Joe Weiner

CMPT220-112-17S

Professor Juan Arias

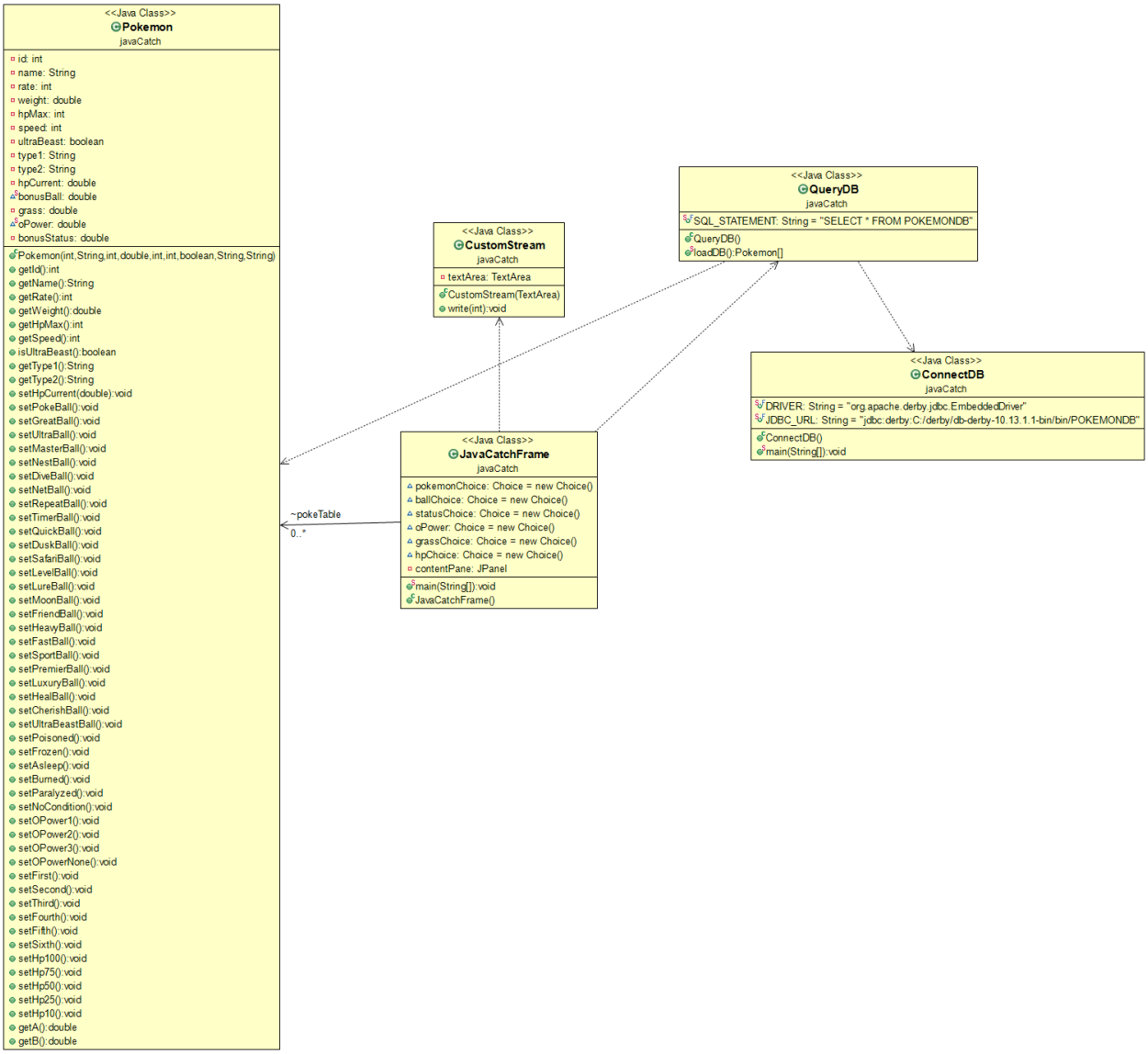
Final Write-up Project 02

JavaCatch is a fully comprehensive current-generation catch calculator for the Pokémon franchise series of games, specifically the newest edition of these games; Pokémon Sun and Pokémon Moon. 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 as well as the amount of times the Poke ball will shake. The GUI works by giving the user of the program various drop down menus with options that will call upon different methods and classes. Overall the user can choose any of the Pokémon, Poke Balls, and status conditions in the game. They can also input the health of the Pokémon which they are trying to catch and the amount of Pokémon that they have already caught; a variable that modifies the algorithm used in the Pokémon games and in this program.

The program has 5 major components: the GUI itself, wherein users can actually make their calculations, a local database wherein all of the data on each individual Pokémon is stored, the *Pokemon* class, which creates an object of our database that we can use to make our calculations, and two classes to communicate with the embedded database, *ConnectDB*, and *QueryDB*. The database, through much trial and tribulation, has been implemented into this program for the convenience of not needing internet connectivity to use this program. The average Pokémon player plays their games on a handheld console and does not need internet to play their game, so why should they need internet to help them calculate their catch chances?

The database has a table which contains Pokémon stats that must be plugged into the algorithm in order for it to work. This is 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 is built in Swing Window Builder for Eclipse, and the project as a whole has been coded in Eclipse. The classes *ConnectDB*, *QueryDB*, *Pokemon*, *JavaCatchFrame*, and a class called *CustomStream* that routes console output to the GUI text box, work together to connect to the embedded database, read the values in that database, and convert them into a usable object that will make our program functional.

On The following page a detailed UML diagram of the entire project can be found.


<<Java Class>>
Pokemon
javaCatch
id: int
name: String
rate: int
weight: double
hpMax: int
speed: int
ultraBeast: boolean
type1: String
type2: String
hpCurrent: double
bonusBall: double
grass: double
oPower: double
bonusStatus: double
Pokemon(int,String,int,double,int,int,boolean,String,String)
getId():int
getName():String
getRate():int
getWeight():double
getHpMax():int
getSpeed():int
isUltraBeast():boolean
getType1():String
getType2():String
setHpCurrent(double):void
setPokeBall():void
setGreatBall():void
setUltraBall():void
setMasterBall():void
setNestBall():void
setDiveBall():void
setNetBall():void
setRepeatBall():void
setTimerBall():void
setQuickBall():void
setDuskBall():void
setSafariBall():void
setLevelBall():void
setLureBall():void
setMoonBall():void
setFriendBall():void
setHeavyBall():void
setFastBall():void
setSportBall():void
setPremierBall():void
setLuxuryBall():void
setHealBall():void
setCherishBall():void
setUltraBeastBall():void
setPoisoned():void
setFrozen():void
setAsleep():void
setBurned():void
setParalyzed():void
setNoCondition():void
setOPower1():void
setOPower2():void
setOPower3():void
setOPowerNone():void
setFirst():void
setSecond():void
setThird():void
setFourth():void
setFifth():void
setSixth():void
setHp100():void
setHp75():void
setHp50():void
setHp25():void
setHp10():void
getA():double
getB():double
<<Java Class>>
CustomStream
javaCatch
textArea: TextArea
CustomStream(TextArea)
write(int):void
<<Java Class>>
QueryDB
javaCatch
SQL_STATEMENT: String = "SELECT * FROM POKEMONDB"
QueryDB()
loadDB():Pokemon[]
<<Java Class>>
ConnectDB
javaCatch
DRIVER: String = "org.apache.derby.jdbc.EmbeddedDriver"
JDBC_URL: String = "jdbc:derby:C:/derby/db-derby-10.13.1.1-bin/bin/POKEMONDB"
ConnectDB()
main(String[]):void
<<Java Class>>
JavaCatchFrame
javaCatch
pokemonChoice: Choice = new Choice()
ballChoice: Choice = new Choice()
statusChoice: Choice = new Choice()
oPower: Choice = new Choice()
grassChoice: Choice = new Choice()
hpChoice: Choice = new Choice()
contentPane: JPanel
main(String[]):void
JavaCatchFrame()
~pokeTable
0..*

The requirements for this project are as follows: 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. These requirements make the program work, however, for a user to properly implement this, they must take the "csv" file with the database information and batch load it onto their own computer's "ij" interface. Without the database, nothing works, so if a user wishes to download this project for home use off of this GitHub page, I the programmer strongly recommend that you download the *poketable.csv* file located in the code folder. This file should be loaded into an apache derby interface, and then the embedded address in the code for *ConnectDB,* and *QueryDB* should be changed to match the path of your database. The only other system requirement for this program would be a java installation and the ability to run a .jar file.

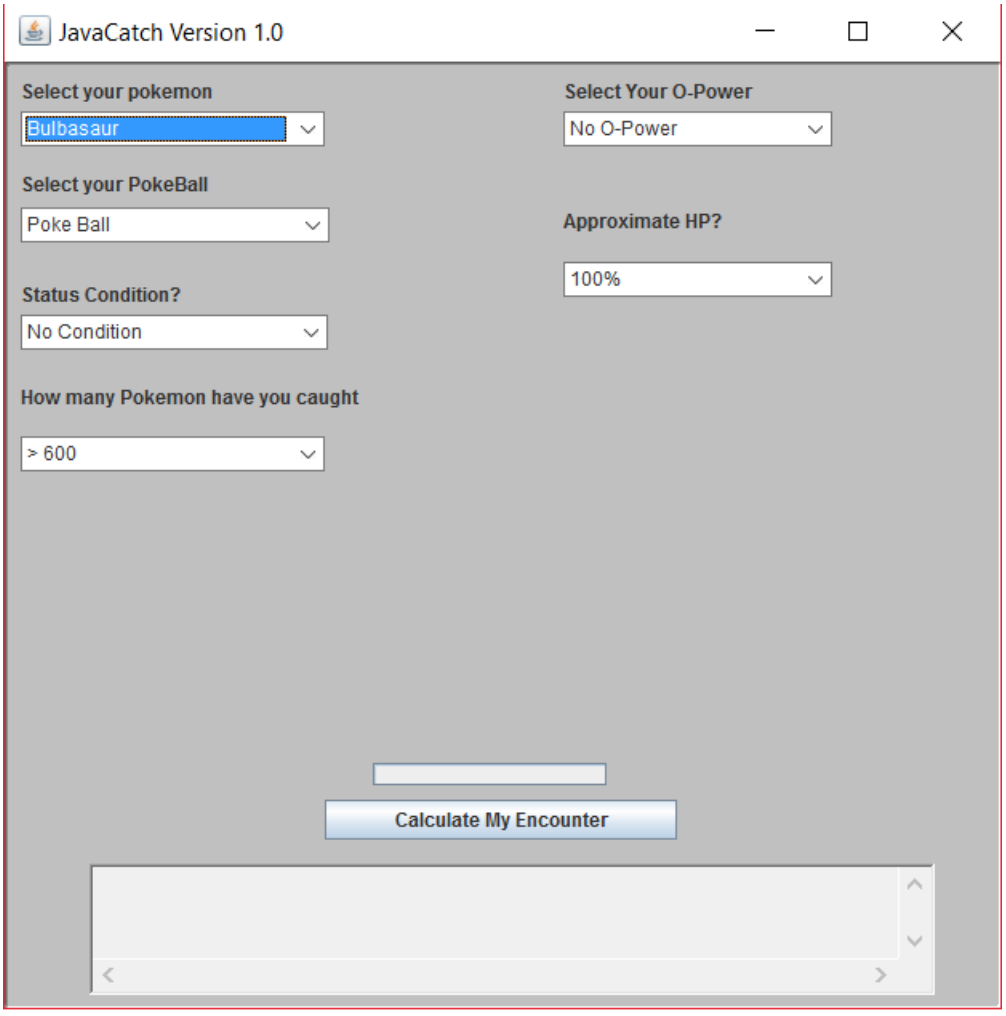
There are a few other projects out on the web that seek to solve similar issues. One that I thought was interesting and pretty novel was the Generation 6 catch calculator made by a

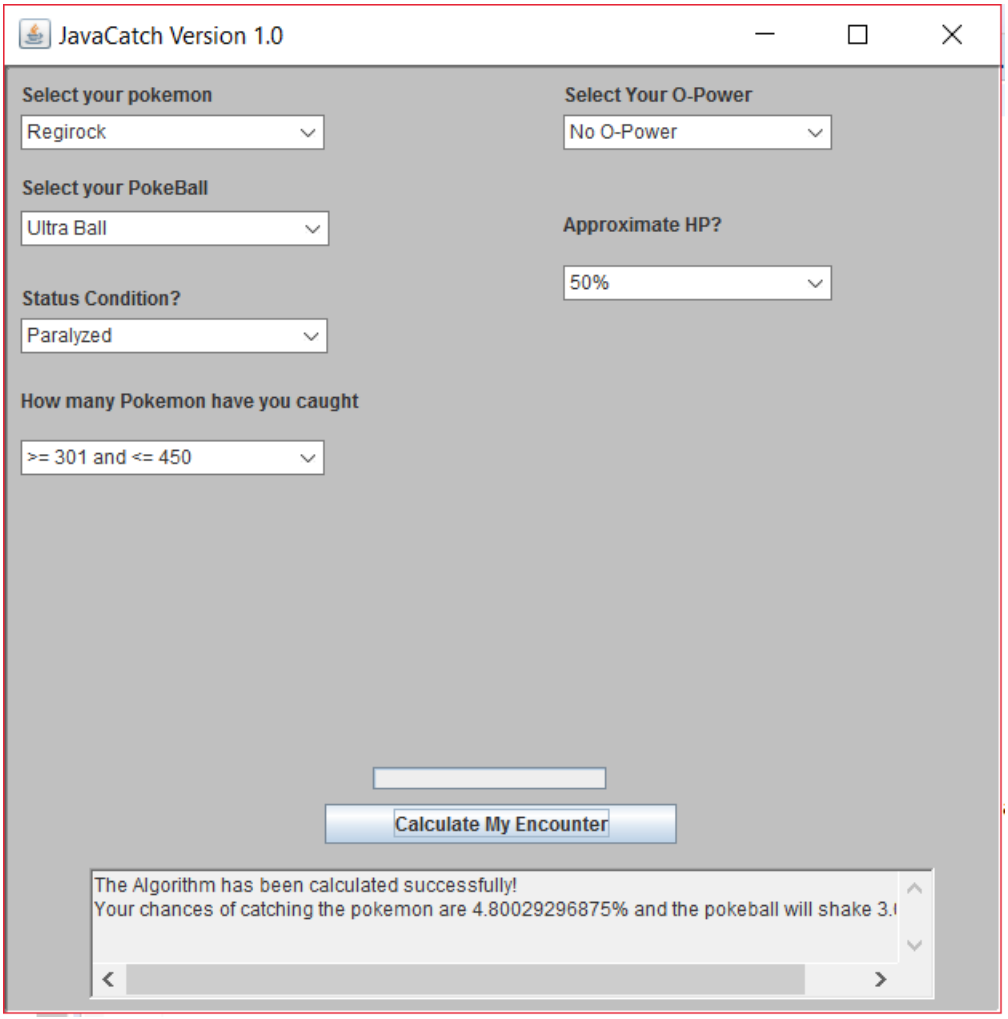
website called “A Cave of Dragonflies.” Their project unlike mine is for generation 6 and not the current generation 7, it is also web based and is programmed using HTML, CSS, and JavaScript. The way these folks solved the database problem I had was incredibly simple. JavaScript arrays do not need to 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. These IV and EV values are like the genetics of Pokémon, while base stats work in theory perfectly, each Pokémon can have potentially different IVs and EVs.

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. The fields are all labeled accordingly and they are self-explanatory. Simply follow the prompts and fill out exactly what you want to test, once all the drop down menus are selected to your liking, press the “Calculate My Encounter” button at the bottom of the interface and the results will show up in the console.

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 also grateful that this project turned out the way I would have wanted it to from the beginning.

Finally we see two screen shots of the program, the first is the default look of the GUI upon program start, and the second is a calculation of catching the Pokémon “Regirock” with some set of user inputs.




JavaCatch Version 1.0
Select your pokemon
Regirock
Select Your O-Power
No O-Power
Select your PokeBall
Ultra Ball
Approximate HP?
50%
Status Condition?
Paralyzed
How many Pokemon have you caught
>= 301 and <= 450
Calculate My Encounter
The Algorithm has been calculated successfully!
Your chances of catching the pokemon are 4.80029296875% and the pokeball will shake 3.

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