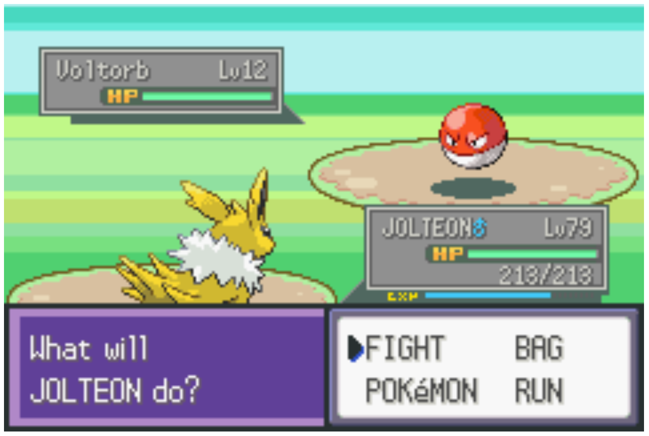
Computer Science 12 **Name: ____________________**
Pokemon Game **Date: ____________________**

Create a Pokemon game using object oriented programming. Your game will simulate a pokemon match. The user will battle against the computer. You will each start with a team of 4 pokemon. The battle is won when either the user or computer has no pokemon left to battle.



What will JOLTEON do?

Several csv files have been provided.

The **pokemon.csv** file contains information for each pokemon: number, name, type, hp, attack, defense.

The **moves.csv** file contains information for each move: name, type, power.

The amount of damage a move does against a defending pokemon depends on the attacking pokemon's level, attack, defence, the move's power and a multiplier.

$$Damage = \left(\left(\frac{2 \times Attacker_{Level}}{5} + 2\right) \times Move_{Power\ Level} \times \frac{Attacker_{Attack\ Level}}{Attacker_{Defense\ Level}} \div 50 + 2\right) \times Multiplier \times 5$$

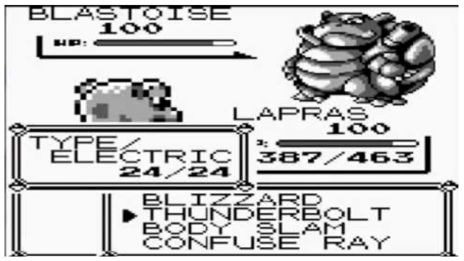
The multiplier depends on the type of the move and the type of the defending pokemon.

The **multiplier.csv** file contains information to determine the multiplier.

Damage lowers the opponent's hp. When a pokemon's hp reaches 0 it is said to have feinted and can no longer be used in battle.

Game

At each turn in the game – the player must choose to Fight, Bag, Pokemon or Run



Fight

When you select fight, you will get the option of choice which move to use. After you select the specific move, damage will be calculated and applied. A summary will show

Bag

Enemy SQUIRTLE used TAIL WHIP!

Each player's bag will be initialized to contain status condition healing items. Some examples of these items are Antidote, Awakening, Burn Heal, Full Heal, Ice Heal, Paralyze Heal, Persim Berry. Each of these items heal certain status conditions.

Item	Status Condition
Antidote	Poison
Awakening	Sleep
Burn Heal	Burn
Full Heal	All
Ice Heal	Frozen
Paralyze Heal	Paralysis
Persim Berry	Confusion

Status Condition

Certain moves cause certain status conditions. When a pokemon has a status condition they are prevented from fighting until their status condition is healed.

The **special_status_pokemon.csv** file contains information about which moves cause which status conditions.

Pokemon

When you select pokemon you can call another pokemon from you team to battle – as long as they have not feinted.

Run

The user will select run to exit the game (or play again).

GUI

Your game will have a gui. You can choose to build your gui using the Tkinter library. When you are using a gui, events (button clicks, key presses) control the flow of your program. Consider disabling buttons when the user cannot use them. To disable a button change the `state` of the button to `DISABLED`.

HP

A pokemon's HP will be visually indicated with a health bar on the gui. Consider using a `ProgressBar` widget for the health bat.

```
from tkinter import *
from tkinter.ttk import Progressbar

def add():
    if progress_bar['value'] < 100:
        progress_bar['value'] += 10

root = Tk()
root.geometry('500x300')
# creating a progress bar 100 px long
progress_bar = Progressbar(root, length = 100, orient = HORIZONTAL, mode = 
'determinate')
progress_bar.place(x = 200, y = 100, width = 100, height = 50)

# making a button that when click updates progress_bar
button = Button(root, text = 'Click Me')
button.place(x = 200, y = 200, width = 100, height = 50)
button['command'] = add

root.mainloop()
```

Planning

Create a UML diagram that gives an overview of each class involved and the relationship between them.

Expectations & Achievement Categories		Level 4	Level 3	Level 2	Level 1
Communication	**Class docstrings include description and description of each instance variable.** **Function docstrings include type contract, description, sample calls.**	Docstrings are well written and clearly explain how the function is used.	Docstrings are mostly well written and mostly explain how the function is used.	Docstrings are provided but missing criteria and/or unclear.	Minimal/no docstrings.

Knowledge	**Approaches software design using object oriented programming (as shown in UML)**	UML class diagram is clear and complete. Classes are	Functions are used and mostly cohesive. Minimal duplication of algorithms and/or data structures. Conditionals, lists and loops (for and while) are mostly used appropriately. User interface is mostly clear and easy to use. Program is mostly robust and mostly handles errors, try/except mostly used appropriately.	Few functions are used. Some duplication of algorithms and/or data structures. Conditionals, lists and loops (for and while) are somewhat used appropriately. User interface is somewhat clear and easy to use. Program is somewhat robust and somewhat handles errors, try/except somewhat used appropriately.	Program does not use functions effectively. Significant duplication of algorithms and/or data structures. Conditionals, lists and loops (for and while) are not used appropriately. User interface is not clear and easy to use. Program is not robust and does not handle errors, try/except not used appropriately.

Thinking and Application	**Source code well designed using a modular design.**	Functions are highly cohesive (do one thing) and designed for reusability. No unnecessary duplication of algorithms and/or data structures. Conditionals, lists and loops (for and while) are used well and appropriately. User interface is clear and easy to use. Program is robust and handles errors, try/except, used appropriately and specifically.	Functions are used and mostly cohesive. Minimal duplication of algorithms and/or data structures. Conditionals, lists and loops (for and while) are mostly used appropriately. User interface is mostly clear and easy to use. Program is mostly robust and mostly handles errors, try/except mostly used appropriately.	Few functions are used. Some duplication of algorithms and/or data structures. Conditionals, lists and loops (for and while) are somewhat used appropriately. User interface is somewhat clear and easy to use. Program is somewhat robust and somewhat handles errors, try/except somewhat used appropriately.	Program does not use functions effectively. Significant duplication of algorithms and/or data structures. Conditionals, lists and loops (for and while) are not used appropriately. User interface is not clear and easy to use. Program is not robust and does not handle errors, try/except not used appropriately.