Register | Login | Logout | Contact Us

Java-Success.com

Industrial strength Java/JEE Career Companion to open more doors

search here … Go

Home | Java FAQs | 600+ Java Q&As | Career | Tutorials | Member | Why?

Can u Debug? | Java 8 ready? | Top X | Productivity Tools | Judging Experience?

Home › Interview › Core Java Interview Q&A › Collection and Data structures › ♥♦ HashMap & HashSet and how do they internally work? What is a hashing function?

♥♦ HashMap & HashSet and how do they internally work? What is a hashing function?

Posted on February 9, 2016 by Arulkumaran Kumaraswamipillai

HashMap is not only one of the frequently used data structures, but also one of the popular interview topics.

Q1. How does a HashMap store data?
A1. As key/value pairs. You can store and retrieve values with the keys.

Q2. What is the HashMap lookup time in Big O notation?
A2. It is O(**n**) = O(k * n). On average it is O(1) if the

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

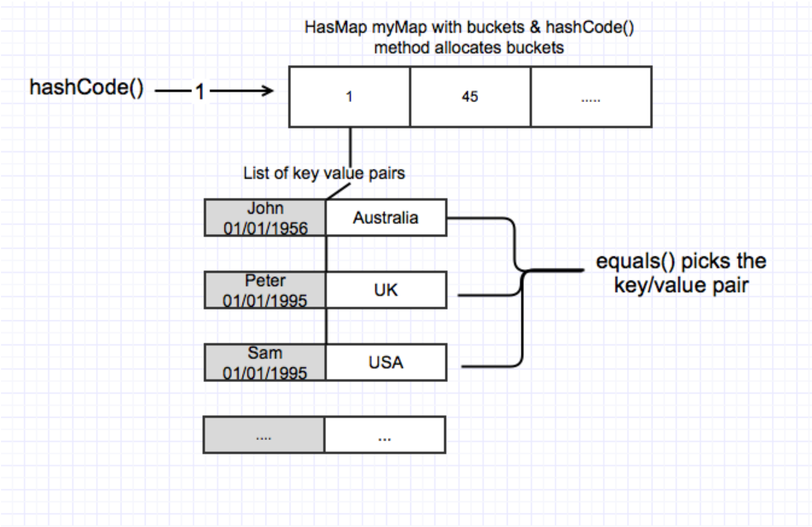
open all | close all

- Ice Breaker Interview
- Core Java Interview (
  - Java Overview (4)
  - Data types (6)
  - constructors-methc
  - Reserved Key Wor
  - Classes (3)
  - Objects (8)
    - ▶ Beginner Jav
    - ♥♦ HashMap & H
    - ♦ 5 Java Object
    - ♦ Java enum inte
    - ♦ Java immutabl
    - ♦♥ Object equals
    - Java serializatio
    - Mocks, stubs, dc
  - OOP (10)
  - GC (2)
  - Generics (5)
  - FP (8)
  - IO (7)

hashCode() method spreads out the buckets as discussed below.

Q3 How does a HashMap internally store data?
A3. A backing array for the buckets and a linked list to store key/value pairs.

Backing Array with buckets: as shown below.



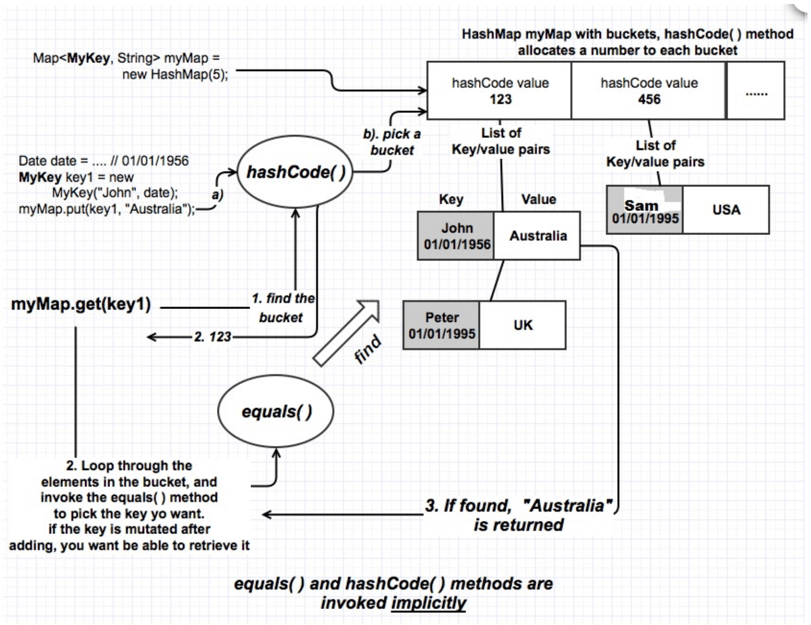
hashCode() returns 1, 45, etc

1) When you put an object into a map with a key and a value, **hashCode()** method is implicitly invoked, and hash code value say 123 is returned. Two different keys can return the same hash value. A good hashing algorithm spreads out the numbers. In the above example, let's assume that ("John",01/01/1956) key and ("Peter", 01/01/1995) key return the same hash value of **123**.

⊞ Multithreading (12)
⊞ Algorithms (5)
⊞ Annotations (2)
⊟ Collection and Data
♦ Find the first n
♦ Java Collection
♥ Java Iterable V
♥♦ HashMap & H
♦ Sorting objects
02: ♦ Java 8 Stre
04: Understandi
4 Java Collection
If Java did not ha
Java 8: Different
Part-3: Java Tree
Sorting a Map by
When to use whi
⊞ Differences Betwee
⊞ Event Driven Progr
⊞ Exceptions (2)
⊞ Java 7 (2)
⊞ Java 8 (24)
⊞ JVM (6)
⊞ Reactive Programn
⊞ Swing & AWT (2)
⊞ JEE Interview Q&A (3
⊞ Pressed for time? Jav
⊞ SQL, XML, UML, JSC
⊞ Hadoop & BigData Int
⊞ Java Architecture Inte
⊞ Scala Interview Q&As
⊞ Spring, Hibernate, & I
⊞ Testing & Profiling/Sa
⊞ Other Interview Q&A f
⊞ ▶ Free Java Interview

16 Technical Key Areas

open all | close all


HashMap myMap with buckets, hashCode( ) method allocates a number to each bucket
Map<MyKey, String> myMap = new HashMap(5);
hashCode value 123
hashCode value 456
......
b). pick a bucket
List of Key/value pairs
List of Key/value pairs
Date date = ... // 01/01/1956
MyKey key1 = new MyKey("John", date);
myMap.put(key1, "Australia");
a)
hashCode( )
Key
Value
John 01/01/1956
Australia
Sam 01/01/1995
USA
myMap.get(key1)
1. find the bucket
2. 123
find
Peter 01/01/1995
UK
equals( )
2. Loop through the elements in the bucket, and invoke the equals( ) method to pick the key yo want. if the key is mutated after adding, you want be able to retrieve it
3. If found, "Australia" is returned
equals( ) and hashCode( ) methods are invoked implicitly

Java equals vs hashCode

2) Now when a hashCode value of say 123 is returned and the initial HashMap capacity is say 10, how does it know which index of the backing array to store it in? This is where the HashMap internally invokes the **hash(int)** & **indexFor(int h, int length)** methods. This is known as the **hashing** function. This function in a very simple explanation is like

```

hashCode() % capacity

123 % 10 = 3
456 % 10 = 6

```

This means the "hashcode = 123" gets stored in index 3 in the backing array. So you get numbers between **0** and **9** for the capacity of 10. Once the HashMap reaches its 75% of the capacity indicated by the default hash factor of 0.75, the backing array's capacity is doubled and the **rehashing** takes place to reallocate the buckets for the new capacity of 20.

```

hashCode() % capacity

123 % 20 = 3
456 % 20 = 16

```

⊞ Best Practice (6)
⊞ Coding (26)
⊞ Concurrency (6)
⊞ Design Concepts (7)
⊞ Design Patterns (11)
⊞ Exception Handling (3
⊞ Java Debugging (21)
⊞ Judging Experience I
⊞ Low Latency (7)
⊞ Memory Managemen
⊞ Performance (13)
⊞ QoS (8)
⊞ Scalability (4)
⊞ SDLC (6)
⊞ Security (13)
⊞ Transaction Managen

80+ step by step Java Tutorials

open all | close all

⊞ Setting up Tutorial (6)
⊞ Tutorial - Diagnosis (2
⊞ Akka Tutorial (9)
⊞ Core Java Tutorials (2
⊞ Hadoop & Spark Tuto
⊞ JEE Tutorials (19)
⊞ Scala Tutorials (1)
⊞ Spring & HIbernate Tu
⊞ Tools Tutorials (19)
⊞ Other Tutorials (45)

100+ Java pre-interview coding tests

open all | close all

⊞ Can you write code? (

Now the above modulus operator approach of rehashing has a flaw. What if the hashCode is a negative number? You don't want a negative index. Hence, an improved hashing formula would be to shift out the sign bit and then calculate the remainder with the modulus (i.e. %) operator.

```

(123  & 0x7FFFFFFF) % 20 = 3
(456 & 0x7FFFFFFF) % 20 = 16

```

This ensures that you get a positive index value. If you look at the Java 8 HashMap source code, its implementation uses

a). Protect against poorer hashes by only extracting the important lower bits.

```

static int hash(int h) {
     // This function ensures that hashCodes that
     // constant multiples at each bit position h
     // number of collisions (approximately 8 at
     h ^= (h >>> 20) ^ (h >>> 12);
     return h ^ (h >>> 7) ^ (h >>> 4);
 }

```

b). Determine the index based on the **hashCode** and **capacity**.

```

static int indexFor(int h, int length) {
     return h & (length-1);
 }

```

Actual name value pairs are stored as a key/value pair LinkedList

As shown in the diagram, the key/value pairs are stored as linked list. It is important to understand that two different keys can result in the same hashcode say 123, and get stored in the same bucket. For example, "John, 01/01/1956" & "Peter, 01/01/1995" in the above example. So, how do you retrieve

⊞ ♦ Complete the given
⊞ Converting from A to I
⊞ Designing your classe
⊞ Java Data Structures
⊞ Passing the unit tests
⊞ What is wrong with th
⊞ Writing Code Home A
⊞ Written Test Core Jav
⊞ Written Test JEE (1)

How good are your .....?

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

just “John, 01/01/1956”? This is where you key class’s **equals()** method gets invoked. It loops through the items in the LinkedList bucket “123” to retrieve the key “John, 01/01/1956” by using the equals() method to find a match. This is why it is important to implement the **hashCode()** & **equals()** method in your key class. If you are using an existing wrapper class like Integer or the String as a key, it is already implemented. If you write your own key class like “MyKey” with name and DOB as the attributes as in “John, 01/01/1956”, you are responsible for properly implementing these methods.

Q5. Why is it a good practice to appropriately set the initial capacity of a HashMap?
A5. To minimize the occurrences of **rehashing**.

Q6. How does a HashSet internally store data?
A6. A HashSet internally uses a HashMap. It stores the element as both key and value.

Q7. What is the effect of implementing a poor hashcode() for an object?
A7. The invocation of hashCode() method should return different values for different objects. If it returns same values for different objects then this results in more key/value pairs getting stored against the same bucket. This **adversely impacts performance** of HashMaps & HashSets.

Relevant topics

1) 5 Java Object class methods interview questions & answers.

2) HashMap from scratch Java example

3) Stack from scratch Java example

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**

‹ 03: Java GC tuning for low latency applications

Huffman coding in Java ›

Posted in Collection and Data structures, Objects

Tags: Free Content, Free FAQs

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.

↑