

**Developer**: Ephellon Grey

**Date**: 24 September 2021

# IT 145 Global Rain Summary Report Template

## Directions

Place your pseudocode, flowchart, and explanation in the following sections. Before you submit your report, remove all bracketed text.

## 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.

new Pet → get Pet Type

if *Boarding* of Pet Type has no *Availability* → Exit Program

Get Pet Name

if Pet Name in Database

“Update Pet Name’s information?”

if *Yes* → Get Pet Age, Pet Weight

if Pet Name not in Database

Get Pet Age, Pet Weight

Get Stay Duration

Get Grooming Data

Assign Pet Name a *Space ID*

Update *Boarding* of Pet Type

## 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.

Diagram

Description automatically generated

## 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.

Using the Specification Document as reference, I’ve already begun implementing a major OOP principle, “Abstraction,” describing how the program should work/behave. Following-up with that, “Encapsulation” is the next most obvious principle, as there are private and public data-types being created and accessed. Lastly (though not used on built-ins), “Polymorphism” is used to create different versions of the Pet class, allowing me to give different inputs to achieve desired results. “Polymorphism” also allows me to change a few lines of code to go from a purely testing environment to a “production” environment, simply by changing the main method into a Pet constructor.