If Java did not have a stack or map, how would you ….

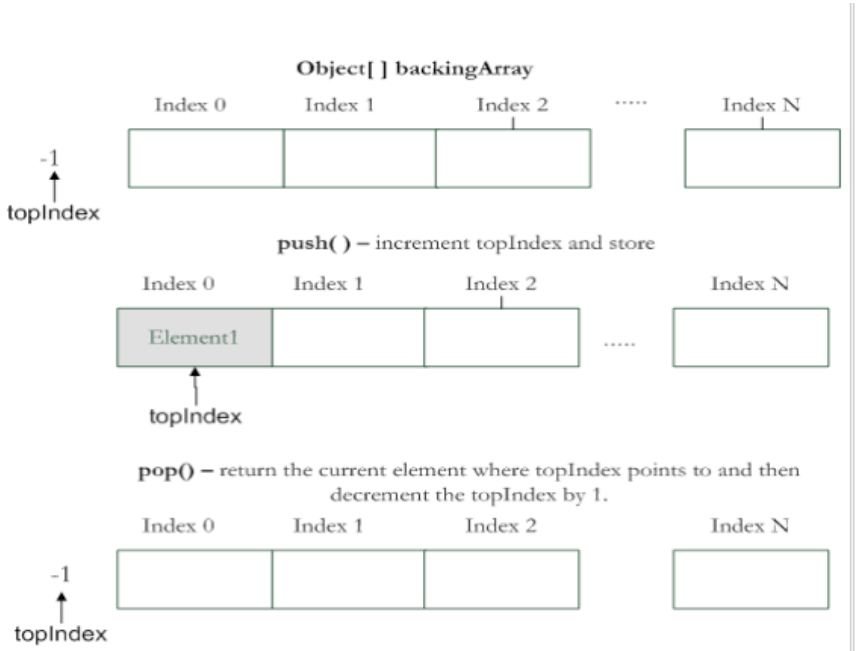
Posted on October 11, 2014 by Arulkumaran Kumaraswamipillai — 1 Comment ↓

Q1 If Java did not have a **Stack** implementation, how would you go about implementing your own?

A1

1) Determine the backing data structure (e.g. array, linked list, etc).
2) Determine the methods that need to be implemented like pop( ), push( ), peek( ), clear( ), empty( ), etc.
3) Determine how you are going to keep track of the last item added. For example, keeping track with an index variable “topIndex”.
4) Determine what to do if the capacity is reached? Double the capacity of the backing array.

5) Drawing a diagram as shown below could make things clearer.



Here are some code snippets to get started. Define the interface first.

```
1 public interface Stack<E> {
2     E push(E item);
3     E pop( );
4     boolean empty( );
5     //other methods like peek( ), etc omitted for
6 }
```

Now provide the implementation class.

```
1  public class MyStack<E> implements Stack<E> {
2
3      private Object[ ] backingArray;
4      private int topIndex;
5      private static final int DEFAULT_CAPACITY =
6
7      public MyStack( ) {
8          this.backingArray = new Object[DEFAULT_C
9          topIndex = -1;
10     }
11
12     @Override
13     @SuppressWarnings("unchecked")
14     public E pop( ) {
15         if (empty( )) {
16             throw new RuntimeException("No data
17         }
18         // return the index element and reset th
```


16 Technical Key Areas

```
19          return (E) backingArray[topIndex--];
20      }
21
22      @Override
23      public E push(E item) {
24          doubleTheArrayIfLimitIsReached( );
25          //increment the index and store the item
26          this.backingArray[++topIndex] = item;
27          return item;
28      }
29
30      @Override
31      public boolean empty( ) {
32          return topIndex == -1;
33      }
34
35      /**
36       * If the array capacity is reached, double
37       */
38      private void doubleTheArrayIfLimitIsReached(
39          if (topIndex + 1 == this.backingArray.le
40              Object[ ] newArray = new Object[back
41              // copy the existing elements to the
42              System.arraycopy(backingArray, 0, ne
43                      backingArray.length);
44              // set the backingArray to the expan
45              backingArray = newArray;
46              System.out.println(
47                          "The backing array has b
48          }
49      }
50  }
```

Q2 If Java did not have a **Queue** implementation, how would you go about implementing your own?
A2 A queue can be implemented in a similar fashion to a **Stack** by declaring a backingArray, a frontIndex, a backIndex, and a currentSize. Initialize the frontIndex to 0 and the backIndex to -1. You will have to provide the relevant methods like enque( ), deque( ), etc.

enqueue( ):

1) If the currentSize == backingArray.length extend the backing array capacity by doubling it.
2) Increment the backIndex by 1. If the incremented index is equal to the array length (backingArray.lenth), set the backIndex to 0 (i.e. imagine it as an endless circular array).
3) Set the element to the backIndex (i.e. bacckingArray[backIndex] = element).
4) Increment the currentSize by 1 (i.e. currentSize++).

dequeue( ):

⊞ Low Latency (7)
⊞ Memory Managemen
⊞ Performance (13)
⊞ QoS (8)
⊞ Scalability (4)
⊞ SDLC (6)
⊞ Security (13)
⊞ Transaction Managen

1) Decrement the currentSize by 1 (i.e currentSize-1).
2) Return the value that frontIndex points to (i.e. backingArray[frontIdex]).
3) Increment the frontIndex by 1. If the frontIndex == backingArray.length, set the frontIndex to 0 (i.e. imagine it as an endless circular array).

Q3 If Java did not have a **HashMap** implementation, how would you go about writing your own one?
A3 Writing a HashMap is not a trivial task. It is also not a good practice to reinvent the wheel. The interviewer is trying to evaluate your level of technical knowledge and thought process. What really important here are, how you approach the problem?, how good your understanding of the data structures are?, and how well you can logically think and code?

1) Firstly, decide on the backing data structure. Arrays are fast and memory efficient. Hence an array can be used to back up the map. This will be an indexed bucket.

2) Decide on a hashing algorithm to index the array and store the entries in a particular slot of the array.

3) Decide on a strategy to store two or more keys that result in the same hash code value. More objects can be linked with each other occupying the same index. This means each bucket can contain 0..N entries. This linking strategy is shown in the diagram.

4) Decide on an approach to evaluate the hash code, which determines the array bucket index to use.

```
1 int index = Math.abs(key.hashCode( ) % buckets.le
```

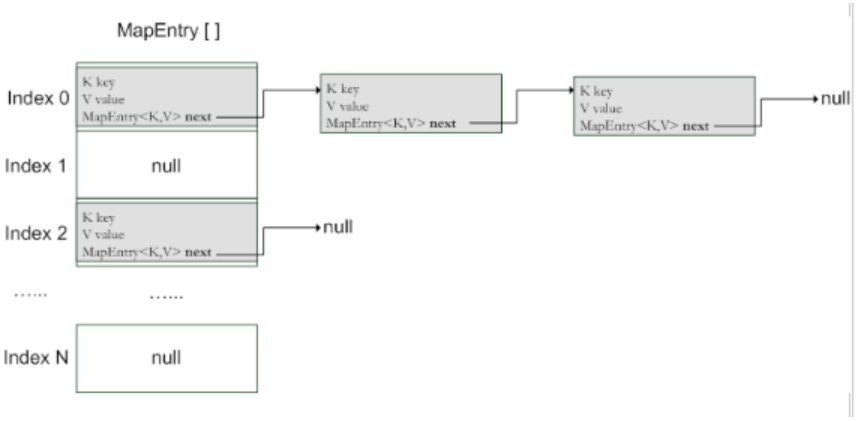
5) Come up with a strategy for resizing the backing array when the capacity is reached. This process is known as rehashing whereby existing items are mapped to new bucket locations.

Written Test JEE (1)

How good are your .....?

open all | close all
Career Making Know-
Job Hunting & Resum

6) Consider implementing the Map interface and fill in the empty methods that need to be implemented.



The following code snippets demonstrate how you could go about implementing your own **HashMap** class.

```
1  public class MapEntry <K, V> {
2
3      private final K key;
4      private V value;
5      private MapEntry<K, V> next;
6
7      public MapEntry(K key, V value){
8          this.key = key;
9          this.value = value;
10     }
11
12     // getters and setters are ignored for brevi
13 }
```

Next, the **HashMap** class that is composed of the **MapEntry**.

```
1  import java.util.Collection;
2  import java.util.Map;
3  import java.util.Set;
4
5  public class MyHashMap<K, V> implements Map<K,
6
7      private MapEntry<K, V>[ ] buckets = null;
8
9      public MyHashMap( ) {
10         this(10);
11     }
12
13     @SuppressWarnings("unchecked")
14     public MyHashMap(int capacity) {
15         buckets = new MapEntry[capacity];
16     }
17
18     @Override
19     @SuppressWarnings("unchecked")
```

```
20      public V get(Object key) {
21          // check for pre-condition.
22          if (key == null)
23              throw new IllegalArgumentException(
24
25          MapEntry<K, V> entry = getMatchingEntry
26          return entry != null ? entry.getValue(
27      }
28
29      @Override
30      public V put(K key, V value) {
31          // check for pre-condition.
32          if (key == null)
33              throw new IllegalArgumentException(
34
35          MapEntry<K, V> entry = getMatchingEntry
36
37          if (entry != null) {
38              entry.setValue(value);
39          }
40          else {
41              int index = evalBucketIndex(key);
42              if (buckets[index] != null) {
43                  entry = linkEntry(buckets[index
44
45
46              } else {
47                  entry = new MapEntry<K, V>(key,
48                  buckets[index] = entry;
49              }
50          }
51          return entry.getValue( );
52      }
53
54     /**
55       * Hashing function. The more spaced out th
56       * performance is. Only positive values can
57       */
58      private int evalBucketIndex(K key) {
59          int index = Math.abs(key.hashCode( ) %
60          return index;
61      }
62
63      /**
64       * Return the matching entry if found, othe
65       */
66      private MapEntry<K, V> getMatchingEntry(K k
67          MapEntry<K, V> entry = buckets[evalBuck
68
69          // same index can have more entries lin
70          // in the MapEntry class. So find the o
71          while (entry != null && !key.equals(ent
72              entry = entry.getNext( );
73          }
74
75          // conditional operator
76          return entry != null ? entry : null;
77      }
78
79      /**
80       * Create a new entry and link it to the gi
81       */
82      private MapEntry<K, V> linkEntry(MapEntry<K
83          boolean linked = false;
84
85          while(!linked){
```

```
86              if(entry.getNext( ) == null){
87                  entry.setNext(new MapEntry<K, V
88
89                  linked = true;
90              }
91              else{
92                  entry = entry.getNext( );
93              }
94          }
95
96          return entry;
97      }
98
99      //other methods that need to be implemented
100     //Map interface not shown for brevity.
101 }
```

Popular Posts

♦ 11 Spring boot interview questions & answers

823 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

765 views

18 Java scenarios based interview Questions and Answers

399 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ ♥♦ Regular Expressions (aka Regex) by examples for Java developers

♦ Sorting objects in Java interview Q&As ›

Posted in Collection and Data structures, member-paid

One comment on "If Java did not have a stack or map, how would you ...."

hzstanley says:

April 4, 2016 at 11:32 am

Hi Arun

Please would you be able to post a demo class for the above Stack example.

Thanks

Reply

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.

↑