THE UNIVERSITY OF SYDNEY

INFO1113 Week 3 Tutorial

Classes, Encapsulation, Objects, 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.

```
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.

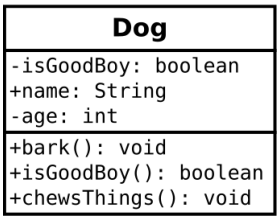

Figure 1: UML Class Diagram of the `Dog` class

Question 1: Class Definition

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?
- How can we access the data in the class?
- What are the issues of having the instance variables marked as `public`?
- What does it mean to create a `new Book` in our code?

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 publically, 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**.

Question 2: `Static`, `final`, `this`, `public`

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

```
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 3: Create a class from UML

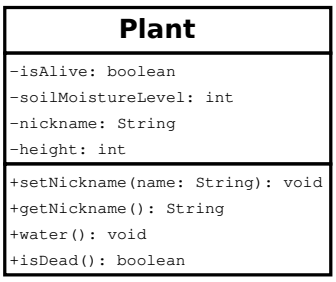

Figure 2: UML Class Diagram of the `Plant` class

Given the above UML diagram, construct the class. You should test your setters and getters with your own `Driver` or `Test` class. Specifically, we are asking you to exercise a degree of creativity when testing the class. You may suggest your method of how `soildMoistureLevel` works.

`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 4: Loading Athlete Times

Your program must 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.

After parsing the data, you must create an `Athlete` class and instantiate an instance for each correct entry in the data.

The data will be in the format:

```
name,time
```

Example data:

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

Extension: You can use your solution from last week's question **Ranking athletes** to rank the athletes from the file.

Question 5: Writing a shopping list

Write a program that will allow the user to input shopping items and quantity. The data will be structured in the following format:

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

Example:

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

Similarly to the previous question, attempt to map the data from file to a class in your program.

Question 6: 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 7: 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 8: Oldest Pet

Create an `OldestPet` class with a main method. In your 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 9: Manhattan Drive

You are given a grid (`CityPoint[][]`) and two sets of points (A and B), You must create a method which will return an array of CityPoint objects which correspond to the path from A to B. The `CityPoint[][]` matrix corresponds to the city and each `CityPoint` object contains data related to to that point.

Your program is divided up into two segments. You will need to implement the CityPoint class which contains the following information.

- Name: String ("Jeff's bar and grill", "Subway Entrance A", "Mary's Theatre", etc)
- Coordinates: int[2] (x, y)
- Description: string

The above attributes should be accessible through getter methods:

- getName(): String
- getCoords(): int[]
- getDescription(): String

Afterwards you are to create a `static` method which returns the path between two points.

```
static CityPoint[] drive(int x1, int y1, int x2, int y2,
    CityPoint[][] city)
```

Example Path 1:

```
//Point A = (1, 1) and Point B = (3, 5)
+------+
|      |
| A**  |
|   *  |
|   *  |
|   *  |
|   B  |
+------+
```

Question 10: 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 Assessment to find out the task and the due date. This is a marked task.

Extension

Binary I/O

Binary-based I/O is where the data in itself is not human readable due to how it is encoded. The `byte` type is extremely useful in this scenario as we are able to treat the data encoded as bytes and interpret them based on a specification.

```
import java.io.*;
// - snip -

String inputFile = "some_binary_file.bin";
try {
        InputStream inputStream = new FileInputStream(inputFile);

        byte[] buffer = new byte[128]; //

        inputStream.read(buffer);

        //Buffer will be populated with 128 bytes of data from the file
        //The file cursor would have moved 128 bytes
} catch (IOException ex) {
        ex.printStackTrace();
}
```

Unlike C and C++, Java does not allow directly writing to memory to arbitrary addresses (at least not through any safe API method). Java provides safe api functions to interpret data that corresponds to a type. However, it is useful to understand how to convert an array of bytes to an integer.

A big danger we have to watchout here though is the signed bit in regards to how the byte is stored. Remember, java has a sign bit for all primitive types.

```
byte[] b = { -1, -1, -1, 127};
int a = (b[0] & 0xFF) | ((b[1] & 0xFF) << 8)
    | ((b[2] & 0xFF) << 16) | ((b[3] & 0xFF) << 24);
```

Question 11: Reading the bits we have written (bytes to int)

You will need to write a program that will write binary data to a file. After you have done this, you will need to read that data back and interpret it.

Java is a staticly typed system and you will need to convert a byte array to integers. If you consider the memory layout of an integer, your

```
static int bytesToInt(bytes[] b)
```

Do not use the `java.nio` for this question.

Question 12: Take the bits from an int

From the previous question we assembled an integer from a byte array. We now want to perform the inverse and take the bites from an integer to a byte array.

```
static bytes[] intToBytes(int a)
```