Chatbots

Create a small, networked chat application that is populated by bots.

Introduction

On an old server park, filled with applications from the early days of the internet, a few servers still run one of the earliest chat applications. The number of servers has dwindled, so everyone who uses it knows all the servers there are. Humans have long since left the place, but for some reason the bots never left.

While the servers run bots themselves, either for moderation or simply to keep the chat lively, there are also migratory bots that fly to the southern servers when the chats get cold, as well as free bots that participate in many chats at a time.

These bots, despite being old, manage interesting levels of interaction. They copy each others messages, and have a remarkable level of variety in the number of messages they produce.

Concrete Processes

Your application will consist of a number of chatrooms. These chatrooms have a feed. If a message is sent to the feed, that message is forwarded to every participant. Chatrooms are listed as a global resource, and are accessible by everyone. They are accessible via Port and IP-address.

A local bot can be in only one particular chatroom. Local bots have an AI that is limited to

- Changing its message
- Sending a message
- Crashing

They are added to a chatroom, and never leave again.

A migratory bot can also migrate to other chatrooms. When it does, it forgets the old chatroom, and upon arriving in the new chatroom, the chatroom informs it of its feed, so it can send messages to the feed and receive messages as well.

Independent bots are run of separate machines, and as such need to connect to each chatroom with a handler, that handles traffic between the remote bot and the server by forwarding posts from the bot to the feed, and sending updates from the feed to the bot. Independent bots expand upon migratory bots through:

- Not migrating, instead
- Joining a server
- Leaving a server

This provides a lively ecosystem of bots. Naturally all bots are independent, which means that you need a thread for every single one.

Changing Messages

Changing a message is one of the ways bots try to keep things fresh in their chatrooms. Of course a good bit of spamming is never off, so copying the message of another bot verbatim is always an option. In some cases however, a bot will try a more sophisticated algorithm.

At the start, the bot will select a number close to the average number of words in a sentence among the messages it receives (but at most the maximum). Then, it will create a new sentence by, for each physical position in the sentence, copying a word in that place. For example, if it chooses 3, and receives the sentences ”I see green”, ”I am not” and ”Do you understand?” It will generate: ”I am understand?”.

View

It will be interesting to see what kind of spam bots see and create. Because multiple threads printing to one console is difficult to track, make a **simple** UI: A frame displaying print statements, used for each bot or chatroom. *Do not spend any more time than necessary on your UI. Do not make a fancy configurator. Only a simple view is sufficient.*

Requirements

In summary, the concise functionality is:

- Chatrooms
- Three kinds of bots with random AIs
- A view for bots and chatrooms.

Achieved by means of:

- A Client-Server pair
- Copious amounts of Threads

Some useful resources: Server-Client pair , Server Socket , Socket , ObjectOutputStream , ObjectInputStream , Thread , Synchronized Methods , Random .

As extras you could think of adding human interaction, more sophisticated message generation or chat modes such as r9k.

Assignments in 2017 LuL > LUL