AST-Assignment #2

Prof: Nico Hochgeschwender
T.A.: Samuel Parra: samuel.parra@smail.inf.h-brs.de

May 2020

Last assignment, we designed and implemented a system that behaved like Facebook. Now, it's time to visualize how the entities of your program interact with one another. In this assignment, we will use the *Graphviz* library to generate a graph that reflects the state of your system at a specific time. Your visualizer should be able to generate from the command line a "snapshot" of the objects in the system and how those objects connect to others.

1 Useful concepts and resources

Before we go into the specifics, let's look at a couple of concepts and resources that will help you with this assignment. These are simple introductions that you can skip if you are familiar with the concepts, but it doesn't hurt to check them out just in case!

1. Object Introspection: `https://book.pythontips.com/en/latest/object_introspection.html`
2. UML Object Diagrams: `https://www.tutorialspoint.com/uml/uml_object_diagram.htm`
3. Graphviz Documentation: `https://graphviz.readthedocs.io/en/stable/manual.html`

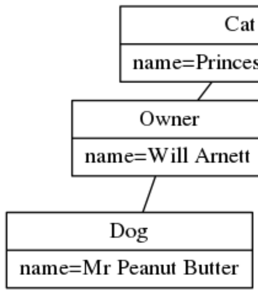
2 Example

Before introducing the requirements of the assignment, let's look at this short and simple example for you to get an idea of what you have to deliver. Consider this code:

```
import Dog, Cat, Owner

peanut_butter = Dog("Mr Peanut Butter")
peanut_butter.bark()
```

```
princess_carolyn = Cat("Princess Carolyn")
princess_carolyn.meow()

will = Owner("Will Arnett")
will.add_pet(peanut_butter)
will.add_pet(princess_carolyn)

draw_owner_with_pets(will)
```

In this program, the function `draw_owner_with_pets()` will generate this diagram using introspection and the *graphviz* library:



This way, we get a picture of the objects in the system, their values and their relationships.

3 Requirements

1. Create a module called `grapher` that generates an object diagram.
2. The nodes in the graph should follow the standard notation of UML object diagrams. it should show the object's names and their current attributes and values, as well as links to other objects.
3. You must use introspection in order to get the name of the attributes and their values that appear in the diagram. This should *not* be hard coded.
4. you must use the system you developed for assignment #1.
5. you must develop the `grapher` module on a branch called grapher using git. Merge your changes with the master when you finish.
6. You must save the generated diagram like `diagram.png` on the `./documentation` folder.

4 assignment submission

- This assignment can be submitted by individuals or pairs only.

- The deadline for this assignment is **01.06.2020 - 08:00 CEST**. Please *do not* make more commits on your GitHub repository after the deadline.
- All the code should be place on the `./code` folder.
- Place all CRC cards, UML class diagrams, and object diagram in a folder called `./documentation`.
- Submission is made through GitHub. Please, try to commit your progress constantly as commits after the deadline won't be taken into consideration.

5 Notes

- We expect a command line interface (CLI) to use your program, not a GUI.
- Only one member of the team has to open a repository.
- Use Python 3.
- Your code should be readable, well commented, organized and representative of your final model. Please, take your time to create **docstrings**.
- You should be able to explain your code and your design decisions.
- Update `requirements.txt` before submitting.
- You can contact the T.A. for any questions you may have with the email address provided at the beginning of this document. Please, write [AST-Assignment #] in the subject and your name(s) in the body of the email.

6 Resources

- Creating docstrings: `https://www.geeksforgeeks.org/python-docstrings/`