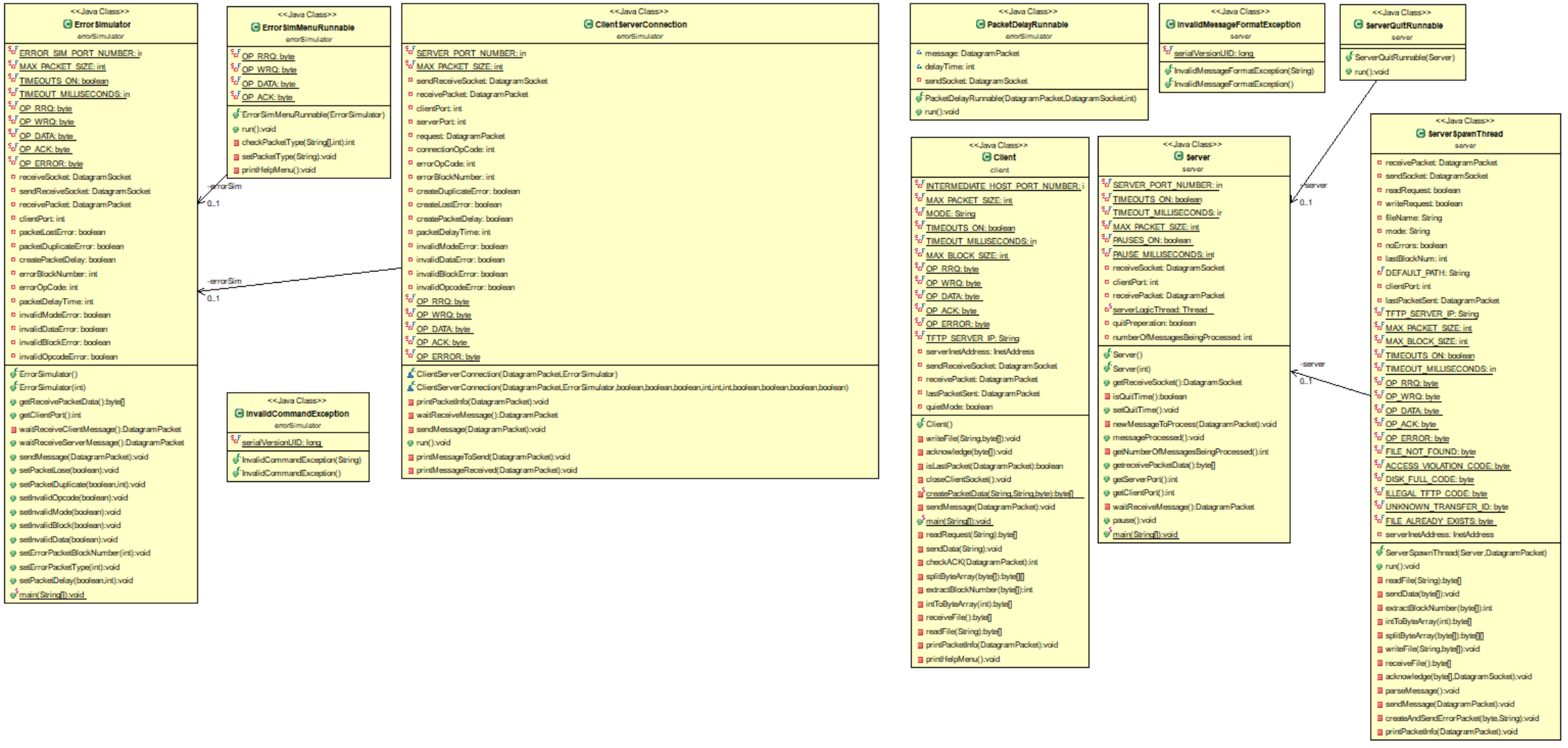
<<Java Class>>
ErrorSimulator
errorSimulator
ERROR_SIM_PORT_NUMBER: in
MAX_PACKET_SIZE: int
TIMEOUTS_ON: boolean
TIMEOUT_MILLISECONDS: in
OP_RRQ: byte
OP_WRQ: byte
OP_DATA: byte
OP_ACK: byte
OP_ERROR: byte
receiveSocket: DatagramSocket
sendReceiveSocket: DatagramSocket
receivePacket: DatagramPacket
clientPort: int
packetLostError: boolean
packetDuplicateError: boolean
createPacketDelay: boolean
errorBlockNumber: int
errorOpCode: int
packetDelayTime: int
invalidModeError: boolean
invalidDataError: boolean
invalidBlockError: boolean
invalidOpcodeError: boolean
ErrorSimulator()
ErrorSimulator(int)
getReceivePacketData():byte[]
getClientPort():int
waitReceiveClientMessage():DatagramPacket
waitReceiveServerMessage():DatagramPacket
sendMessage(DatagramPacket):void
setPacketLose(boolean):void
setPacketDuplicate(boolean,int):void
setInvalidOpcode(boolean):void
setInvalidMode(boolean):void
setInvalidBlock(boolean):void
setInvalidData(boolean):void
setErrorPacketBlockNumber(int):void
setErrorPacketType(int):void
setPacketDelay(boolean,int):void
main(String[]):void
<<Java Class>>
ErrorSimMenuRunnable
errorSimulator
OP_RRQ: byte
OP_WRQ: byte
OP_DATA: byte
OP_ACK: byte
ErrorSimMenuRunnable(ErrorSimulator)
run():void
checkPacketType(String[],int):int
setPacketType(String):void
printHelpMenu():void
-errorSim
0..1
<<Java Class>>
ClientServerConnection
errorSimulator
SERVER_PORT_NUMBER: in
MAX_PACKET_SIZE: int
sendReceiveSocket: DatagramSocket
receivePacket: DatagramPacket
clientPort: int
serverPort: int
request: DatagramPacket
connectionOpCode: int
errorOpCode: int
errorBlockNumber: int
createDuplicateError: boolean
createLostError: boolean
createPacketDelay: boolean
packetDelayTime: int
invalidModeError: boolean
invalidDataError: boolean
invalidBlockError: boolean
invalidOpcodeError: boolean
OP_RRQ: byte
OP_WRQ: byte
OP_DATA: byte
OP_ACK: byte
OP_ERROR: byte
ClientServerConnection(DatagramPacket,ErrorSimulator)
ClientServerConnection(DatagramPacket,ErrorSimulator,boolean,boolean,boolean,int,int,int,boolean,boolean,boolean,boolean)
printPacketInfo(DatagramPacket):void
waitReceiveMessage():DatagramPacket
sendMessage(DatagramPacket):void
run():void
printMessageToSend(DatagramPacket):void
printMessageReceived(DatagramPacket):void
-errorSim
0..1
<<Java Class>>
InvalidCommandException
errorSimulator
serialVersionUID: long
InvalidCommandException(String)
InvalidCommandException()
<<Java Class>>
PacketDelayRunnable
errorSimulator
message: DatagramPacket
delayTime: int
sendSocket: DatagramSocket
PacketDelayRunnable(DatagramPacket,DatagramSocket,int)
run():void
<<Java Class>>
InvalidMessageFormatException
server
serialVersionUID: long
InvalidMessageFormatException(String)
InvalidMessageFormatException()
<<Java Class>>
ServerQuitRunnable
server
ServerQuitRunnable(Server)
run():void
<<Java Class>>
Client
client
INTERMEDIATE_HOST_PORT_NUMBER: i
MAX_PACKET_SIZE: int
MODE: String
TIMEOUTS_ON: boolean
TIMEOUT_MILLISECONDS: in
MAX_BLOCK_SIZE: int
OP_RRQ: byte
OP_WRQ: byte
OP_DATA: byte
OP_ACK: byte
OP_ERROR: byte
TFTP_SERVER_IP: String
serverInetAddress: InetAddress
sendReceiveSocket: DatagramSocket
receivePacket: DatagramPacket
lastPacketSent: DatagramPacket
quietMode: boolean
Client()
writeFile(String,byte[]):void
acknowledge(byte[]):void
isLastPacket(DatagramPacket):boolean
closeClientSocket():void
createPacketData(String,String,byte):byte[]
sendMessage(DatagramPacket):void
main(String[]):void
readRequest(String):byte[]
sendData(String):void
checkACK(DatagramPacket):int
splitByteArray(byte[]):byte[][]
extractBlockNumber(byte[]):int
intToByteArray(int):byte[]
receiveFile():byte[]
readFile(String):byte[]
printPacketInfo(DatagramPacket):void
printHelpMenu():void
<<Java Class>>
Server
server
SERVER_PORT_NUMBER: in
TIMEOUTS_ON: boolean
TIMEOUT_MILLISECONDS: in
MAX_PACKET_SIZE: int
PAUSES_ON: boolean
PAUSE_MILLISECONDS: int
receiveSocket: DatagramSocket
clientPort: int
receivePacket: DatagramPacket
serverLogicThread: Thread
quitPreperation: boolean
numberOfMessagesBeingProcessed: int
Server()
Server(int)
getReceiveSocket():DatagramSocket
isQuitTime():boolean
setQuitTime():void
newMessageToProcess(DatagramPacket):void
messageProcessed():void
getNumberOfMessagesBeingProcessed():int
getreceivePacketData():byte[]
getServerPort():int
getClientPort():int
waitReceiveMessage():DatagramPacket
pause():void
main(String[]):void
-server
0..1
-server
0..1
<<Java Class>>
Server$spawnThread
server
receivePacket: DatagramPacket
sendSocket: DatagramSocket
readRequest: boolean
writeRequest: boolean
fileName: String
mode: String
noErrors: boolean
lastBlockNum: int
DEFAULT_PATH: String
clientPort: int
lastPacketSent: DatagramPacket
TFTP_SERVER_IP: String
MAX_PACKET_SIZE: int
MAX_BLOCK_SIZE: int
TIMEOUTS_ON: boolean
TIMEOUT_MILLISECONDS: in
OP_RRQ: byte
OP_WRQ: byte
OP_DATA: byte
OP_ACK: byte
OP_ERROR: byte
FILE_NOT_FOUND: byte
ACCESS_VIOLATION_CODE: byte
DISK_FULL_CODE: byte
ILLEGAL_TFTP_CODE: byte
UNKNOWN_TRANSFER_ID: byte
FILE_ALREADY_EXISTS: byte
serverInetAddress: InetAddress
Server$spawnThread(Server,DatagramPacket)
run():void
readFile(String):byte[]
sendData(byte[]):void
extractBlockNumber(byte[]):int
intToByteArray(int):byte[]
splitByteArray(byte[]):byte[][]
writeFile(String,byte[]):void
receiveFile():byte[]
acknowledge(byte[],DatagramSocket):void
parseMessage():void
sendMessage(DatagramPacket):void
createAndSendErrorPacket(byte,String):void
printPacketInfo(DatagramPacket):void