

**Developer**: Brandon Hobbs

**Date**: July 23, 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.

A Method for Pet Check-in

Main function ()

**Loop**

**Output** “Are you checking in a (c)at or a (d)og?”

**Store** entry in variable petType

**Continue** Loop if not ‘c’ OR ‘d’

//Need to check if the space for that pet type is available

**If** PetType is cat and spaces available for cat equals 0

**Warn** the user “No cat spaces available”

**Exit**

**Else** if dog spaces available equals 0

**Warn** the user “No dog spaces available”

**Exit**

**Loop**

**Output** “Is this a (n)ew customer or a (r)eturning customer”

**Store** entry in variable customerType

**Continue** Loop if not ‘n’ OR ‘r’

//Need to determine the pet’s name no matter if new or returning

**Output** “Please enter the pet’s name”

**Store** entry in variable petName

**If** new customer

**Set** pet name with variable petName

**Output** “Please enter the pet’s age”

**Store** entry in variable petAge

**Set** pet’s age with variable petAge

**If** petType is dog

**Output** “Please enter the pet’s weight”

**Store** entry in variable dogWeight

**Set** pet’s weight with variable petWeight

**Else** //returning customer

**Output** petAge

**If** petType is dog

**Output** dogWeight

**Output** “Does all of the data look correct?”

**Store** entry in variable dataUpdate

**If** dataUpdate is ‘No’

**Loop**

**Output** “What data do you need to update: (n)ame, (a)ge, (w)eight, (q)uit?

**Store** entry in variable dataUpdateType

**Case** statement

**Case** weight

**If** petType is dog

**Output** “Please enter the correct weight”

**Store** entry in dogWeight

**Set** dog weight with variable dogWeight

**Else**

**Output** “Weight is not needed for cats”

**Case** age

**Output** “Please enter the correct age”

**Store** entry in petAge

**Set** pet age with variable petAge

**Case** name

**Output** “Please enter the correct name”

**Store** entry in petName

**Set** pet name with variable petName

**Loop** until dataUpdateType is quit

**Output** “How many days to be board?”

**Store** entry in daysStay

**If** petType is dog and daysStay >=2

**Output** “Does the customer want grooming: (y)es or (n)o?”

**Store** entry in petGrooming

//Now we have all the required information; book the stay

**Call** method to

**If** petType is dog

**Set** grooming status with variable petGrooming

**Set** dogSpaceNumber

//Will need to implement the method in a way to overload via polymorphism. Better to have the method split instead of implementing tons of logic in the method and in main

**Set** amountDue passing dogWeight, daysStay, petGrooming

**Else**

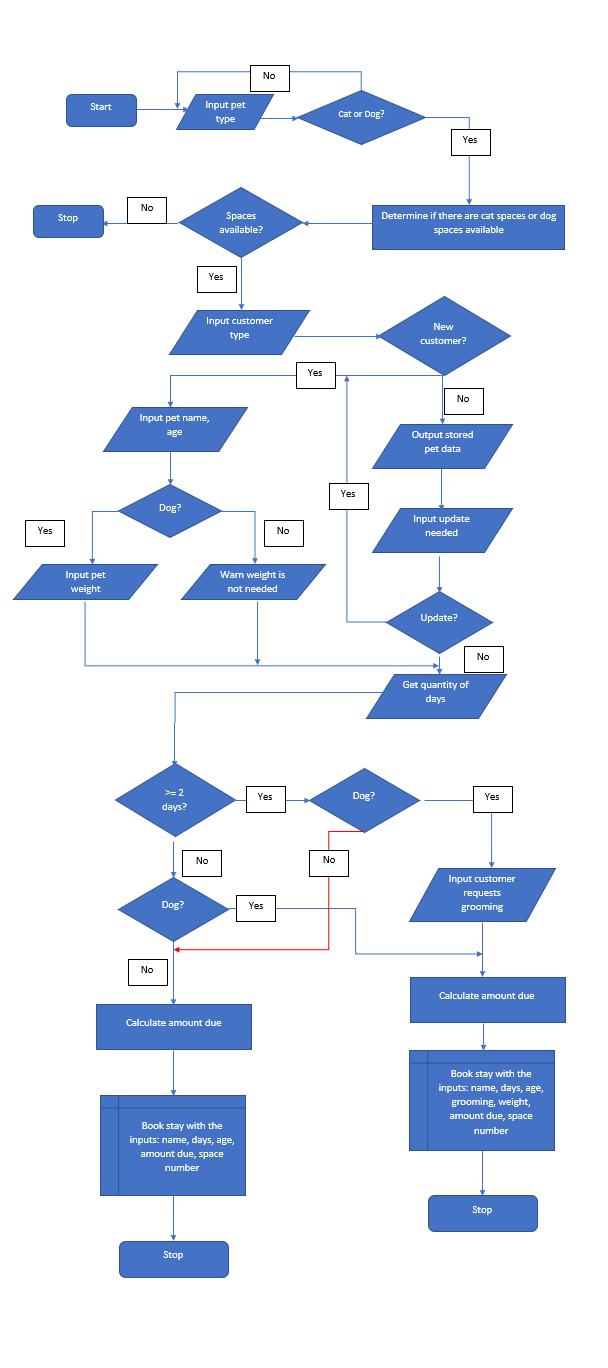
**Set** cat Space Number

**Set** amountDue passing daysStay

## 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.

**The flowchart for Check-in follows:**



## 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.

The 4 OOP Principles are Inheritance: children classes share the parent properties and take on their own unique properties; Encapsulation: implementation details are hidden and only methods are exposed (GET/SET often); Abstraction: programming codes are hidden away from the user; Polymorphism\*: "... a method or subclass can define its behaviors and attributes while retaining some of the functionality of its parent class... ".

Within this project I used inheritance twice: once to extend Pet to Cat and once to extend for Dog. This allows the Cat and Dog derived classes to share the common attributes, e.g., name, age, length of stay, etc. Encapsulation was achieved by implementing the GET/SET methods that access and mutate the object properties. For example, if I instantiate a Cat object named newCat and want to set its catSpaceNumber I cannot just say catSpaceNumber equals 15 (because that attribute is private); instead, the Public assessor method, setCatSpaceNumber, is called which then sets the attribute. Abstraction is used in the project because the end user has no idea how I am setting or manipulating the object attributes – they only interact with the object by the Abstracted methods. Polymorphism isn’t used in the Pet class but assuming I tried to make a list of all pets staying at the facility the list could contain Cats and Dogs and the JVM would assign the object properties correctly to each of the objects in the list.

These principals allow the code to be easier to read, visually cleaner, and be less cluttered. This cuts down on duplicate code, chances for bugs, and reduces the places needed to be re-coded if a bug is found.

\*<https://www.nerd.vision/post/polymorphism-encapsulation-data-abstraction-and-inheritance-in-object-oriented-programming>