SOFTWARE REQUIREMENTS SPECIFICATION

José Manuel Navarro Castillo & Juan De La Cierva Benavent

MUSIC WORLD

Group C

1.-Requirements Definitions and Specifications

1.a) General definition of the software project

General Idea

Our project goal is to create a software, inspired in the design of the web page GarageBand, capable of reproducing different kinds of sounds of instruments, in a basic keyboard and sending those to a second computer, connected and listening to the arriving sounds.

We use imported code from MyMIDIPlayer, in order to be able to use instruments sounds in our code. The use of it will be specified, and used only in the Keyboard Interface (we will talk about it later) to create the effect of a real piano keyboard experience. Also, the objects sent in the messages while the connection between two computers is taking place, are MIDI sounds.

Objectives

We want to create a easy to use and to understand interface, and a playable keyboard, able to reproduce sounds with the lowest latency possible in the main computer, and in the sending.

Users

The applicattion has been made for any kind of user (that´s the objective of the easy to understand interface), so everyone who has a code developer such as Java Eclipse or NetBeans can use it easily. Also we added commentaries to the code to make it simple to read and comprehend.

1.b) Project requirements specification

General Requirements

Music synthesizer that reproduces different kinds of sounds making use of a MIDI library of the class MyMIDIPlayer1. It is also a goal and objective, to be able to send and receive sounds between both of the computers participating in the project.

Functional Requirements

Our main functional requirement is to reproduce sounds in our computers with a piano keyboard interface, making use of the MIDI library imported previously from the MIDI class.

The other important functional requirement we work with is the capability of connecting two computers with the applicattion, in order to create a bond between them and making them able to become a server (broadcaster) and client (reproducer), with the objective of sending and receiving musical MIDI sounds, and reproducingg them in both computers.

Project Legacy

The idea of the project became from the interface of the web page GarageBand, and we developed it with the only objective of presenting it in the classroom for our subject. We are talking about an original project, that probably hasn´t been made before, without any kind of retro-compatibility because its the first version.

System Scopes

The main limitation an user can find in our applicattion is the fact that we use the UDP connection for the interactions with the music sounds. We send the objects one by one, so we have a limited size of message. Also, we have limited musical instruments, due to the fact that we dont use the whole MIDI library, so the user won´t find all the instruments he will probably want.

1.c) Installation and testing procedures

Development procedure

Used Tools

In the creation moment, we have used the Java Eclipse development environment, exactly the version 2018-09. The tools we used, also were NetBeans in the beginning, but after we traslated the project to Eclipse, we started also using WindowsBuilder, the interface designer for Eclipse. That has been the main tool we have been using during the most part of the development.

Planification

In the first moment, we started with the design of the interface, trying to make it as similar as possible to the interface of GarageBand (the web page we used as model), in NetBeans, but after we changed to Eclipse, and started using WindowsBuilder. In the moment we finished our interface, we started creating the auxiliary functions and the jumps between the panels. We corrected our problems of focusing in the keys and the latency in the moment of sending the sounds. The last part, was one of the more important, we developed the connection between both computers, and sending the message with all the content, with initial problems, but corrected them with the time.

Testing and Installation Procedures

Non-Functional Requirements

We can find two big restrictions in our project. In order to work, we need internet web connection, and if we can count with it, then we will need (in the moment of the communication of the computers) the routers to have the ports opened.

Obtaining and Installation

Our main idea, is to provide the user with an executable JAR, that gives the user the final instance of the project, to use it.

If the user wants to install the software, s/he will only need to install the latest version of Java Eclipse s/he will find on the web. After that, importing the project inside Eclipse (Right click in the left panel of the screen --> import ---> existing projects into workspace --> browse --> select the folder where you have downloaded the project). If you want to modify the interface, you will also need to download the eclipse tool WindowsBuilder, from the Java Store inside Eclipse. To execute the programm, you will only need to go to the "main" package and select the "main" class, and run it in Eclipse in order to initiate the applicattion.

Testing and Execution Specification

In this case, the user doesn´t need to satisfy any kind of specification, because our application uses JAR archives, so the importation and implementation will be easy.

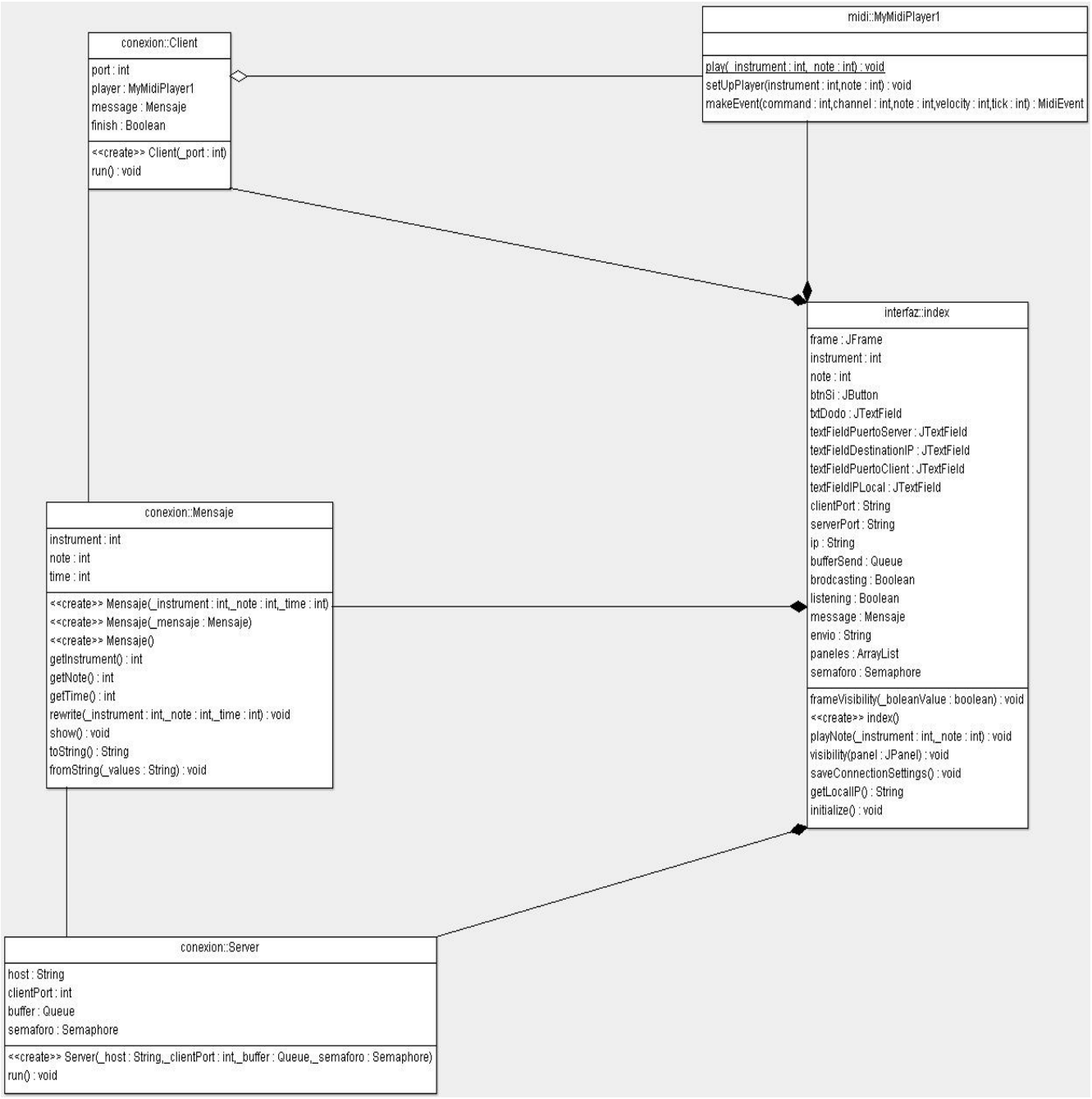
2.-System Architecture

Hierarchical description.

This software is organized in four packages. Each of these packages contain the files necesary to an specific requierement of the software. A brief description of the function of package could be:

- **main:** this package content just the main class and its only function is launch the graphic interface .
- **interfaz:** here we can find the file index.java; this file has all the code of the graphical interface and is the class that manage all the progam,
- **midi:** this is the responsable of the midi player to reproduce the sounds.
- **conexion:** here we have the three necesary clases to send and recieve the music.

Module Diagram



Module Description

Index

- **Purpose:** To provide a graphical interface which the user can manage all the functions of the program in a easy way.
- **Dependencies:**
  - midi.MymidiPlayer1
  - conexion.Client
  - conexion.Server
  - conexión.Mensaje
- **Implementation:** \interfaz\index.java

MyMidiPlayer1

- **Purpose:** This class content the logic required to play the sounds of the instruments.
- **Dependencies:**
  - java.sound.midi
- **Implementation:** \midi\MyMidiPlayer.java

Mensaje

- **Purpose:** Here we can find all the mechanisms to send the information and to deal when we recieve it.
- **Dependencies:**
  - None
- **Implementation:** \conexion\Mensaje.java

Client

- **Purpose:** It allows the reception and to play the music sended by the server.
- **Dependencies:**
  - Java.net
  - Midi.MyMidiPlayer
- **Implementation:** \conexion\Client.java

Server

- **Purpose:** It give the chance to the user of send the music what he is playing to other user, using a network
- **Dependencies:**
  - java.net
  - java.util
- **Implementation:** \conexion\Server.java