Taylor Hennessy
CPSC 4720 - Section 001
Paige Rodeghero
Mini Project 2

May 1, 2020

Application Activity Monitor

NOTE: Due to slow upload constraint in uploading the video, it is available at the link below.

https://drive.google.com/file/d/1he2yualXQO6mJDiI3Y0yL-h8aLH32brJ/view?usp=sharing

Design of the Application

1. Design Concept

The concept of this application's general GUI design and its design philosophy comes from the trend of programs called "Ghosts" in English, but in Japanese are referred to as "Ukagaka" (伺か) loosely meaning "Something You Inquire With." They are essentially desktop programs featuring mascots that sit in the corner of a user's screen and that the user usually interacts with in a leisurely manner. Oftentimes they take the form of a user's favorite character or characters, or a cute animal. They are meant to provide a sense of comfort and companionship as a sort of "virtual friend" on your computer. Normal interactions with these programs often includes short dialogue trees and sometimes small games.

As a small note, "Ghosts" typically run as script-based libraries on other software, but for this project I decided to create the engine for this software on my own.

When we discussed the isolation of freelancing positions in many cases, I thought that a more advanced "Ghost" assistant could be an alternative to having to work alone for hours upon hours a day, and so with this program I wanted to combine the friendly, encouraging face of a "Ghost" program with the practicality of an application activity monitor. Therefore, the design of this program seems fun but follows this rudimentary purpose.

2. Design of the GUI

The GUI of this virtual assistant is designed to appear in the bottom right of the user's screen, and to remain there, unable to be moved other than minimized. This is because I wanted the feeling of a digital character being "actually there" rather than being dragged around, and also because this was the simplest way to create a transparent window without a top control bar as is standard in Mac applications.

I kept the options on the screen fairly simple since the window is designed to remain small: the two options are to "Start Tracking Activity" (Changes to "Stop" once tracking is started) or "View Tracking Activity". The first one starts the tracking (begins polling AppleScript) and the second one displays a second window that shows some charts generated based on the most recent recorded day's current activity.

I would like to add more options to this design in the future and continue developing this program but for now I believe these options and design choices are adequate for a "Mini Project."

Using the Application

1. *System Requirements*

This program must be run on OS X (Mac). This is because the way that it polls the operating system for the active applications is through AppleScript. Working on the implementation of AppleScript in Java is what took up the greatest amount of my time in creating this program.

Upon running the "Start Tracking Activity" button, you may be prompted to allow accessibility utilities to allow the program to run (not doing so will cause it to error). These must be allowed in order to allow the program to poll AppleScript for the list of active applications.

2. *Setup*

This application contains 4 Java files within the “Template” package:

Main.java
Character.java
View.java
ApplicationData.java

There is a Makefile that will compile all of these outside of the “Template” folder, inside the “Virtual Assistant” folder along with this document and my demo “activity.csv” file. With this file, you can open the application straightaway and click on “View Tracking Activity.” This will open the secondary window that has tracking activity listed and illustrated.

YOU MAY NEED TO FIX THE SEPARATOR ERROR IN THE MAKEFILE: when I upload them from my Mac it always seems to become 4 spaces.

3. *Compilation and Running the Program*

There is a Makefile that will compile all of these outside of the “Template” folder, inside the “Virtual Assistant” folder along with this document and my demo “activity.csv” file. With this file, you can open the application straightaway and click on “View Tracking Activity” to view the chart window.

4. *Typical Usage*

The typical usage of this program should be to run it on “Start Tracking Activity” for some time first, then to stop the tracking after a good amount of time by clicking “Stop Tracking Activity.” After the activity is stopped, this means the *activity.csv* file has been updated.

Implementation

1. *Code Organization*

To organize parts of the code, I decided to lightly use the "Model View Controller" design pattern. The "Model" codes are the Character class and the ApplicationData class. The "View" code is the View class, and the "Controller" code is the Main class. This allows for simplistic code organization and very much helps with debugging.

Workday Video

This workday shows many applications being opened at once, and so all being logged minute by minute in the activity file in the application. I did not have enough disk space to record 4 hours, so it is just about 2 hours and 20 minutes of work. Here is the resulting chart of the activity in the video.

The video can be viewed at this link

https://drive.google.com/file/d/1he2yualXQO6mJDiI3YoyL-h8aLH32brJ/view?usp=sharing

activity

Program	30 April 2020	01 May 2020
FINDER	3	145
OPERA	3	145
PREVIEW	3	2
DISCORD	3	1
IDEA	3	145
SPOTIFY	3	39
TEXTEDIT	3	16
CALENDAR	3	1
JAVA	3	145
QUICKTIME PLAYER	0	144
MEGPOID ENGLISH UNINSTALLER	0	1
UTAU-SYNTH	0	18
PAGES	4	83
MOTION	1	2
KEYNOTE	1	2
TERMINAL	1	2

References

Ghosts/伺か

zarla. “Some Info about Ghosts.” April 29, 2020. http://www.ashido.com/ukagaka/ghostinfo.html

JavaSwing learned entirely from StackOverflow, but no code was copied to be used, so in the end a sincere thank you to all the confused Java programmers before me.