INFO1113/COMP9003 Week 3 Tutorial

Classes, Encapsulation, Objects and I/O

Classes

Java heavily uses the concept of a class as part of its language. A class can be considered a template or a blueprint of an object. When defining a class in java you define properties that will be associated with the objects of that class.

```java
public class Dog {

        private boolean isGoodBoy = true;
        public String name;
        private int age;

        public Dog(String dogName, int dogAge) {
                name = dogName;
                age = dogAge;
        }

        public void bark() {
                System.out.println("Woof!");
        }

        public boolean isGoodBoy() {
                return isGoodBoy;
        }

        public void chewThings() {
                isGoodBoy = false;
        }

}
```

We can also use the unified modelling language (UML) to illustrate a class to assist with designing our application. The class has been represented as a UML Class Diagram. The class diagram shows the attributes, methods and access modifiers of the class.

Figure 1 represents the Dog class with UML. We are able to annotate the access modifiers with + `public`, - `private` and the two not seen # `protected` and ~`package` access modifiers.

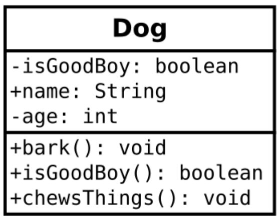

Figure 1: UML Class Diagram of the `Dog` class

Discussion

Consider the following class definition:

```
public class Book {

    public String title;
    public String author;
    public int year;
    public String url;

}
```

- What do each of the components/keywords of this class mean?
- What are the issues of having the instance variables marked as `public`?
- What does it mean to create a `new Book` in our code?
- Draw out a UML Diagram for the class

Encapsulation

You may have noticed the use of the `public`, `private`, and the lesser used `protected` access modifiers. These modifiers refer to how the data is exposed to other objects and classes.

- `public` allows the variable/method to be accessible publicly, meaning that the variable or method is able to be accessed by other objects.
- `private` is only accessible within the instance or class itself.
- `protected` Similar to `private` but can be accessed by **sub types**.
- No access modifier keyword (sometimes referred to as default), is accessible to any class within the same **package**.

Discussion

Discuss with your peers and tutor about the issues with the following code:

```java
public class VHSTape {

        private static final String title;
        public boolean rented;

        public VHSTape(String title, boolean checkedOut) {
                title = title;
                rented = checkedOut;
        }

        private String getTitle() {
                return this.title;
        }

        public static boolean isRented() {
                return this.rented;
        }
}
```

Question 1: Create a class from UML

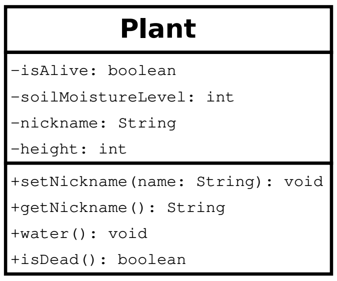

Figure 2: UML Class Diagram of the `Plant` class

Given the above UML diagram, construct the class. You should test your setters and getters with your own `Driver` or `Test` class. Make sure to increase the `soilMoistureLevel` every time the plant gets watered.

`this` keyword

The `this` keyword is similar to the `self` variable in python. However unlike self it is integrated into the language and corresponds to the instance of the class. For example, we are unable to use the `this` keyword within a static method as it does not relate a specific instance.

I/O

Java standard library contains classes and methods that allow the programmer to interact with the file system and other I/O interfaces (such as networking). The input/output API is an abstraction of the platform's filesystem functions. This allows Java programs to not require recompilation to work on other platforms since the IO classes map to the platform specific functions.

Text-based I/O

Text-based I/O or sometimes referred to as human-readable format, involves text parsing. Part of parsing is to create a method of interpreting the structure of the data and mapping it to our own program's constructs.

```
File f = new File("some_file.txt");
Scanner scan = new Scanner(f);

while(scan.hasNextLine()) {
        //Reads each line of the file
        String s = scan.nextLine()
}
```

Question 2: Creating an Athlete Class

Construct an Athlete class with two properties - the athlete's name and their time. You should also include a getter and setter for each attribute.

Question 3: Loading Athlete Times

Extend your program to read a text file that will contain the athlete's name and their time and instantiate a new Athlete instance for each line that exists. Your program must handle ill-formatted data such as missing commas, times or names.You will only need to instance an Athlete instance for each correct entry in the data.

The data in the text file will be in the format:

```
name,time
```

Example data:

```
Johnson,11.22342
Iron,13.445431
Lucas,10.3001
Fielding,10.0012
```

Extension: Can you now print to standard out a ranking of the athletes from fastest to slowest time? Make sure you print out both the name and time for each Athlete.

Question 4: Let's Go Shopping

You've written out a shopping list in a text file with the items you plan to buy and the associated quantities. A sample shopping list looks like this:

```
<item>:<quantity>
```

Example:

```
Conditioner:1
Apple:3
Light Bulb:3
Soda:1
```

Write a program to read in the data from the file and attempt to map each item to a new ShoppingItem object.

Question 5: Pet

Create a class for the Pet object. Your class should contain instance variables (fields), appropriate get/set methods and at least one constructor. Do not worry about the implementation of the rest of the class.

A `Pet` object contains the following:

- a name
- an array of nicknames
- age
- species (animal type)
- whether or not the pet is house trained

Question 6: Extend Pet

- An `equals` method that checks if one `Pet` is the same as another. If two pets have the same name, species and age, they are considered to be equal.
- An `addNickname` method that adds a new nickname to the pet (but only if the pet doesn't already have that nickname).
- An `hasNickname` method that checks if the pet has a given nickname.

Question 7: Oldest Pet

Create an `OldestPet` method. In the main method, create a few `Pet` instances (at least 3) with different names, nicknames, species, age, house trained or not, in an array of `Pet` objects.

Then, iterate through the `Pet` objects to find the oldest one. Once it's found, print its detailed information (name, species, age, etc.).

Question 8: Assessed Task: Online Task 2

Remember you are required to complete a Online Task within the due date. Go to EdStem for this unit and click on Lessons to find out the task and the due date. This is a marked task. Note that you are allowed to submit multiple times but only the last one will be marked.