Indiana University
CSCI-C212: Introduction to Software Systems (SP24) Assignment 3
Joshua Crotts & Muazzam Siddiqui February 14, 2024

Arrays, ArrayLists, Java Collections, Generics

Important Dates:

- Assigned: February 14, 2024
- Deadline: February 28, 2024 at 11:59 PM EST

Objectives:

- Students begin to understand the differences between classes in the Java Collections API.
- Students see the advantages and disadvantages over static arrays versus `ArrayList` objects.
- Students learn how generics allow them to write methods that work over any type.

What To Do:

For each of the following problems, create a class named `ProblemX`, where `X` is the problem number. E.g., the class for problem 1 should be `Problem1.java`. Write (JUnit) tests for each (non-`private`) method that you design in corresponding test files named `ProblemXTest`, where `X` is the problem number. For problems that contain multiple parts, put those in the same class file. Additionally, write Javadoc comments explaining the purpose of the method, its parameters, and return value. **Do not round your solutions!**

When testing arrays, use the `assertArrayEquals` method; using just `assertEquals` will not work because it compares the memory addresses of the arrays and not their contents!

If you are doing corrections for this assignment, the extra credit is worth only 10 points instead of 20, meaning the maximum score for corrections is 100%.

You must write sufficient tests and adequate documentation.

All problems are listed in *Teaching Java - A Test-Driven Approach*.

1. Page 114, Exercise 3.5
2. Page 115, Exercise 3.7 (*for this problem, you cannot sort the array*).
3. Page 116, Exercise 3.15
4. Page 117, Exercise 3.17 (*for this problem, the hint references a "linear search from problem 1". Problem* 1 *is Exercise* 3.3 *in the textbook and was assigned last semester, so disregard this hint*).
5. Page 118, Exercise 3.20
6. Page 118, Exercise 3.21
7. Page 119, Exercise 3.22
8. Page 120, Exercise 3.29 (*for this problem, the input examples are incorrect*; *the first and second inputs are*:)

   ```
   postfixEvaluator({"5", "2", "*", "5", "+", "2", "+"}) => 17
   postfixEvaluator({"1", "2", "3", "4", "+", "+", "+"}) => 10
   ```

9. Page 123, Exercise 3.47
10. Page 123-124, Exercise 3.49
11. **Extra credit:** It's black history month, and to honor this, we will offer an extra credit worth 20 points. You will implement a system with querying functionality similar to a database language such as SQL. In particular, we have a 2D array of strings whose first row contains the following column headers: `"ID"`, `"Name"`, `"BirthYear"`, `"Occupation"`, and `"Salary"`. The remaining rows contain several black contributors to not only computer science but other sciences, engineering, and mathematics as well.

    Design the `List<String> query(String[][] db, String cmd)` method that, when given a "database" and a "Command", returns the names of all people that satisfy the criteria enforced by the command.

    ```
    A Command is "SELECT <count> WHERE <predicate>"
    ```

    The `SELECT` command receives a `<count>`, which is a number between 1 and n, or the asterisk to indicate everyone in the database. The `WHERE` clause receives a "Predicate".

```
A Predicate is "<header> <comparator> <value>"
```

Headers are one of the column headers of the database, and comparators are either =, !=, <, <=, >, or >=, or LIKE. Values are either numbers, floats, or strings.

Parsing a LIKE command is more complicated. There are four possible types of values:

```
'S'
'%S'
'S%'
'%S%
```

The first matches an exact string, namely S. The second matches any string that ends with S. The third matches any string that begins with S. The fourth matches any string that contains S.

The template that we provide contains one entry; you should add many more to test your code. An appropriate amount would be at least twenty values in the database. You can use arbitrary values for the salary.