Syntax and Utilities II (Page 1 of 2)

The first four (Lists, Maps, printf, using generics in your own methods) are the most important. I threw in some other problems for the very few developers that already have some experience with these topics.

1. Make a List of Circle objects. Use a random radius. Keep adding circles to the list until Math.random() returns less than 0.01. Then, loop down the list and print out each area. If you do not have a good Circle class to use, change the problem to make a List of Strings, where each String is made like this:

```
String word = "word" + i; // i is your loop variable
// Add the word to the List
```

Use the same logic as with Circles, where you continue until Math.random() returns less than 0.01. Then, loop down the List and print out each word.

2. Make a hash table (Map) that associates the following employee IDs with names. To make things simpler, you can use String for both the ID and the name, rather than bothering to create a Name or Person class. The point here is to associate keys with values, then retrieve values later based on keys.

ID	Name
a1234	Steve Jobs
a1235	Scott McNealy
a1236	Jeff Bezos
a1237	Larry Ellison
a1238	Bill Gates

Make test cases where you test several valid *and* invalid ID's and print the associated name.

3. Change your circle list example (problem 1) so that the areas are printed with the decimal points aligned and with exactly three digits after the decimal point.

4. **[Harder]** Make a static method called "lastEntry" to which you pass a List and get back the last entry of that list. If you pass it an List of Strings, you should get back a String. If you pass it a List of Circles, you should get back a Circle. E.g.:

```
List<Circle> listOfCircles = ...;
Circle lastCircle = ListUtils.lastEntry(listOfCircles);
```

Next, support arrays as well as Lists. That is, you should be able to call ListUtils.lastEntry(someList) or ListUtils.lastEntry(someArray). Hint: very easy once you think of a minor "trick".

5. Make a hash table (Map) that maps numbers (e.g., 2) to words (e.g, "two" or "dos"). Test the Map out by passing in a few numbers and printing out the corresponding words. Note: Map keys in Java cannot be primitives; they must be objects. But, if you declare the key to be of type Integer, autoboxing will be in effect, so that you can use an int and the conversion will occur automatically.

6. Take the state lookup example and modify it so that it works the same no matter what case the user supplies for the state. I.e., Maryland and MARYLAND should work identically.

7. Switch the state lookup example so that it maps state abbreviations to full state names, instead of the other way around.

8. Make a routine that accepts any number of state abbreviations and prints out the corresponding state names.

9. Do some timing tests to compare ArrayList to LinkedList for accessing the middle element. Hints:
   - Use System.currentTimeMillis or System.nanoTime to lookup the current time. Compute a delta and divide to get an elapsed time in seconds. Use printf to control the number of decimal places displayed.
   - To ensure meaningful results, use very long lists and access the middle element many times.
   - Have your program repeat the test several times, and throw out the first result.

10. Do similar timing tests to compare the performance of the two versions of padChar.