GLOBALRAIN

Developer: Tashyra Adams

Date: 02/01/2024

IT 145 Global Rain Summary Report Template

Pseudocode

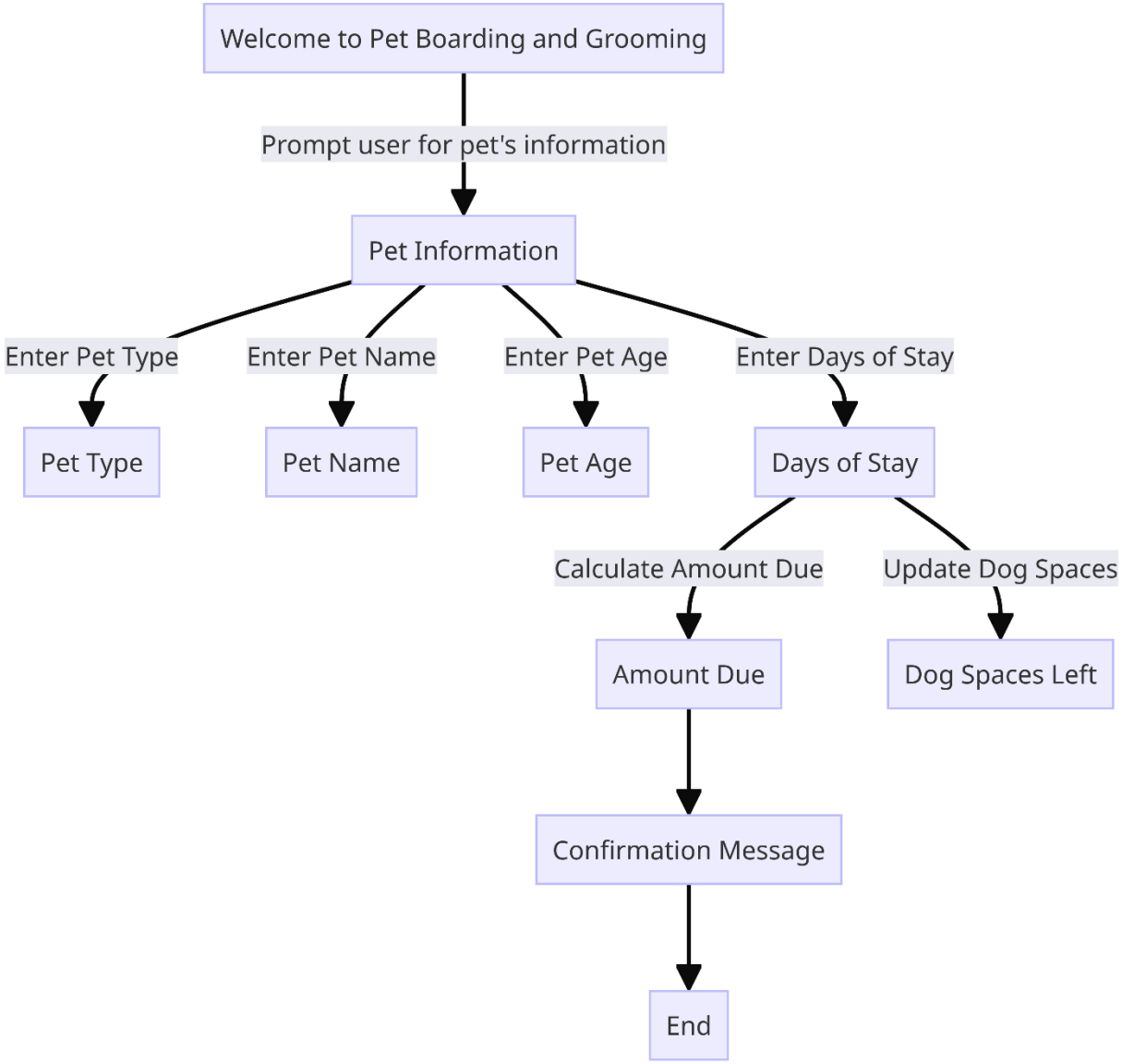
When you are done implementing the Pet class, refer back to the Pet BAG Specification Document and select either the pet check in or check out method. These methods are detailed in the Functionality section of the specification document.

Write pseudocode that lays out a plan for the method you chose, ensuring that you organize each step in a logical manner. Remember, you will *not* be creating the actual code for the method. You do *not* have to write pseudocode for both methods. Your pseudocode must not exceed one page.

```
BEGIN checkIn() method

    DISPLAY "Welcome to Pet Boarding and Grooming!"
    DISPLAY "Please enter your pet's information:"

    PROMPT user for pet type
    READ petType from user input

    PROMPT user for pet name
    READ petName from user input

    PROMPT user for pet age
    READ petAge from user input

    PROMPT user for number of days of stay
    READ daysStay from user input

    SUBTRACT 1 from dogSpaces (assuming a dog is checked in)

    CALCULATE amountDue as daysStay * $20 (assuming $20 per day)

    DISPLAY "Your pet, " + petName + ", has been checked in successfully!"
    DISPLAY "Dog spaces left: " + dogSpaces
    DISPLAY "Amount due: $" + amountDue

END checkIn() method
```

Flowchart

Based on the pseudocode you wrote, create a flowchart using a tool of your choice for the method you selected. In your flowchart, be sure to include start and end points and appropriate decision branching, and align the flowchart to the check in or check out process. Your flowchart must be confined to one page.



OOP Principles Explanation

Briefly explain how you applied object-oriented programming principles and concepts (such as encapsulation, inheritance, and so on) in your software development work thus far. Your explanation should be one paragraph, or four to six sentences.

In developing the Pet Boarding and Grooming software, object-oriented programming (OOP) principles were integral. Encapsulation was implemented by encapsulating data within the Pet class, allowing attributes like petType, petName, petAge, daysStay, dogSpaces, and amountDue to be accessed and modified through getter and setter methods, ensuring data integrity and abstraction. Inheritance was utilized to inherit common behaviors and attributes from the parent class, promoting code reusability and minimizing redundancy across subclasses. Additionally, polymorphism was demonstrated through method overriding, where specific behaviors could be tailored to subclasses while adhering to a common interface. This OOP approach facilitated modular design, scalability, and maintainability of the software, ensuring it could easily adapt to evolving requirements and support future enhancements.