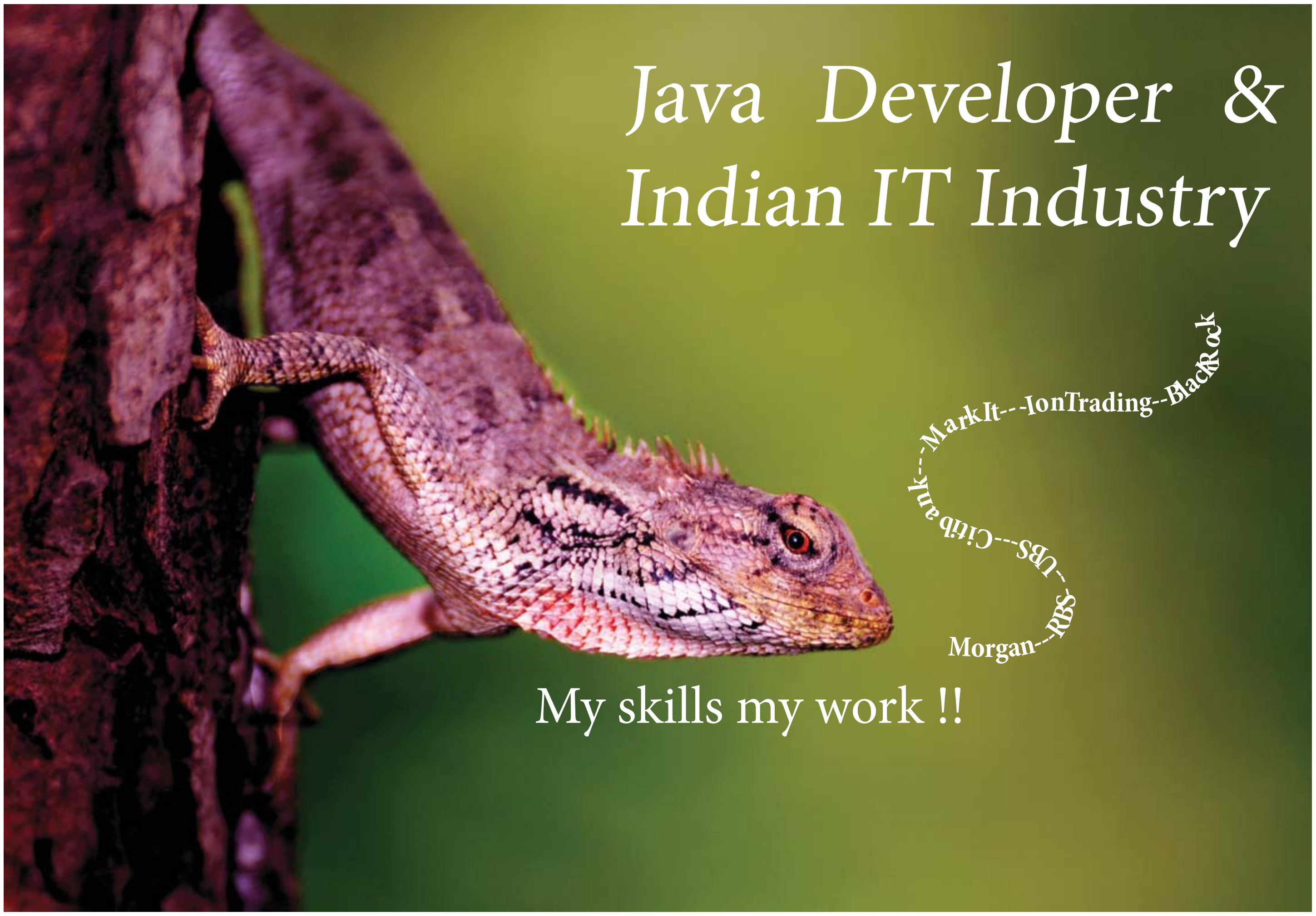
# Java Developer & Indian IT Industry


Morgan--RBS--UBS---Citibank---MarkIt---IonTrading--BlackRock
My skills my work !!

## Cultivate skills to create your own Path ...

This e-book covers following topics

1. Core Java (JDK 1.6)
2. Concurrency and Java Collections Framework
3. Algorithms, Data structures and Puzzles
4. Object Oriented Design Problem for the open discussion
5. Sample Interview Questions

*About Author*
**Munish Chandel**

Munish is currently working as a Java software developer for an investment bank in India.

Linked In Profile
http://linkedIn.com/munish.chandel

1st edition, 2013

# Java Developer & Indian IT Industry

This book attempts to address common problems faced by a Java Developer in Indian IT Industry

Topics Covered In This Book

(Core Java, Algorithms, Data Structures, Puzzles & Concurrency Problems)

# Preface

This work is my sincere effort to consolidate solutions to some basic problems faced by my fellow mates in their day to day work.

## This Book Isn't

- A research work, neither it is intended to be.
- Of much help to a fresher in IT industry as it expects some level of hands on experience. It doesn't even cover all the topics required by a newbie to start developing software from scratch.
- A reference book, one time read should be enough.

## This Book Is

- Collection of excerpts discussing the common problems faced by an experienced Java Developer in his day to day work. The intent is not to provide with the concrete solution to a given problem, but to show the approach to get the problem solved. And there could definitely be more efficient ways to solve the given problem compared to what is mentioned in this book. The approach shown here is limited to the knowledge of the author.
- Collection of topics in core Java, OOD, algorithms, data structures & Puzzles.

## Who should read this book ?

- Experienced candidates who are appearing for Java interviews specifically in investment banking domain (having approach for enterprise level applications)
- Experienced Java developers who want to brush up their skills to solve their day to day software problems.

## What does this book cover ?

See Index for the coverage of topics

## What this book does not cover ?

- It does not cover much stuff related to frameworks like Spring, Hibernate, Struts, EJB's, etc
- It does not cover much of database stuff like SQL & PLSQL.
- It does not provide with the concrete solution, though the authors tried his best to provide you with the most relevant pointers to the solution.

I hope this book adds value to your skills. Your reviews will be most valuable to me.

Munish Chandel
cancerian0684@gmail.com
http://linkedIn.com/munish.chandel
January 2013

This page is Intentionally left blank.

# Contents

# IT Industry and Psyche of Developers

## My Opinion in Indian Context

**Question : What is overall picture of Indian IT Industry ?**
IT is outcome of Intellectual Minds from western civilization in modern era. The knowledge is still sourced from west in this domain east at the time of this writing. The main reason for outsourcing IT related work to India is economically cheap labour, though the trends are changing now. Typically there are two types of companies in India -

1. Product Based (Adobe, Google, Amazon, Oracle, etc)
2. Services Based (TCS , Infosys, Cognizant, Wipro, HCL, Sapient, etc) - Broker Companies

There is a big cultural and compensation differences in both of these types. Product based company provides better culture for the employee in terms of personal satisfaction. Compensation offered in Product based companies is higher than Service based companies, but service based companies compensate this by short/long term on site opportunities.

**Question : What type of work people do in India IT Industry?**
We can categorize typical IT work into two types - Green Fields and Maintenance & Support.
Most people in Indian IT industry work for services based companies where major fraction of work is of Maintenance & Support type. Even most product based companies build their basic block outside and then outsource work to India. You could find exceptions to my words at some places.

**Question : What causes a typical developer to switch his/her job so frequent, Is that bad, Why that is not the case in best ?**
A thought to switch Job comes to one's mind when one finds blockage in growth opportunities in their current organization. I could list few reasons for the same -

1. Salary disparity and scope for higher salaries in new company is the major reason for job switch. Most service based companies tries to maximize profit on their side by offering lower salaries to its employees (anything starting from 5000 $ a year in typical service based company), but as people acquire more skills they switch for higher roles in new company. Demand and supply typically governs the salaries in India.
2. The quality of work is another reason for switch. Work quality directly relates to stress (more manual work more stress)
3. Shift Timings and location preference also causes people to switch their jobs
4. Very few people switch their job because they are unable to meet expectations of their company or because they got an unexpected work.

To some extent this switch is fair because we can't expect someone to work for a company few thousand dollars a year for his lifetime (considering the increments that we currently get), isn't ? As the Industry will mature, job shift will reduce. Moreover, IT companies are not supposed to hinder one's growth in current Caste System. The IT companies in west are mature, salaries are already saturated, people don't take much work stress, So western employes do not find many reasons for their Job switch.

**Question :  What is growth Pyramid of a typical Developer in Indian IT Industry ?**
There are two main streams in It companies - Technical & Management. Most Indian people start their carrier as a software developer/qa, but over the time they shift to Management Side. At least that is the trend in Service Based companies. This trend has started changing now, because of the saturation on number of jobs in management side. So people will experience longer stretch working as a developer in the coming decade. Employees working for startup and product based companies, tend to remain on technical side, because Individual Contribution gives them better growth opportunities.

**Question : What is typical psychology of an average Indian Developer ? What kind of chaos pollute their mind ?**
Most Indian opt for IT, not by choice but for money, because of large unemployment in India. Most others think that earning money in IT industry is easy and effortless compared to other opportunities. A great proportion wants to take IT as the jumping ground for their higher studies (MBA, MS, etc). An average fresher is polluted with the thoughts about his career development, and is unsure about his interests in IT field, trying various alternates in first few years.

**Question : What is the Problem with Most Indian Developers in terms of Skills ?**

Majority of IT crowd does not have good hold over their primary skills which are required for the work. The underlying cause for the faded skills is the type of Work which is fed to India. The majority of work does not require high quality of skills on developer's part. But still people can learn by their own, build their skills and fight for better quality work. One should have a very good hold over his primary skill set and look for work which is matching those skills, instead otherwise.

**Question : What are advantages of acquiring skills ?**

There are many. Listing few of them -
1. Very good understanding the basic computer science along the core language skills helps write very efficient software applications which can utilize the available hardware effectively, with minimum bugs.
2. Skills alleviates tension, because only skills level help us automate the mundane tasks.
3. We are on the better side of the crowd and thus remain positive minded.
4. Skills help us estimate the time for software development effectively.
5. When you have good understanding of hardware and software then you get a deep penetration into software development which otherwise is not possible.

**Question : Would it help if I memorize all the questions ?**

No, it will not. But if you memorize the most common **Patterns of software development,** that will definitely help to ease day to day work. A single pattern resolves n number of problems emerging from that pattern, and we should always look forward finding the patterns instead of solution to a particular problem.

**Question : Why do interviewers ask rocket science questions in interviews even if the new role does not require any such skills ?**
Hiring in IT industry is not regulated by any means, it is solely up to the interviewer to choose the topic for discussion by the interviewee. In today's intellectual world, people like intellectual war, and interview is a good place for that. I do not find any harm by such interview process unless interviewer hides the real picture of work that one needs to perform after joining the new role. For sure there is one plus point to such interview process that it will definitely raise our skill set.

**Question : Why people take so many offers at the time of Job change, doesn't it add to chaos to the system ?**
The main reason for doing so, is the disparity between work and salary across the companies, and this will continue to happen till a saturation point come in India. People feel insecure at financial level and try their best to grab the most paying Job opportunity, and that's fair. On the other hand, companies tend to maximize their profit by limiting the salary offer as per individual's previous company's salary. So it is a game, where both the employer and the employee are fighting to maximize their own profit. Ultimately, the Demand and Supply equation balances the fight between employer and the employee.

## Java Developer Road Map

A deep understanding of basic software components is must for a developer who wants to write Scalable Enterprise/Internet Applications from scratch using Java. Below is the matrix showing a hierarchy of skills which are required for a senior software developer who are aiming a long term carrier in Software Development.

| Priority | Category | Topics |
|---|---|---|
| 6 | Web Tier | Servlet basics, HTTP Basics, REST, MVC Design Pattern, Struts & Spring |
| 5 | Database | SQL, database Indexes, JPA/Hibernate QL, Table to Entity Mapping, Inheritance in JPA (Table per class, joined, single table), Transaction Isolation Level, embeddable, Mapped Super Classes, All relationships - OneToOne, OneToMany, ManyToMany. |
| 4 | Core Java | Inheritance, Generics, Garbage Collector, good hold over Concurrency (synchronizer, non-blocking algorithms using atomic package, executor service, Java Memory Model, Immutability, volatile, Fork/Join), good understanding of internals of Java Collections Framework (HashMap, LinkedList, ConcurrentHashMap, PriorityQueue, TreeMap) |
| 3 | Algorithms | Tree Traversal, Tree Insertion, Tree balancing using left & right rotation, Sorting (quick, merge, external, selection), Binary Search, BFS, DFS, Topological Sorting using Graph, Dijkstra's algorithm |
| 2 | Data Structures | ArrayList, LinkedList, Stack, Queue, Tree (Binary Search Tree, Red Black Tree, Binary Heap), Hashtable, Prefix Tree (Trie), Suffix Tree, Graphs. |
| 1 | Concepts | Object Oriented Programming & Design<br>Test Driven Development (JUnit, Mockito, etc)<br>Familiarity with Version Control System, Continuous Integration<br>Design Patterns (Abstract Factory, Builder, Singleton, Observer, Template, Strategy, Visitor, Decorator, Facade, Flyweight, etc)<br>Memory (Heap, Stack, Memory Generations in Java)<br>Logarithm, Big-O notation, Bitwise Manipulation & Recursion<br>Number System (2's complement, binary, hexadecimal, etc) |

**Skills Matrix for a Java Developer**

## Good Software Practices

1. Follow good software development practices like Test Driven Development. A very good test coverage (End To End and Unit Tests) keeps a developer away from last minute stress at production deployment time.

2. Automate all the mundane tasks related to development/deployment.

3. Take a software change request only if it is really required.

4. Keep refactoring your code base time to time, don't leave any duplicate code inside code base. Follow DRY (don't repeat yourself) strictly. Every object must have a single authoritative representation in the system.

5. Add an automated test case for every new bug found.

6. Document interfaces and reasons instead of implementation.

7. Use profiler to identify bottlenecks.

8. Use pair programming when bringing someone new up to speed and when tackling particularly tricky problems

9. Use tools to find the duplicates and refactor to reuse the existing code.

10. Work in small steps with frequent feedback and correction to avoid the last minute surprises.

11. Continuous Integration environment is must for rapid bug free development.

Chapter 1

# Concepts

## Question : What are the Pros and Cons of Java (as of JDK 1.7)?

### SOLUTION

#### Java Pros

It's free of cost, download it and start creating your applications
Open source with quite large community base
Lots of available third party libraries & frameworks for fast development cycles
Platform independent, works on most modern platform (Unix, Windows, Mac, 32/64 bit Hardware)
Supports Object Oriented Programming, easy to model real life scenarios into object model
In built support for multi-threading, It can very efficiently utilize maximum of given hardware (Threads, Fork/Join, non-blocking algorithm using CAS, etc)
Very good support for Internationalization
Memory management is automatic by use of garbage collector
Pure Java Byte code running on 32 bit jvm works perfectly fine on 64 bit platform
Its Improving year by year

#### Java Cons

Is not a good fit for desktop applications because of heavy memory footprint and huge vm startup time
Normal Java Is not good for real time systems because of "stop the world garbage collector pauses".
Memory foot print is large compared to C++
Does not provide very good support for functional programming as of JDK 1.7

## Notes

#### Difference between 32 bit and 64 bit JVM

The Java language specifications are same for both the platform i.e. int will remain to be 4 bytes signed two's complement, char will remain single 16-bit Unicode, long will remain 64-bit signed two's complement, and so on. Hence any Pure Java code will not see any difference provided external native calls are not used. All that changes is  the amount of addressable memory (good) and the amount of memory per Object (not that good). The size of the reference variables doubles from 32 bit to 64 bit, thus all the reference variable will take double the size when running on 64 bit JVM.

Theoretically, there are no class file differences between code compiled with the 32 bit and 64 bit versions of the same revision of Java.

For 32 bit JVM, the maximum memory is limited to 4GB, the memory limit for 64 bit JVM is very high.

Please note that 64 bit JVM requires more memory compared to 32 JVM for the same application because now each reference starts consuming 64 bit instead of 32 bit.

## Question : What are four principles of OOP ?

### SOLUTION

There are 4 major principles that make an language Object Oriented.  These are Encapsulation, Data Abstraction, Polymorphism and Inheritance.

### Encapsulation

Encapsulation is the mechanism of hiding of data implementation by restricting access to public methods.

### Abstraction

Abstract means a concept or an Idea which is not associated with any particular instance. Using abstract class/ interface we express the intent of the class rather than the actual implementation. In a way, one class should not know the inner details of another in order to use it, just knowing the interfaces should be good enough.

### Inheritance

Inheritances expresses "is a" relationship between two objects. Using proper inheritance, In derived classes we can reuse the code of existing super classes.

### Polymorphism

It means one name many forms. It is further of two types - static and dynamic. Static polymorphism is achieved using method overloading and dynamic polymorphism using method overriding.

### What is aggregation, how is it different from composition[1] ?

Both of these are special type of association and differ only in weight of relationship.
Composition is stronger form of "is part of" relationship compared to aggregation "has a".
In composition, the member object can not exist outside the enclosing class while same is not true for Aggregation.

1 http://design-antony.blogspot.in/2007/07/aggregation-vs-composition.html

## Question : What is Logarithm ? Why is it relevant in Software Development ?
## SOLUTION

A logarithm[1] tells what exponent (power) is needed to make a certain number. In a way it is opposite of exponentiation. For example,

$\log_2(8) = 3$ and $2^3 = 8$
$\log_{10}(1000) = 3$ and $10^3=1000$

| Number | Logarithm (base 10) |
|---|---|
| 1 | 0 |
| 10 | 1 |
| 100 | 2 |
| 1000 | 3 |
| 10000 | 4 |
| 100000 | 5 |

### Why do we need Logarithm ?

Logarithm converts big number values into smaller, human readable format.

- It is easy to handle small numbers compared to very large numbers, when our motive is just to compare them. Logarithm converts big values to small numbers.
- It makes multiplication and division of large numbers easy because adding logarithms is the same as multiplying and subtracting logarithms is same as dividing.

In pre modern era, when calculators were not there, logarithm tables were used for division and multiplication of large astronomical numbers.

Sum of logs = log of product
Subtraction of logs = log of division

## Notes

Logarithm was discovered in India in ancient times around 2 BC and was used to express astronomical units. It is called as **Laghuganak** in hindi.

Logarithmic spirals are common in nature. Examples include the shell of a nautilus or the arrangement of seeds on a sunflower.

The Richter scale measures earthquake intensity on a base 10 logarithmic scale.

In astronomy, the apparent magnitude measures the brightness of stars logarithmically, since the eye also responds logarithmically to brightness.

1 http://simple.wikipedia.org/wiki/Logarithm

## Question : What do you understand by Big O Notation, Why is it important in software development ?

## SOLUTION

Big O Notation[1] is a mechanism used to measure the relative efficiencies of Algorithms in terms of space and time. It makes us understand how execution time & memory requirements of an algorithm grow as a function of increasing input size.

In this notation, O stands for the order of magnitude.
Following are the examples of Big O, in increasing order of their magnitude.

| Big O Notation | Name | Example |
|---|---|---|
| O (1) | Constant-time | Searching from a HashMap, check a number for even/odd |
| O (log n) | Logarithmic | Find an item inside sorted array using Binary Search |
| O (n) | Liner | Printing all elements from an array |
| O (n log n) | Loglinear | Sorting using Merge Sort |
| O ($n^2$) | Quadratic | Bubble Sorting Algorithm |
| O ($2^n$) | Exponential | Shortest Path Problem Djigstraw Algorithm |
| O (n!) | Factorial | Solving Travelling Sales Man Problem |

### Importance of Big O

We should always keep time efficiencies in mind while designing an algorithm using existing data structures, otherwise there could be sever performance penalties for using wrong data structure for a given scenario.

## Notes

### Time efficiency in Big O notation for few Java Collections

**ArrayList** (ignoring the time taken by array resize operation)
O(1) for add, size and get
O(n) for toString() method

**PriorityQueue**
O(1) for peek, element and size
O(log n) for offer, poll, remove() and add
O(n) for remove(Object) & contains(Object)

**HashMap (with no collisions)**
O(1) for get operation
O(1) for put operation

**LinkedList**
O(1) for removal
O(1) for add & poll method
O(n) for toString() method

1 http://en.wikipedia.org/wiki/Big_O_notation

## Question: List down sorting algorithms by their time & memory complexity in Big O notation ? When do we call a sorting algorithm stable ?

**SOLUTION**

Sorting is an algorithmic technique to put all the collection elements in certain order.[1]

| Algorithm | Average Time Complexity | Worst Time Complexity | Memory Complexity |
|---|---|---|---|
| Quicksort | n log n | $n^2$ | log n |
| Binary Tree Sort | n log n | n log n | n |
| Merge Sort | n log n | n log n | n |
| Selection Sort | $n^2$ | $n^2$ | 1 |
| Bubble Sort | $n^2$ | $n^2$ | 1 |
| Heap Sort | n log n | n log n | 1 |

### When is a sorting algorithm Stable ?

An algorithms is said to be stable if it maintains the relative order of records where keys are equal. For example, suppose the following key-value pair need to be sorted based on key in ascending order

```
INPUT       -->     [(2,3)  (1,2)   (1,3)  (3,1)]
```

Now there are two solution possible for the first two elements

```
OUTPUT1     -->     [(1,2)  (1,3)  (2,3)  (3,1)]  --> stable sort because order is maintained
OUTPUT2     -->     [(1,3)  (1,2)  (2,3)  (3,1)]  --> unstable sort because order changed from the original
```

Examples of Stable Sort algorithms are : Binary Tree Sort, Bubble Sort, Merge Sort, Insertion Sort, etc
Unstable Sorting Algorithms : Heap Sort, Selection Sort, Quick Sort

*TIP - Which Sorting Algorithm is used by Java's Collections.sort(List<E>) in Java 1.7*
*Collections.sort() uses Iterative merge sort that requires far fewer than n lg(n) comparisons when the input array is partially sorted and is guaranteed to be Stable.*

1 http://en.wikipedia.org/wiki/Sorting_algorithm

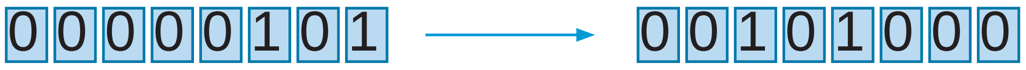
## Question : What is left shift <<, right shift >> and Unsigned right shift operator in Java? How are these useful ?

**SOLUTION**

All Integer in Java are of signed type (negative numbers are represented in 2's complementary notation), hence Java provides both signed and unsigned bit shift operators to support signed and unsigned shift of bits.

### Left Shift Operator << (Signed)

It shifts the underlying bits of an integer to left by the given distance filling the right most bits with zero always.
X = a << b means the same as X = a*2^b
*a is given Number and b is the shift amount.*
Here is an example of 8 bit representation of number 5. and when we left shift it's bit by 3 then the right most 3 bits are filled by zero.
And the number becomes
$5*2^3$ = 40.



The same thing happens for negative numbers which are represented in 2's complementary notation. for example -5 becomes -40 as follow
11111011 becomes 11011000

### Right Shift Operator >> (Signed)

Shifts the bits to left by specified amount maintaining the sign of underlying integer i.e. It fills the left most bits with 0 if the number is positive otherwise with bit 1.
X = a >> b means same as arithmetic operation X = a / $(2^b)$

### Unsigned right shift Operator >>> (does not respect sign of Number)

Unsigned right shift operator >>> is effectively same as >> except that it is unsigned, it fills the left most positions with bit 0 always. (Irrespective the sign of the underlying number)
For example,
So 10000000 >>> 3 becomes 10000 in binary
256 >> 3 becomes 256 / 2^3 = 16.

## Notes

- Eight-bit type ***byte*** is promoted to ***int*** in shift-expressions. To mitigate such effects we can use bit masking to get the result as byte for example, (b & 0xFF) >>> 2. Casting can also help achieving the same.

- **Why there is no need of unsigned left shift ?**
  Because there is no need to have that. Sign bit is the right most bit of an integer and shifting bits to right only require the decision of sign. Logical and arithmetic left-shift operations are identical so << signed solves the purpose of unsigned left shift as well.

- **Uses of bitwise operators:** bitwise operators are used for few very efficient mathematical calculations in Big O(1). Bloom Filter, fast mathematical calculations, hashing functions of HashMap are some of applications.

## Question : What is 2's complement notation system for Binary Numbers ?

### SOLUTION

It is a number format for storing negative numbers in Binary[1]. This system is the most common method of representing signed numbers on computers. An N-bit two's-complement numeral system can represent every integer in the range −(2N−1) to +(2N−1 − 1)

### Why 2's Complement ?

The two's-complement system has the advantage that the fundamental arithmetic operations of addition, subtraction, and multiplication are identical to those for unsigned binary numbers (as long as the inputs are represented in the same number of bits and any overflow beyond those bits is discarded from the result). This property makes the system both simpler to implement and capable of easily handling higher precision arithmetic. Also, zero has only a single representation, obviating the subtleties associated with negative zero, which exists in ones'-complement systems.

### Calculating 2's complement

Positive numbers are represented as the ordinary binary representation in 2's complementary notation. The most significant bit (leftmost bit) is always 0 for positive number, otherwise number is negative.

To get 2's complement of a negative numbers, the bits are inverted (using bitwise NOT operator) and then value of 1 is added to get the final result.

For example, **Let's convert 5 to -5 using 2's complement notation**

Using 8 bit system, number 5 is represented as
00000101
Now invert all the bits (1's complement)
11111010
Finally add 1 to get the result
11111011

So this is binary representation of -5 in 2's complement notation.

To convert it back to a Positive Number, calculate 2's complement of a negative number

For example, lets convert -5 to 5
11111011 represents -5

Invert all the bits
00000100

Then add 1 to get the result
00000101

That is +5.

1 http://en.wikipedia.org/wiki/Two's_complement

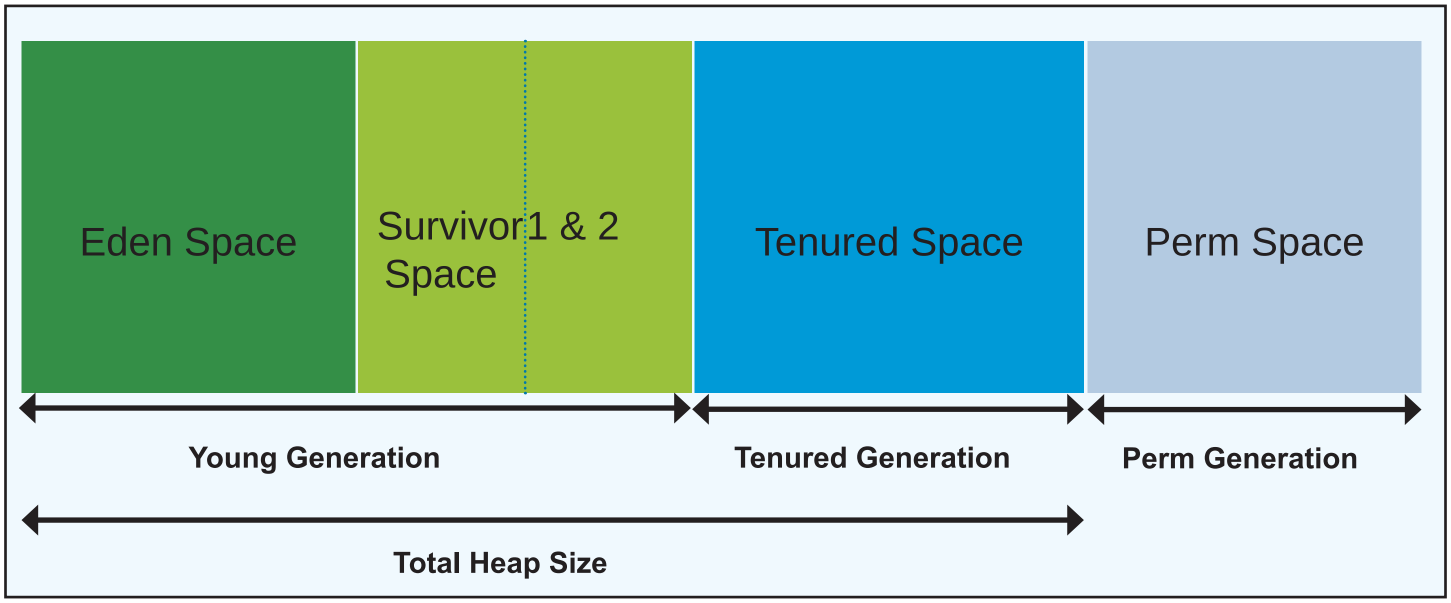
## Question : How Heap space is divided in Java. How does Garbage Collector cleans up the un-used Objects ? Why shouldn't we use System.gc() command in production code ?

## SOLUTION

Memory taken up by the JVM is divided into Stack, Heap and Non Heap memory areas. Stacks are taken up by individual threads for running the method code while heap is used to hold all class instances and arrays created using new operation. Non-heap memory includes a method area shared among all threads and is logically part of the heap but, depending upon the implementation, a Java VM may not GC or compact it.

### Java HotSpot VM defines two generations[1]

**The Young generation** - This further consists of one Eden Space and ***two*** survivor spaces. The VM initially assigns all objects to Eden space, and most objects die there. When VM performs a minor GC, it moves any remaining objects from the Eden space to one of the survivor spaces.



**Tenured/Old Generation** - VM moves objects that live long enough in the survivor spaces to the "tenured" space in the old generation. When the tenured generation fills up, there is a full GC that is often much slower because it involves all live objects.

**Permanent Generation** - The permanent generation holds all the reflective data of the virtual machine itself, such as class and method objects.

*Please note that the contents of this article are likely to change with the version change.*

1 http://docs.oracle.com/javase/7/docs/technotes/guides/management/jconsole.html
http://www.oracle.com/technetwork/java/javase/gc-tuning-6-140523.html

*continued on 19*

Chapter 3

# Concurrency

## Question : What is Concurrency, how is it implemented in Java Programs ?

### SOLUTION

Concurrency is the property of an software program to run several computations in parallel. Java provides us with the multiple mechanisms to create Threads so as to utilize the multiple processor cores of a given hardware in order to achieve high throughput.

Java provides various ready to use utilities for writing concurrent programs, which otherwise is difficult to implement.

Some of the Utility classes provided as of JDK 1.6 are[1]
Executors, Queues, TimeUnit, Synchronizers (semaphore, latch, barrier, exchanger), Concurrent collections (ConcurrentHashMap, CopyOnWriteArrayList, ConcurrentSkipListMap), Atomic package for non-blocking algorithms, locks (ReentrantLock), and well defined Java Memory Model for memory consistency in concurrent environment.

### What is Thread-Safety and how to achieve it ?

A Class is thread safe when it behaves correctly when accessed & modified from multiple threads in parallel without any changes in the code calling it.

There are certain ways to make our class thread-safe, as follow
1. Use synchronization mechanism (intrinsic or explicit) on the accessor methods
2. Make the class Immutable
3. Don't expose the shared state across threads (for e.g. Keep objects local to thread using ThreadLocal)

### What is Synchronization ?

Synchronization avoids thread interference and memory consistency errors by providing serialized access to the shared state.

Synchronization has two major aspects
1. It makes sure that the compound actions executes atomically by providing mutually exclusive access to the shared state across the threads.
2. It ensures the memory consistency by making the changes visible to all the threads upon method exit.

Lets examine the following program for its correctness in concurrent environment, It maintains the integer counter.

**Counter.java**

```
@NotThreadSafe
class Counter {
        private int c = 0;
        public void increment() {
                c++;
        }
        public int value() {
        return c;
        }
}
```

1 http://www.ibm.com/developerworks/java/library/j-5things4/index.html

continued on 74

This program will work absolutely fine in single threaded environment but will not behave correctly in multi-threaded environment, because

1. increment() method will not be executed atomically so data race may corrupt the counter value.
2. value() method may not return the latest value of counter because of caching in processor's registers.

So lets make this program thread-safe.

```
Counter.java
@ThreadSafe
class Counter {
        private int c = 0;
        public synchronized void increment() {    //this will make the operation execution atomic across the threads
                    c++;
        }
        public synchronized int value() {           //this will make sure the changes are visible to the calling thread
        return c;
        }
}
```

**Please Make Sure**

That you make the getter method as synchronized until the getter returns an immutable object. Otherwise the memory effects may not be consistent and the calling thread may see a stale value of the variable.

**Weaker form of synchronization using volatile**

Volatile variable can not make the method execution atomic, but it can make sure that the updates to the variable are propagated predictably to the other threads. volatile variables basically, are not cached into the processor registers, rather they are always fetched and written to the main memory on heap.

## Question : There are two Threads A and B operating on a shared resource R, A needs to inform B that some important changes has happened in R. What technique would you use in Java to achieve this ?

## SOLUTION

Object R's method wait(), notify() & notifyAll(), can be used for inter-thread communication. This will allow all threads which hold lock over R, to communicate among them selves. You can explore a typical Producer-Consumer problem to see how it works.

## Question : What are the different states of a Thread ? What does those states tells us ?

## SOLUTION

A thread in JVM can have 6 different states as defined in Thread.State enum. At any given time, thread must be in any of these states.

**NEW**
This state is for a thread which has not yet started.

**RUNNABLE**
This state is for the currently running thread which is executing in java virtual machine, but it may be waiting for the other resources from operating system such as processor.

**BLOCKED**
Thread state for a thread blocked waiting for a monitor lock. A thread in this state can be waiting for a monitor lock to enter a synchronized block/method or reenter a synchronized method after calling Object.wait.

**WAITING**
A thread is waiting due to calling on one of the method -
Object.wait with no timeout
Thread.join with no timeout
LockSupport.park

A Thread in this state is waiting for another thread to perform a particular action. For example, a thread that has called Object.wait() on an object is waiting for another thread to call Object.notify() or Object.notifyAll() on that object. A thread that has called Thread.join() is waiting for a specified thread to terminate.

**TIMED_WAITING**
Thread state for a waiting thread with a specified waiting time. A thread is in the timed waiting state due to calling one of the following methods with a specified positive waiting time -

Thread.sleep
Object.wait with timeout
Thread.join with timeout
LockSupport.parkNanos
LockSupport.parkUntil

**TERMINATED**
Thread state for a terminated thread. The thread has completed execution.

### References

This content has been taken directly from the Java 7 Docs - Thread.State enum.

## Question: What do you understand by Java Memory Model ? What is double-checked locking? What is different about final variables in new JMM ?

## SOLUTION

**Interviewer's Intent** - *Interviewer wants to understand if you can write concurrent code.*

Java Memory Model[1] defines the legal interaction of threads with the memory in a real computer system. In a way, it describes what behaviors are legal in multi-threaded code. It determines when a Thread can reliably **see** writes to variables made by other threads. It defines semantics for volatile, final & synchronized, that makes guarantee of visibility of memory operations across the Threads.

Let's first discuss about Memory Barrier which are the base for our further discussions. There are two type of memory barrier instructions in JMM - read barriers & write barrier.

***A read barrier*** invalidates the local memory (cache, registers, etc) and then reads the contents from the main memory, so that changes made by other threads becomes visible to the current Thread.
***A write barrier*** flushes out the contents of the processor's local memory to the main memory, so that changes made by the current Thread becomes visible to the other threads.

### JMM semantics for synchronized

When a thread acquires monitor of an object, by entering into a synchronized block of code, it performs a read barrier (invalidates the local memory and reads from the heap instead). Similarly exiting from a synchronized block as part of releasing the associated monitor, it performs a write barrier (flushes changes to the main memory)
Thus modifications to a shared state using synchronized block by one Thread, is guaranteed to be visible to subsequent synchronized reads by other threads. This guarantee is provided by JMM in presence of synchronized code block.

### JMM semantics for Volatile fields

Read & write to volatile variables have same memory semantics as that of acquiring and releasing a monitor using synchronized code block. So the visibility of volatile field is guaranteed by the JMM. Moreover afterwards Java 1.5, volatile reads and writes are not reorderable with any other memory operations (volatile and non-volatile both). Thus when Thread A writes to a volatile variable V, and afterwards Thread B reads from variable V, any variable values that were visible to A at the time V was written are guaranteed now to be visible to B.

Let's try to understand the same using the following code

```
Data data = null;
volatile boolean flag = false;

Thread A
-------------
data = new Data();
flag = true;              <-- writing to volatile will flush data as well as flag to main memory

Thread B
-------------
if(flag==true){           <-- reading from volatile will perform read barrier for flag as well data.
use data;                 <--- data is guaranteed to visible even though it is not declared volatile because of the JMM
                               semantics of volatile flag.
}
```

1 http://www.ibm.com/developerworks/library/j-jtp03304/

*continued on 77*

Chapter 4

# Algorithms & Data Structures

## Question : Given a collection of 1 million integers ranging from 1 to 9, how would you sort them in Big O(1) time ?

## SOLUTION

This is a typical Integer Sorting problem with a constraint that the number range to sort is very limited in spite 1 million total entries. Integer Sorting with limited range is achieved efficiently with Bucket Sorting.

*TIP What does Wiki Says about Sorting ?*
*Bucket sort, counting sort, radix sort, and van Emde Boas tree sorting all work best when the key size is small; for large enough keys, they become slower than comparison sorting algorithms…*
*Integer Sorting Techniques : http://en.wikipedia.org/wiki/Integer_sorting#Algorithms_for_few_items*
*Sorting Algorithms : http://en.wikipedia.org/wiki/Sorting_algorithm*

### Algorithm

Create a array of size 9 and at each index store the occurrence count of the respective integers. Doing this will achieve this sorting with time complexity of Big O(1) and even the memory requirements are minimized. In Order to print the output just traverse the above created array.

### Source Class

```
public class BucketSort {
    public int[] sort(int[] array, int min, int max) {
        int range = max - min +1;
        int[] result = new int[range];
        for (int i : array) {
            result[i]++;
        }
        return result;
    }
}
```

### Test Class

```
public class BucketSortTest {
    @Test
    public void testBucketSortFor1To9() {
        int[] array = {2, 1, 5, 1, 2, 3, 4, 3, 5, 6, 7, 8, 5, 6, 7, 0};
        int[] sort = new BucketSort().sort(array, 0, 8);
        for (int i = 0; i < sort.length; i++) {
            for(int j=0;j<sort[i];j++){
                System.out.println(i);
            }
        }
    }
}
```

**Program output :** 0,1,1,2,2,3,3,4,5,5,5,6,6,7,7,8

## Notes

Bloom Filter[1] could help us achieve something similar.

1 http://en.wikipedia.org/wiki/Bloom%5Ffilter

## Question : Given 1 million trades objects, you need to write a method that searches if the specified trade is contained in the collection or not. Which collection would you choose for storing these 1 million trades and why ?

### SOLUTION

HashSet is a good choice for storing this collection because it will offer Big O(1) time complexity. In order to use HashSet we must override equals() and hashcode() method for the Trade Object. If that's not possible then we should created a Trade Wrapper class which overrides these methods.

```
public class Trade{
...
@Override
public boolean equals(Object o) {...}
@Override
public int hashCode() {...}
}
```

## Question : I have an Integer array where every number appears even number of time except one. Find that number.

### SOLUTION

**Approach**
This problem can be solved by utilizing bitwise operators in O(1) space and O(n) time complexity.
XOR all the number together and the final result would the odd number.

How does XOR works ?

**Here is the complete solution using XORing**

```
public class OddNumberProblem {
    private int[] array = {1,1,2,3,4,5,2,3,4};
    public int findSingleOdd(){
        int result =0;
        for (int i : array) {
          result=result^i;
        }
        return result;
    }

    public static void main(String[] args) {
        OddNumberProblem test = new OddNumberProblem();
        int singleOdd = test.findSingleOdd();
        System.out.println("singleOdd = " + singleOdd);
    }
}
```

Output:

```
singleOdd = 5
```

Chapter 5

# Object Oriented Design

## Question : How would you resolve task's inter dependency, just as in maven/ant.

Let's consider the following task dependencies.

| Task | Dependent On |
|---|---|
| 3 | 1,5 |
| 2 | 5,3 |
| 4 | 3 |
| 5 | 1 |

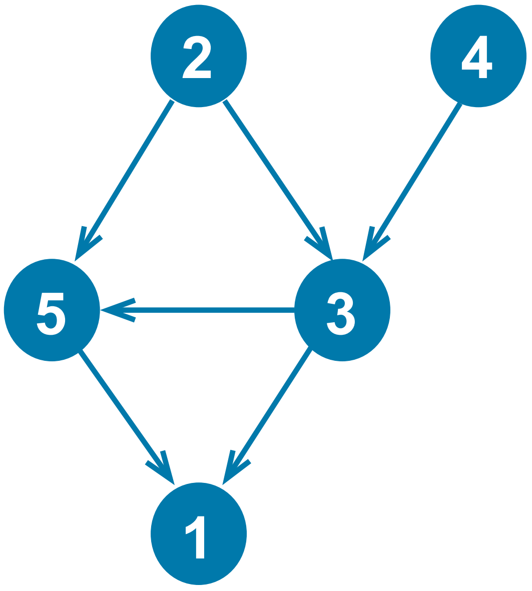
Here first row states that task 3 is dependent on task 1 and task 5, and so on. If the two consecutive tasks have no dependency, then they can be run in any order.

The output should look like - [1, 5, 3, 2 ,4] or [1, 5, 3, 4, 2]

## SOLUTION

### Approach 1

It is a typical Graph traversal problem, that can be solved using Topological Sorting Algorithm[1] in linear time O(|V| + |E|),
Where
V = number of nodes
E = number of edges



**Direct Acyclic Graph for the Task Dependencies**

1 http://en.wikipedia.org/wiki/Topological_sorting

## Question : Toolkit & Resources for a Java Developer.

### SOLUTION

**Essential Tool kit for Java Developer**
IntelliJ IDE
Java DB
Twitter Bootstrap CSS library
Jquery
Freemarker
Struts 2
Servlets
Tortoise SVN
Apache Web Server
Jetty Server
HTML 5
Firebug extension for mozilla Firefox
Cygwin for Unix simulation

**Books**
Design Patterns in Java - Head First
Concurrency In Practice by Brian Goetz
Effective Java 2nd Edition by Joshua Bloch
Algorithms 4th edition : http://algs4.cs.princeton.edu/home/
Cracking the coding interview at CareerCup

**Technology Forums**
http://www.geeksforgeeks.org/fundamentals-of-algorithms/
http://www.careercup.com
http://www.stackoverflow.com

**Great Tutorials Articles**
Few articles on Java 6 at IBM website
http://www.ibm.com/developerworks/views/java/libraryview.jsp?search_by=5+things+you+did
Concurrency In Practice by Brian Goetz - http://www.briangoetz.com/pubs.html
Java Articles : http://www.oracle.com/technetwork/articles/java/index.html
http://www.oracle.com/technetwork/articles/java/index.html
Java SE Tutorial : http://docs.oracle.com/javase/tutorial/index.html
Java 7 docs: http://docs.oracle.com/javase/7/docs/

**Video Tutorials**
http://www.youtube.com/user/nptelhrd

**Topics To Cover**
Bitwise operators, Binary trees, Number System