CSCI 1133, Fall 2019
Programming Assignment 12
Due: 11:55pm, Wednesday December 4, 2019

Due Date: Submit your solutions to GitHub by 11:55 p.m., Wednesday, December 4th. We will do a pull from this time point. Do not upload anything to Canvas and PLEASE be sure to use proper naming conventions for the file, classes, and functions. We will NOT change anything to run it using our scripts.

Unlike the computer lab exercises, this is not a collaborative assignment. You must design, implement, and test your code on your own without the assistance of anyone other than the course instructor or TAs. In addition, you may not include solutions or portions of solutions obtained from any source other than those provided in class with one exception: this homework allows you to freely use an extra online resource: effbot.org/tkinterbook (you should also look at docs.python.org/3/library/tk, which is allowed by default). Otherwise obtaining or providing solutions to any homework problem for this class is considered Academic Misconduct. See the syllabus and read section “Academic Dishonesty” for information concerning cheating. Always feel free to ask the instructor or the TAs if you are unsure of something. They will be more than glad to answer any questions that you have. We want you to be successful and learn so give us the chance to help you.

Instructions: This assignment consists of 1 problem, worth 60 points (and up to 10 points of extra credit). Solve the problem below by yourself, and put all functions in a single file called `hw12.py`. Use the signatures given for each class and function. We will be calling your functions with our test cases so you must use the information provided. If you have questions, ask!

Because homework files are submitted and tested electronically, the following are very important:

- You follow all naming conventions mentioned in this homework description.
- You submit the correct file, hw12.py, through Github by the due date deadline.
- You follow the example input and output formats shown.
- Regardless of how or where you develop your solutions, your programs should execute using the python3 command on CSELabs computers running the Linux operating system.

Push your work into Github under your own repo. The specific hosting directory should be: repo-<username>/hw12, where you replace <username> with your U of M user name. For instance, if your email address is bondx007@umn.edu, you should push your hw12 to this directory: repo-bondx007/hw12

The following will result in a score reduction equal to a percentage of the total possible points:

- Incorrectly named/submitted source file, functions, or classes (20%)
- Constraints not followed (40%)
- Failure to execute due to syntax errors (30%)

Constraints:

- Do not use the input() function anywhere in your code. It will break our grading scripts and result in major point penalties.

- You must actually use the Pokemon class that you create by instantiating it and accessing it as some point in your code (i.e., don't just create the class but then do the whole assignment without using it)

Problem A. (60 *points*) **Pokemon Safari GUI**

In this assignment, you will be creating a graphical user interface (GUI) that allows the user to play Pokémon Safari, a simple minigame within Pokémon in which the player has a set number of Safari balls to use to catch as many Pokémon as possible. There are no battles within this minigame, so the user must do one of the following each turn:

- Throw a Safari ball at the Pokémon to try and capture it, which has a random chance of success that depends on the Pokémon's species.
- Run away and find another Pokémon.
- Extra Credit: Throw a rock at the Pokémon, which increases the chance of capture but also increases the chance that the Pokémon will run away on its turn.
- Extra Credit: Throw bait at the Pokémon, which decreases the chance of capture but also decreases the chance that the Pokémon will run away on its turn.

You only need to worry about the first two options for now; the last two can be implemented for extra credit, as explained below.

The `hw12.zip` file on Canvas contains everything you need to get started on this assignment, including:

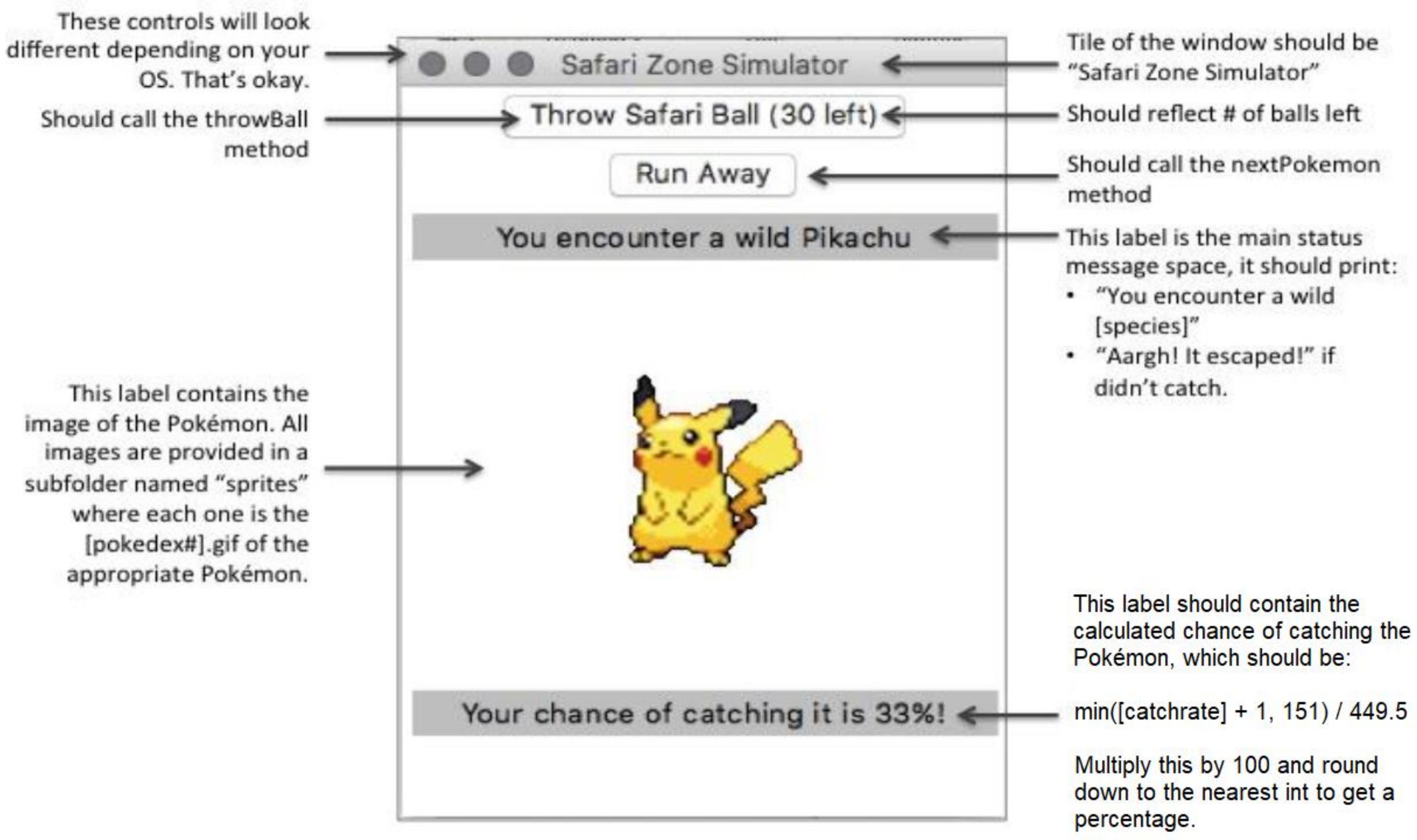
- `hw12.py`, a python file which contains the skeleton of your program, outlining the classes and methods required and providing details and hints for each one.
    - Note: currently there is a print statement at the top of every method in the SafariSimulator class that just declares that the method was run. You are encouraged to keep the print statements there, and add your own to help debug: we grade based on the GUI and not the printed output, so put in as many print statements as you want.
- `pokedex.csv`, a CSV file which contains the dex number, name, catch rate, and speed of each of the 151 original species of Pokémon.
- `sprites`, a folder which contains GIF images of each Pokémon, ordered by dex number.

Your program should do the following things (see hw12.py for additional details):

- You should create and use a `Pokemon` class with instance variables for the species name, dex number, catch rate, and speed. It should have a `__str__` method that returns the species name. Other methods and variables are up to you. Hint: you should design and test this class first, before writing any other code.
- Read in `pokedex.csv` to get Pokémon names, dex numbers, catch rates, and speeds.
- Keep track of the remaining safari balls, starting at 30, and stop the Safari adventure once that number reaches 0.
- Keep track of all captured Pokémon.
- Create appropriate buttons and label widgets as shown in the screenshot below.

- Create a `nextPokemon` method that will display the image, catch probability, and name of another Pokémon, with the species chosen at random from the 151 possibilities listed in `pokedex.csv`. This method should be called immediately after initializing the widgets, if the user runs away from a Pokemon, or if the user caught the previous Pokémon.
- Create a `throwBall` method that will attempt to capture the Pokémon using the probability value. The probability that a Pokemon will be caught can be determined using this formula:
  `min((catchRate+1), 151) / 449.5`
  (where `catchRate` is this Pokémon's catch rate from the csv file).
    - If the Pokémon is caught, it will store the Pokemon in the list of captured Pokemon, and call `nextPokemon` method to move onto the next wild Pokemon.
    - If it is not caught, it will display appropriate status message and let the user keep trying. If the number of balls reaches 0, it should call the `endAdventure` method.
- Create an `endAdventure` method that will display the caught Pokémon or the message that no Pokémon were caught.

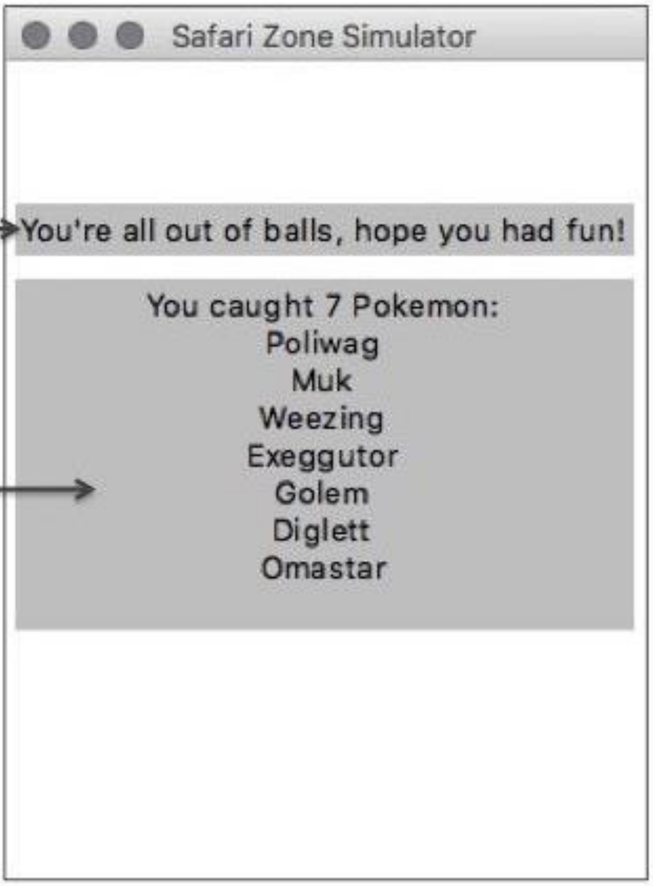
Your standard GUI should look like this:


These controls will look different depending on your OS. That's okay.
Should call the throwBall method
This label contains the image of the Pokémon. All images are provided in a subfolder named "sprites" where each one is the [pokedex#].gif of the appropriate Pokémon.
Safari Zone Simulator
Throw Safari Ball (30 left)
Run Away
You encounter a wild Pikachu
Your chance of catching it is 33%!
Tile of the window should be "Safari Zone Simulator"
Should reflect # of balls left
Should call the nextPokemon method
This label is the main status message space, it should print:
• "You encounter a wild [species]"
• "Aargh! It escaped!" if didn't catch.
This label should contain the calculated chance of catching the Pokémon, which should be:
min([catchrate] + 1, 151) / 449.5
Multiply this by 100 and round down to the nearest int to get a percentage.

Your end-of-adventure screen should look like this:

This message should always print when the adventure is complete.

This label contains the list of caught Pokémon. If appropriate, the same species may appear multiple times.
If 0 Pokémon were caught, it should say "Oops, you caught 0 Pokémon."



```
You're all out of balls, hope you had fun!
```

```
You caught 7 Pokemon:
Poliwag
Muk
Weezing
Exeggutor
Golem
Diglett
Omastar
```

Checklist (the Automated Feedback tests can't do much other than confirm that your file was named correctly and has no syntax errors for this assignment, so consider this a replacement for that system):

- ❏ Pokemon Class
  - ❏ Has instance variables for species name, dex number, catch rate, and speed
  - ❏ Has an overloaded __str__ method that returns the species name
  - ❏ Has any other instance variables/methods you think would be useful
- ❏ SafariSimulator __init__ method
  - ❏ Read in the information from pokedex.csv, and store it in some collection (list, dictionary, set, etc) of Pokemon objects.
  - ❏ Have some way of keeping track of the number of Safari balls left
  - ❏ Have some way of keeping track of the caught Pokemon
  - ❏ Initialize any other instance variables you might need
  - ❏ Call the nextPokemon method (after createWidgets) to start the first Pokemon encounter
- ❏ SafariSimulator createWidgets method
  - ❏ Create a Button for throwing a Safari Ball, and bind it to the throwBall method
  - ❏ Create a Label for the Pokemon sprite image
  - ❏ Create a Label for the Catch Probability
- ❏ SafariSimulator nextPokemon method
  - ❏ Pick a random Pokemon out of the 151 possibilities
  - ❏ Update the message label to "You encounter a wild [species name]"
  - ❏ Update the catch probability label to the correct percentage
  - ❏ Make a PhotoImage object for the sprite of the current Pokemon, and save it to some instance variable
  - ❏ Update the image label to use the given PhotoImage object
- ❏ SafariSimulator throwBall method
  - ❏ Generate a random number between 0 and 1 using random.random()
  - ❏ Use that number to determine whether the Pokemon was caught (see comments in throwBall for details)
  - ❏ Update the throwButton text to reflect the fact that one Safari ball was used
  - ❏ Change the message label to "Aargh! It escaped" if the Pokemon was not caught
  - ❏ Call endAdventure if this was the last Safari Ball
  - ❏ Call nextPokemon if the Pokemon was caught
- ❏ SafariSimulator endAdventure method
  - ❏ Display "You're out of balls, I hope you had fun!" in one of the labels
  - ❏ Display the number of Pokémon caught, and the list of their names, in another label
  - ❏ Remove all other buttons and labels so that the user can't keep playing

(*Available extra credit: up to 10 points*) **Additional Pokémon Safari Features**
Note: you may do as many or as few of these as you want, but they have to be done in the order specified below. This does not require a separate submission: all of your code should be submitted as a single `hw12.py` file, but you must include a comment at the top of the file specifying which Extra Credit parts you have completed.

(Part 1: 3 points possible) In the real Pokémon Safari, Pokémon had a chance to run away from the trainer. Add this feature to the simulator, by calculating a Pokémon's chance of running from the trainer and calling the nextPokemon method if the Pokémon runs (this happens after the trainer's turn: so if the Pokémon is caught then it doesn't get a chance to run). The method for calculating the likelihood of running is explained on this page (http://www.dragonflycave.com/safarizone.aspx), but to summarize, the chance of running is going to be (2*[pokemon's speed]/256) for this part of the extra credit (though Part 2 will change this value).

The .csv includes the "actual speed" stat that you should use in your calculation (basically, I calculated this assuming that each Pokémon is the same level). You should add another text label below the capture probability label specifying the likelihood of the Pokémon running away. You must also have some way to tell the difference within the GUI between moving on to a new Pokémon because the current one was caught, and moving on to a new one because the current one ran away: how to do this is up to you. You may ignore the rock throw or bait if you're only doing part 1 of the extra credit.

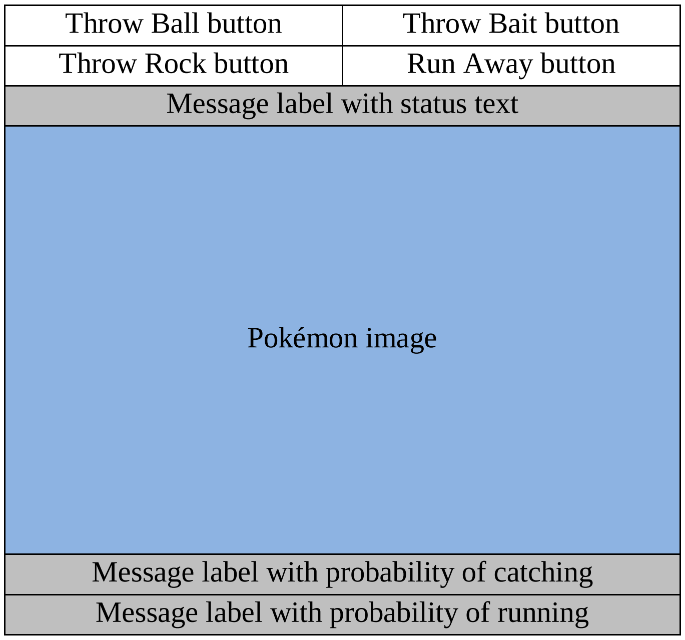
(Part 2: 5 points possible) In the real Pokémon Safari, the trainer has the opportunity to throw bait and/or rocks at the Pokémon, affecting its likelihood of running and capture rate. How each probability is affected is explained on this page (http://www.dragonflycave.com/safarizone.aspx), but to summarize:

- You have infinite rocks and bait to throw, but doing so will take up your turn (so the Pokémon gets a chance to run away before the player can throw another ball). Rocks make the Pokémon more likely to be caught but also more likely to run away, and Bait does the opposite.
- Throwing a Rock makes the Pokémon angry for a random number of turns between 1 and 5 (if it was already angry, then the number of turns that it has left being angry increases by that number)
- Throwing a Bait makes the Pokémon eating for a random number of turns between 1 and 5 (if it was already eating, then the number of turns that it has left being angry increases by that number)
- A Pokémon can't be simultaneously eating and angry, so doing one replaces the other (in reality, it's a bit more complex than this, but you can ignore that if you want).
- If a Pokémon is angry, then:
  - Their chance to be caught doubles with each rock thrown, up to a maximum of 151/449.5, or about 33%.
  - Their chance of running away each turn is set to min(255, 4*[pokemon's speed])/256 (this does NOT get higher with each additional rock thrown while already angry).
  - If the Pokémon ever stops being angry (because they run out of "angry turns" left), then the run and catch chances reset to the original values.
- If a Pokémon is eating, then:
  - Their chance to be caught is halved with each bait thrown.
  - Their chance of running away each turn is set to int([pokemon's speed]/2)/256 (this does NOT get lower with each additional bait thrown while already eating).

- If the Pokémon ever stops eating (because they run out of "eating turns" left), then the run and catch chances reset to the original values.

You should add a "Throw Rock" and "Throw Bait" buttons and appropriate callback methods to respond to each action. The new probabilities of capture and running away should be reflected on each appropriate label message.

(Part 3: 2 points possible) So far, our program uses the simple pack() layout manager and everything is laid out top-to-bottom. It would look nicer if everything could be laid out using the grid() layout as sketched out below:


Throw Ball button
Throw Bait button
Throw Rock button
Run Away button
Message label with status text
Pokémon image
Message label with probability of catching
Message label with probability of running

You can modify the screen width in the DO NOT MODIFY area of the code if necessary to fit all of the buttons in neatly.