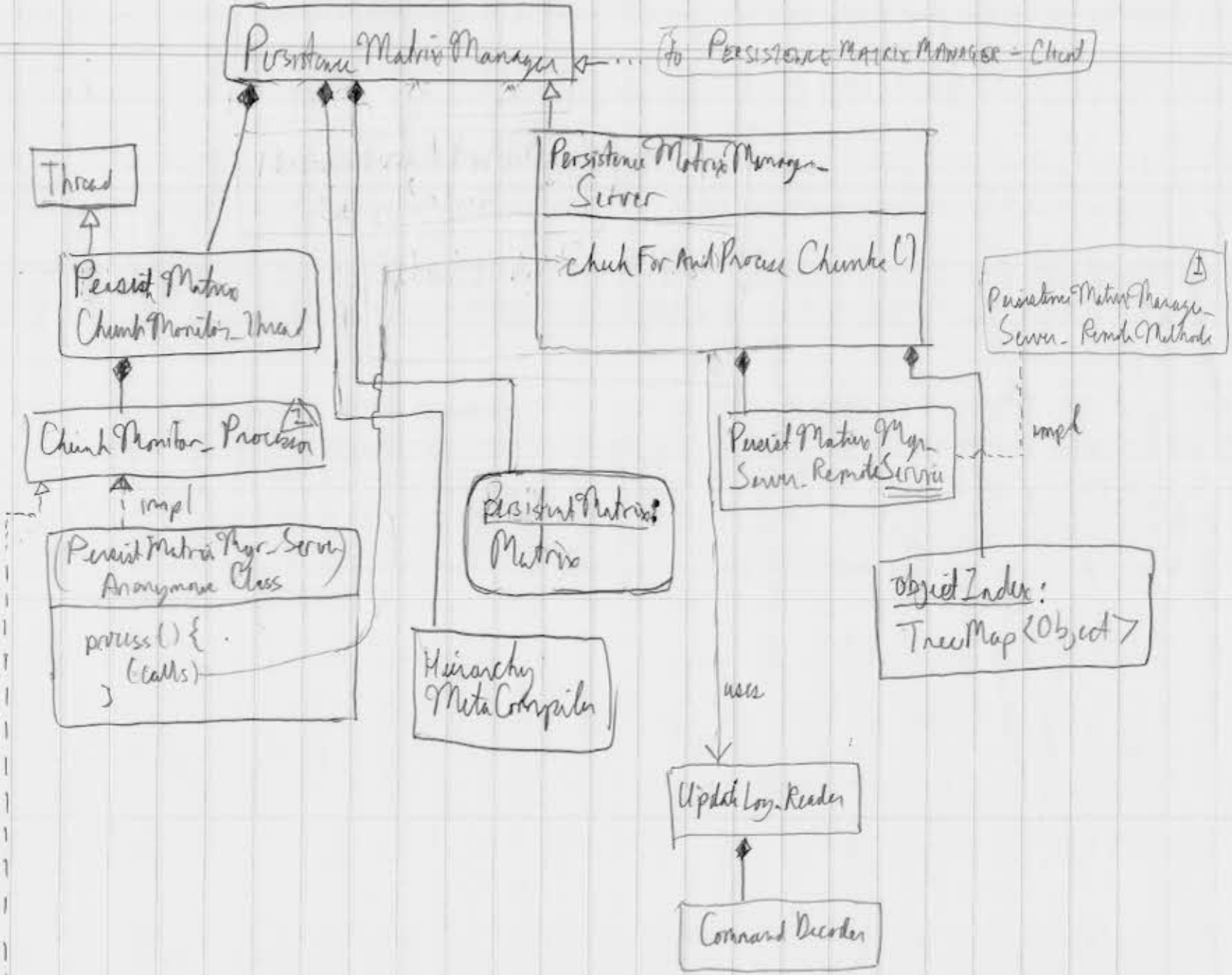
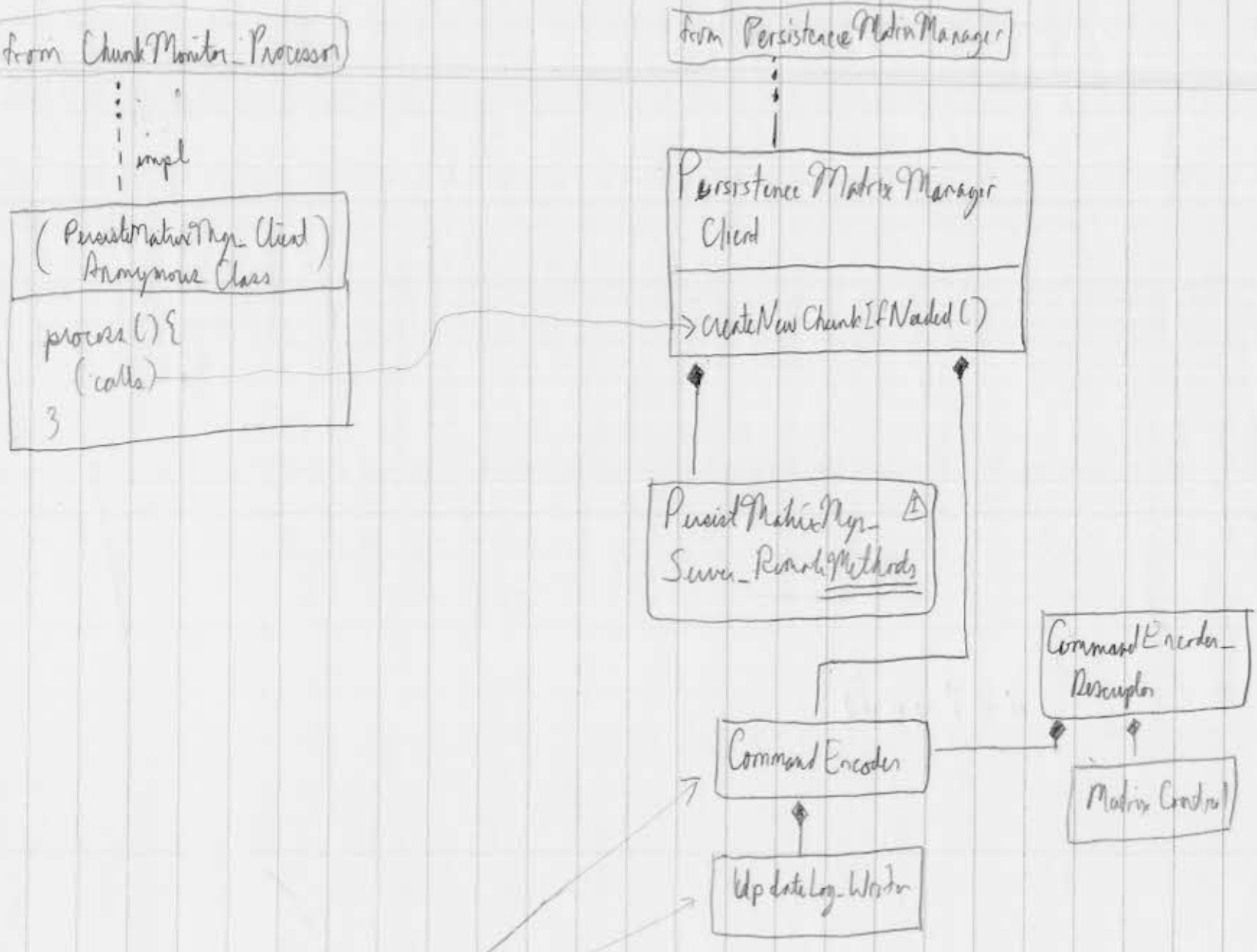
[Persistent Class Diagrams] 1 | Persistence Matrix Manager - Server

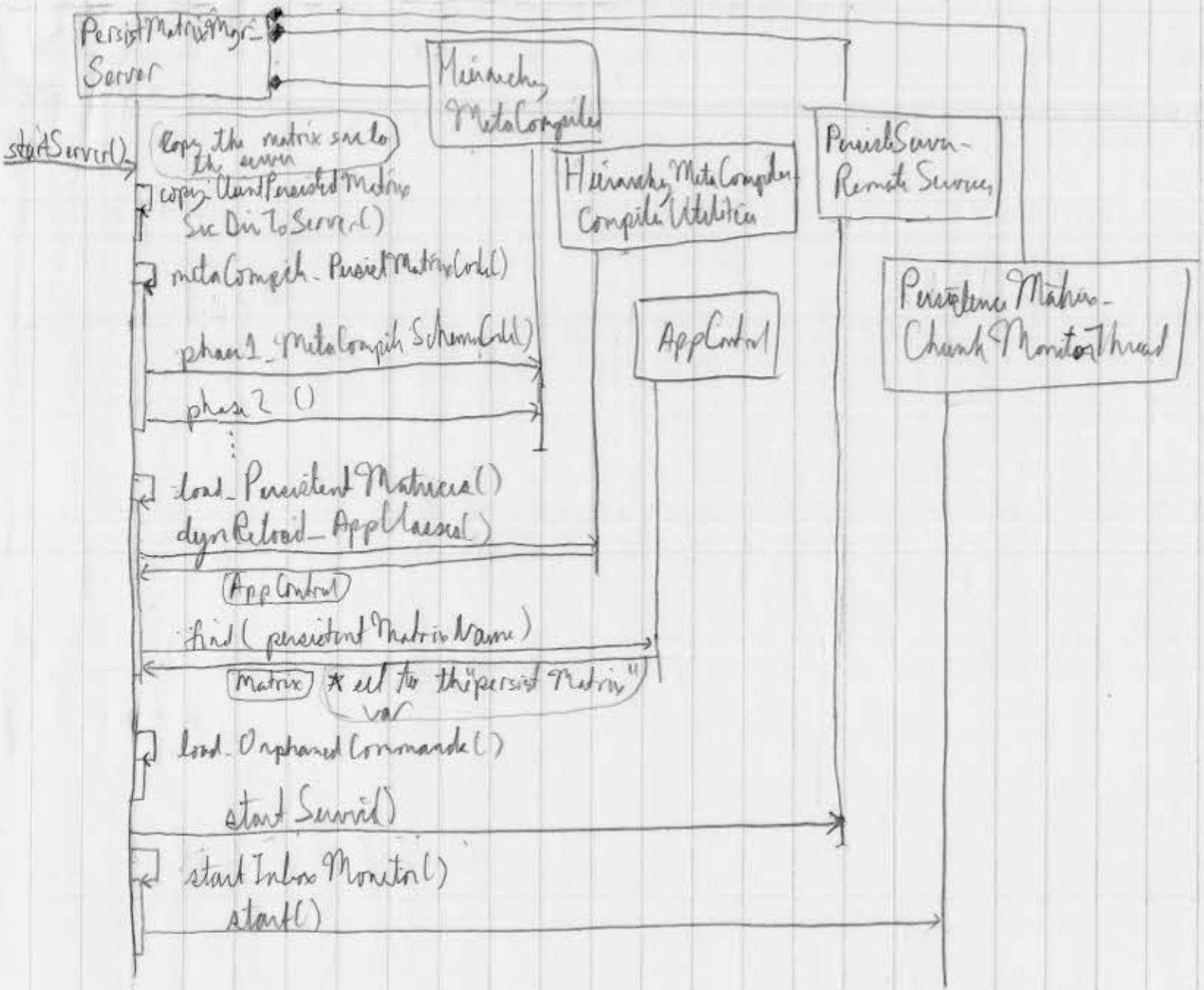


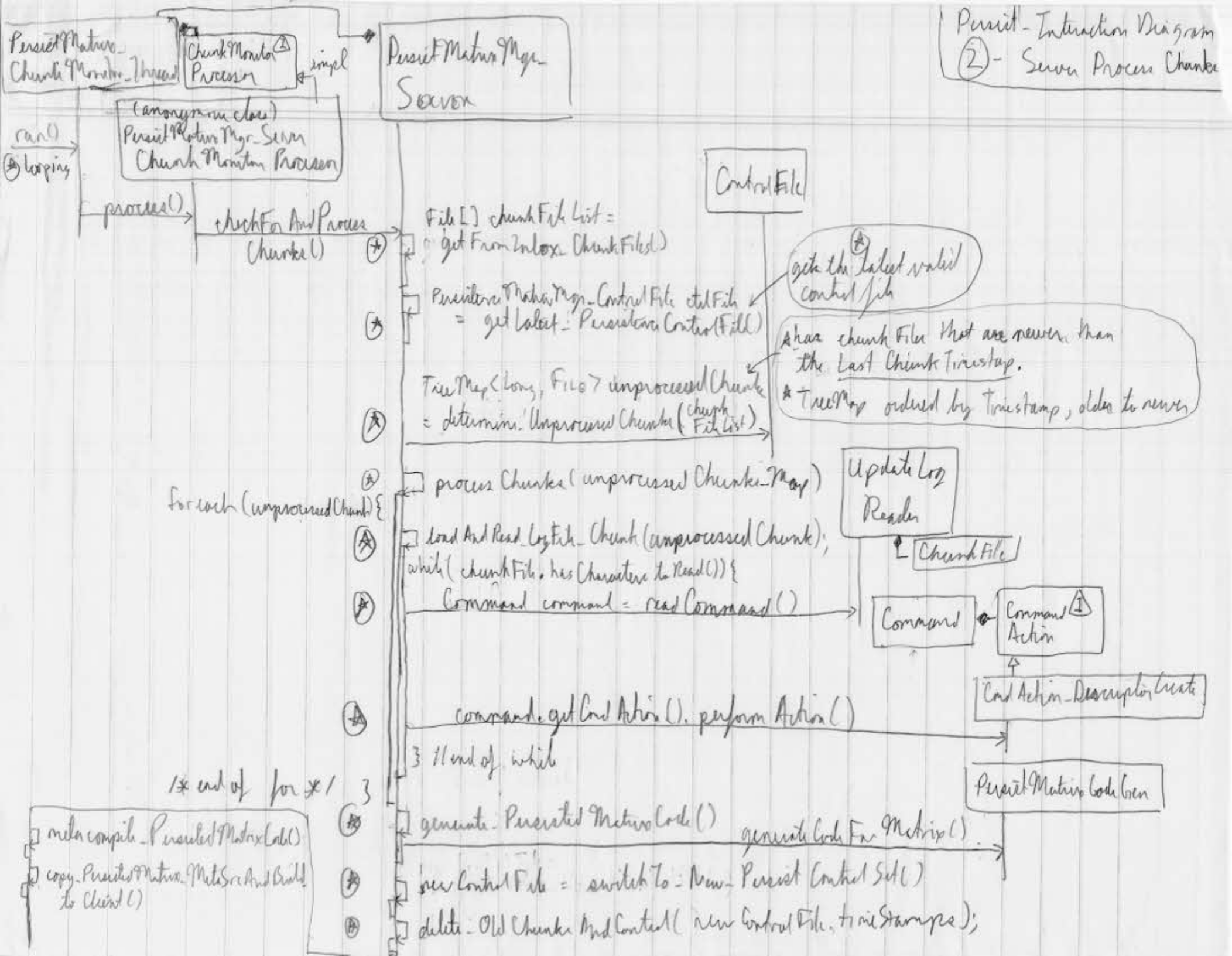
PERSISTENCE CLASS DIAGRAM | 2 | PERSISTENCE MATRIX MANAGER - CLIENT



☆ Note: The Client has a CommandEncoder which has an UpdateLog_Writer.
But the Server has an UpdateLog_Reader which has a CommandEncoder.
} This makes sense, because the client needs to first encode commands (which get written to the log), while the srv. reads the log (which then gets decoded)

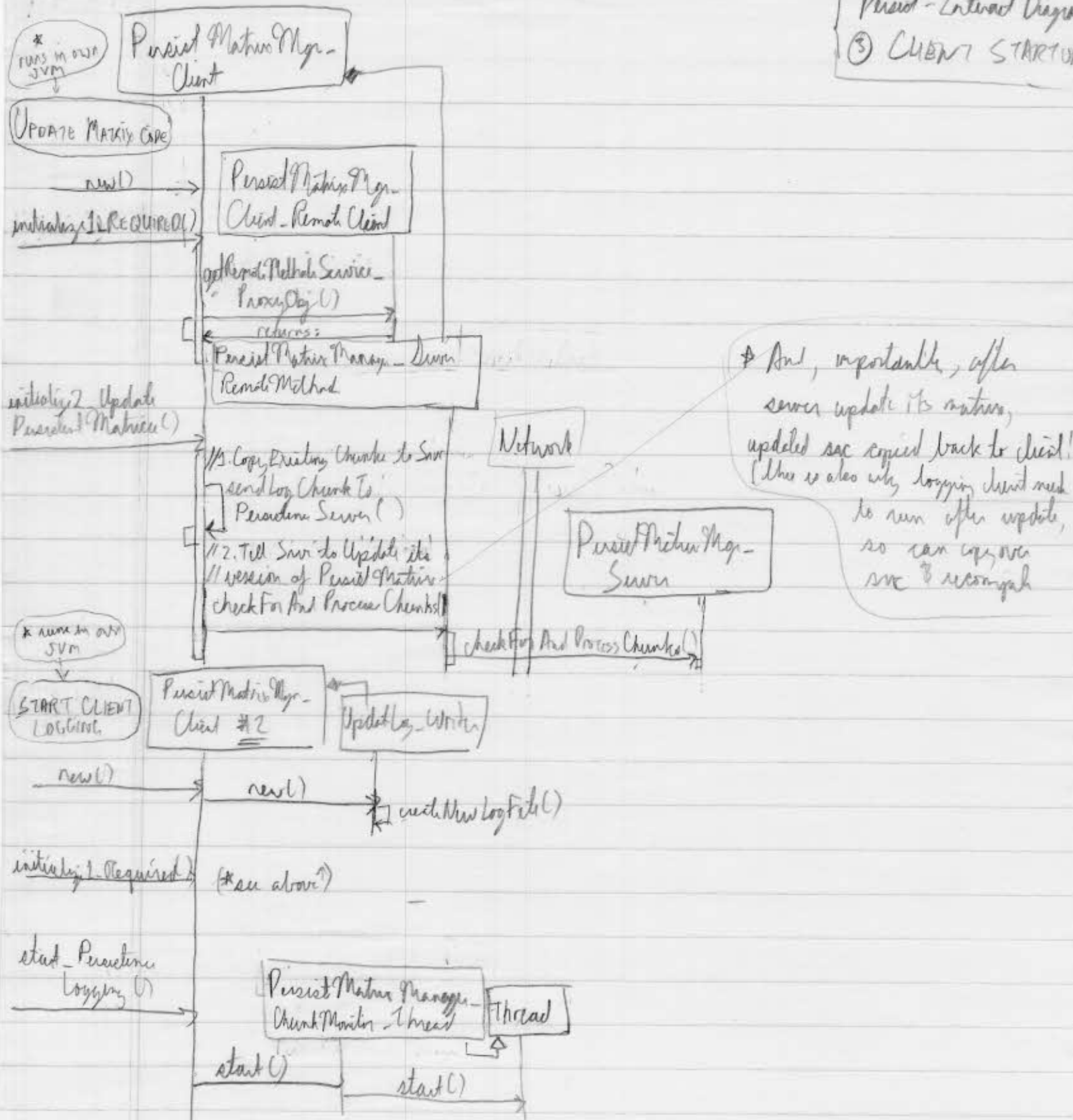
Persist - Interaction Diagram
(1) - Server Startup




Persist - Interaction Diagram (2) - Server Process Chunks
PersistMatrixMgr_Server
ChunkMonitorProcessor
impl
PersistMatrix_ChunkMonitor_Thread
run()
looping
(anonymous class) PersistMatrixMgr_Server ChunkMonitorProcessor
process()
checkForAndProcessChunks()
ControlFile
get the latest valid control file
File[] chunkFileList = getFromIndex_ChunkFiles()
PersistenceMatrixMgr_ControlFile ctlFile = getLatestPersistenceControlFile()
than chunk Files that are newer than the Last Chunk Timestamp.
* TreeMap ordered by Timestamp, older to newer
TreeMap<Long, File> unprocessedChunks = determineUnprocessedChunks(chunkFileList)
processChunks(unprocessedChunks_Map)
UpdateLogReader
ChunkFile
for each (unprocessedChunk) {
loadAndReadLogFile_Chunk(unprocessedChunk);
while(chunkFile.hasCharactersToRead()) {
Command command = readCommand()
Command
CommandAction
CmdAction_DescriptorCreate
command.getCmdAction().performAction()
} // end of while
/* end of for */ }
PersistMatrixCodeGen
generate_PersistedMatrixCode()
generateCodeForMatrix()
metaCompile_PersistedMatrixCode()
copy_PersistedMatrix_MetaSrcAndBuild to Client()
newControlFile = switchTo_NewPersistControlSet()
deleteOldChunksAndControl(newControlFile.timeStamp());

Persist - Interact Diagram

③ CLIENT STARTUP



PERSIST Interval DIAGRAM-
④ Client Write to Log