

**Developer**: Vinessa Deliberto

**Date**: January 30, 2025

# 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.

BEGIN CheckInProcess

// 1. Collect Pet Details

PROMPT "Enter pet name:" → petName

PROMPT "Enter pet age:" → petAge

PROMPT "Enter pet type (Dog/Cat):" → petType

PROMPT "Enter number of days for stay:" → daysStay

// 2. Create the Appropriate Pet Object Using a Factory Method

// The factory method createPetObject() returns a new Dog object if petType equals "Dog",

// or a new Cat object if petType equals "Cat". No additional question is asked.

SET currentPet = createPetObject(petType)

// 3. Set Pet Attributes (Encapsulation)

CALL currentPet.setPetName(petName)

CALL currentPet.setPetAge(petAge)

CALL currentPet.setPetType(petType)

CALL currentPet.setDaysStay(daysStay)

// 4. Check Boarding Space Availability

IF petType equals "Dog" THEN

IF currentPet.getDogSpaces() equals 0 THEN

DISPLAY "No spaces available for dogs."

TERMINATE CheckInProcess

END IF

ELSE IF petType equals "Cat" THEN

IF currentPet.getCatSpaces() equals 0 THEN

DISPLAY "No spaces available for cats."

TERMINATE CheckInProcess

END IF

END IF

// 5. Calculate Base Amount Due

IF petType equals "Dog" THEN

SET amountDue = daysStay \* DOG\_BOARDING\_RATE

ELSE

SET amountDue = daysStay \* CAT\_BOARDING\_RATE

END IF

CALL currentPet.setAmountDue(amountDue)

// 6. Offer Grooming Option for Dogs (if stay > 3 days)

IF petType equals "Dog" AND daysStay > 3 THEN

PROMPT "Add grooming for $70? (Yes/No):" → groomingChoice

IF groomingChoice equals "Yes" THEN

CALL currentPet.setGrooming(true)

SET amountDue = amountDue + 70

CALL currentPet.setAmountDue(amountDue)

ELSE

CALL currentPet.setGrooming(false)

END IF

END IF

// 7. Update Available Boarding Spaces

IF petType equals "Dog" THEN

CALL currentPet.setDogSpaces(currentPet.getDogSpaces() - 1)

ELSE IF petType equals "Cat" THEN

CALL currentPet.setCatSpaces(currentPet.getCatSpaces() - 1)

END IF

// 8. Confirm Check-In

DISPLAY "Check-in successful for " + currentPet.getPetName()

DISPLAY "Total amount due: $" + amountDue

END CheckInProcess

## 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.

A diagram of a 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.

In my development, I applied OOPs (object-oriented programming principles by designing classes that encapsulate both data and behaviors. The Pet class encapsulates attributes like petType, petName, petAge, and others, ensuring that these properties are accessed and modified only through controlled methods (getters and setters). I also employed inheritance by having Dog and Cat classes extend the Pet class, which allowed me to inherit common properties while also introducing their unique attributes, such as grooming for dogs. This structured approach not only reduces code duplication but also improves maintainability, scalability, and clarity in the overall software design.