Pet simulator

# Aims

The pet simulator will simulate looking after and training a pet. You will then be able to compete in a competition. They will then be able to use the money from the competition to buy items to get better food and toys.

* To be able to create a digital pet
* Look after the pet’s simulated stats
* To train the pet
* To battle other pets
* To have more then one pet at once
* To be able to save
* To buy items from a store
* Easy for a user to use

# Analysis

### Command-line vs GUI

A command-line interface could be easier to program, but a graphical interface will be easier to use by the end user. It will also be easier when there are a large amount of items in the store as it is easier to select an item.

### Imports and principles

The program will use OOP to store and save. The program will use some functions of python to complete its aims.

### Pickle

Will allow the game to be saved and opened. The advantage of OOP means it is easy to save the objects as the player will be contained in within the player object and the store in its own project.

### Tkinter

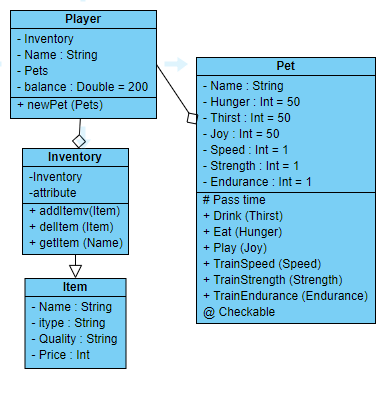
is a graphical user interface which makes it easier for a user to use the program then a command line interface.

### OOP

Object oriented programming will allow for multiple pets to be created and stored at the same time, and store the inventory of the player. It allows for easy expansion of the program for more features and saving the program.

# Design

## Class diagram



### Description of key functions

Private checkable function will look to see if the pet’s stats are the minimum required to be able to complete actions like train and battle.

Pass time will simulate the needs of the pet increasing as an action is completed by the pet.

New pet will create a new object contained in the pets list.

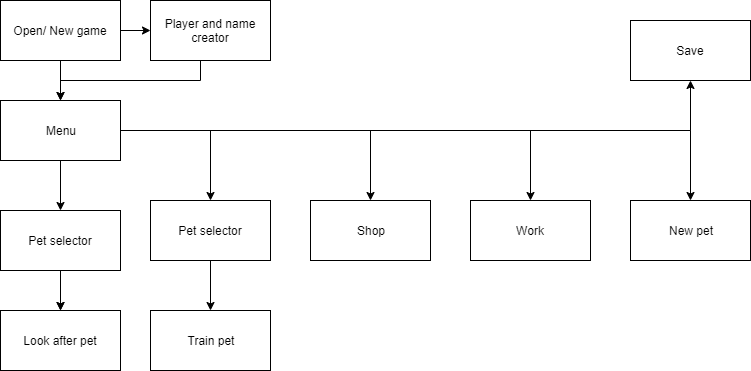
### Objectives of key GUI

Look after pet will simulate the player looking after the pet, they will use the objects in their inventory to complete the action.

Shop will allow the user to buy from the shop. The data from the shop will be a constant when a new game is created and will be saved along with the player data.

Work will allow the user to compete against other pets these pets are created by using data from the constant files which are independent of the save file therefore if it was to change then different pets could be competed with. They can then ‘fight’ against speed, strength and endurance if the player wins then they will earn money.

## GUI flow diagram



## GUI design

The graphical interface uses a grid layout below are some of those designs:

### Look after GUI

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pet (0,0) | {name} |  |  | Hunger | {value} |
| Joy | {value} |  |  | Thirst | {value} |
| Feed | | Drink | | Play | |
| Name: | {item name} |  |  |  |  |
| Quality | {value} | <Repeats |  |  |  |
| Use me! | |  |  |  |  |

### Battle GUI

|  |  |  |  |
| --- | --- | --- | --- |
| Pet | {name} | PC | {name} |
| Select | | Select | |
| Compete in? | Strength | Speed | Endurance |
| Player: | {value} | Status (win or lose) | |
| PC: | {value} | Balance | {value} |

# Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Data | Expected results | Results |
| To be able to create a digital pet. Check look after interface | Player: Nick  Pet: Champ | GUI will show player with the stats as 50 and name as Champ | As expected, figure 1 |
| Look after the pet’s simulated stats. | As above  Drink: Juice quality 5 | GUI will show thirst 46 and hunger and joy will show 51 | As expected, figure 2 |
| To battle other pets | Champ as created  Easy pet  Strength | You: 1  Comp: 10  You lose. | As expected, figure 3 |
| To train the pet by selecting strength | As test 1 | Strength 4  Others still show 1 | As expected, figure 4 |
| To have more then one pet at once | As test 1  New pet function with name Ash | Will now show in selector and be able to be used in training | As expected, figure 5,6 |
| To be able to save | As test 1  Save name game | Should reopen as the same. | As expected, figure 7 |

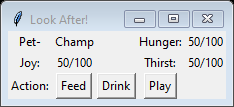


Figure 2

Figure 1

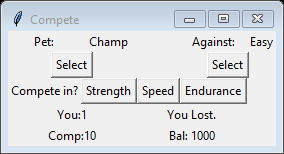
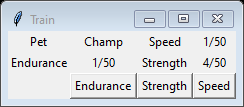
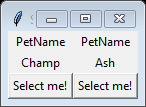


Figure 4

Figure 3

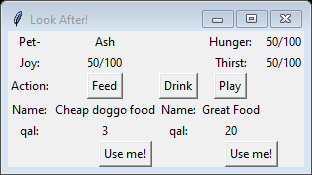


Figure 7

Figure 5

Figure 6

# Critique

The program worked well. The use of the graphical interface made testing the program much simpler and helped avoid issues. Testing against the aims of the program helped ensure it met the goals I had established. Use of object orientated design made the data structure much simpler and reduced the size of the program as well as prevented old data from being kept in memory.

More features could have been added to the program, for example you could have had items for the pet to wear. The interface could have been improved greatly adding graphical images to the program would have made it easier for users to distinguish which pet they were interacting with.

The use of a graphical interface was the most time-consuming aspect of the program, the design and implementation took time, the Tkinter design took time to understand and its use of its own variables and the way it uses functions is not standard to python and I spent several hours trying to understand how it works.

The pets competing could have been improved by a tri-compete function where all the ‘skills’ of the user were tested and a overall winner would be established. The pets selection process could have been improved. There are separate windows for each of the graphical interfaces to select a pet, a function could have been used with an argument to tell the function which window to open.