Software Developer Hiring Questionnaire

## About The Company

[Videology](http://www.videologygroup.com/) is an online advertising company based in Baltimore, MD. We run advertising campaigns for our customers effectively and efficiently. Effective campaign execution means that our customers are guaranteed brand safety when they run their campaigns on our platform, as well as higher return on investment. Efficient campaign execution comes about because of the support built into the platform to eliminate not just routine tasks, but also sophisticated data analysis that often cannot be done quickly enough by humans.

Our company relies on our software platform not just for campaign execution, but also for other company-wide functions. This platform is a high-performance, always-available, low-latency, high-throughput platform built on top of open-source technologies. We are looking to hire smart, creative software engineers who have hands-on high performance experience, and who have programmed in a cutting-edge Java environment for several years.

## About This Questionnaire

You received this questionnaire because you inquired about software development careers with Videology. To be considered for a position, please answer the questions on the pages that follow. You may type your answers directly into the document. Use as many pages as necessary.

The questions are not intended to be “gotcha” questions. If you need to make reasonable assumptions to provide an answer, let us know what those assumptions are. You are welcome to consult whatever resource you like – books, web, friends – as long as you can claim to yourself honestly that you solved the problem.

## Your Contact Details

Name: Matthew Lesko

Phone: 410.935.6551

Email: orangepips@gmail.com

Address: 1917 E Pratt Street Baltimore, MD 21231

## Your Links

If you would like to share links to your website or to your previous projects and accomplishments, please mention them below. Include a sentence or two explaining the significance of each.

Best place to look is my LinkedIn profile: <https://www.linkedin.com/in/matthewlesko> and resume, which should accompany this questionanire.

## Update 2/26/2015: code, as appropriate now follows each question in 10 point Courier. Links to that code on Google Drive remain as they originally appeared.

## Telephone Directory

Suppose I have a very long list of alphabetically-sorted unique names, along with exactly one phone number next to each name. The list is very long, containing ~10M names and numbers. Given a name, how would you find the associated number without hitting disk every time? (Okay to hit disk for one-time pre-processing.)

Assuming the problem is maintaining the data in memory for multiple lookups the problem to solve is choosing a data structure that can fit all the data into memory and supports fast lookups. Also, I assumed that the data is either static or changes infrequently, such that insert time is not as important. I considered three options, their time and space complexities listed in the table that follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Space** | **Insert** | **Search Average** | **Search Worst** |
| Hashtable | O(n) | O(1) | O(1) | O(n) |
| Binary Search Tree (BST) | O(n) | O(log n) | O(log n) | O(n) |
| Trie | O(a\*k\*n) | O(k) | O(k) | O(k) |

n - number of entries | k - key length | a - alphabet size

In my own tests using a Python dict - that is a Hashtable - 1 million records used 175 MB of memory. This suggests 10 million would consume less than 2 GB; a reasonable amount of memory to have available on a server. From there I examined a BST, and ruled it out of hand because because without an ordering requirement is has no advantage over a Hashtable. Finally, I considered a Trie. While on paper it does not look like a desirable option due to space complexity, that is a worst case measure. In practice key prefixes overlap significantly (e.g. many people have the name “John”) producing a sparse matrix. Such that a Trie representation that takes up considerably less space than a Hashtable. I benchmark those two options in the link below using Python.

Benchmark (code inline at the end of the question): <https://drive.google.com/file/d/0B_ceT2Lz-EMvb0paTG9YS19kbEk/view?usp=sharing>

What the benchmark shows is that a “good” Trie implementation - i.e. a non-naive implementation - uses about 40% less memory than Python’s dict - a Hashtable - while taking significantly longer to insert. Going back to the assumption insertion time is not important a Trie is the option I would choose where 10 million records should easily fit into 2 GB of RAM.

Code name\_phone\_lookup.py:

# Based on tests below the decision point boils down to fast insert times or small memory footprint.

# The two candidate data structure options being a hashtable (a.k.a. dictionary, map) or a trie

# with the former being much faster to populate and the latter much more memory efficient. Note, no practical

# retrieval difference was noted, albeit depending on the character set of the data being stored or the hash function

# involved may impact that retrieval speed.

#

# The hashtable used is Python's built in dict while the trie is from https://github.com/kmike/datrie

#

# Benchmarks were done with the anacondas Python 2.7.8 distribution on a Macbook Pro OSX 10.9.5 2.7 GHz Intel Core i7

# 16 GB 1600 Mhz DDR3

#

# The following data gathering shows the different in population performance and memory consumption

#

# dictionary

#

# iterations|MBs|population secs|retrieval secs

# 100,000|27.93359375|3.8673|0.0000

# 200,000|57.7578125|7.9814|0.0000

# 300,000|75.63671875|11.7909|0.0000

# 400,000|117.453125|15.5348|0.0000

# 500,000|135.30078125|19.6350|0.0000

# 1,000,000|272.484375|38.6464|0.0000

#

# trie

#

# iterations|MBs|population secs|retrieval secs

# 100,000|17.7578125|9.3730|0.0000

# 200,000|34.9765625|43.6296|0.0000

# 300,000|45.21484375|97.1127|0.0000

# 400,000|65.10546875|149.0669|0.0000

# 500,000|73.0859375|193.6686|0.0000

# 1,000,000|175.06640625|381.6497|0.0000

import os

import datrie

import string

import random

import time

trie\_characters = string.printable

last\_full\_name = u''

def full\_name\_gen():

return ''.join(random.choice(trie\_characters) for x in range(2, 15)) + ' ' \

+ ''.join(random.choice(trie\_characters) for x in range(2, 15))

def phone\_num\_gen():

return ''.join(random.choice(string.digits) for x in range(10))

def scrub(value):

return unicode(filter(lambda x: x in trie\_characters, value))

def memory\_usage\_psutil():

# return the memory usage in MB

import psutil

process = psutil.Process(os.getpid())

mem = process.get\_memory\_info()[0] / float(2 \*\* 20)

return mem

def timeit(f):

def timed(\*args, \*\*kw):

ts = time.time()

result = f(\*args, \*\*kw)

te = time.time()

#print 'func:%r args:[%r, %r] took: %2.4f sec' % \

# (f.\_\_name\_\_, args, kw, te-ts)

print 'func:%r took: %2.4f sec' % (f.\_\_name\_\_, te-ts)

return result

return timed

@timeit

def populate(container, iterations):

global last\_full\_name

start = memory\_usage\_psutil()

for index in range(0, iterations):

full\_name = full\_name\_gen()

full\_name = str(index) + full\_name if index % 2 == 0 else full\_name + str(index)

full\_name = scrub(full\_name)

phone\_number = phone\_num\_gen()

container[full\_name] = phone\_number

if index % (iterations / 10) == 0:

print str(index) + ':\t' + str(memory\_usage\_psutil() - start)

last\_full\_name = full\_name

print 'Total Memory:\t' + str(memory\_usage\_psutil() - start)

@timeit

def retrieve(container):

return container[last\_full\_name]

def main():

# hashtable

#container = {}

# trie

container = datrie.Trie(trie\_characters)

populate(container, 200000)

print last\_full\_name + ':\t' + str(retrieve(container))

return

if \_\_name\_\_ == "\_\_main\_\_":

main()

## Java Signature

Write down the signature of a Java method that will take in a word and return a bunch of words that are synonyms of the input word. The implementation of the method is irrelevant.

public SynonymsContainer getSynonyms(String word)

Elaborate on the choices you made when selecting the type for the return value. Specifically, what benefits does choosing this type over another give you?

See <http://www.thesaurus.com/browse/love?s=t> OR <http://www.thesaurus.com/browse/lead?s=t> for example data that informs the answer that follows.

The synonyms for a given word may:

1. Come from more than one part of speech - nouns and verbs for example
2. Be contextualized further - for example “lead” may be a noun in the sense of an actor or an advantage. And a given context may be more common than another.
3. Have a strength of affinity - that is how common a usage.

As a result, encapsulating the response in a custom Java object provides the opportunity to make those distinctions to aid a client in choosing the best option.

Example Java interface with nested elements: <https://drive.google.com/open?id=0B_ceT2Lz-EMvZEgwaXNVbm00RE0&authuser=0>

Code SynonymsContainer.java:

import java.util.List;

/\*\*

\* Aggregation of all known synonyms for a given word broken down by the part of speech and affinity.

\*/

interface SynonymsContainer {

/\*\*

\* How closely related a given word is to the original word used.

\*/

enum Affinity {

STRONG,

MEDIUM,

WEAK

}

enum PartOfSpeech {

ADJECTIVE,

ADVERB,

NOUN,

VERB

}

interface Synonym {

Affinity getAffinity();

String getWord();

}

interface Context {

/\*\*

\* Phrase to contextualize the usage of the synonyms

\* @return phrase to contextualize usage

\*/

String getDescription();

PartOfSpeech getPartOfSpeech();

/\*\*

\* Synonyms in order of affinity from strong to weak

\* @return synonyms in order of affinity

\*/

List<Synonym> getSynonyms();

}

/\*\*

\* Get the word synonyms were put into the container for

\* @return the word used to populate this container

\*/

String getWord();

/\*\*

\* Gets all contexts in order of most to least common usage

\* @return all contexts in order of most to least common usage

\*/

List<Context> getContexts();

/\*\*

\* Get all contexts in order of most to least common usage for the given part of speech

\* @param partOfSpeech part of speech to get synonyms for

\* @return contexts for all synonynms for a given part of speech

\*/

Context getContexts(PartOfSpeech partOfSpeech);

}

## Power Of 2

Implement a method / function that takes an int n as input and returns a boolean, whose value is true, if and only if n is a power of 2. Pseudocode is sufficient.

Both methods can be found in this Java program. Both are bitwise approaches, which are usually the fastest:

<https://drive.google.com/file/d/0B_ceT2Lz-EMvRVpnMENyU01yZFE/view?usp=sharing> (code below)

Implement a second algorithm for the same problem.

A second method is also in the link above.

Code PowerOfTwo.java:

/\*\*

\* Shamelessly stolen from http://javarevisited.blogspot.com/2013/05/how-to-check-if-integer-number-is-power-of-two-example.html

\*

\* Note this could also be done with brute force methods like multiplying two by itself recursively or in a loop,

\* but bit shifting is generally faster.

\*

\* Recall that zero - i.e. "0" - is NOT a power of 2.

\*/

public class PowerOfTwo {

/\*

\* checking if number is power of 2 using bit shift operator in java

\* e.g. 4 in binary format is "0000 0000 0000 0000 0000 0000 0000 0100";

\* and -4 is "1111 1111 1111 1111 1111 1111 1111 1100";

\* and 4&-4 will be "0000 0000 0000 0000 0000 0000 0000 0100"

\*/

public static boolean isPositiveNegative(int number) {

return ((number > 0) && (number & -number) == number);

}

/\* See: http://www.exploringbinary.com/ten-ways-to-check-if-an-integer-is-a-power-of-two-in-c/

\* x x – 1 x & (x – 1)

\* 00000001 00000000 00000000

\* 00000100 00000011 00000000

\* 00010000 00001111 00000000

\*/

public static boolean isMinusOne(int number)

{

return ((number > 0) && (number & (number - 1)) == 0);

}

public static void main(String... args) {

for (int x = 0; x <= 100; x++) {

System.out.println(x + ":\t" + isPositiveNegative(x) + "\t" + isMinusOne(x));

}

}

}

## Storing Trees

Suppose you had a tree structure, where each node contains a geographic name (e.g., continent, country, state, etc.), and each node can have many children but exactly one parent. What database schema would you use to store such a structure? Given a particular node, how would you enumerate all of that node’s siblings?

Not many times I get to brag, but in this particular case I did write the definitive StackOverflow question on the topic, which in this case is really the answer as well:

<http://stackoverflow.com/questions/4048151/what-are-the-options-for-storing-hierarchical-data-in-a-relational-database>

That noted, these days I’d want to use a graph database, such as neo4j where the Cypher query to do what is asked would look like this:

MATCH (node {name:'{NAME\_OF\_SIBLING’})<-[]-(parent)   
WITH node, parent MATCH (parent)-[]->(sibling)   
WHERE NOT node = sibling   
RETURN DISTINCT sibling.name ORDER BY sibling.name;

But, I suspect this about using a relational database. To choose an approach a process of elimination to make a choice. To start, it’s unclear from the question if the nodes would always follow a particular named hierarchy, for instance instead of going continent->country->state the question seems to leave open the possibility of continent->state. Also the question implies the nodes have a type as well as a name. This excludes both Lineage Column approaches, which wants a named consistent hierarchy. A Flat table doesn’t really work well either because geographies do change as the result of wars, politics and acts of God implying updates. And, a Flat Table format is not easy to update.

This leaves two Nested approaches and Adjacency List. Here, eliminate Adjacency List because geography changes infrequently. As a result, fast changes, the forte of Adjacency List are not really needed and its shortcomings are still considerable - pathfinding, ancestry , descendants, level, et al.

Between the last two Nested Sets and Nested Intervals, Nested Sets are an easier implementation and because geography changes infrequently the easier updates Nested Intervals offers are unnecessary. This leaves us with a Nested Set whose data would look something like this:

|  |  |  |  |
| --- | --- | --- | --- |
| node | lft | rgt | type |
| Earth | 1 | 18 | Planet |
| North America | 2 | 17 | Continent |
| United States | 3 | 8 | Country |
| Delaware | 4 | 5 | State |
| Maryland | 6 | 7 | State |
| Canada | 9 | 16 | Country |
| Alberta | 10 | 11 | Province |
| Ontario | 12 | 13 | Province |
| Saskatchewan | 14 | 15 | Province |

Before getting to the SQL statement below, it bears some explanation that an Adjacency List would certainly be easier for this particular problem, but the Nested Set provides the ability to answer a number of other questions much more easily. Questions that seem likely to be asked about such a data model, for instance “What are all the cities in North America?” Therefore I choose it as the way to represent the data.

From here the SQL statement to get a node’s siblings - let’s assume SQLite with a table name of “geography” and we’re looking for the United State’s siblings - is as follows:

SELECT sibling.node

FROM

(

SELECT parent.lft, parent.rgt, parent.node, child.node as child\_node

FROM geography child, geography parent

WHERE

child.node = 'United States'

AND child.lft - 1 BETWEEN parent.lft AND parent.rgt

ORDER BY parent.lft DESC

LIMIT 1

) AS parent,

geography sibling

WHERE

sibling.lft - 1 BETWEEN parent.lft AND parent.rgt

AND sibling.node <> parent.child\_node

AND NOT EXISTS ( -- remove nodes beneath siblings

SELECT \* FROM geography mid

WHERE

mid.lft BETWEEN parent.lft AND parent.rgt

AND sibling.lft BETWEEN mid.lft AND mid.rgt

AND mid.node NOT IN (parent.node, sibling.node)

)

That SQL produces the following output based on the dataset above:

Canada

## Faulty Program

What’s wrong with the program below?

public static scrollDown (integer a)

{

(a = 0) ? scrollDown(100) : scrollDown(a-1);

}

main(int arg1, arg2)

{

if (arg2 <= 0)

if (arg2 < 0)

print("Too small.")

else

print("Just right.")

scrollDown(arg1, "start");

}

See FaultyProgram.java: <https://drive.google.com/file/d/0B_ceT2Lz-EMvbEdva00yXy1GZ2c/view?usp=sharing>

Code FaultyProgram.java:

// ISSUE: missing class declaration

public class FaultyProgram {

// ISSUE: missing return type & "integer"

// is not a valid Java type

// OLD\_CODE: public static scrollDown (integer a)

public static void scrollDown (int a)

{

// ISSUE: (a = 0) is a declaration not a boolean test

// ISSUE: a ternary operator is used for making

// an assignment in Java

// ISSUE: the algorithm expressed produces an infinite

// loop - intent is unclear so fix is not knowable

// OLD\_CODE: (a = 0) ? scrollDown(100) : scrollDown(a-1);

}

// ISSUE: missing return type & argument type for 'arg2'

// ISSUE: also the expectation is probably that this is treated

// as "public static void main(String args...)"

// but this will not work where Java when given a class to start

// with from the command line looks for a method on

// that class, unless told otherwise, that matches exactly that signature.

// OLD\_CODE: main(int arg1, arg2)

void main(int arg1, int arg2)

{

// ISSUE: while not wrong, the nested if block below is probably

// better expressed as

// if (arg2 < 0) {

// System.out.print("Too small");

// } else if (arg2 == 0) {

// System.out.print("Just right.");

// }

// In addition this test does not seem to be meaningful because

// it does not affect program execution in any way

// and arg2 is not used for anything else

if (arg2 <= 0)

if (arg2 < 0)

// ISSUE: no "print" function and missing a terminating

// semicolon

// OLD\_CODE: print("Too small.")

System.out.print("Too small");

else

// ISSUE: no "print" function and missing a terminating

// semicolon

// OLD\_CODE: print("Just right.")

System.out.print("Just right.");

// ISSUE: no overloaded 'scrollDown' function taking an integer

// and String argument

// OLD\_CODE: scrollDown(arg1, "start");

scrollDown(arg1);

}

}

## Adding Two Numbers

Write a Java program to add two numbers.

See AddTwoNumbers.java on Google drive (code inline below): <https://drive.google.com/file/d/0B_ceT2Lz-EMvLU1LTTM0blVqUHM/view?usp=sharing>

Anticipate edge-case problems and future requirements for this program, and explain how you will adapt.

The “add” method takes two, and returns one, String. This allows for some degree of future proofing along the lines I outline next.

**Complex Numbers**: these would come in the format “a+bi” with a and b both allowed to be in the form of valid floating point numbers. And b representing the imaginary piece. Because it’s addition implementing the formula is straightforward:

(m + ni) + (p+qi) = (m + p) + (n + q)i

As such a simple parsing and combining algorithm should do. And, allow for a continued backing BigDecimal implementation - a pair for real numbers and another pair for imaginary. A review of the Apache Math Commons Complex object’s source code reveals a similar approach, so this should leave things on good footing: <https://git-wip-us.apache.org/repos/asf?p=commons-math.git;a=blob;f=src/main/java/org/apache/commons/math3/complex/Complex.java;h=c8bd2113bcb117a6ea443bae1f8d1369fa567333;hb=ef6e0f882819e7c5230aece1610297e67775cca2#l156>

**Fractions**: that is a value with a numerator, bar and denominator - e.g. 100/225. Before doing the addition these types of values would need conversion to a decimal by dividing the numerator by the denominator. A parsing algorithm detecting the bar is where I would start for this type of value.

**Locale Specific Number Formats**: some locales use a comma to indicate a decimal instead of a period. One approach would be to use the default locale of the JVM in combination with the Java DecimalFormat class: <http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html> Another might be to provide a way to overload the method with a locale as an additional argument.

Code AddTwoNumbers.java:

import java.math.BigDecimal;

public class AddTwoNumbers {

public static void main(String... args) {

System.out.println(add(args[0], args[1]));

}

// Relies upon BigDecimal's built in number parsing to convert to an appropriate value

// this handles sign, integers, fractions and exponents

// see http://docs.oracle.com/javase/7/docs/api/java/math/BigDecimal.html#BigDecimal(java.lang.String)

public static String add(String a, String b) {

return new BigDecimal(a).add(new BigDecimal(b)).toString();

}

}

## Joint Membership

Given two long lists of integers, A and B, use whatever tool/program you like to list out:

* All elements present in both lists (elements must be present in list A and list B)
* All elements present in exactly one list, but not the other (“in A but not in B” as well as “in B but not in A”)
* All elements present in any list, but discarding duplicates

The problem description makes unclear if duplicates matter for the first two items. I assumed no and used sets to solve these problems. I also assume both lists can fit into main memory.

See sets.py: <https://drive.google.com/file/d/0B_ceT2Lz-EMvcGxXc0Itb1p2MEk/view?usp=sharing>

Code sets.py:

import random

def main():

a = set([random.randint(0, 10) for n in range(random.randint(5, 10))])

b = set([random.randint(0, 10) for n in range(random.randint(5, 10))])

print "a:\t" + str(a)

print "b:\t" + str(b)

print ""

print "All elements present in both lists (elements must be present in list A and list B)"

print a.intersection(b)

print ""

print "All elements present in exactly one list, but not the other (""in A but not in B"" as well as ""in B but not in A"")"

print set(list(a.difference(b)) + list(b.difference(a)))

print ""

print "All elements present in any list, but discarding duplicates"

print set(list(a) + list(b))

return

if \_\_name\_\_ == "\_\_main\_\_":

main()

## Mean Time

Write an algorithm to find the mean (average) of an extremely large list of numbers. The list could contain trillions or quadrillions of numbers, but each number is (relatively) manageable, e.g., in the hundreds, thousands or millions.

I treated this problem as the ability to get an average at any time while processing a stream of numbers. I made two assumptions:

1. memory is a constraint such that the it is impossible to hold onto all of the values seen.
2. it would be too slow to hold onto a single sum and number of instances using objects that can hold onto an arbitrarily large values - Java’s BigDecimal or BigInteger for instance - because the division would be a very slow operation to actually get the average as the number of observations grew.

Assuming the above is true, you end up making a tradeoff in precision where interim average calculations may have infinite decimals (e.g. ⅓) not accurately capturable. The following shows how to keep a running average using built-in Java types:

Google Drive Average.java: <https://drive.google.com/file/d/0B_ceT2Lz-EMvc3pGcU9yQlRUYkE/view?usp=sharing>

Note, as a sanity check I also used the Apache Math Commons Mean object and came up with similar results to my own algorithm:

<http://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math3/stat/descriptive/moment/Mean.html>

Code Average.java:

public class Average {

int observations = 0;

double average = 0;

public void update(double x) {

average = (average \* observations + x) / (++observations);

}

public double getAverage() {

return this.average;

}

public static void main(String... args) {

Average oAvg = new Average();

int[] numbers = new int[]{10,20,15,25,16,60,100};

int sum = 0;

for(int i=0; i < numbers.length ; i++) {

sum = sum + numbers[i];

oAvg.update(numbers[i]);

}

//calculate average value

double average = sum / numbers.length;

System.out.println("Average value of array elements evaluated once is : " + average);

System.out.println("Average value of array elements evaluated dynamically is : " + oAvg.getAverage());

}

}

## Anagrams

How would you determine if one string is an anagram of another (ignoring whitespace, case and punctuation)? For example, “A man’s rag” and “anagrams” are anagrams.

1. For both strings
2. Strip all non-alphanumeric values - including spaces and punctuation.
3. Lowercase the resulting string.
4. Iterate through the string counting the number of times each character appears in a map.
5. Compare the maps for each string for equality

See AnagramSolver.java: <https://drive.google.com/file/d/0B_ceT2Lz-EMvSWFfMlNQcWREUWs/view?usp=sharing>

Code AnagramSolver.java:

import java.util.HashMap;

import java.util.Map;

public class AnagramSolver {

public static boolean isAnagram(String a, String b) {

a = a.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();

b = b.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();

return (getCharCounts(a).equals(getCharCounts(b)));

}

private static Map<Character, Integer> getCharCounts(String value) {

Map<Character, Integer> counts = new HashMap<Character, Integer>();

for (Character c : value.toCharArray()) {

if (counts.containsKey(c)) {

counts.put(c, counts.get(c) + 1);

} else {

counts.put(c, 1);

}

}

return counts;

}

public static void main(String... args) {

String a = "A man’s rag";

String b = "anagrams";

System.out.println(isAnagram(a, b));

String c = "test";

String d = "wrong";

System.out.println(isAnagram(c, d));

String e = "Reach Excel Nettle";

String f = "Excellent Teacher";

System.out.println(isAnagram(e, f));

}

}

## Persistence

Given a class like the one below, suggest some ways to store thousands of instances of it on disk. Augment the class with whatever you need.

public class Superhero

{

private String name;

private Date debut;

private int numVillainsFought;

private short numAlterEgos;

private boolean masked;

private boolean female;

private boolean retired;

}

Absent additional requirements, for instance searchable or accessible outside of Java I would lead with Java serialization, a file per object, because it’s a readily accessible option. I think a file per instance is safe because the question is for thousands of instances where the reading I have done is most operating systems (OS) are good with thousands of files in a single directory.

In short the changes to the class are to implement the Serializable interface and a add “static final long serialVersionUID = 1L” variable to version the class. The example referenced below does not write out to a file for expediency’s sake, instead using an in memory byte stream, but should make things clear otherwise.

Superhero.java (code inline below): <https://drive.google.com/file/d/0B_ceT2Lz-EMvaW9vdjFyV19wYWs/view?usp=sharing>

In terms of writing out files, I would use a file per object. The file name would either be hashCode based with some check for conflicts or use a UUID to provide a practical guarantee of no collisions in the case of thousands of instances. This should be a workable solution because there’s no search requirement.

Finally, going back to my original statements regarding searchable or accessible, I would probably in practice build this out using a backing relational database and annotate the class with the Java Persistence API (JPA). But I wanted to write out a complete runnable code example in a single file and serialization was the most straightforward option.

Code Superhero.java:

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

public class Superhero implements Serializable

{

public static final long serialVersionUID = 1L;

private String name;

private Date debut;

private int numVillainsFought;

private short numAlterEgos;

private boolean masked;

private boolean female;

private boolean retired;

public static void main (String... args) {

Superhero superhero = new Superhero();

superhero.setName("Superman");

try {

Date debut = new SimpleDateFormat("MMMM dd, yyyy").parse("April 18, 1938");

superhero.setDebut(debut);

} catch (ParseException e) {

throw new RuntimeException(e);

}

superhero.setNumVillainsFought(Integer.MAX\_VALUE);

superhero.setNumAlterEgos((short)2);

superhero.setMasked(false);

superhero.setFemale(false);

superhero.setRetired(false);

ByteArrayOutputStream baos = new ByteArrayOutputStream();

try {

ObjectOutputStream oos = new ObjectOutputStream(baos);

oos.writeObject(superhero);

oos.close();

} catch (IOException e) {

throw new RuntimeException(e);

}

ByteArrayInputStream bais = new ByteArrayInputStream(baos.toByteArray());

Superhero superhero2 = null;

try {

ObjectInputStream ois = new ObjectInputStream(bais);

superhero2 = (Superhero) ois.readObject();

ois.close();

bais.close();

baos.close();

} catch (Exception e) {

throw new RuntimeException(e);

}

System.out.println(superhero.equals(superhero2));

}

/\*\*

\* Default no-arg constructor used for deserialization

\*/

public Superhero() {

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Date getDebut() {

return debut;

}

public void setDebut(Date debut) {

this.debut = debut;

}

public int getNumVillainsFought() {

return numVillainsFought;

}

public void setNumVillainsFought(int numVillainsFought) {

this.numVillainsFought = numVillainsFought;

}

public short getNumAlterEgos() {

return numAlterEgos;

}

public void setNumAlterEgos(short numAlterEgos) {

this.numAlterEgos = numAlterEgos;

}

public boolean isMasked() {

return masked;

}

public void setMasked(boolean masked) {

this.masked = masked;

}

public boolean isFemale() {

return female;

}

public void setFemale(boolean female) {

this.female = female;

}

public boolean isRetired() {

return retired;

}

public void setRetired(boolean retired) {

this.retired = retired;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Superhero superhero = (Superhero) o;

if (female != superhero.female) return false;

if (masked != superhero.masked) return false;

if (numAlterEgos != superhero.numAlterEgos) return false;

if (numVillainsFought != superhero.numVillainsFought) return false;

if (retired != superhero.retired) return false;

if (debut != null ? !debut.equals(superhero.debut) : superhero.debut != null) return false;

if (name != null ? !name.equals(superhero.name) : superhero.name != null) return false;

return true;

}

@Override

public int hashCode() {

int result = name != null ? name.hashCode() : 0;

result = 31 \* result + (debut != null ? debut.hashCode() : 0);

result = 31 \* result + numVillainsFought;

result = 31 \* result + (int) numAlterEgos;

result = 31 \* result + (masked ? 1 : 0);

result = 31 \* result + (female ? 1 : 0);

result = 31 \* result + (retired ? 1 : 0);

return result;

}

@Override

public String toString() {

return "Superhero{" +

"name='" + name + '\'' +

", debut=" + debut +

", numVillainsFought=" + numVillainsFought +

", numAlterEgos=" + numAlterEgos +

", masked=" + masked +

", female=" + female +

", retired=" + retired +

'}';

}

}

## Technologies

For each of the technologies below, place an X in the column that best describes your level of experience with it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Technologies | Just Googled it / Have not used | Used 2+ years ago | Used it actively recently | Patched it |
| Hibernate / JPA |  |  | **X** |  |
| JSON |  |  | **X** |  |
| MySQL |  | **X** |  |  |
| InfoBright | **X** |  |  |  |
| Spring |  |  | **X** |  |
| Spring MVC | **X** |  |  |  |
| jQuery / Ext JS / Flot |  | **X** |  |  |
| Eclipse / IntelliJ |  |  | **X** |  |
| JMS / ActiveMQ / RabbitMQ | **X** |  |  |  |
| Unix / Linux command-line |  |  | **X** |  |
| Perl / awk / shell scripts |  |  | **X** |  |
| Ant / Maven / make |  |  | **X** |  |
| CVS / SVN / Perforce / … |  |  | **X** |  |
| Jenkins / Hudson / CruiseControl / … |  |  | **X** |  |
| Cassandra / Redis / MongoDB / memCached | **X** |  |  |  |
| Tomcat |  |  | **X** |  |
| JUnit |  |  | **X** |  |
| JaxB | **X** |  |  |  |
| AWS / Cloud Computing | **X** |  |  |  |
| Graphite | **X** |  |  |  |
| Nagios | **X** |  |  |  |

## Following Up

When you have completed the questionnaire, please email your resume to [careers@videologygroup.com](mailto:careers@videologygroup.com?subject=Software%20Development%20Candidate) with this document attached. Include the words "Software Development Candidate" in the subject line of the email. We will reply to confirm receipt. If you don't receive confirmation within three business days, please inquire by email or phone.

If you prefer, you may mail a hard copy of your resume and questionnaire to the address below. We will not be able to confirm receipt.

There is no need to contact us to follow up. We will contact you for an interview at our discretion. Thank you for your application.

**Videology, Inc.**1500 Whetstone Way

Suite 500

Baltimore, MD 21230

Phone: (410) 276-8700

11490 Commerce Park Drive

Suite 220

Reston, VA 20191

Phone: (703) 207-0040

<http://www.videologygroup.com>  
[careers@videologygroup.com](mailto:careers@videologygroup.com?subject=Software%20Development%20Candidate)