Taboola
Content You May Like.

R&D Engineer - Java & Architecture Home Test (V16)

Introduction

The purpose of this test is to evaluate your technical abilities in various aspects of Software Engineering, Optimizations, Concurrency Issues and Architecture. Also allowing common grounds for discussion in the interview that takes place after you complete this test.

The test is a home test and you are encouraged to use any resource at your disposal - online etc. You will be asked to explain your solution in the interview, emphasizing on correctness, design and coding decisions.

You are asked to submit via email, working and tested code. We expect the code to be "Production Grade" code. The code should be in Java.

Questions

Question 1a - Parse invoice numbers

We have recently decided to digitize our old invoice archives. Since finding a volunteer for such an arduous task was impossible, an employee was selected at random and instructed to type in all invoice numbers into a text file.

Little did we know that the employee we picked is an aspiring ASCII artist. Instead of handing us a file containing a set of numbers, we ended up with 7-segment display representations of the invoice numbers.

This is where you come in. Write a program that given an input text file of 7-segment invoice numbers, outputs a text file with the parsed invoice numbers.

Input:

A text file containing several hundreds of invoice numbers in the following form (look at ***input_Q1a.txt*** for exact input file):

```
    _  _     _  _  _  _  _  (line 1)
  | _| _||_||_ |_   ||_||_| (line 2)
  ||_  _|  | _||_|  ||_| _| (line 3)
                            (line 4)
    _  _  _  _  _  _     _  (line 5)
|_||_|| ||_||_   |  |  ||_  (line 6)
  | _||_||_||_|  |  |  | _| (line 7)
                            (line 8)
```

Invoice number format:

Each invoice number is constructed of 9 digits [0..9]
Invoice number is written using '_' and '|' characters.
Invoice number input takes 4 lines.
The first 3 lines contain 27 characters.
The fourth line is blank.
Note: You can use ***input_Q1a.txt*** to feed your program.

Output:

A text file with the parsed invoice numbers. One number per row.

Example:

```
123456789
490867715
```

Note: You can use ***output_Q1a.txt*** to test that the output file generated by your program is correct.

Question 1.b - Illegal column

Typing in invoices proved rather tedious for our hapless employee, so it is no wonder he had the occasional typo. We need to track these errors, and you're just the person for the job. Extend the functionality implemented in the previous user story to handle illegal digits. Replace the illegal digits with a ? and add a second column indicator ILLEGAL.

Input:

Same as above, but in this case there might be invoices with illegal digits.

Example:

```
    _  _     _  _  _  _  _  (line 1)
  | _| _||_||_ |_   ||_||_| (line 2)
  ||_  _|  | _||_|  ||_| _| (line 3)
                            (line 4)
    _  _  _  _  _  _     _  (line 5)
|_||_|| ||_||    |  |  ||_  (line 6)
  | _||_||_||_|  |     | _| (line 7)
                            (line 8)
```

Note: You can use ***input_Q1b.txt*** to feed your program.

Output:

A text file with the parsed invoice numbers. One invoice per row.

Example:

```
  123456789
  4908?7?15 ILLEGAL
```

Note: You can use ***output_Q1b.txt*** to test that the output file generated by your program is correct.

Question 2

The following class has several memory and runtime inefficiencies and bugs. Locate and fix as many as you can.

```
import java.util.Date;
import java.util.List;

public class MyClass
{
      private Date time;
      private String name;
      private List<Long> numbers;
      private List<String> strings;

      public MyClass(Date time, String name, List<Long> numbers, List<String>
strings) {
            this.time = time;
            this.name = name;
            this.numbers = numbers;
            this.strings = strings;
      }

      public boolean equals(Object obj) {
            if (obj instanceof MyClass) {
                  return name.equals(((MyClass)obj).name);
            }
            return false;
      }

      public String toString() {
            String out = name;
            for (long item : numbers) {
                  out += " " + item;
            }
            return out;
      }

      public void removeString(String str) {
            for (int i = 0; i < strings.size(); i++) {
                  if (strings.get(i).equals(str)) {
                        strings.remove(i);
                  }
            }
      }
```

```
        public boolean containsNumber(long number) {
                for (long num : numbers) {
                        if (num == number) {
                                return true;
                        }
                }
                return false;
        }

        public boolean isHistoric() {
                return time.before(new Date());
        }
}
```

Question 3

Words Counter

In this exercise, you are asked to write a Class (WordsCounter.java) that will receive few text files and will count (and later print) all the words that exist in these files (together) and the number of times each one of them appears.

The class should contain a single map that will hold the words and the number of items of each. As can be seen in the example below, the method “load” can get few files. Each file should be handled in separate threads (in parallel).
The method “displayStatus” will print the map.

Specific guidance:
please use java.util.concurrent.ConcurrentHashMap and also it’s method putIfAbsent()

Note: When splitting the text into separate words, There is no need to deal with special characters such as “,.-:” etc.

```
public class WordsCounter {
java
      public static void main (String [] args) {

      WordsCounter wc=new WordsCounter ();
      // load text files in parallel
         wc.load("file1.txt","file2.txt","file3.txt");
      // display words statistics
      wc.displayStatus();
      }
}
```

Small example (it is recommended to use files with many words):

File1.txt:

```
this is the first file
```

File2.txt:

```
this one is the second file
```

File3.txt:

```
and this is the third file
```

Output of displayStatus method:

```
and  1
file 3
first 1
is   3
one  1
second      1
the  3
third 1
this 3

** total: 17
```

You should send the working code as well as the input text files you used, and also the displayStatus output as you got it when executed the code on these files.

Taboola
Content You May Like.

~~Question 4~~

~~You're tasked with writing a spec for a generic local cache with the following property: If the cache is asked for a key that it doesn't contain, it should fetch the data using an externally provided function that reads the data from another source (database or similar).~~

~~What features do you think such a cache should offer? How, in general lines, would you implement it?~~

~~Question 5~~

~~You are tasked with improving the efficiency of a cache heavy system, which has the following properties/architecture:~~

~~The system has 2 components, a single instance backend and several frontend instances.~~

1. ~~The backend generates data and writes it to a relational database that is replicated to multiple data centers.~~

2. ~~The frontends handle client requests (common web traffic based) by reading data from the database and serving it. Data is stored in a local cache for an hour before it expires and has to be retrieved again. The cache's eviction policy is LRU based.~~

~~There are two issues with the implementation above:~~

1. ~~It turns out that many of the database accesses are redundant because the underlying data didn't actually change.~~
2. ~~On the other hand, a change isn't reflected until the cache TTL elapses, causing staleness issues.~~

~~Offer a solution that fixes both of these problems. How would the solution change if the data was stored in Cassandra and not a classic database?~~