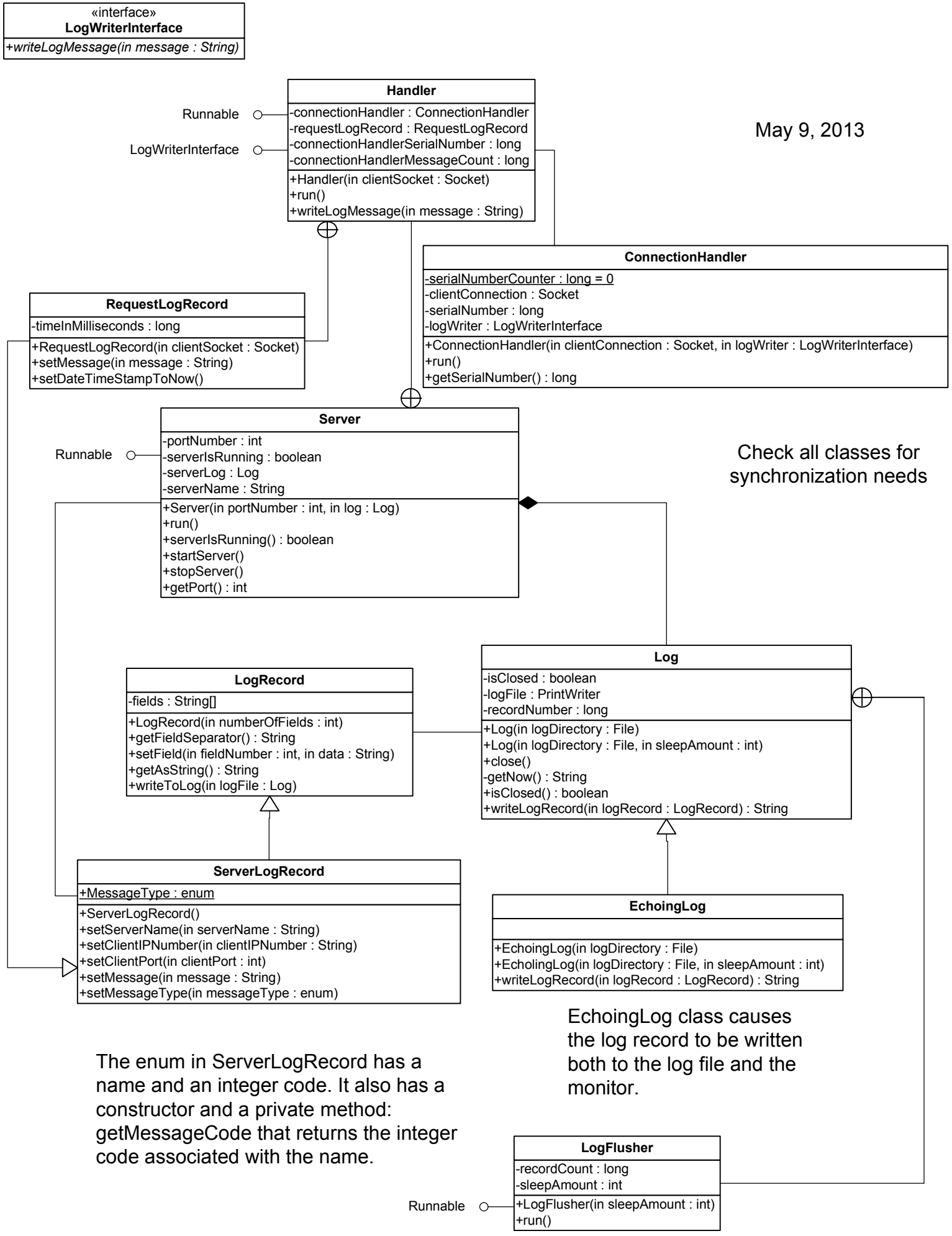
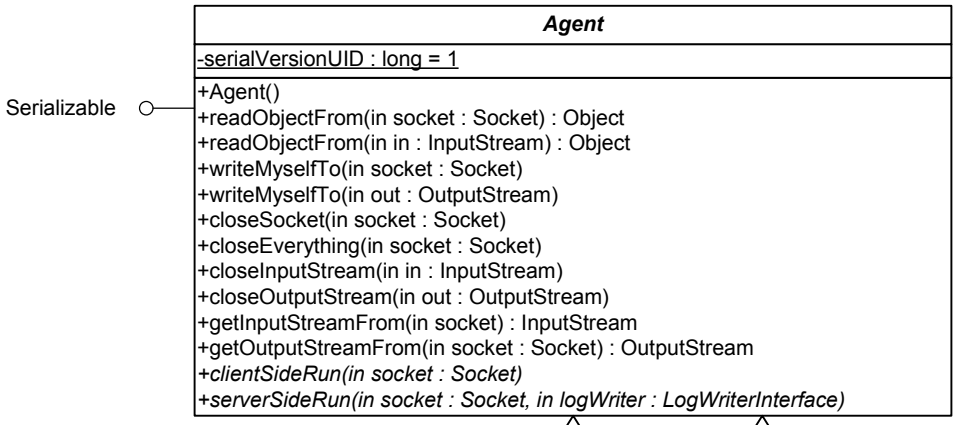
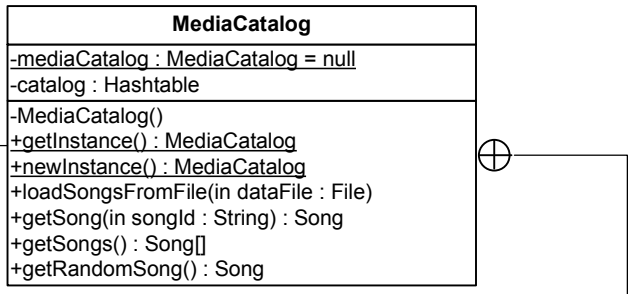

«interface»
LogWriterInterface
+writeLogMessage(in message : String)
Handler
-connectionHandler : ConnectionHandler
-requestLogRecord : RequestLogRecord
-connectionHandlerSerialNumber : long
-connectionHandlerMessageCount : long
+Handler(in clientSocket : Socket)
+run()
+writeLogMessage(in message : String)
Runnable
LogWriterInterface
May 9, 2013
ConnectionHandler
-serialNumberCounter : long = 0
-clientConnection : Socket
-serialNumber : long
-logWriter : LogWriterInterface
+ConnectionHandler(in clientConnection : Socket, in logWriter : LogWriterInterface)
+run()
+getSerialNumber() : long
RequestLogRecord
-timeInMilliseconds : long
+RequestLogRecord(in clientSocket : Socket)
+setMessage(in message : String)
+setDateTimeStampToNow()
Server
-portNumber : int
-serverIsRunning : boolean
-serverLog : Log
-serverName : String
+Server(in portNumber : int, in log : Log)
+run()
+serverIsRunning() : boolean
+startServer()
+stopServer()
+getPort() : int
Runnable
Check all classes for synchronization needs
Log
-isClosed : boolean
-logFile : PrintWriter
-recordNumber : long
+Log(in logDirectory : File)
+Log(in logDirectory : File, in sleepAmount : int)
+close()
-getNow() : String
+isClosed() : boolean
+writeLogRecord(in logRecord : LogRecord) : String
LogRecord
-fields : String[]
+LogRecord(in numberOfFields : int)
+getFieldSeparator() : String
+setField(in fieldNumber : int, in data : String)
+getAsString() : String
+writeToLog(in logFile : Log)
ServerLogRecord
+MessageType : enum
+ServerLogRecord()
+setServerName(in serverName : String)
+setClientIPNumber(in clientIPNumber : String)
+setClientPort(in clientPort : int)
+setMessage(in message : String)
+setMessageType(in messageType : enum)
EchoingLog
+EchoingLog(in logDirectory : File)
+EcholingLog(in logDirectory : File, in sleepAmount : int)
+writeLogRecord(in logRecord : LogRecord) : String
EchoingLog class causes the log record to be written both to the log file and the monitor.
The enum in ServerLogRecord has a name and an integer code. It also has a constructor and a private method: getMessageCode that returns the integer code associated with the name.
LogFlusher
-recordCount : long
-sleepAmount : int
+LogFlusher(in sleepAmount : int)
+run()
Runnable

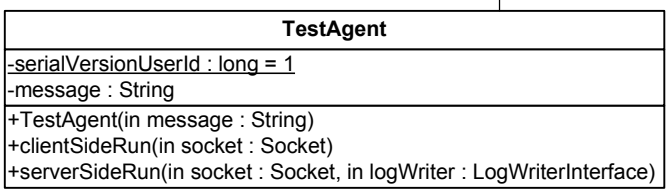
All methods in the Agent class must advertise the applicable exceptions except the close methods and the abstract methods. The close methods and abstract methods do not advertise.

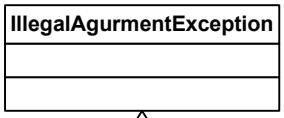


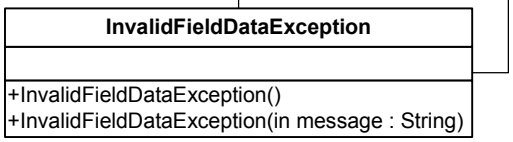
May 9, 2013

MediaCatalog is implemented as a Singleton

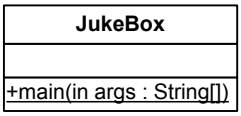


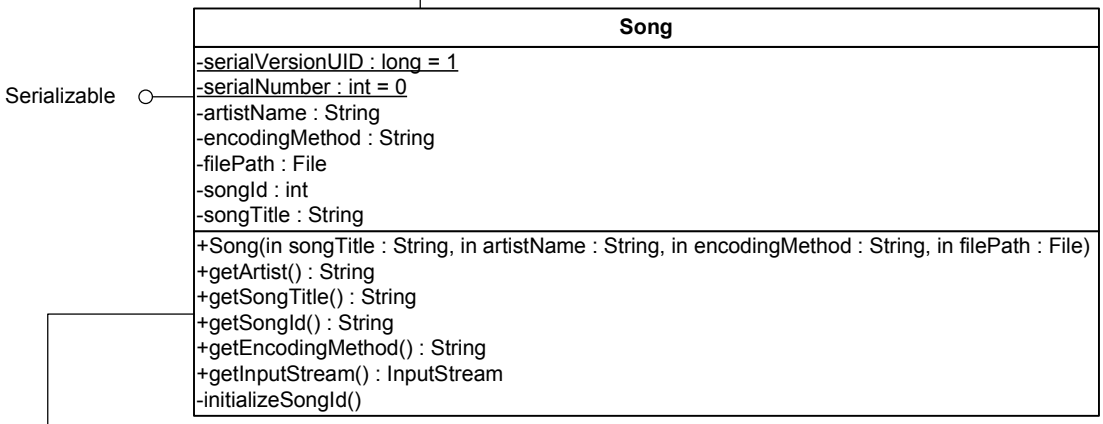




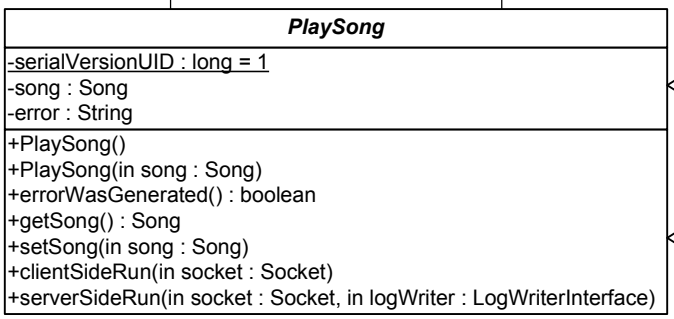


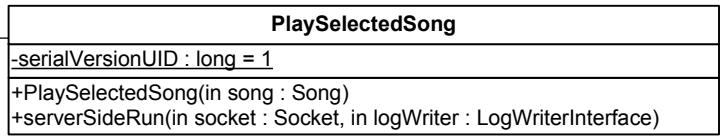
JukeBox is the driver for the Server end of the system. Based on the values in the xml Properites file, it starts the server and log.



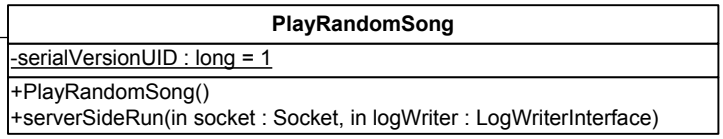


The file containing the song information has one record per song. Each record contains the following fields: song title, artist name, encoding (WAV. MP3, etc), and file path



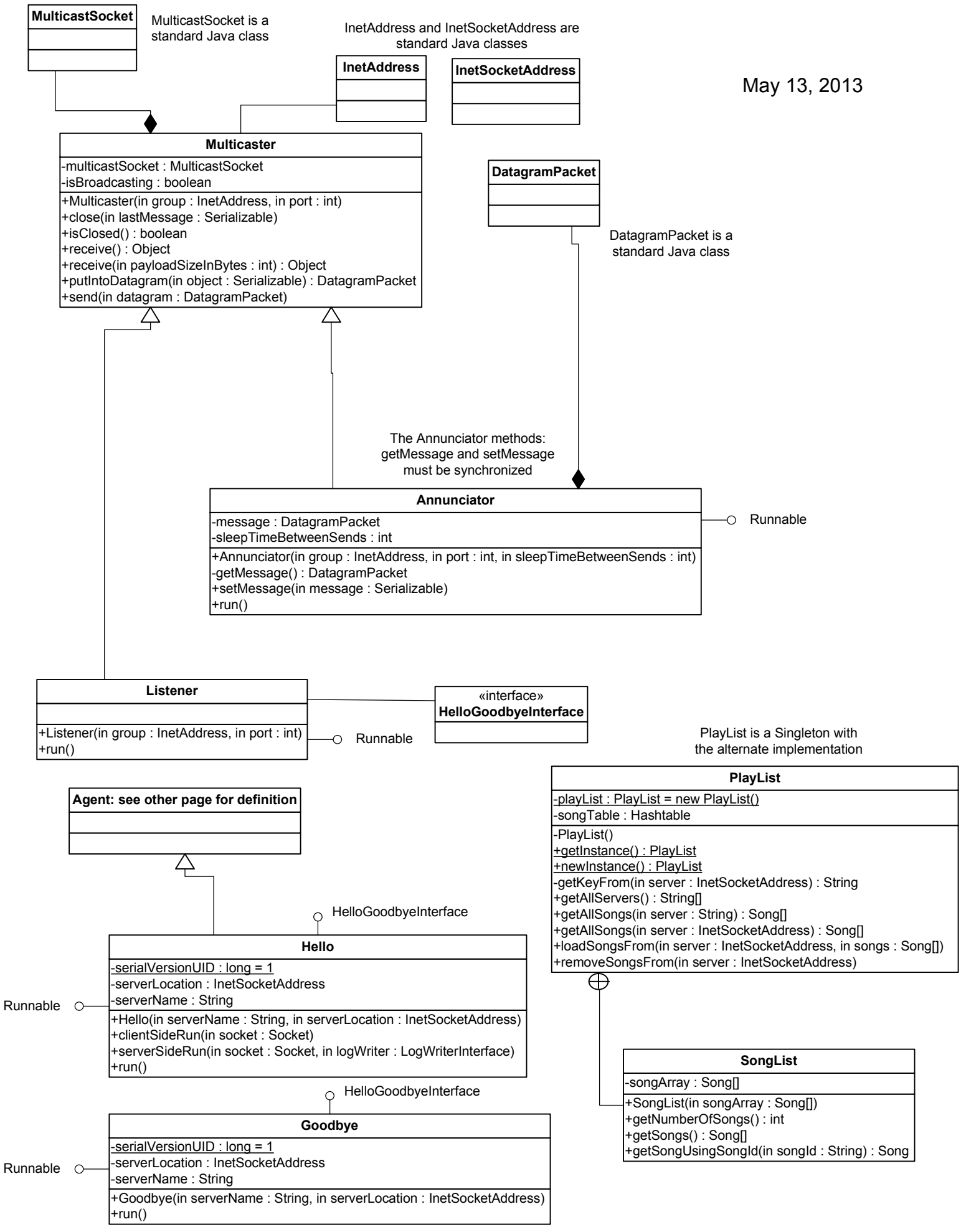


Overriding method in PlaySelectedSong calls super.setSong passing it the matching song from the MediaCatalog and then calls the super class serverSideRun method.



See clientSideRun and serverSideRun flowcharts

Overriding method in PlayRandomSong calls super.setSong passing it a random song from the MediaCatalog and then calls the super class serverSideRun method.


MulticastSocket
MulticastSocket is a standard Java class
InetAddress and InetSocketAddress are standard Java classes
InetAddress
InetSocketAddress
May 13, 2013
Multicaster
-multicastSocket : MulticastSocket
-isBroadcasting : boolean
+Multicaster(in group : InetAddress, in port : int)
+close(in lastMessage : Serializable)
+isClosed() : boolean
+receive() : Object
+receive(in payloadSizeInBytes : int) : Object
+putIntoDatagram(in object : Serializable) : DatagramPacket
+send(in datagram : DatagramPacket)
DatagramPacket
DatagramPacket is a standard Java class
The Annunciator methods: getMessage and setMessage must be synchronized
Annunciator
-message : DatagramPacket
-sleepTimeBetweenSends : int
+Annunciator(in group : InetAddress, in port : int, in sleepTimeBetweenSends : int)
-getMessage() : DatagramPacket
+setMessage(in message : Serializable)
+run()
Runnable
Listener
+Listener(in group : InetAddress, in port : int)
+run()
Runnable
«interface»
HelloGoodbyeInterface
PlayList is a Singleton with the alternate implementation
PlayList
-playList : PlayList = new PlayList()
-songTable : Hashtable
-PlayList()
+getInstance() : PlayList
+newInstance() : PlayList
-getKeyFrom(in server : InetSocketAddress) : String
+getAllServers() : String[]
+getAllSongs(in server : String) : Song[]
+getAllSongs(in server : InetSocketAddress) : Song[]
+loadSongsFrom(in server : InetSocketAddress, in songs : Song[])
+removeSongsFrom(in server : InetSocketAddress)
Agent: see other page for definition
HelloGoodbyeInterface
Hello
-serialVersionUID : long = 1
-serverLocation : InetSocketAddress
-serverName : String
+Hello(in serverName : String, in serverLocation : InetSocketAddress)
+clientSideRun(in socket : Socket)
+serverSideRun(in socket : Socket, in logWriter : LogWriterInterface)
+run()
Runnable
SongList
-songArray : Song[]
+SongList(in songArray : Song[])
+getNumberOfSongs() : int
+getSongs() : Song[]
+getSongUsingSongId(in songId : String) : Song
HelloGoodbyeInterface
Goodbye
-serialVersionUID : long = 1
-serverLocation : InetSocketAddress
-serverName : String
+Goodbye(in serverName : String, in serverLocation : InetSocketAddress)
+run()
Runnable

May 13, 2013

Partial UML of Client Side Driver
The interface and all classes are static nested classes within Customer