Joseph Weiner

CMPT220L-112-17S

Professor Juan Arias

*Milestone for JavaCatch*

JavaCatch is a fully comprehensive current-generation catch calculator for the Pokémon series of games, specifically the newest edition of these games. It works inside of a GUI and once the user inputs the parameters of the Pokémon encounter, the program will plug the user inputs into an algorithm that will give the percent chance of catching this Pokémon. The GUI will work by essentially acting as an interface to call methods and classes from the various files of the project.

So far the main algorithm program and many of the individual class programs that control variables in the algorithm have been successfully created and debugged. There is also a database, which through much trial and tribulation, has been completed and implemented into the program. The table contains Pokémon stats that must be plugged into the algorithm in order for it to work. This will be accessed through java and the database itself has been built in Apache Derby. The “ij” interface has been used to configure the database. The GUI has been built in Swing Window Builder for Eclipse, and the project as a whole has been coded in Eclipse. The GUI to date has been started, design is going quite well and the next step is the GUI’s functionality. This involves mapping the buttons and inputs of the GUI to the methods and classes I have written in Java. The current issues with the program are in getting the actual Pokémon to load from an Array object. This will be the most difficult part, but once this data can be gathered and implemented without compilation or syntax errors, the project is basically finished and functional.

A UML diagram of the JavaCatch project can be found in the adjacent file which is labeled “DiagramOfJavaCatch.pdf.” I thought it would be more efficient to outline the project with the GUI as a focal point since this project is many files long with over 10 classes and many methods. This is merely a milestone and the project is not finished. The number of classes should be increased by as many as 3 by the time this project is completed. Once the number of classes is completely set in stone a UML diagram would make more sense. Also my Visual Paradigm free trial has run out. This will be fixed by the project’s completion.

The requirements for this project are all essentially outlined in the Algorithm class of the project. For the algorithm to be satisfied and for a user of the system to calculate their chances at catching a Pokémon, the system needs 14 variables, and a database with the individual Pokémon’s stats and a list of the variables needed from each Pokémon to satisfy the algorithm. The Pokémon database within the project from which this system pulls its variables contains 9 of the 14 variables. The other 5 variables are filled from either user input or a variation of one of the other variables based on user input and other things from the Pokémon game that are also set parameters such as the “heavy ball” item which has a catch rate based on the weight of a Pokémon, its parameters to determine catch can be expressed as a series of if else statements based on the database Pokémon object that the user selects from the drop down list. Therefore the main system requirements are as follows: the database from which to pull the Pokémon and statistic variables, the GUI to invoke methods and classes, and the variables that satisfy the algorithm which are either inputted through the GUI or obtained through the database. This gives the user a functional system and a way to use it.

There are a few other projects out on the web that seek to solve similar issues. One that I thought was interesting and pretty novel was the Generation 6 catch calculator made by a website called “A Cave of Dragonflies.” Their project unlike mine is for generation 6 and not the current generation 7, it is also web based and is programmed using HTML, CSS, and JavaScript. The way these folks solved the database problem I had was incredibly simple. JavaScript arrays do not need to have a primitive type, so what is the CSV file in my derby database could have been one fluid array in Javascript. One problem with a web based service like the one on “A Cave of Dragonflies” is that a user must have an internet connection. If a potential user doesn’t have a connection to the internet they cannot use the service. Having a standalone application made in Java has a few benefits, one of those being no need for internet connectivity, and also a standalone application cannot fall victim to DDOS attacks or anything else of that nature. Their project does not use the shake check algorithm or at least not in the foreground of the program for the user to see. The project on the web also does not include all pokeballs available in the game like my standalone application. However this project has clear advantages to mine. It is web based and has language compatibility with python because JavaScript and python are both interpreted languages and this project didn’t have to be entirely in Java. It also has more detailed HP description and delves into IV and EV values which are far too advanced and irrelevant for my project.

The user manual can be defined as first fill out the fields and drop down menus of the GUI then press the Calculate My Encounter button.

In conclusion this project has shown me so far how complex databases can sometimes be and it has shown me the wonders of programming a GUI. This project has taken quite some time to complete and I am grateful for the experience and hopefully this project turns out the way I would have wanted it to from the beginning. The project should be done before the deadline which is a positive thing for sure.

Works Cited

The other project that I studied

<https://www.dragonflycave.com/calculators/gen-vi-catch-rate>

Source code

view-source:https://www.dragonflycave.com/calculators/gen-vi-catch-rate

https://www.dragonflycave.com/gen6capture.js