Final Project: Tamagotchi Pet Game

Prompt:
Create a virtual pet care game where players can choose a cute pet and then feed, train, or play-with it. The game is based on a simple game loop, where at the beginning of each loop, the game notifies the player of the current pet status (hungry, sleepy, bored, etc.) The player should then select the corresponding actions to improve the status of the pet. After each action, the game loop is repeated and the pet values are updated. The status should be displayed along with the corresponding messages as well as the menu of actions.

Design Requirements:
Implement the Pet class

- Create the base Pet class. It should manage variables that keep track of the pet's state, such as how hungry/sleepy/bored/happy the pet is currently.
- Include a minimum of three (3) interaction functions that change those variables. For example, playing with a pet will make it less bored but might make it more hungry.
- Include a function that updates the pet's state values to reflect the passage of time. For example, you can make a nextHour() that will make the pet more hungry, bored, and sleepy by increasing those variables by some amount. If any of the variables are above some threshold (such as 40 out of 100), the function should output the corresponding message such as "<Pet name> is hungry!"
- Each Pet class should have a method to save and load all of its data onto or from a file, to allow the player to continue with an existing pet.

Implement Types of Pets

- Create at least three (3) new child classes that inherits from the above Pet class.
- Each new Pet type (class) should be unique, from printing a unique message for each interaction, to changing how fast each pet will get hungry or bored. You can add other interactions like playing, battles, or more (be creative!).
- If a Pet class has extra variables, override the file save and load methods to make sure the new variables are updated in the save file.

Implement the Game Loop

- After the game starts, the player should be presented with a menu to load a pet, or to create a new pet from the list of available pet types to choose from (perhaps with a brief description).
- After choosing a new pet type, the player should be asked to give it a name and it may be assigned some random starting attributes. Any further game message for the pet should address it by its new name.
- When loading a pet, the game should correctly restore the name, stats, and status of the pet.
- The game will then present the player with a list of possible interactions (play, feed, …), save or load files, or exit the game.
- It should handle player actions, update the pet status, and repeat until the player exits.

Program Requirements:

- Must use file I/O for saving and loading pet data.
- Must implement a pet base class and at least three (3) different pet classes derived from the base class, with each pet having some unique attributes, abilities, and/or interactions.
- Must be at least three (3) different player interactions for each pet.

Final Project: Tamagotchi Pet Game

- Consider use polymorphism for easier programming.
- Be creative. Try using text (ASCII) art to create an aesthetic menu system and draw the shape of a pet

Submission:

1. **Prepare a Preliminary Project Report**. This report should include a description of how the program is (or will be) designed, including the design of the classes and their inheritance relationships. Be sure to describe the main functionality (in the form of pseudo code or step-by-step instructions) of the program. Please also provide the menu options and their use cases for your program. You can refer to ZyBooks 11.10 to learn how to use UML to describe your OOP design.
2. **Prepare a short two to three-page report** explaining how the program was designed and developed, what classes were used, any special instructions to operate the program, and how the final implementation differed from the plan. **Submit the report on Blackboard.**
3. **Submit your source code as a single .zip file on Blackboard.**
4. **Provide a link to a video clip (less than 5 minutes in length) demonstrating all the above required functionality. Submit this to Blackboard.**

Grading:

- Mid-term report **25%**.
- Peer-grading (based on the uploaded video) **25%**.
- Final product graded by instructors and TAs **50%**.
    - Final report 20% (out of 50)
    - Demo via video clip 40% (out of 50)
    - Source code 40% (out of 50)