Spike: 10
Title: Spike – Tactical Analysis with PlanetWars

Author: Adonias Pedro, 104463681

Goals / deliverables:

- Code
- Report

Technologies, Tools, and Resources used:

- Latest Version of the Visual Studio Code or the Python IDE
- GeeksforGeeks: https://www.geeksforgeeks.org/introduction-to-pyglet-library-for-game-development-in-python/
- PythonDocs: https://docs.python.org/3/tutorial/modules.html
- Computer/Laptop

Tasks undertaken:

- Download and install Git bash terminal
- Download and install the latest version of the Python IDE or Visual Studio Code
- Use the git bash terminal for and running the code while the code is inside a folder by using the python command
- The whole program was object-oriented while each file had their own role. Which was the main, entities, map_generator, planet_wars_draw, planet_wars, players, & and folders to form the whole GUI
- The program has a bot folder that stores all the bots that can be used in the program and 2 bots were created and implemented in the program
- The bot finds a target planet with the least amount of ships and the second bot shoots to the strongest planet
- The command to use one of the bots in the program is: `python main.py --gui -p Blanko Bot2 -m map001`

What we found out:

The tactical analysis being used in the program bots are detecting and shooting a planet the minimum number of ships and detecting and shooting the strongest planet. And the tactical analysis showed in the outcome.

The first bot tactical analysis is done by:

- Sorting through the available planets
- Looking for a target with the least amount of ships

- Launching a new fleet if more than 10 are visible and only launching 75% of the ships that are available

```python
from random import choice

class Bot(object):
    def update(self, gameinfo):

        src = max(gameinfo._my_planets().values(), key=lambda p: p.ships)

        #Find a target planet with the minimum number of ships.
        dest = min(gameinfo._not_my_planets().values(), key=lambda p: p.ships)
        #Launch new fleet if there's enough ships
        if src.ships > 10:
            gameinfo.planet_order(src, dest, int(src.ships * 0.75))
```

The second bot tactical analysis is done by:
- First sorting through the available planets
- Detecting the strongest target source planet
- Launching only 75% of the ships that are available

```python
from random import choice

class Bot2(object):
    def update(self, gameinfo):

        src = max(gameinfo._my_planets().values(), key=lambda p: p.ships)

        #Detect the strongest target planet
        dest = max(gameinfo._not_my_planets().values(), key=lambda p: p.ships)

        #Launch new fleet if there are enough ships
        gameinfo.planet_order(src, dest, int(src.ships * 0.75))
```

Table of Comparsing bots performance:

Match	Results
Bot vs Rando	Bot had more of advantage against Rando because of its targeted strategy of launching ships to the planet with its ships applied to the planet.
Bot2 vs Rando2	Rando2 had a disadvantage against Bot2 due to its number of ships it had applied to it, while Bot2 had more advantage due to its aggressive strategy of attacking the stronger planet.
Bot vs Bot2	Bot2 was still at dominance as it for it's plan to the destroy a great number of ships while Bot was having the hard of destroying more ships

Rando vs Rando2	Rando2 had a disadvantage towards Rando due to attacking fewer ships while Rando was leveraging its strategy of attacking more random selected ships.
Bot2 vs OneSlowMove	Bot2 was still able to dominate again against OneSlowMove due to OneSlowMove having a disadvantage of only sending ships greater than a counter that increases by 0.5 each time.