1. **Maven** - a tool that can be used for building and managing any Java-based project.

- Types of Repositories:

local - /.m2 (C:/Users/superdev/.m2)

central - is located at <http://repo.maven.apache.org/maven2/>

Whenever you run build job, maven first try to find dependency from local repository.

remote – belong to specific company such as Apple and Facebook, any developer in the group can download the repository by remote.

(Git Hub as our remote repository)

add dependency on the pom.xml file. (Maven -> refresh)

- life cycle: validate, compile, test, package (jar), verify, install, deploy. (7 steps)

command line: mvn clean, mvn test, mvn install …

2. **Git** - distributed version control system, handle projects with speed and efficiency.

- 2 ways to push: IntelliJ: Version Control -> Create Git repository …

Terminal: git add .

git commit -m "add read me file"

git push

3. Eight Basic **Data Types**

- primitive type:

byte, short, int, long, float, double, char, boolean

- wrapper class -

Byte, Short, Integer, Long, Float, Double, Character, Boolean

- pri -> wrap: autoboxing, wrap -> pri: unboxing.

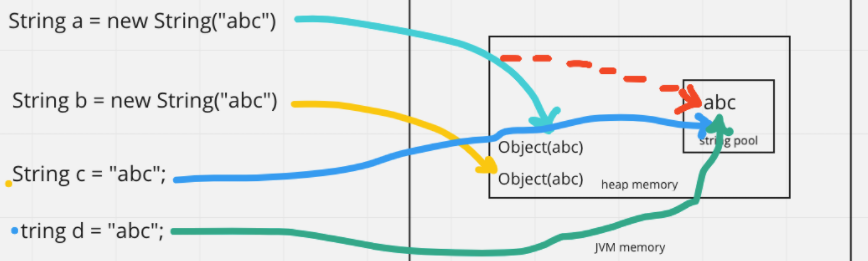
4. **String**/StringBuilder/StringBuffer

- String: **immutable**, thread safe ( + “”; // wouldn’t change content)

- StringBuilder: mutable, not thread safe (.append())

- StringBuffer: mutable, thread safe (.append())

- String constant pool



EX: every primitive data type (Wrapper Class) has constant pool.

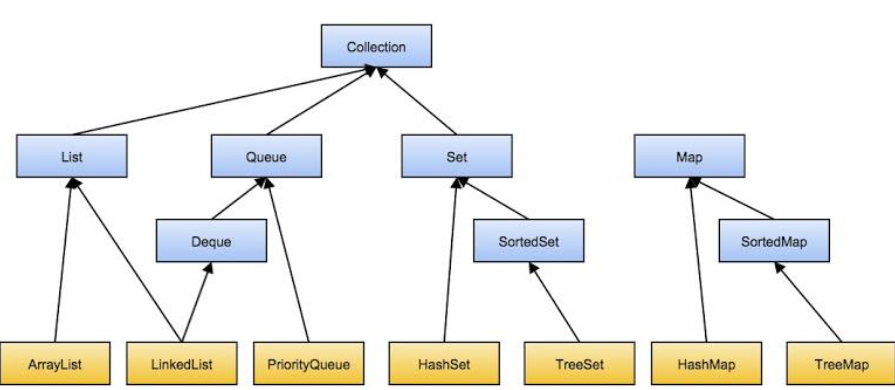
Integer limited range -127~128. Over range, different reference.

- == vs equals(): == compare reference, equals() compare content.

5. **equals** / **hashcode**

- override for return true value of equal(), or just shallow equal (reference).

6. **Collection**



- List : allow duplicated elements. Insertion Order

ArrayList (insert/remove takes O(n) time, retrieve data O(1) since random access)

LinkedList (insert/remove takes O(1) time, retrieve data O(n))

- Set : unique elements

HashSet (don’t keep insertion order)

LinkedHashSet (keep insertion order)

TreeSet (sorted)

- Deque:

ArrayDeque: first [ ] last

replace Stack: deque.offerFirst(), deque.pollFirst()

push(), , pop()

- vector & stack (deprecated)

thread safe, push pop (stack), FILO (stack)

- Map: (not implements Collection)

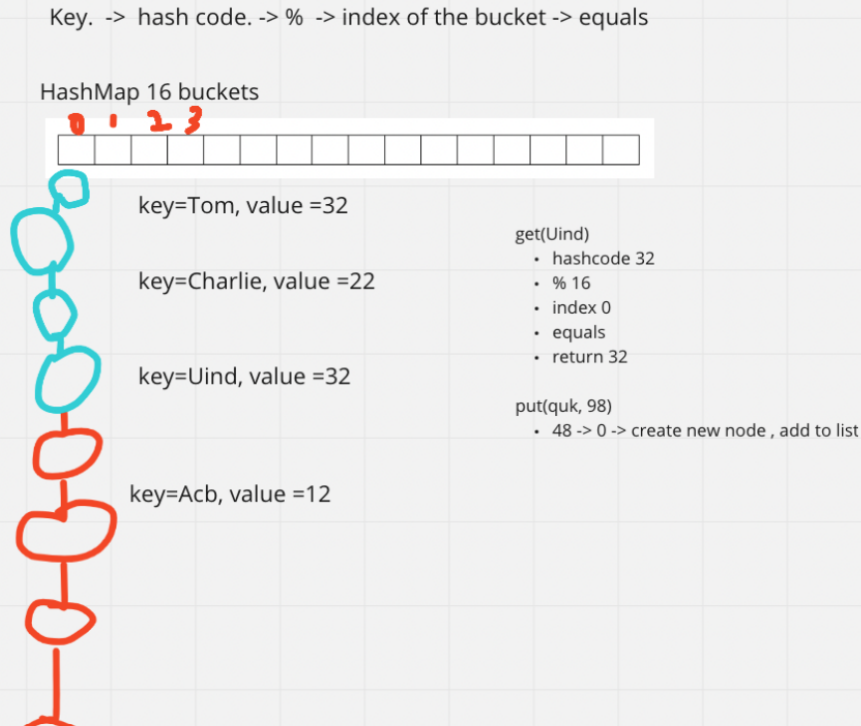
HashMap: not thread safe

**create 16 buckets -> hash(key) generates hashCode -> %16 to cal idx**

**-> equal(key) -> put or get val of key.**

**Same bucket using separate chaining. (Linked List) add new data to the “first” position.**

OR, open addressing which add data to next bucket (NOT in JAVA)

****

After Java 8, size of linked list (O(n)) over 8, will transform to red-black tree to improve performance (O(logn))

LinkedHashMap: Insertion Oder

TreeMap: Sorted (by Key)

HashTable: synchronized, thread safe (1 lock)

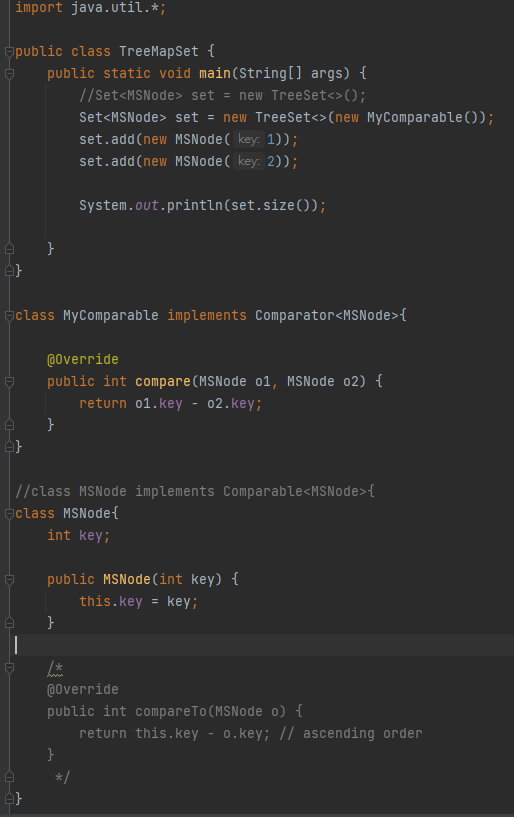
ConcurrentHashTable: synchronized, thread safe (16 lock)

- Queue : FIFO

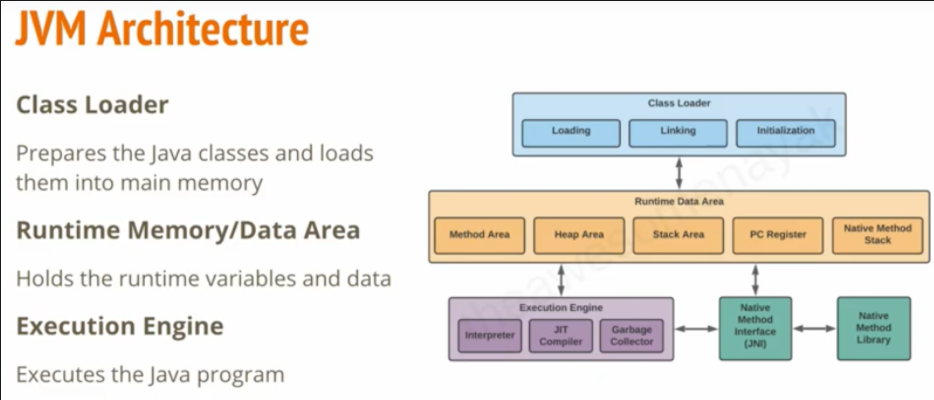
- Heap : PriorityQueue, maxHeap (parent node > child nodes, root max), minheap.

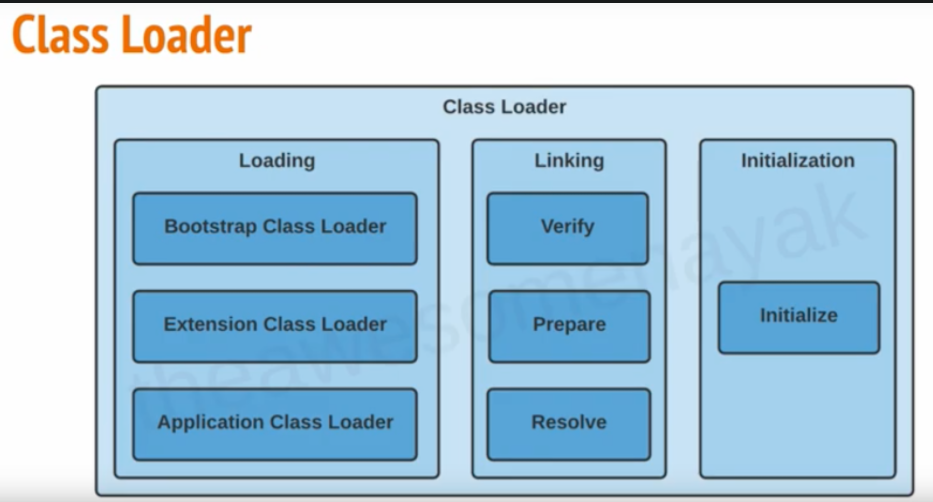
HashSet internal work by HashMap.

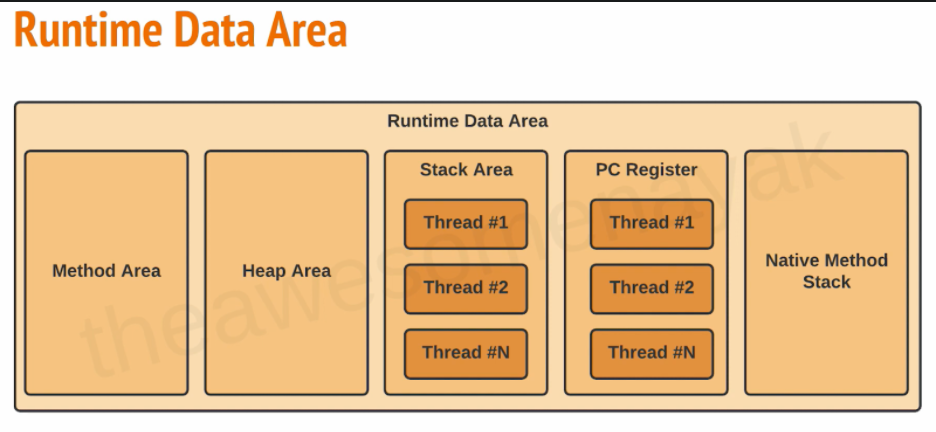
7. **Comparable** vs **Comparator**:



8. **JVM**







一張含有 文字 的圖片

自動產生的描述

* Class Loader (main memory)
  + Loading (parent delegation mechanism, recursive method, super class of lower level class))
    - Bootstrap Class Loader (java.util, java,lang, …)
    - Extension Class Loader (JDBC driver, ODBC driver)

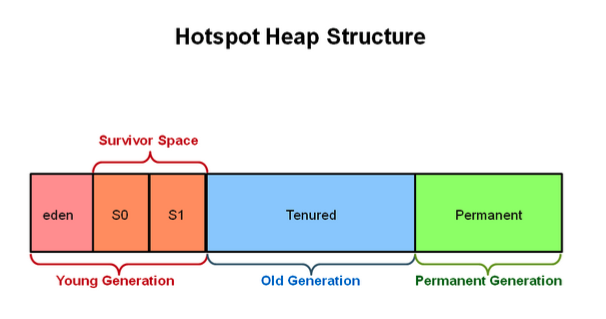
* + - Application Class Loader (User define class)

ClassLoader Sub System

* + Linking (3 steps)
    - Verify (correctness of .class file)
    - Prepare (allocated memory for static fields for class or interface)
    - Resolve (the process of dynamically determining concrete values from symbolic references in the run-time constant pool.)
  + Initialization
    - initialize (initialize static block, static class, constant pool…)
* Runtime Memory/Data Area
  + Method Area (class level data, static field)
  + Heap Area (object, new keyword)
  + Stack Area (private by single thread, call method))
    - Threads
  + PC Register (execution address of current thread)
    - Threads
  + Native Method Stack (implement by other language like c/c++)
* Execution Engine
  + Interpreter (execute byte code line by line)
  + JIT Compiler (find hotspot to improve performance of Interpreter)
  + Garbage Collector
* Native Method Interface (JNI) (bridge between execution engine and Native Method Library)
* Native Method Library (implement by other language like c/c++)

11. Garbage Collector

* serial GC (single thread)
* parallel GC (multithread, default in JAVA)
* G1 GC (by rank of chunk)
* CMS GC(Concurrent Mark Sweep) G1
  + deprecated since java 9
  + completely removed in java 14



* minorGC -
  + Young Generation
  + more frequently
* majorGC -
  + Old Generation
  + sometimes

12. Keywords

* for data types
  + byte, short, int, long, float, double, char, boolean
* flow control
  + (if, else), (switch, case, default), for, (do, while), (break, continue), return
* modifiers
  + (public, private, protected), static, final, abstract, synchronized, native, strictfp, transient, volatile
* exception handling
  + (try, catch, finally), (throw, throws), assert
* class related
  + class, package, import, extends, implements, interface
* Object related keywords,
  + new, instanceof, super, this
* final finally finalize()
  + variable - constant, must be initialized
  + method - can’t be overridden
  + class - can’t be extended
  + container (like list) – can modify the content, can’t address to different reference (new container).
* Immutable class
  + final class
  + private final fields
  + no setter
  + return deep copy of the collections for getter
* static (method area)
  + block
  + variable
  + methods
  + classes

13. OOP

* Abstraction
  + abstract class
  + interface
* Encapsulation
  + private variables
  + getter & setter
* Inheritance
  + extends (single class)
  + implements (multiple interfaces)
* Polymorphism
  + override (method of parent class)
  + overload (same name with diff inputs)

14. Exception

* checked exception
  + IO or Compile time exception
* unchecked exception
  + Runtime or Null Pointers Exception
* Handle Exception
  + try catch (finally)
  + throws
  + throw vs throws
* customize exception
  + user class extends Exception
* handle multiple exceptions
  + catch(E1){}

catch(E2){}…

* + catch(E1|E2|E3…){}

15. Generics - <K, V, E> ..

* easier and less error-prone
* enforce type correctness at compile time
* without causing any extra overhead to your application

16.IO Stream

* Byte Stream
  + 1 Byte
  + InputStream, OutputStream
* Character Stream
  + 2 Bytes
  + Reader, Writer
* File
  + part of java.io
  + access to underlying file systems

17. Serialization and deserialization

18. Java 8 features

* lambda
  + (arguments) -> {body}
  + functional programming
  + less code
* Functional Interface
  + with only one abstract method
  + allow # of default (concrete) method
  + can define by lambda expression in main()
  + Predicate - test(T t), Fuction - apply(T t),

Consumer - accept(T t), Supplier - get()

* Stream API (can define by lambda)
  + intermedia - return a stream
    - map( -> ), filter( -> )...
  + terminal - return nun-stream
    - forEach( -> ), collect( -> )…
* Optional
  + to handle NullPointerExcception

19. MultiThread:

thread vs process:

process:

independent memory space, heap, OS resources

thread”

shared memory space

private stack, program counter, register

thread states:

new - thread create, not yet start

runnable - executing in JVM

blocked - wait for a monitor lock to enter synchronized block or method

waiting - Object.wait with no timeout

Thread.join() with no timeout

park()

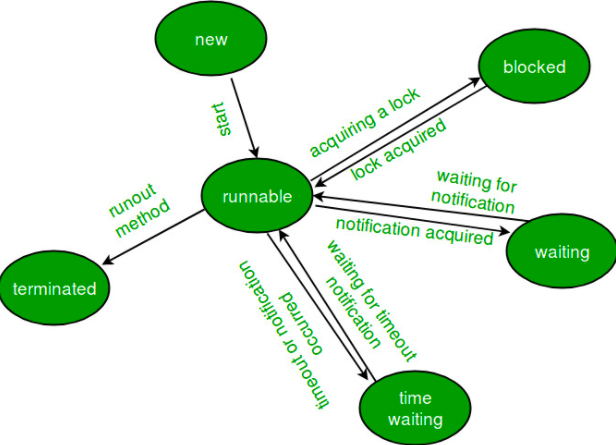
timed\_waiting - thread sleep

Object.wait() with timeout

thread.join with timeout

park

terminated - thread has completed



thread creation:

extends Thread

implements Runnable

implements Callable

thread pool

一張含有 文字 的圖片

自動產生的描述

Thread pool:

customized thread pool

一張含有 文字 的圖片

自動產生的描述

ThreadPoolExecutor (

corePoolSize,

maximumPoolSize,

KeepAliveTime,

Time unit,

work queue,

thread factory,

handler -

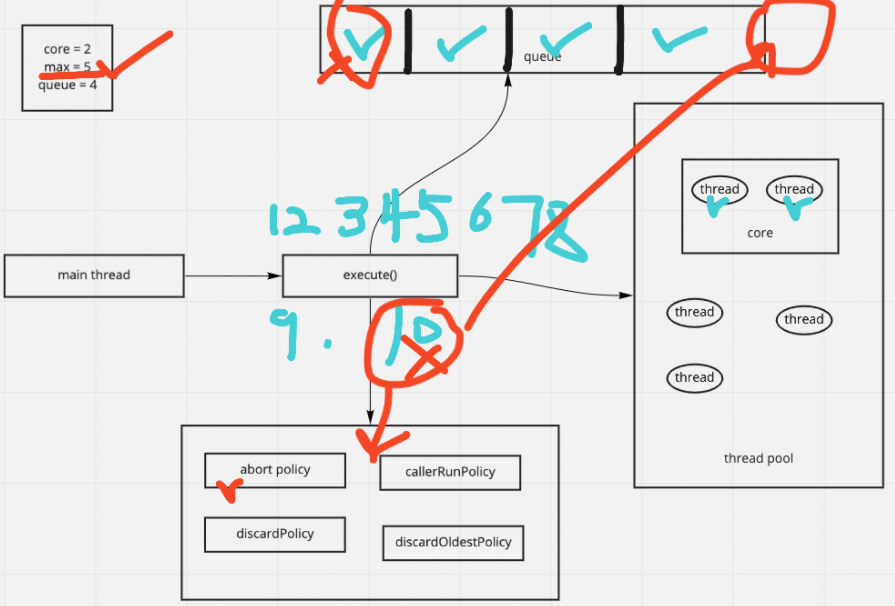
abortPolicy

callerRunPolicy

discardPolicy

discardOldestPolicy

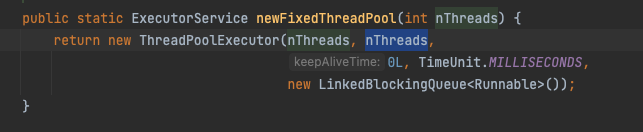
);



一張含有 文字 的圖片

自動產生的描述

built-in thread pool (not often use in real job)



一張含有 文字 的圖片

自動產生的描述

OutOfMemoryError (Since unlimited # of data can put in LinkedBlockingQueue)

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

自動產生的描述

Lock:

* synchronized
* Lock interface

synchronized:

* block
* method
* static method
* class

class Demo {

public void method() {

synchronized(Demo.class) {

}

}

public synchronized void method() {

}

public synchronized static void method() {

}

public void method () {

synchronized(this) {

}

}

}

Lock interface

* lock(), unlock(), newCondition(), tryLock(), lockInterruptibly()
* ReentrantLock class

ReadWriteLock interface

* method
  + Lock readLock();
  + Lock writeLock();
* class
  + reentrantReadWriteLock