Lab 7 : Classes – Selected Solutions

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

- ...
- Song (music)
- ...

For each example, consider the following questions:

- What data, including data type, should be stored in each object?

  A song generally contains a set of metadata (information about the song) as well as the sound data itself. For example the song might store the following:

  - a title: `String title`
  - a set of artists: `String[] artists`
  - an album name: `String album`
  - a genre: `String genre`
  - the year it was released: `int year`
  - the actual sound samples: `byte[] dataSamples`
  - the length of the song: `long trackLength`

- How should the data be accessed?

  Depending on the application, the only data that needs special access is the sound samples. All the other attributes could be read or changed directly. An example of accessing the samples data:

```
1  /** return a copy of the sound data from a given position and
2    * for a fixed duration
3    * position - milliseconds from start
4    * duration - number of milliseconds to extract
5    */
6  public byte[] getDataDuration(long position, long duration) {
7     if (position < 0 || duration <= 0)
8        return null;
9     if ((position + duration) >= trackLength)
10       return null;
11
12    byte[] samples =
13       Arrays.copyOfRange(dataSamples, position, duration);
14    return samples;
15 }
```

- Should someone be able to read/write the data?

  - the title, artists, album and genre can potentially change and have no special access requirements. These could be marked as `public`, or `private` with both `get` and `set` methods.
  - the year a song was released is not likely to change, and so should be marked as `private` with only a `get` method.
  - the sample data should definitely be hidden from direct access, and so should be `private` with special access methods.

- the length of the song in milliseconds is potentially not so useful. This field could be hidden as `private`, however have a `get` method that converts the time into a readable string (for example in minutes and seconds).

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.

```
1  import java.util.Arrays;
2
3
4  public class Pet {
5     private String name;
6     private int age;
7     private String species;
8     private boolean houseTrained;
9
10    private int numNicknames;
11    private String[] nicknames;
12
13    public Pet(String name, int age, String species, boolean houseTrained) {
14       this.name = name;
15       this.age = age;
16       this.species = species;
17       this.houseTrained = houseTrained;
18
19       this.numNicknames = 0;
20       this.nicknames = new String[numNicknames];
21    }
22
23    public String getName() {
24       return name;
25    }
26    public void setName(String name) {
27       this.name = name;
28    }
29
```

```
30      public int getAge() {
31          return age;
32      }
33      public void setAge(int age) {
34          this.age = age;
35      }
36
37      public boolean isHouseTrained() {
38          return houseTrained;
39      }
40      public void setHouseTrained(boolean houseTrained) {
41          this.houseTrained = houseTrained;
42      }
43
44      public String getSpecies() {
45          return species;
46      }
47
48      public String[] getNicknames() {
49          return nicknames;
50      }
51
52      public void addNickname(String nickname) {
53          if(hasNickname(nickname)) {
54              return;
55          }
56          // Adding a new array position requires creating an entirely new array!
57          // copy the old nicknames into a new array that is 1 bigger
58          nicknames = Arrays.copyOf(nicknames, numNicknames + 1);
59          // set the last position of the array to the new nickname
60          nicknames[numNicknames] = nickname;
61          numNicknames = numNicknames + 1;
62      }
63
64      public boolean hasNickname(String nickname) {
65          int index = 0;
66          while(index < nicknames.length) {
67              if(nicknames[index].equalsIgnoreCase(nickname)) {
68                  return true;
69              }
70              index++;
71          }
72          return false;
73      }
74
75      public boolean equals(Pet other) {
76          if(other == null) {
77              return false;
78          }
79          return (this.name.equals(other.name) &&
80                  this.species.equals(other.species) &&
81                  this.age == other.age);
82      }
83  }
```

Exercise 6: : Create a A `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.).

```
import java.util.Arrays;

public class OldestPet {
   public static void main(String[] args) {
      Pet[] pets = new Pet[3];
      Pet oldestPet = null;

      //create three Pet objects
      pets[0] = new Pet("Lucy", 1, "Dog", true);
      pets[0].addNickname("Lulu");

      pets[1] = new Pet("Milo", 3, "Cat", true);
      pets[1].addNickname("MyMy");
      pets[1].addNickname("MoMo");

      pets[2] = new Pet("wolfie", 2, "Kangaroo", false);

      //find the oldest pet
      for (int i = 0; i < pets.length; i++) {
         if (i == 0) {
            oldestPet = pets[i];
         }else {
            if (pets[i].getAge() > oldestPet.getAge()){
               oldestPet = pets[i];
            }
         }
      }

      //print information of the oldest pet
      System.out.println("The oldest pet is " + oldestPet.getName() + "!");
      System.out.println("It is a " + oldestPet.getSpecies() + ", and " +
                     oldestPet.getAge() + " years old.");
      System.out.println("Its nciknames are " +
                     Arrays.toString(oldestPet.getNicknames()));
   }

}
```