Lab 7 : Classes

Topics covered objects and classes, "getter" and "setter" methods

With the aid of variables, loops and decisions, methods, arrays, files and exceptions, you can now write just about any program you want (with a bit of determination).

With the help of Object Oriented Design, however, you'll be able to make your programs more portable, extendable and re-usable.

Exercise 1: Your tutor will lead a discussion on the importance of objects and classes. Why do we use them, rather than just keep using what we know already?

Exercise 2: Consider the following class definition:

```
1 public class Book {
2    public String title;
3    public String author;
4    public int year;
5    public String url;
6 }
```

- 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?

Consider the following rule about the url of a book:

The url of any book is given by `www.books.com/<year>/<author>/<title>`, where `<year>`, `<author>` and `<title>` are all properties of the book.

What access modifiers do we need to have to make sure that this can be reflected in the `Book` class?

Exercise 3: Describe each of the following real-world objects or concepts as a class.

- Polygon
- Song (music)
- Student
- ATM
- Table (furniture)
- Country

For each example, consider the following questions:

- What data, including data type, should be stored in each object?
- How should the data be accessed?
- Should someone be able to read/write the data?

Exercise 4: 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
- an age
- a species (animal type)
- whether or not the pet is house trained

Exercise 5: Implement the following methods for the `Pet` class created in exercise 4:

- An `equals` method that checks if one `Pet` is the same as another. A two pets must only have the same name, species and age to be considered 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.

Exercise 6: 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.).

Exercise 7: If you haven't finish last week's lab exercises, please continue to finish them.

Extensions

Extension 1: Write a program that reads and writes pet information from/to a `Pet` object. The program will be run like this:

```
> java ReadPet wolfie
```

If `wolfie.txt` does not exist, then the program will ask the user questions to fill out the fields in the `Pet` object. The information will then be stored in `wolfie.txt`.

```
> java ReadPet wolfie
Unknown pet: "wolfie"
What are wolfie's nicknames? (comma separated list)
  wolf, fluffy
How old is wolfie?
  6
What animal is wolfie?
  kangaroo
Is wolfie house trained?
  yes
```

If `wolfie.txt` does exist, read the contents of the file and print them to the screen.

```
> java ReadPet wolfie
Pet name: wolfie
Nicknames: wolf, fluffy
Age: 6
Species: kangaroo
House trained: yes
```

After printing to the screen, ask the user if they wish to change the data and allow them to enter new information to save to the file.

```
Would you like to update this information? (y/n)
  y
What are wolfie's nicknames? (comma separated list)
...
```