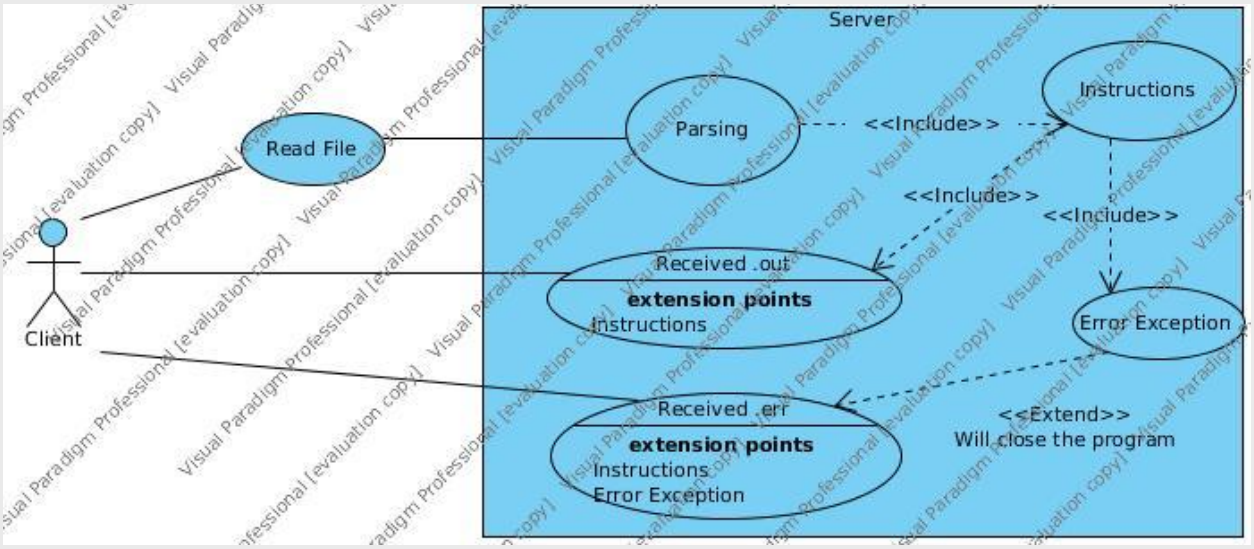
UML SURVEY AND DESIGN JUSTIFICATION – REPORT

SID GIDELA, DAVID LIANG, JULIUS MAZER

Jack Baskin School of Engineering, University of California, Santa Cruz, 95064

CASE DIAGRAM

Overall in the UML design, there was not much changes to the integrity of the program itself from the time the first phase was due. There were some significant changes mainly in sequence and class diagrams, but minor changes in the case diagram. To reiterate, in the case diagram, the general idea is that the client will read and check for trivial errors before sending to the server. In which, the server will read and perform the instructions. All the results will go into ".out" until an error occurs in which it would go into a ".err" file. The client will receive the two files when the program terminates. Interesting design to note is that error checking will be "include[d]" while the result of an error will be "exclude[d]." This stems from the fact that sending the result as an error is a condition instead of a mandatory notion.

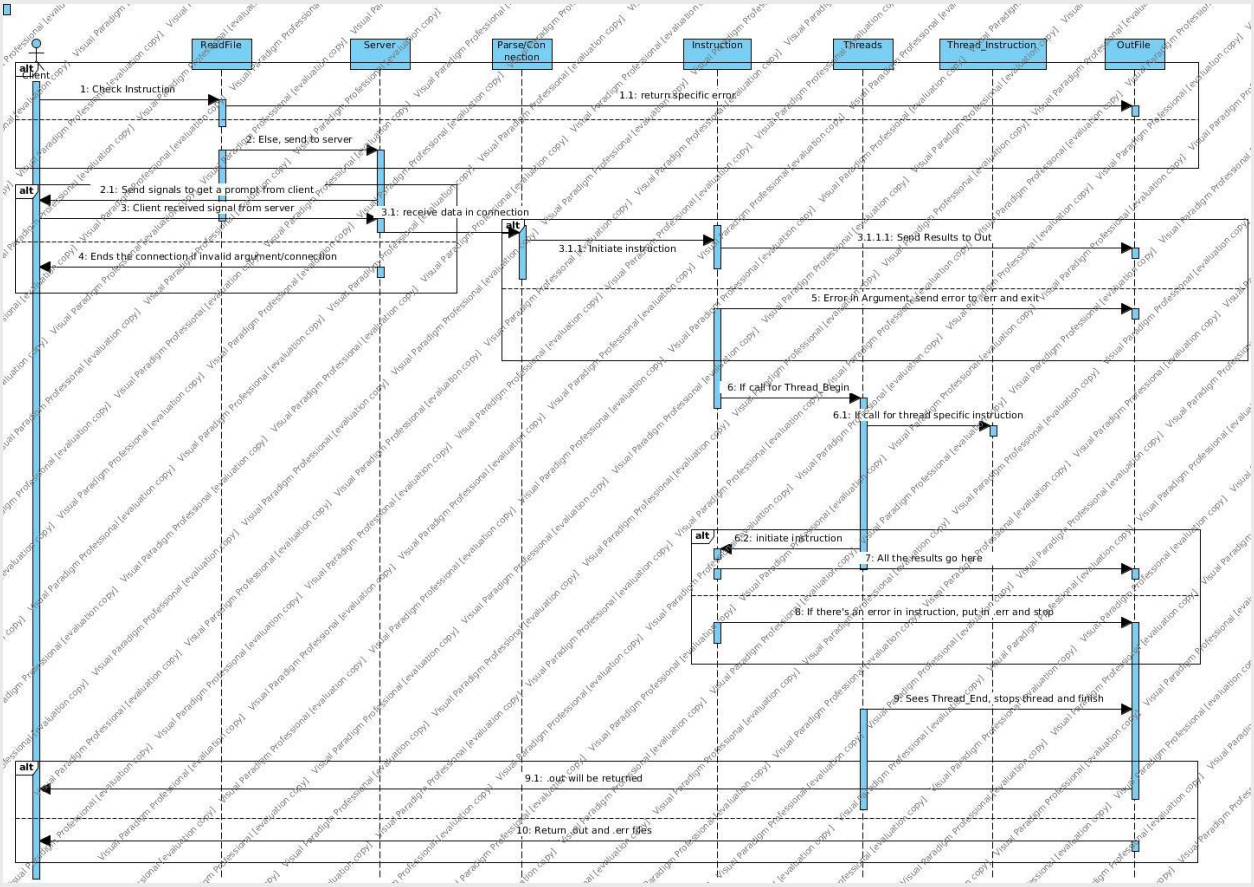


SEQUENCE DIAGRAM DISCUSSION

In the sequence diagram, it is a seemingly larger and more dynamic version of case diagram. The client will self-check the file (or files) before sending the server a signal with the packets. This server will response back to the client that it got the connection and the user will response back to inform the server it received the signal before it begins the execution. The server will then create a POSIX thread for each “Thread_Begin” it reads from the file. At the same time, if there are any instructions in between “Thread_End” of one part and “Thread_Begin” of another, it will begin the method for them with the main thread alongside the “Thread_Begin” thread. For each instruction, the threads go through, there are built-in error checking within all methods. If there is any error to be found, the program will exit with a specific error designating the location of said error to the user in “.err” file. Either until then or if there is no error, the program/threads will continue sending the results of each instruction into the output file called “.out” through the use of the standard sixteen instructions.

There are specific events for threads like “Thread_End”, “Thread_Begin”, “Unlock”, “Lock”, and “Barrier”. In “Thread_End”, since it marks the end of a certain thread of a section, it acts like a mini instruction from “connection.cpp.” In this case, it will send the results of the instruction into the “.out” file regardless of the positioning of the main thread. “Thread_End” marks the beginning of the thread in which case the client would send a certain list of instruction to that thread and be stored in a double vector. Unlock and lock depends on each other to a slight extent. “Lock” would shut off a variable of any sort from being accessed until an “unlock” commands it to be turn on. “Barrier” only works through the main program and it blocks the main program from finishing until all running threads are done. If there are no threads, the main program will continue as normal.

UML Design


Client
ReadFile
Server
Parse/Connection
Instruction
Threads
Thread_Instruction
OutFile
alt
1: Check Instruction
1.1: return specific error
2: Else, send to server
2.1: Send signals to get a prompt from client
3: Client received signal from server
3.1: receive data in connection
4: Ends the connection if invalid argument/connection
3.1.1: Initiate instruction
3.1.1.1: Send Results to Out
5: Error in Argument, send error to .err and exit
6: If call for Thread_Begin
6.1: If call for thread specific instruction
6.2: initiate instruction
7: All the results go here
8: If there's an error in instruction, put in .err and stop
9: Sees Thread_End, stops thread and finish
9.1: .out will be returned
10: Return .out and .err files

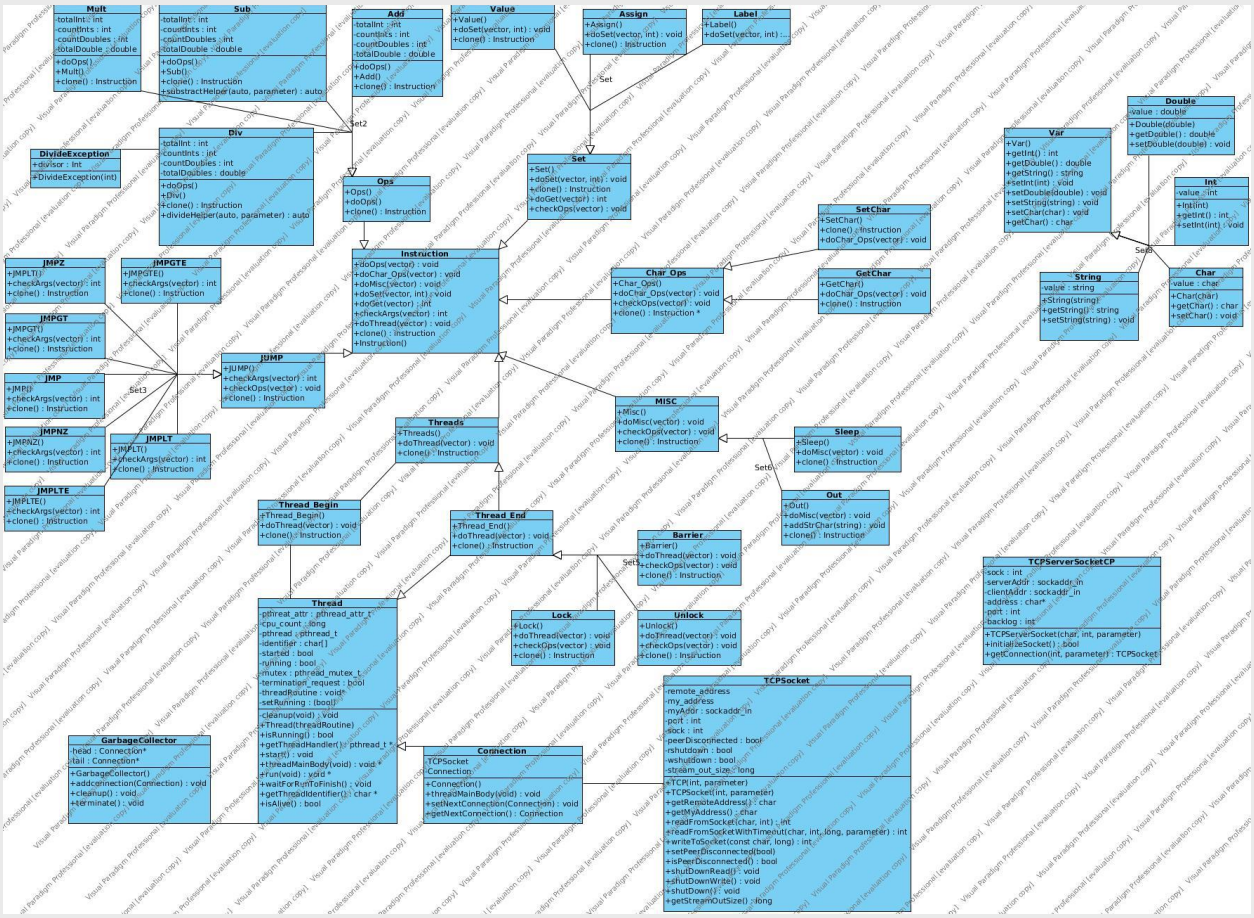
CLASS DIAGRAM DISCUSSION

In the Class diagram, there are the standard sixteen instructions from the first phase with an addition of five more instructions from the second phase. Asides from the standard instructions, there are a few new updates to the first phase. Mainly, there are seven jump functions to make it easier to read in terms of coding, removal of error exception handling classes to make the resulting errors more specific, and the addition of "VAR" class to centralized the data types into one object for the creation of one map. From the second phase, there are two main thread classes, "Threads" and "Thread." "Thread" inherits from the "Instruction" class, and "Threads" acts as a superclass for "Connection." They utilize POSIX threads instead of the C++ library ones. "Thread.h" depends on "Connection" class for it to connect to the server and the client and to manage multi-threads. All the four instructions excluding "Thread_End" will inherit "Thread_End" while it will inherit from both "Threads" and "Thread." The reasoning is that "Thread_End" has to perform the transaction of the results from "Instruction" and to take in multiple threads as directed from the user in "Connection". There are two socket classes that the "Connection" relies on. However, it does not rely on "TCPServerSocket" since "client.cpp" (another main.cpp) relies on that hence it being alone. They both allow the connection of server to client with the help of "Connection" to solidify it.

"Instruction" class is the main class that holds the five parents class of similar functions. "Var" class has its own set of classes they inherit from, and the TCP socket classes do not inherit as they are "helper" with the main program.

An interesting side note while designing the blueprints. There was no option to use destructors in each of the class nor was there any option to use a "virtual" placement word to signify that the function is virtual.

UML Design


Mult
-totalInt : int
-countInts : int
-countDoubles : int
-totalDouble : double
+doOps()
+Mult()
+clone() : Instruction
Sub
-totalInt : int
-countInts : int
-countDoubles : int
-totalDouble : double
+doOps()
+Sub()
+clone() : Instruction
+subtractHelper(auto, parameter) : auto
Add
-totalInt : int
-countInts : int
-countDoubles : int
-totalDouble : double
+doOps()
+Add()
+clone() : Instruction
Value
+Value()
+doSet(vector, int) : void
+clone() : Instruction
Assign
+Assign()
+doSet(vector, int) : void
+clone() : Instruction
Label
+Label()
+doSet(vector, int) : ...
Set
Set2
Div
-totalInt : int
-countInts : int
-countDoubles : int
-totalDoubles : double
+doOps()
+Div()
+clone() : Instruction
+divideHelper(auto, parameter) : auto
DivideException
-divisor : int
+DivideException(int)
Ops
+Ops()
+doOp()
+clone() : Instruction
Set
+Set()
+doSet(vector, int) : void
+clone() : Instruction
+doGet(vector) : int
+checkOps(vector) : void
Double
-value : double
+Double(double)
+getDouble() : double
+setDouble(double) : void
Var
+Var()
+getInt() : int
+getDouble() : double
+getString() : string
+setInt(int) : void
+setDouble(double) : void
+setString(string) : void
+setChar(char) : void
+getChar() : char
Int
-value : int
+Int(int)
+getInt() : int
+setInt(int) : void
Set3
JMPZ
+JMPLT()
+checkArgs(vector) : int
+clone() : Instruction
JMPGTE
+JMPGTE()
+checkArgs(vector) : int
+clone() : Instruction
JMPGT
+JMPGT()
+checkArgs(vector) : int
+clone() : Instsruction
JMP
+JMP()
+checkArgs(vector) : int
+clone() : Instruction
JMPNZ
+JMPNZ()
+checkArgs(vector) : int
+clone() : Instruction
JMPLTE
+JMPLTE()
+checkArgs(vector) : int
+clone() : Instruction
JMPLT
+JMPLT()
+checkArgs(vector) : int
+clone() : Instruction
JUMP
+JUMP()
+checkArgs(vector) : int
+checkOps(vector) : void
+clone() : Instruction
Instruction
+doOps(vector) : void
+doChar_Ops(vector) : void
+doMisc(vector) : void
+doSet(vector, int) : void
+doGet(vector) : int
+checkArgs(vector) : int
+doThread(vector) : void
+clone() : Instruction
+Instruction()
Char_Ops
+Char_Ops()
+doChar_Ops(vector) : void
+checkOps(vector) : void
+clone() : Instruction *
SetChar
+SetChar()
+clone() : Instruction
+doChar_Ops(vector) : void
GetChar
+GetChar()
+doChar_Ops(vector) : void
+clone() : Instruction
String
-value : string
+String(string)
+getString() : string
+setString(string) : void
Char
-value : char
+Char(char)
+getChar() : char
+setChar() : void
Set6
Threads
+Threads()
+doThread(vector) : void
+clone() : Instruction
MISC
+Misc()
+doMisc(vector) : void
+checkOps(vector) : void
+clone() : Instruction
Sleep
+Sleep()
+doMisc(vector) : void
+clone() : Instruction
Set5
Out
+Out()
+doMisc(vector) : void
+addStrChar(string) : void
+clone() : Instruction
Thread_Begin
+Thread_Begin()
+doThread(vector) : void
+clone() : Instruction
Thread_End
+Thread_End()
+doThread(vector) : void
+clone() : Instruction
Barrier
+Barrier()
+doThread(vector) : void
+checkOps(vector) : void
+clone() : Instruction
Set4
TCPServerSocketCP
-sock : int
-serverAddr : sockaddr_in
-clientAddr : sockaddr_in
-address : char*
-port : int
-backlog : int
+TCPServerSocket(char, int, parameter)
+initializeSocket() : bool
+getConnection(int, parameter) : TCPSocket
Thread
-pthread_attr : pthread_attr_t
-cpu_count : long
-pthread : pthread_t
-identifier : char[]
-started : bool
-running : bool
-mutex : pthread_mutex_t
-termination_request : bool
-threadRoutine : void*
-setRunning : (bool)
-cleanup(void) : void
+TThread(threadRoutine)
+isRunning() : bool
+getThreadHandler() : pthread_t *
+start() : void
+threadMainBody(void) : void *
+Fun(void) : void *
+waitForRunToFinish() : void
+getThreadIdentifier() : char *
+isAlive() : bool
Lock
+Lock()
+doThread(vector) : void
+checkOps(vector) : void
+clone() : Instruction
Unlock
+Unlock()
+doThread(vector) : void
+checkOps(vector) : void
+clone() : Instruction
TCPSocket
-remote_address
-my_address
-myAddr : sockaddr_in
-port : int
-sock : int
-peerDisconnected : bool
-rshutdown : bool
-wshutdown : bool
-stream_out_size : long
+TCP(int, parameter)
+TCPSocket(int, parameter)
+getRemoteAddress() : char
+getMyAddress() : char
+readFromSocket(char, int) : int
+readFromSocketWithTimeout(char, int, long, parameter) : int
+writeToSocket(const char, long) : int
+setPeerDisconnected(bool)
+isPeerDisconnected() : bool
+shutDownRead() : void
+shutDownWrite() : void
+shutDown() : void
+getStreamOutSize() : long
GarbageCollector
-head : Connection*
-tail : Connection*
+GarbageCollector()
+addConnection(Connection) : void
+cleanup() : void
+terminate() : void
Connection
TCPSocket
Connection
+Connection()
+threadMainBody(void) : void
+setNextConnection(Connection) : void
+getNextConnection() : Connection