Java Knowledge Framework

February 25, 2019

1 Basics

1.1 Object-Orientated Programming (OOP)

1.1.1 OOP Basics

1. OOP vs Imperative Programming
2. Three Properties of OOP
3. Five Basic Rules of OOP

1.1.2 Platform Independence

1. How does Java achieve platform independence?
2. JVM language support

1.1.3 Pass-by-Value

1. Pass-by-value *v.* pass-by-reference
2. Why does Java only have pass-by-value?

1.1.4 Encapsulation, Inheritance and Polymorphism

1. What is polymorphism, method overloading and overriding?
2. Java inheritance and implementation
3. Constructor method and default constructor
4. Class variables, instance variables and local variables
5. Instance variables and method scope

2 Java Basics

2.1 Fundamental Data Types

1. The 8 fundamental data types
2. Integer-type value ranges
3. Floating-point numbers
4. Single-precision *v.* double-precision floating-point numbers
5. Why shouldn't one represent money with floating-point numbers?

2.2 Auto-boxing

1. Boxed types (Wrappers), Primitive types and Auto-boxing
2. `Integer` caching mechanism

2.3 Strings

1. `String` immutability
2. `String.substring` implementation and differences in JDK6 vs JDK7
3. Overloading of `+` operator and ways of performing `String` concatenation
4. `String.valueOf` *v.* `Integer.toString`
5. `switch` support for `String`
6. `String` Pool
7. Constant Pool
   - Run-Time Constant Pool
   - `Class` Constant Pool
   - `intern`

2.4 Java Keywords

1. The mechanism behind and usage of:
   - `transient`
   - `instanceof`
   - `final`
   - `static`
   - `volatile`
   - `synchronized`
   - `const`

2.5 Java Collections Framework

1. Usage of common `Collection` classes
2. `ArrayList` *v.* `LinkedList` *v.* `Vector`
3. `SynchronizedList` *v.* `Vector`
4. `HashMap` *v.* `HashTable` *v.* `ConcurrentHashMap`
5. `Set` *v.* `List`
6. How does `Set` guarantee *uniqueness* of elements?
7. Java 8 `streams` API usage
8. Apache collections processing tool usage
9. Different implementations of `HashMap` across different versions of JDK and cause of such difference
10. `Collection` vs `Collections`
11. Gotchas of the `List` obtained when using `Arrays.asList`
12. `Enumeration` *v.* `Iterator`
13. `fail-fast` *v.* `fail-safe`
14. `CopyOnWriteArrayList` and `ConcurrentSkipListMap`

2.6 Enumerations

1. Usage of `Enums`
2. `Enum` implementation
3. `Enum` and *Singleton*
4. `Enum` class
5. Comparing `Enums`
6. `switch` support for `Enums`
7. `Enum` serialization implementation
8. `Enum` thread safety issues

2.7 IO

1. `ByteStreams`, `CharacterStreams`, `InputStreams`, `OutputStreams`
2. Synchronous *v.* Asynchronous
3. Blocking *v.* Non-blocking
4. Linux IO models
5. `BIO` *v.* `NIO` *v.* `AIO`
6. Usage of `BIO`, `NIO`, `AIO`
7. `netty`

2.8 Reflection

1. *Reflection* and *Factory*
2. Usage of *Reflection*
3. `Class` class
4. `java.lang.reflect.*` package

2.9 Dynamic Proxy

1. Dynamic proxy *v.* static proxy
2. Dynamic proxy and Reflection
3. Dynamic proxy implementations
4. Aspect-Orientated Programming (AOP)

2.10 Serialization

1. Serialization and deserialization
2. Why serialize?
3. Serialization mechanism
4. Serialization and *Singleton*
5. `protobuf`
6. Why serialization isn't safe?

2.11 Annotations

1. Meta-Annotations
2. Custom Annotations
3. Common Annotations usage
4. Annotation with Reflection
5. `Spring` common Annotations

2.12 JMS

1. Java Message Service
2. `JMS` message delivery model

2.13 JMX

1. `java.lang.management.*`
2. `javax.management.*`

2.14 Generics

1. Generics and inheritance
2. Type erasure
3. Meaning of `K`, `T`, `V`, `E`, `?` and `object`
4. Usage of Generics
5. Constrained wildcard type *v.* unconstrained wildcard type
6. Upper-bounded wildcard `<? extends T>` *v.* Lower-bounded wildcard `<? super T>`
7. `List<Object>` *v.* `List`
8. `List<?>` *v.* `List<Object>`

2.15 Unit Testing

1. `JUnit`
2. `Mock`
3. `Mockito`
4. `h2`

2.16 Regular Expressions

1. `java.lang.util.regex.*`

2.17 Common Utilities Libraries

1. `commons.lang`
2. `commons.*`
3. `guava-libraries`
4. `netty`

2.18 API and SPI

1. `APIs`
2. `API` *v.* `SPI`
3. Defining `SPI`
4. `SPI` implementation

2.19 Exceptions

1. `Exception` type
2. Processing `Exceptions`
3. Custom `Exceptions`
4. `Error` and `Exception`
5. `Exception` chaining (propagation)
6. `try-with-resources`
7. Order of execution between `finally` and `return`

2.20 Time Processing

1. Timezones
2. Daylight Saving Time *v.* Standard Time
3. Timestamps
4. Java `Datetime` API
5. `GMT`
6. `CET`, `UTC`, `GMT`, `CST` definitions and relationships
7. `SimpleDateFormat` thread safety issues
8. Java 8 Datetime processing
9. Obtaining US time from `GMT+8` timezones

2.21 Encoding

1. Unicode
2. Why is `UTF-8` needed even when `Unicode` is present?
3. `GBK` *v.* `GB2312` *v.* `GB18030`
4. `UTF8` *v.* `UTF16` *v.* `UTF32`
5. `URL` encoding/decoding
6. Big Endian *v.* Little Endian
7. Solving garbled messages

2.22 Syntax Sugars

1. Java syntax sugar mechanics
2. Java de-sugaring
3. Syntax Sugars for:
   `switch` supporting `String` and `Enum`
   `Generics`
   Auto-boxing and unboxing
   `Varargs`
   `Enums`
   Nested classes
   Conditional compilation
   Assertions
   Literals
   `for-each`
   `try-with-resources`
   `Lambda` expressions

2.23 Reading Source Code

1. Java classes:
   `String`, `Integer`, `Long`, `Enum`
   `BigDecimal`
   `ThreadLocal`
   `ClassLoader` and `URLClassLoader`
   `ArrayList` and `LinkedList`
   `HashMap`, `LinkedHashMap`, `TreeMap` and `ConcurrentHashMap`
   `HashSet`, `LinkedHashSet` and `TreeSet`

2.24 Java Concurrency

2.24.1 Concurrency and Parallel Programming

1. Concurrency
2. Parallelism
3. Concurrency *v.* Parallelism

2.24.2 Threads *v.* Processes

1. Thread implementation
2. Thread state
3. Thread Priority
4. Thread scheduling
5. Thread creation
6. Guard thread
7. Thread *v.* Process

2.24.3 Thread Pool

1. Designing Thread Pool
2. `submit()` and `execute()`
3. Thread Pool mechanics
4. Why one cannot create thread pool via `Executors`

2.24.4 Thread Safety

1. Deadlocks
2. Diagnosing deadlocks
3. Thread safety and the memory model

2.24.5 Locks

1. CAS
2. Optimistic lock *v.* pessimistic lock
3. Database locks
4. Distributed locks
5. Biased lock
6. Light-weight lock
7. Heavy-weight lock
8. `monitor`
9. Lock optimization
10. Lock elimination
11. Lock coarsening
12. Spin lock
13. Reentrant lock
14. Blocking lock
15. Deadlock

2.24.6 Deadlocks

1. What is a deadlock?
2. Deadlock elimination

2.24.7 `synchronized`

1. Implementation of `synchronized`
2. `synchronzied` and locks
3. Implementing thread-safe singleton without `synchronized`
4. `synchronized` and atomicity, visibility and orderedness

2.24.8 `volatile`

1. `happens-before`
2. Memory Barrier
3. Compiler instruction reshuffling and CPU instruction reshuffling
4. `volatile` implementation
5. `volatile` and atomicity, visibility and orderedness
6. Why is `volatile` needed when `synchronized` exists?

2.24.9 `sleep` and `wait`

2.24.10 `wait` and `notify`

2.24.11 `notify` and `notifyAll`

2.24.12 ThreadLocal

2.24.13 Solving the Consumer-Producer problem

2.24.14 Java Concurrency Package

1. Java Concurrency and `java.util.concurrent`
   - `Thread`
   - `Runnable`
   - `Callable`
   - `ReentrantLock`
   - `ReentrantReadWriteLock`
   - `Atomic*`
   - `Semaphore`
   - `CountdownLatch`
   - `ConcurrentHashMap`
   - `Executors`

3 Underlying Fundamentals

3.1 Java Virtual Machine (JVM)

3.1.1 JVM Memory Structure

1. `class` file format

2. Run-time data sectors:
   - Heap memory
   - Stack memory
   - Method partition
   - Direct memory
   - Run-time constant pool
   - Heap memory *v.* Stack memory
3. Must Java objects be allocated on the Stack?

3.1.2 Java Memory Model

1. Computer memory model
2. Cache coherency
3. `MESI` protocol
4. Visibility
5. Atomicity
6. Orderedness
7. `happens-before`
8. Memory Barrier
9. `synchronized`
10. `volatile`
11. `final`
12. Lock

3.1.3 Garbage Collection

1. Garbage Collection Algorithms
   - Mark-and-sweep
   - Reference counting
   - Copy
   - Mark-and-compact
   - Generational GC
   - Incremental GC
2. Garbage Collection Parameters
   - Determining survivors
   - GC: `CMS`, `G1`, `ZGC`, `Epsilon`

3.1.4 JVM Parameters and Optimization

1. `-Xmx`, `-Xmn`, `-Xms`, `-Xss`, `-XX:SurvivorRatio`, `-XX:PermSize`, `-XX:MaxPermSize`, `-XX:MaxTenruingThreshold`

3.1.5 Java Object Model

1. `oop-klass`
2. Object head

3.1.6 HotSpot

1. JIT compiler
2. Compiler optimizations

3.1.7 JVM Performance Monitoring and Debugging Tools

1. JPS, JStack, JMap, JStat, JConsole, JInfo, JHat, javap, btrace, TProfiler, Arthas

3.2 Class Loading Mechanism

1. `ClassLoader`
2. Class loading process
3. Parent Delegation Model (and breaking it)
4. Modularization (JBoss modules, osgi, jigsaw)

3.3 Compiling and decompiling

1. Compilation (front-end, back-end)
2. Decompilation
3. JIT
4. JIT optimizations (escape analysis, stack allocation, scalar replacement, lock optimization)
5. Compilation tooling: `javac`
6. Decompilation tooling: `javap`, `jad`, `CRF`

4 Intermediate

4.1 Java Underlying Mechanics

1. Bytecode
2. `class` file format
3. CPU caching, L1, L2, L3 and pseudo-sharing
4. Tail recursion
5. Bit manipulations

4.2 Design Patterns and Principles

1. Class-level Principles:
   - Single-responsibility Principle
   - Open-Close Principle
   - Liskov-Substitutability Principle

Interface-Segregation Principle
Dependency-Inversion Principle
Composite-Reuse Principle

4.2.1 Design Patterns

1. Creational Patterns:
   Singleton
   Factory
   Abstract Factory
   Builder
   Prototype
2. Structural Patterns:
   Adapter
   Bridge
   Decorator
   Composite
   Facade
   Flyweight
   Proxy
3. Behavioral Patterns:
   Template Method
   Command
   Iterator
   Observer
   Mediator
   Memoir
   Interpreter
   State
   Strategy
   Chain of Responsibility
   Visitor

4.2.2 Common Variants of Design Patterns

1. Singleton Variants:
   Thread-unsafe singleton
   Thread-safe singleton
   Hungry singleton
   Hungry singleton variant
   Static inner class
   Enumeration
   Double validation lock

4.2.3 Implementing thread-safe singleton without `synchronized` and `lock`

4.2.4 Implementing AOP

4.2.5 Implementing IOC

4.2.6 `nio` and `reactor` design patterns

4.3 Networking

4.3.1 TCP, UDP, HTTP, HTTPS Protocols

1. Triple Handshake + Quadruple Shutdown
2. Data flow control
3. Congestion management
4. OSI seven-layered model
5. TCP packet construction and deconstruction

4.3.2 HTTP/1.0 *v.* HTTP/1.1 *v.* HTTP/2.0

1. HTTP `get` *v.* `post`
2. Status codes

4.3.3 HTTP/3.0

4.3.4 Java RMI, Socket, HttpClient

4.3.5 Cookies and Sessions

1. Implementing sessions without cookies

4.3.6 Implementing simple static file HTTP server

4.3.7 Understand `nginx` and `apache` servers and build a corresponding server

4.3.8 Implement FTP, SMTP protocols in Java

4.3.9 Inter-Process Communication (IPC)

4.3.10 CDN

4.3.11 DNS

1. DNS
2. DNS Record type: A, CNAME, AAAA
3. DNS pollution

4. DNS hijacking
5. Public DNS: 114 DNS, Google DNS, OpenDNS

4.3.12 Reverse Proxy

1. Forward proxy *v.* reverse proxy
2. Reverse proxy server

4.4 Frameworks

4.4.1 Servlet

1. Life cycle
2. Thread safety
3. `filter` and `listener`
4. `web.xml` configurations

4.4.2 Hibernate

1. OR Mapping
2. Hibernate lazy loading
3. Hibernate caching
4. Hibernate *v.* IBatis *v.* MyBatis

4.4.3 Spring

1. `Bean` initialization
2. AOP mechanics
3. Spring IOC
4. Spring dependency injection methods (4)

4.4.4 Spring MVC

1. MVC
2. Spring MVC *v.* Struts MVC

4.4.5 Spring Boot

1. Spring Boot 2.0
2. Startup dependency
3. Automatic configuration
4. Spring Boot starter mechanism and implementation

4.4.6 Spring Security

4.4.7 Spring Cloud

1. Service discovery and registration:
   Eureka
   Zookeeper
   Consul
2. Load balancing:
   Feign
   Spring Cloud LoadBalance
3. Service Configuration:
   Spring Cloud Config
4. Service Throttling and Breaker
   Hystrix
5. Service Chain Tracer:
   Dapper
6. Service Gateway, Safety and Messages

4.5 Application Server

4.5.1 JBoss

4.5.2 Tomcat

4.5.3 Jetty

4.5.4 WebLogic

4.6 Tooling

4.6.1 git and svn

4.6.2 maven and gradle

4.6.3 IntelliJ IDEA

5 Advanced

5.1 Java

5.1.1 Java 8

1. Lambdas
2. Stream API
3. Time API

5.1.2 Java 9

1. Jigsaw, JShell, Reactive Streams

5.1.3 Java 10

1. Local type inference
2. G1 parallel full GC
3. ThreadLocal handshake mechanism

5.1.4 Java 11

1. ZGC
2. Epsilon
3. Enhanced `var`

5.1.5 Spring 5

1. Reactive programming

5.1.6 Spring Boot 2.0

5.1.7 HTTP/2

5.1.8 HTTP/3

5.2 Performance Optimization

1. Singleton
2. Future mode
3. Thread pool
4. Select Ready
5. Reduce context switch
6. Reduce lock granularity
7. Data compression
8. Result caching

5.3 Debugging

5.3.1 Obtaining Dump

1. Thread dump
2. Memory dump
3. GC report

5.3.2 Dump Analysis

1. Deadlock analysis
2. Memory leak analysis

5.3.3 Dump Tooling

1. Dump tooling:
   JStack
   JStat
   JMap
   JHat
   Arthas

5.3.4 Writing `OutOfMemory` and `StackOverflow` Programs

1. `HeapOutOfMemory`
2. `YoungOutOfMemory`
3. `MethodAreaOutOfMemory`
4. `ConstantPoolOutOfMemory`
5. `DirectMemoryOutOfMemory`
6. `StackOutOfMemory`
7. `StackOverflow`

5.3.5 Arthas

1. JVM-related
2. Class/ClassLoader-related
3. Monitor, watch, trace
4. Options,
5. Pipeline
6. Background async tasks

5.3.6 Common Problems

1. Memory leaks
2. Thread deadlock
3. ClassLoad conflict

5.4 Compilation

5.4.1 Compilation and Decompilation

5.4.2 Java Decompilation Tooling

1. Tools:
   javap, jad, CRF

5.4.3 JIT Compiler

5.4.4 Compilation Analysis

1. Lexical analysis
2. Syntactical analysis (parsing)
   LL, recursive descent, LR
3. Semantical analysis
4. Run-time environment
5. Intermediate code generation
6. Code generation
7. Optimization

5.5 Operating System

5.5.1 Linux Commands

5.5.2 Inter-Process Communication (IPC)

5.5.3 Process Synchronization

1. Producer-consumer problem
2. Dinning-philosopher problem
3. Reader-writer problem

5.5.4 Buffer Overflow

5.5.5 Segmentation and Paging

5.5.6 Virtual Memory and Main Memory

5.5.7 Virtual Memory Management

5.5.8 Page-Switch Algorithms

5.6 Database

5.6.1 MySQL Executing Engine

5.6.2 MySQL Execution Plan

1. Checking plan
2. Optimization

5.6.3 Indexing

1. Hash index
2. B-tree index (B+ tree, Sum B tree, R tree)
3. Normal index
4. Unique index
5. Covering index
6. Leftmost prefix principle
7. Index condition pushdown (ICP)

5.6.4 SQL Optimization

5.6.5 Database Task and Quarantine Level

1. Quarantine levels
2. Can tasks mimic locks?

5.6.6 Database Locks

1. Row lock
2. Table lock
3. Database lock

5.6.7 Connections

1. Internal connection
2. Left connection
3. Right connect

5.6.8 Database Main and Backup

5.6.9 binlog

5.6.10 redolog

5.6.11 In-Memory Database

1. h2

5.6.12 Database Table Partitioning

5.6.13 Command Query Separation

5.6.14 NoSQL

1. Redis
2. Memcached

5.6.15 Database and Locks

1. Database lock and NoSQL to implement distributive lock

5.6.16 Performance Optimization

5.6.17 Database Connection Pool

5.7 Data Structures and Algorithms

5.7.1 Simple Data Structures

1. Simple data structures:
   Stack
   Queue
   LinkedList
   Array
   HashTable
2. Stack *v.* Queue
3. Stack implementation (types of storage)

5.7.2 Trees

1. Binary tree
2. Dictionary tree
3. Balanced tree

4. Sort tree
5. B tree
6. B+ tree
7. R tree
8. Multipath tree
9. Red-black tree

5.7.3 Heap

1. Max heap
2. Min heap

5.7.4 Graph

1. Directed graph
2. Undirected graph
3. Topology

5.8 Sorting Algorithms

1. Stable sorting algorithms:
   - Bubble sort
   - Insertion sort
   - Cocktail sort
   - Bucket sort
   - Counting sort
   - Merge sort
   - In-place merge sort
   - BST sort
   - Pigeonhole sort
   - Radix sort
   - Dwarf sort
   - Library sort
   - Block sort
2. Unstable sorting algorithms:
   - Selection sort
   - Shellsort
   - Clover sort
   - Comb sort
   - Heapsort
   - Smooth sort
   - Quick sort
   - Introsort
   - Patience sort

5.8.1 Implementing Queue with Two Stacks

5.8.2 Implementing Stack with Two Queues

5.8.3 Depth-First Search and Breadth-First Search

5.8.4 Permutations

5.8.5 Greedy Algorithms

5.8.6 KMP Algorithm

5.8.7 Hashing Functions

5.8.8 Big Data Processing

1. Divide and conquer
2. Hash projection
3. Heap sort
4. Double bucket division
5. Bloom filter
6. bitmap
7. Database index
8. MapReduce

5.9 Big Data

5.9.1 `Zookeeper`

5.9.2 `Solr`, `Lucene`, `ElasticSearch`

5.9.3 `Storm`, Stream Calculation, `Spark`, `S4`

5.9.4 `Hadoop`, Offline Calculation

1. `HDFS`
2. `MapReduce`

5.9.5 Distributed Log Collection `flume`, `kafka`, `logstash`

5.9.6 Data mining `mahout`

5.10 Network Safety

5.10.1 XSS

1. XSS defense

5.10.2 CSRF

5.10.3 Injection Attacks

1. SQL injection
2. XML injection
3. CRLF injection

5.10.4 File Upload Vulnerability

5.10.5 Encryption and Decryption

1. Symmetric-key encryption
2. Asymmetric-key encryption
3. Hashing
4. Salted hashing
5. MD5
6. SHA-1
7. DES
8. AES
9. RSA
10. DSA
11. Rainbow table

5.10.6 DDoS

1. DoS attack
2. DDoS attack
3. `memcached` and DDoS
4. Reflectional DDoS

5.10.7 SSL, TLS, HTTPS

5.10.8 OpenSSL Certificate

6 Software Architecture

6.1 Distributed Systems

Data integrity, Service Governance, Service Downgrade.

6.1.1 Distributed Tasks

1. 2PC
2. 3PC
3. CAP
4. BASE
5. Reliable Message Eventual Integrity
6. Max-Effort Notice
7. TCC

6.1.2 `Dubbo`

1. Service registration
2. Service discovery
3. Service governance

6.1.3 Distributed Database

1. Implementing Distributed Database
2. When is distributed database needed?
3. `mycat`
4. `otter`
5. `HBase`

6.1.4 Distributed File System

1. `mfs`
2. `fastdfs`

6.1.5 Distributed Caching

1. Cache integrity
2. Cache accuracy
3. Cache redundancy

6.1.6 Throttling and Downgrading

1. `Hystrix`
2. `Sentinal`

6.1.7 Algorithms

1. Consensus algorithm
2. Raft protocol
3. Paxos algorithm
4. Raft algorithm
5. Byzantine problem and algorithm, 2PC, 3PC

6.2 Microservices

SOA, Conway's Law.

6.2.1 SeviceMesh

1. sidecar

6.2.2 Docker and Kubernets

6.2.3 Spring Boot

6.2.4 Spring Cloud

6.3 High Concurrency

6.3.1 Database Table Partition

6.3.2 CDN

6.3.3 Message Queues

1. `ActiveMQ`

6.4 Monitoring

6.4.1 Targets

1. CPU
2. Memory
3. Disk I/O
4. Network I/O

6.4.2 Methods

1. Process monitoring
2. Semantics monitoring
3. Machine resource monitoring
4. Data fluctuations

6.4.3 Data Collection

1. Logging
2. Implants

6.4.4 `Dapper`

6.5 Load Balancing

6.5.1 `tomcat` Load Balancing

6.5.2 `Nginx` Load Balancing

6.5.3 Four-tier Load Balancing

6.5.4 Seven-tier Load Balancing

6.6 DNS

6.6.1 DNS Principles

6.6.2 DNS Design

6.7 CDN

6.7.1 Data Integrity

7 Extensions

7.1 Cloud Computing

1. IaaS
2. SaaS
3. PaaS
4. Virtualization
5. openstack
6. Serverless

7.2 Search Engine

1. Solr
2. Lucene
3. Nutch
4. ElasticSearch

7.3 Permissions Management

1. Shiro

7.4 Blockchains

7.4.1 Hashing

7.4.2 Merkle Tree

7.4.3 Public Key Cryptography

7.4.4 Consensus Algorithm

7.4.5 Byzantine Problem and Algorithm

7.4.6 Message Authentication Code (MAC)

7.4.7 Digital Signature

7.4.8 Bitcoins

1. Mining
2. Consensus mechanism
3. Lightning network
4. Side chain
5. Hotspot problem
6. Branching

7.4.9 Ethereum

7.4.10 Hyperlodger

7.5 Artificial Intelligence

7.5.1 Mathematical Foundations

7.5.2 Machine Learning

7.5.3 Neural Networks

7.5.4 Deep Learning

7.5.5 Applications

7.5.6 TensorFlow, DeepLearning4J