Assignment 7
Problem #2 of 3
Parks
Lists And Arrays
4/19/19

Getting Started: If you haven't already *"Assignment 7 Files.zip"* provided into your working directory. You will see a directory named "Park". That is the directory we'll use for this problem.

See "General Notes" and "General Hints" in Problem #1 document.

All **classes** should be designated as **public** unless otherwise noted.
All **methods** should be designated as **public** unless otherwise noted.
All **instance variables** should be designated as **private** unless otherwise noted

Problem 2 -- Park Management System

For this problem our goal is to build software to manage a park.

The software will let park-goers know what park features are open.

A park has a collection of park features. These park children represent different features in the park. For example, a beach, picnic area, water park, mountain bike trail, climbing wall, etc.

Initially a feature is closed (when first constructed) but can be opened by sending it the "open" message.

Concept

We have two classes:

- Park
- ParkFeature

The park owns a collection (list) of park features.

A park object will manage and report about it's features. E.g. is a feature open, close a feature, which features are open, etc.

Hint: Let the park features do most of the work. Keep the "Park" object as light as possible.

ParkFeature

This class is already coded and ready for use. You will find it in the “Park” directory.
The method headers are:

```
public ParkFeature(String newName)

//Commands (directives) we can give to a feature
public void open() //sets feature open
public void close() //sets feature closed
public void setClosed(boolean newClosed)    //sets the closed value of the feature per the param

//Questions we can ask a feature
public boolean isClosed()
public boolean isOpen()
public boolean hasName(String aName) //Answers true if our name matches param "aName"
public String toString()
```

Park

Add a class named **Park**
(see top of document for the directory where the class should go).

Instance variables

Name: “features”
Type: List<ParkFeature>
Description: An array of ints (our numbers)

Constructors

Park (constructor)
Constructor Method Parameters: None
Description: Empty constructor. It must initialize the “features” instance variable to a new list.
Return Type: Not Applicable

Instance Methods

Method Name: getFeatures
Method Parameters: None
Return Type: List<ParkFeature>
Description: Simple getter that returns our features

Method Name: addFeature
Method Parameters:
	Param 1: Type “String” that is the name of a feature (e.g. “Beach”, “Disk Golf Course”, etc)
Return Type: None
Description: Adds a new feature to our features

Method Name: closeAll
Method Parameters: None
Return Type: None
Description: Close all the features

Method Name: openAll
Method Parameters: None
Return Type: None
Description: Open all the features

Method Name: getAllOpen
Method Parameters: None
Return Type: List<ParkFeature>
Description: Return a list of all of the features that are open

Method Name: countAllOpen
Method Parameters: None
Return Type: int
Description: Return a count of all of the features that are open

Method Name: isOpen
Method Parameters: Param of type String that is a feature name
Return Type: boolean
Description:

- Return true if feature is open (for feature that has name matching method param feature "name")
- If feature is not found for param feature "name", return false

Hint: This method (and the methods "setStatus" and "hoursRemaining") all could use a helper method that would make things easier. If you think about the algorithms for all three, you'll see the helper method that would be handy. As always, if you want, you may add zero, one, or many "helper methods".

Method Name: setStatus
Method Parameters:
    Param 1: Type "String" that is the name of a feature (e.g. "Beach", "Disk Golf Course", etc)
    Param 2: Type boolean that if true if the feature should close, else false
Return Type: None
Description:

- For the "name" method parameter, we find the feature and set it's closed setting with the method parameter that controls if it should close

Method Name: toString
Method Parameters: None
Return Type: String
Description: Return a String that describes the object (any String you like -- it is helpful for debugging). One possibility would be something like:

```
return "Open Park Features: " + this.countAllOpen();
```

Note: It is good practice to generally add a "toString" method to all your Java classes