IST400/600 Social Web Technologies, Spring 2010
Lab 4.
Instructor: Keisuke Inoue

Description:

In this lab, you will work on implementing concepts for a social web site using object-oriented programming. All the information you will need is covered in Cooper, up to Chapter 7. The following instructions will help you to accomplish the task, but will not tell you all the small steps. **Note: this lab assignment is designed as an individual work. You may ask questions to instructors and/or friends (but keep it quiet) and look up resources, but you are not allowed to copy-and-paste any resources other than you created.**

Scenario:

You are implementing a Chat Bot for the users of your social web systems. (Because of the other assignment, this week's lab is open-ended. You will need to complete

Lab Instructions:

1. **Setup and basic Linux operations**

   First, login to your Linux system, and create a folder called "Lab4". If you don't remember how to do it, review the steps 1.1 to 1.6 of Lab 1.

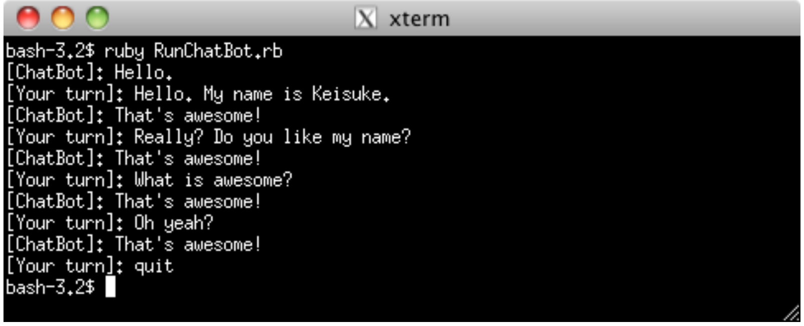
   1.1. Now on the terminal, move to the Lab4 directory, by typing "cd Lab4". Then download a ruby files that have been created by the instructor by typing:

   ```
   wget http://sanka.syr.edu/files/ChatBot.rb
   wget http://sanka.syr.edu/files/RunChatBot.rb
   ```

   This command downloads the specified file through HTTP to the current directory.

   1.2. Launch your favorite editor and open the new files you just downloaded. Take a quick look at the files. ChatBot.rb contains a definition of the ChatBot class, and RunChatBot.rb creates an instance of the ChatBot class and run the chatbot.

   1.3. Just to get the idea, let's run the ChatBot. Go to the terminal window, type 'ruby RunChatBot.rb'. You should be able to chat with the ChatBot, except that the ChatBot has a very few vocabulary… specifically, it only tell you "That's awesome!" whatever you say… Type 'quit' to quit the ChatBot.



```
bash-3.2$ ruby RunChatBot.rb
[ChatBot]: Hello.
[Your turn]: Hello. My name is Keisuke.
[ChatBot]: That's awesome!
[Your turn]: Really? Do you like my name?
[ChatBot]: That's awesome!
[Your turn]: What is awesome?
[ChatBot]: That's awesome!
[Your turn]: Oh yeah?
[ChatBot]: That's awesome!
[Your turn]: quit
bash-3.2$ 
```

ChatBot in action… not exactly the best ChatBot around…

You tasks in this Lab 4 will be to create improved versions of the ChatBot, by creating subclasses of ChatBot.

2. CopyBot

The first ChatBot subclass, called “CopyBot”, is a very simple subclass of ChatBot, which copies whatever the user said and say it as the response. Here is how you implement it:

1. Create a file called “CopyBot.rb”.

2. The first line of the file should be as follows:

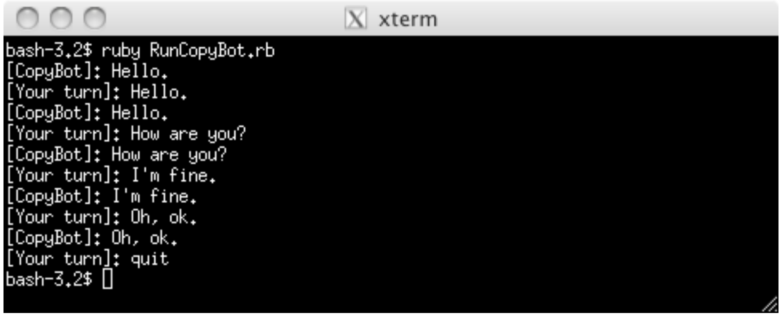
   ```
      require ‘ChatBot’
   ```

   This is so that you don’t need to repeat whatever the stuff in ChatBot.rb.

3. Then, define a subclass of ChatBot, named CopyBot.

4. For this class, you only need to override one method: response_to. Take a look at ChatBot.rb, and see how the method is implemented. You need to redefine the method in CopyBot.rb class, so that it does what it is supposed to do.

5. Once the class is defined, create an intance, and run the start_conversation method. Take a look at RunChatBot.rb method, if you don’t know how to do it.



```
bash-3.2$ ruby RunCopyBot.rb
[CopyBot]: Hello.
[Your turn]: Hello.
[CopyBot]: Hello.
[Your turn]: How are you?
[CopyBot]: How are you?
[Your turn]: I'm fine.
[CopyBot]: I'm fine.
[Your turn]: Oh, ok.
[CopyBot]: Oh, ok.
[Your turn]: quit
bash-3.2$ 
```

CopyBot in action

3. KennyBot

The next ChatBot subclass is called “KennyBot. KennyBot tells you everything in a mumble, so you can’t really communicate.

Here is how you implement it:

1. Create a file called “KennyBot.rb”.

2. The first line of the file should be as follows:

   ```
      require ‘ChatBot’
   ```

   This is so that you don’t need to repeat whatever the stuff in ChatBot.rb.

3. Then, define a subclass of ChatBot, named KennyBot.

4. For this class, you only need to override two methods: initialize and response_to. Take a look at ChatBot.rb, and see how the methods are implemented.

   For the initialize method, you can call the initialize method of the super class by saying “super” in the beginning of the method definition block. Then, you can change the greeting so that KennyBot doesn’t say “Hello.” articulately.

   For the response method, try to make the method so that the response will be not simply a “Mphmm”, but the length of the response corresponds to the length or the number of the message received. (For example, if the message is three words, e.g. “how are you?”. KennyBot will say “Mphmm mphmm mphmm.”

5. Once the class is defined, create an intance, and run the start_conversation method.

4. PatternBot

The last ChatBot subclass is called “PatternBot. Finally, it has some intelligence. (It’s still very primitive though). This ChatBot uses a technique called regular expression, which is new to some of you, so you can start with a template.

Here is what you will do:

1. Download the file by typing:

   ```
   wget http://sanka.syr.edu/files/PatternBot.rb
   ```

2. Modify the implementation as you like (e.g. another elsif block), to improve the PatternBot. Insert ta comment, so that instructor knows what is the intension of the improvement. You can work on it as much, or less as you want.

5. Submsision

As usual, submit to htpp://sanka.syr.edu/swt/. Because you have three files to submit, concatenate the files first with the following command, and submit your_last_name.rb

```
cat CopyBot.rb KennyBot.rb PatternBot.rb > you_last_name.rb
```