Software Design Document

Chatting

Jongmin Hwang

July 16, 2020

Contents

1. Introduction

1.1. Purpose

The purpose of this document is to describe the implementation details of the Chatting Program. Chatting Program is designed to enable communication between users who are varying in their distance and device. Computer language used in the software is Python.

1.2. Background

I began to learn how to code python about 3 months ago, and this is my first program that can be utilized to do something productive. I was looking for some creative things to do during the quarantine, and my dad, who is an expert computer programmer gave me an idea about this program.

1.3. Overview

This document will contain specifics about the Chatting Program. It explains how the program will function, and which components helped it to do so. It will end with some issues that this program has and what I thought about this program. This program has some differences with the general chatting applications that are used commonly, like WhatsApp and Discord. It does not contain an account management function like them. In popular chatting apps, you just need to make an account only the first time and the program saves the information of yourself and your chat partners. However, to use this program, you need to manually type in the information of your chat partner every time when you connect to them. Those apps can provide you not just the present messages but also previous messages. While you can find the conversation that you had a month ago with your friends on those apps, this program does not provide you any information on the past. The information is reset every time you use it. Among these small problems, there is a big reason why this program cannot be utilized as a chatting app that people can use. This program uses User Datagram Protocol(UDP) to send and receive messages. **However, this protocol does not provide you with the Internet, which means that you cannot communicate with someone who is using a different Wifi-router**. You cannot communicate with someone who is in a different house, so you can only communicate with your family member using this program. The general chatting apps use Transmission Control Protocol(TCP), which provides the Internet and therefore they can be used to communicate with someone who is far away.

2. Scenario

My chatting program is composed of three major states and some events that connect each other. Two user interfaces exist in my program; Chatting UI and Connection UI.

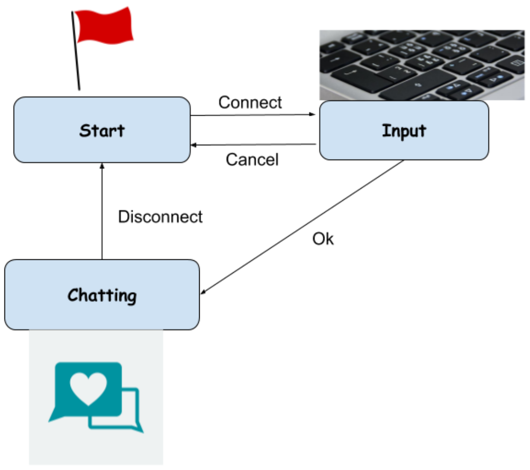
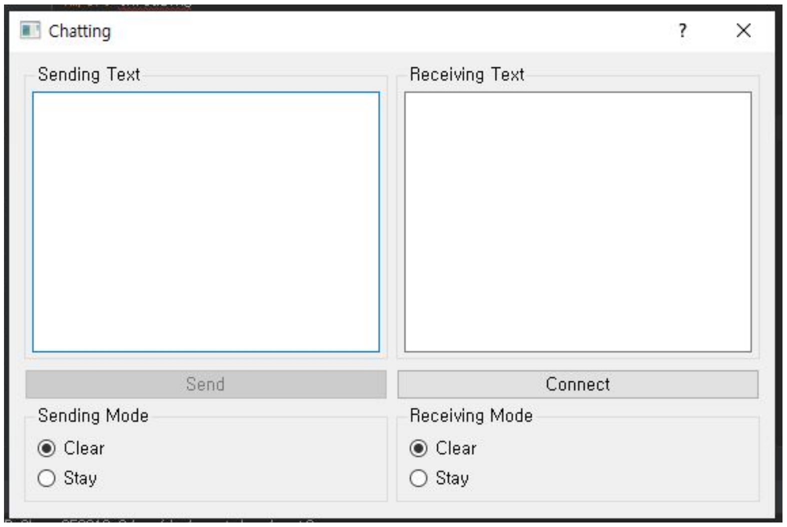


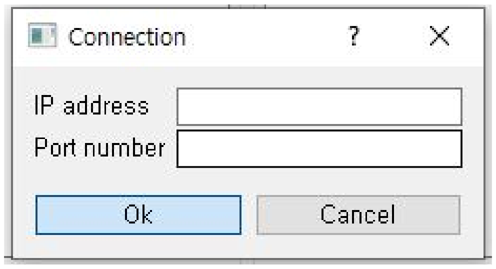
Figure 1. Chatting Program Outline

State

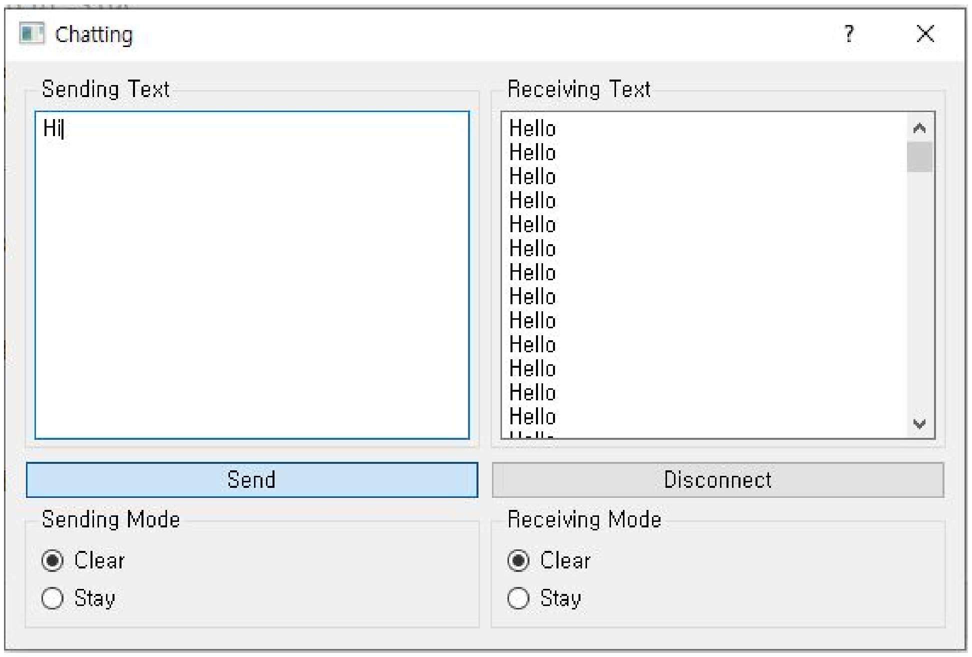
1. **Start**: It is the state in which chatting with someone is not enabled. The Chatting UI is on the screen, but nothing can be done at this state. The picture below is how the Chatting UI looks like.



2. **Input**: During this state, the information about the chat partner is input. IP address and port number should be typed in. The default port number is set as 43210, but it is changeable. The picture below is how the Connection UI looks like.



3. **Chatting**: Chatting is enabled in this state. The program can display messages from the chat partner, and it can also send messages to the chat partner. The picture of how it looks while chatting is posted below.



Hi

Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello

Event

1. **Connect**: It is the button on the Chatting UI. Pressing this button leads the user to the sub UI, where the information about the chat partner is input.
2. **Cancel**: It is the button on the Connection UI. Pressing this button leads the user back to the main UI without enabling the chatting.
3. **Ok**: While Cancel button goes back to Chatting UI, Ok button progresses into Chatting UI. Pressing this button leads the user back to the main UI, and it uses the information that was input to enable the chatting with a chat partner.

4. **Disconnect**: The Connect button changes its text to disconnect when the chatting is enabled. Pressing this button disables the chatting and leads the user to the beginning.

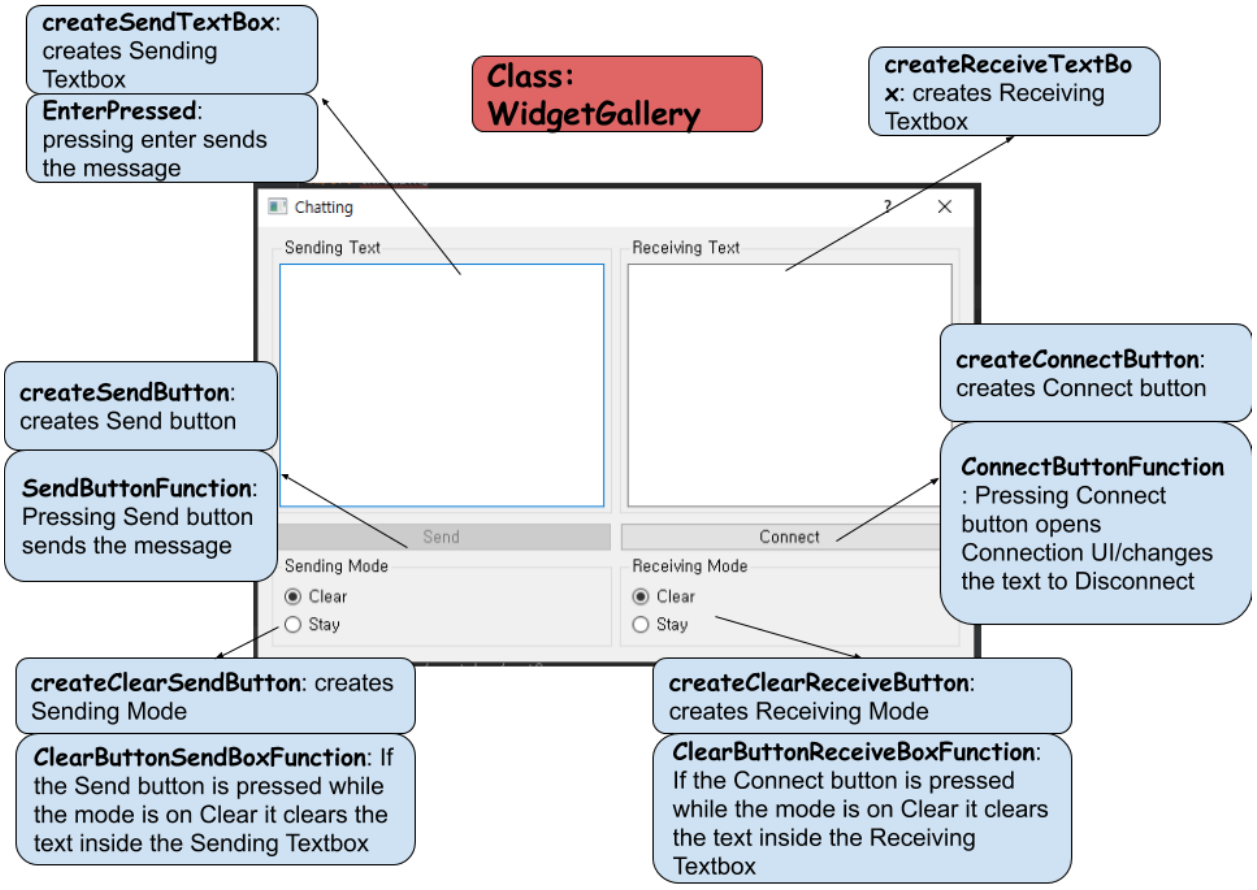
3. Architecture

3.1. Classes

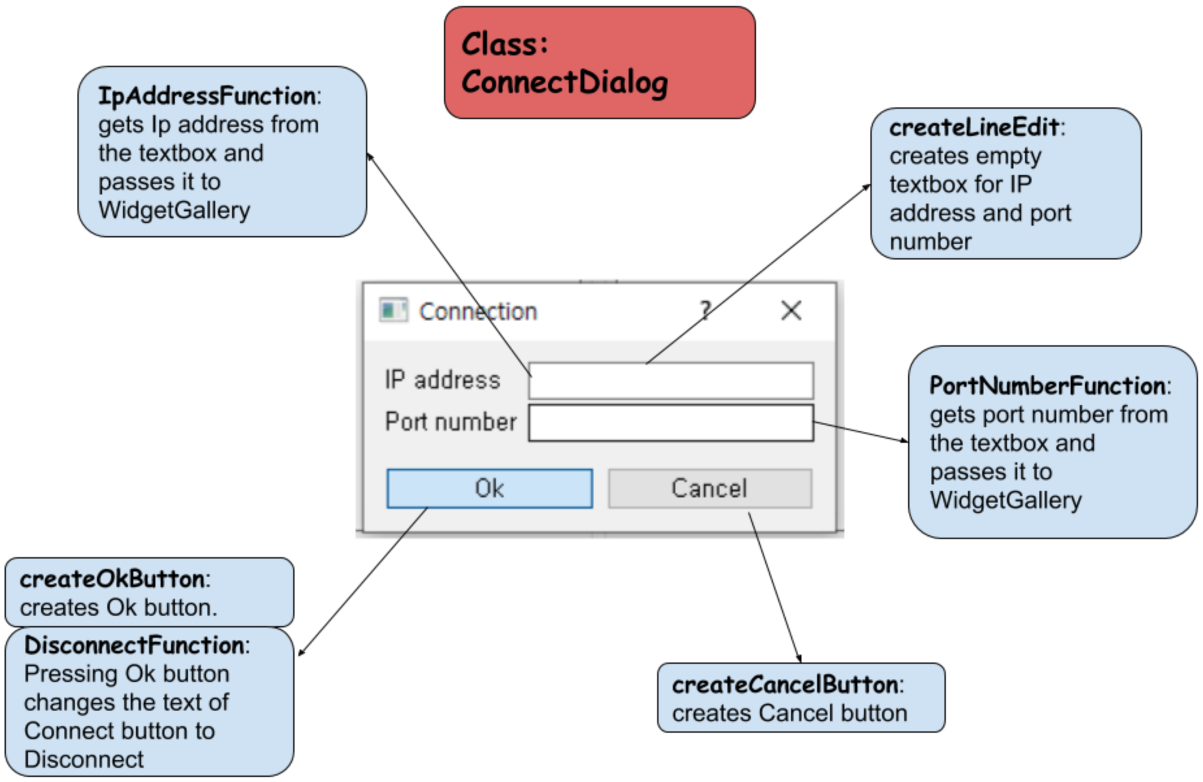
There are three main states in the Chatting program. Each state described above is connected to each class.

State	Start	Input	Chatting
Classes	**WidgetGallery**	**ConnectDialog**	**NetworkConnect**

- WidgetGallery

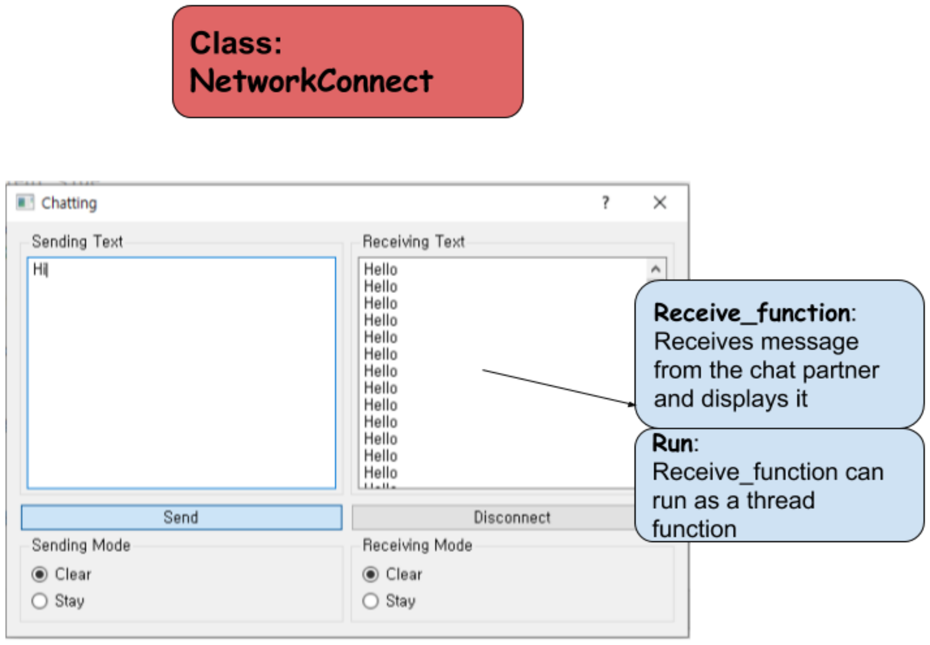

createSendTextBox: creates Sending Textbox
EnterPressed: pressing enter sends the message
Class: WidgetGallery
createReceiveTextBox: creates Receiving Textbox
Chatting
Sending Text
Receiving Text
createSendButton: creates Send button
SendButtonFunction: Pressing Send button sends the message
createConnectButton: creates Connect button
ConnectButtonFunction: Pressing Connect button opens Connection UI/changes the text to Disconnect
Send
Connect
Sending Mode
Clear
Stay
Receiving Mode
Clear
Stay
createClearSendButton: creates Sending Mode
ClearButtonSendBoxFunction: If the Send button is pressed while the mode is on Clear it clears the text inside the Sending Textbox
createClearReceiveButton: creates Receiving Mode
ClearButtonReceiveBoxFunction: If the Connect button is pressed while the mode is on Clear it clears the text inside the Receiving Textbox

- ConnectDialog



Class: ConnectDialog
IpAddressFunction: gets Ip address from the textbox and passes it to WidgetGallery
createLineEdit: creates empty textbox for IP address and port number
Connection
IP address
Port number
Ok
Cancel
PortNumberFunction: gets port number from the textbox and passes it to WidgetGallery
createOkButton: creates Ok button.
DisconnectFunction: Pressing Ok button changes the text of Connect button to Disconnect
createCancelButton: creates Cancel button

- NetworkConnect



3.2. System Architecture

- ❑ **WidgetGallery** is a class used to create the Chatting UI. It displays the text to be sent and received
- ❑ **ConnectDialog** is a class used to create Connection UI. It displays boxes for IP address and Port number
- ❑ **NetworkConnect** is a class used to connect the device to the server and to the device of the chat partner. It uses Input IP address and Port number from Connection UI to connect

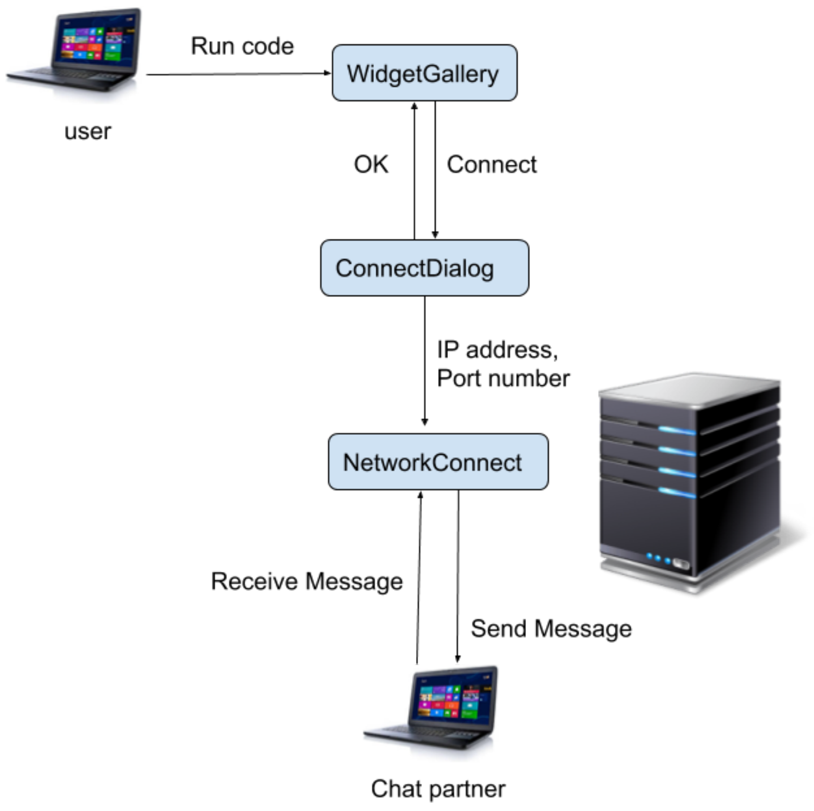


Figure 2. Chatting Program Framework

4. Conclusion

There are some issues about the Chatting Program, but none of them are major.

- **First issue** is the IP address of the user. It is coded to get an IP address automatically from the device, but if there are multiple IP addresses for connection it might not get the one that is working. In that case, the user has to manually input the IP address that is working.
- **Second issue** is the scrollbar of the Receive text box. The scrollbar of the Receive Text box does not move as the messages are written down. Due to lack of time and ideas, the function for auto-scrolling was not added to the Program. The user has to manually scroll down the bar to look at the recent message.
- **Third issue** is the error when the wrong or not existing information about the chat partner is input. If the user tries to send the message to the wrong IP address that does not exist, the program will stop and will have to be run again. This is also due to a lack of ideas.

This program is my first code that actually does something useful, so you might feel that it is clumsy. Finishing this program took about 2 months. It was hard for me to understand how connection between the devices work and which way will make it the most efficient, but as I progressed it made some sense to me. Overall, I am quite satisfied with this final product.