CS202: PROGRAMMING PARADIGMS & PRAGMATICS

Semester II, 2022 – 2023

Lab 3: Python Programming Exercise

- **Aim**: Create Bunny Breeding Game in Python.

• Let's get started!

- Create a directory structure to hold your work for this course and all the subsequent labs:
  - Suggestion: `CS202/Lab4`

• Bunny Breeding Game:

- Write a program (`BunnyBreeding.exe`) that creates a `Linked List` of Bunny `Objects`
- Each bunny object must have:
  - **Gender**: Male / Female (Random at creation: 50/50)
  - **Color**: White, Brown, Black or Spotted
  - **Age**: 0 – 10 (years old)
  - **Name**: Randomly chosen at creation from a list of bunny names
  - **Radioactive_Mutant_Vampire_Bunny**: True / False (Random at creation, 2% chance of true)
- At program initialization 5 bunnies must be created and given random colors.
- After each 'turn', the bunnies age by 1 year.
- So long as there is at least one male age 2 or older, for each female bunny in the list age 2 or older; a new bunny is created each turn. (i.e. if there was 1 adult male and 3 adult female bunnies, three new bunnies would be born in that turn)
- New bunnies born should be the same color as their mother.
- If a bunny becomes older than 10 years old, it dies.
- If a radioactive mutant vampire bunny is born then each turn it will change exactly one non-radioactive bunny randomly into a radioactive vampire bunny. (If there are two radioactive mutant vampire bunnies, two bunnies will be changed each turn and so on...)
- Radioactive vampire bunnies are excluded from regular breeding and do not count as adult bunnies.
- Radioactive vampire bunnies do not die until they reach age 50.
- The program should print a list of all the bunnies in the colony each turn along with all the bunnies' details, sorted by age.
- The program should also output each turns' events such as

```
Bunny Thumper was born!
Bunny Fufu was born!
Radioactive Mutant Vampire Bunny Darth Maul was born!
Bunny Julius Caesar died!
```

- The program should write all screen output to a file (**BunnyBreeding.txt**).
- If the bunny population exceeds 1000 a food shortage must occur killing exactly half of the bunnies (randomly chosen)
- When all the bunnies have died the program terminates.

Suggestions:

- Use OOP Concepts if/wherever possible.
- *Write the program in an incremental fashion, so that even if you run out of time, you can still submit a version of the program that does something useful!*
  - *For example, start with creating the Bunny object followed by incorporating code that 'Creates' new bunnies. Then start thinking of implementing changes between 'Turns' like ageing and breeding!*

Submitting your work:

- All source files and class files as one tar-gzipped archive.
  - When unzipped, it should create a directory with your ID. Example: **P2008CS1001** (NO OTHER FORMAT IS ACCEPTABLE!!! Case sensitive!!!)
  - ***Negative marks if the TA has to manually change this to run his/her scripts!!***
- Source files should include the following: (Case-Sensitive file names!!)
  - Bunny.pl
  - BunnyBreeding.pl
  - Any other supporting or required files
- ***Negative marks for any problems/errors in running your programs***
- *If any aspect of the game / rules are confusing, make an assumption and state it clearly in your* **README** *file! This file should also have instructions on how to use/run your program!*
- Submit/Upload it to Google Classroom