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

**Exercises: Collections**

**1.** Make a List of Circle objects. Use a random radius. Keep adding circles to the list until Math.ran-

dom() 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, steal mine from the shapes3 package of the oop-advanced project.

**2.** Make a Map that associates the following employee IDs with names. Keys and values of Maps can

be any Object type, so in real life you would probably have the key be a String and the associated

value be a Person or Employee object. To make things simpler on this exercise, you can use String

for both the ID and the name, rather than bothering to create a Person or Employee class. The point

**3.** Go back to the previous problem and make your lookup method work with keys in any case. For

here is to associate keys with values, then retrieve values later based on keys.

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

example, both “a1234” and “A1234” should match Steve Jobs. Hint: very easy, so if your solution

is complex, you are overlooking the obvious.

**4.** Make a Map that associates numbers (e.g., 2) with words (e.g, “two” or “dos”). Test the Map out by

passing in a few ints and printing out the corresponding words.

**5.** Do some timing tests to see if you can verify the performance claims of the lecture regarding the

difference between ArrayList and 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.

• To ensure meaningful results, use very long lists and access the middle element many times.

• Run several tests, each with larger and larger lists.