Summative Programming Task (OOP & Console)

Design Problem

The **Al Qurum Veterinary Clinic** has hired you to help create a program for the Pets, Dogs, and Cats under their care.

Requirements and Constraints

- Solution must use OOP principles: objects, inheritance, ...
- All input and output will be through the console/terminal.
- The user interface allow for the user to:
  1. Enter in details for either a Pet, Dog, or Cat.
  2. Continue to input and store Pets until done (while loop)
  3. View all Pet information, sorted by age, formatted clearly
  4. Exit the program when finished.

Extra Challenge: Output all pet details to a file before exiting (JETS - BufferedReader, PrintWriter)

Summative Programming Task (OOP & Console)

The **Al Qurum Veterinary Clinic** asks that you are able to store for each pet, at a minimum, the following attributes:

1. **petAge: int** (must be >= 1)
2. **petName: String**
3. **type: String** (Dog = “Dog”, Cat = “Cat”, Pet = user-defined)
4. **breed: String** (only for Dog or Cat)
5. **houseTrained: boolean** (only for Dogs)
6. **hasClaws: boolean** (only for Cats)
7. **healthy: boolean**
8. **stray: boolean**
9. **ownerName: String** (“N/A” if a stray)
10. **ownerPhone: int** (-1 if a stray)

Consider: What needs to be public/private, methods, and how to ensure data security?

Summative Programming Task (OOP & Console)

Teacher / Developer Expectations

1. The entire program will be explained with one Flowchart (www.diagrams.net).
2. All class relationships will be explained with UML diagrams (www.diagrams.net).
3. The sorting algorithm used is explained using IB pseudocode.
4. The program is coded in Java using Eclipse.
5. Use of comment header, commenting, camelCase, and whitespace throughout.

File 1: designPlan.pdf (3 pages)
Flowchart, UML, Pseudocode

Java Source Code
Driver Class, Object Classes

File 6+: Optional
Additional Classes, Media, ...

You will be **submitting a minimum of 5 files in a .zip file**: (1) designPlan.pdf, (2-5) Java Class Files

A (not completed) Example with the Driver Class (VetMain):

A proof-of-concept example - many of the design requirements are not met (e.g. user input).

VetMain.java | Pet.java | Owner.java | DateOfBirth.java

```
1  import java.util.*;
2
3  public class VetMain {
4
5      public static void main(String[] args) {
6          Pet sooty, harry, strayDog, strayCat;
7          LinkedList<Pet> vetList = new LinkedList<Pet>();
8
9          sooty = new Pet ("Sooty", "Cat", "Black");
10         sooty.setDOB(22, 2, 2020);
11         sooty.setOwner("Mike", "Muscat", "Teacher", 8, 1, 1985);
12
13         harry = new Pet ("Harry", "Bedlington Terrier", "Grey");
14         harry.setDOB(17, 3, 2006);
15         harry.setOwner("Janice", "Canada", "Restaurant Owner", 19, 9, 1957);
16
17         strayDog = new Pet ("Range", "Dog", "Mix");
18         strayDog.setDOB(1, 12, 2021);
19
20         strayCat = new Pet ("Subaru", "Cat", "Orange Mix");
21
22         vetList.add(sooty);
23         vetList.add(harry);
24         vetList.add(strayDog);
25         vetList.add(strayCat);
26
27         while (!vetList.isEmpty()) {
28             System.out.println(vetList.poll().toString());
29             System.out.println();
30         }
31     }
32
33 }
```

Console

<terminated> VetMain [Java Application] /Users/mpoirier/.p2/pool/pl

```
--PET--
Name:   Sooty
Type:   Cat
Colour: Black
dob:    Feb 22 2020
--OWNER--
Name:   Mike
Addr.:  Muscat
Prof.:  Teacher
dob:    Jan 8 1985

--PET--
Name:   Harry
Type:   Bedlington Terrier
Colour: Grey
dob:    Mar 17 2006
--OWNER--
Name:   Janice
Addr.:  Canada
Prof.:  Restaurant Owner
dob:    Sep 19 1957

--PET--
Name:   Range
Type:   Dog
Colour: Mix
dob:    Dec 1 2021
--OWNER--
None (Stray)

--PET--
Name:   Subaru
Type:   Cat
Colour: Orange Mix
dob:    ?
--OWNER--
None (Stray)
```

Note: For this example I also created a **DateOfBirth** class to handle the Day/Month/Year.

DP Computer Science: Programming Task Rubric

	Design Flowcharts, Sketches, Prototypes, Pseudocode	**Readability** Documentation, Conventions, Comments	**Application** Meeting the Success Criteria	**Computational Thinking** Problem Solving and Algorithmic Design
4	Developed a **highly effective** design process, which **consistently** takes into consideration the **success criteria** of the end users in order to develop a **complete** solution.	The documentation is **exceptionally clear**, applies coding conventions **correctly**, and comments code **completely and consistently**.	The product **meets and exceeds** the success criteria, by **consistently applying** programming concepts successfully towards developing a **complete**, **innovative** and **creative** product.	The product and documentation show evidence of an **independent**, **thorough,** and **complete** understanding of **computational thinking skills**, problem solving, and algorithm design.
3	Developed an **effective** design process, which **generally** takes into consideration the success criteria of the end users.	The documentation is **generally clear**, applies coding conventions **accurately with limitations**, and comments code **substantially**.	The product **meets** the success criteria, by **applying** programming concepts towards developing a **complete** product.	The product and documentation show evidence of a **substantial** understanding of **computational thinking skills**, problem solving, and algorithm design **with minimal assistance**.
2	Developed a **moderately effective** design process, which **occasionally** takes into consideration the success criteria of the end users.	The documentation **lacks clarity**, applies coding conventions **inaccurately**, and comments code **inconsistently**.	The product **partially meets** the success criteria, and **inconsistently applied** programming concepts towards developing a product with **noticeable errors**.	The product and documentation show evidence of a **partial** or **incomplete** understanding of **computational thinking skills**, problem solving, and algorithm design **with moderate assistance**.
1	Developed an **ineffective** design process, which **rarely** takes into consideration the success criteria of the end users.	The documentation is **unclear or partially incomplete**, applies coding conventions **incorrectly**, and comments code **rarely**.	The product **does not meet the success criteria**, and is **ineffective at applying** programming concepts with **significant errors**.	The product and documentation show evidence of **a substantial misunderstanding** of **computational thinking skills**, problem solving, and algorithm design, and **requiring considerable assistance**.

Adapted from the IBO DP Computer Science IA Expectations and Guide
Updated: November 9, 2020