Questions | Reading 1: Static Checking | 6.005.1x Courseware

1. Course , current location
2. Progress

Questions

list vs. array

1/1 point (graded)

Rewrite these variable declarations using Lists instead of arrays.

We're only declaring the variables, not initializing them with any value.

correct
List<String> names **or** List<String> names;

Explanation

(edX might not display the answer correctly, it's List<String> names)

The translation from String[] to List<String> is pretty straightforward in Java.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem

list vs. array, part 2

1/1 point (graded)

correct
List<Integer> numbers **or** List<Integer> numbers;

Explanation

(edX might not display the answer correctly, it's List<Integer> numbers)

We can create arrays of primitive types, but not Lists. Use the Integer wrapper.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem

list vs. array, part 3

1/1 point (graded)

correct
List<List<Character>> grid **or** List<List<Character>> grid;

Explanation

(edX might not display the answer correctly, it's List<List<Character>> grid)

There's nothing wrong with a List<List<Character>> -- but if this is a fixed-size grid, it might be simpler to use a 2-dimensional array instead of a list-of-lists.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem

x marks the spot

3/3 points (graded)

Java Maps work like Python dictionaries.

After we run this code:

```
Map<String, Double> treasures = new HashMap<>();
String x = "palm";
treasures.put("beach", 25.);
treasures.put("palm", 50.);
treasures.put("cove", 75.);
treasures.put("x", 100.);
treasures.put("palm", treasures.get("palm") + treasures.size());
treasures.remove("beach");
double found = 0;
for (double treasure : treasures.values()) {
    found += treasure;
}
```

What is the value of...

treasures.get(x)

correct
54

54

treasures.get("x")

correct
100

100.

found

correct
229

229

Explanation

After the first four `put()` calls, the map has stored the pairs ("beach", 25), ("palm", 50), ("cove", 75), ("x", 100). The fifth `put()` call adds the size of the map (4) to the entry for `"palm"`, so that entry is now ("palm", 54). Finally the entry for "beach" is removed from the map, so the final state of the map is ("palm", 54), ("cove", 75), ("x", 100).

Now that we know what the map looks like, we can answer the questions. `treasures.get(x)` returns the value stored for the key `"palm"`, which is 54. `treasures.get("x")` returns the value stored for `"x"`, which is 100. Finally, `found` sums up all the values currently stored in the map, which is 54+75+100 = 229.

You can see this code in action in Online Java Tutor.

Some problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Answers are displayed within the problem