The Objective of the assignment is to create API to use with following measures

[1. Simple API Design 2](#_Toc197348146)

[2. Performance Optimizations 3](#_Toc197348147)

[3. Small Memory Footprint 4](#_Toc197348148)

[4. Thread Safety 6](#_Toc197348149)

[5. Test Coverage 7](#_Toc197348150)

[6. Benchmark 8](#_Toc197348151)

# Simple API Design

* 1. **Single entry point**: Just one FixMessageParser.parseMessage()
  2. **Simple operation:**
     1. return the tag value using get operation in byte[] or String
     2. return the entire bye[] in string format with toString operation
  3. **No setup/teardown**: Stateless operation

**Sample Code:**

|  |
| --- |
| **Output:**  Original Message:8=FIX.4.2|9=176|35=D|49=SenderCompId|56=TargetCompId|34=2|52=20250431-12:30:00|11=12345|21=1|40=2|54=1|60=20250430-12:30:00|10=128|  Parsed Message:8=FIX.4.2|9=176|35=D|49=SenderCompId|56=TargetCompId|34=2|52=20250431-12:30:00|11=12345|21=1|40=2|54=1|60=20250430-12:30:00|10=128|  Value of Tag 8 :FIX.4.2 |

**Interface**

|  |  |
| --- | --- |
| **Method** | **Usage** |
| public String getValueAsString(int key) | Returns value of a tag in String  @param key Fix Message Tag as int  @return value of tag in String |
| public byte[] get(int key) | Returns value of a tag in byte[]  @param key Fix Message Tag as int  @return value of tag in byte[] |
| public String toString() | This method return entire fixMap as String after parsing FIX message  @return the entire fixMap |

# Performance Optimizations

* 1. Processes message in one linear scan O(N)

|  |
| --- |
|  |

* 1. Custom Map implementation for retrieving Tag value in linear scan O(N), which is faster than standard Map operation, implementation also favor CPU cache due to linear memory access pattern

|  |
| --- |
|  |

* 1. Avoid object creation by using primitive arrays and positions
  2. Enhance performance which have tighter memory coupling with the parser instead of standalone class for fixMap (the tradeoff is code maintenance and reusability) . However, inner class is better option for better performance
  3. Use custom parseTag method to parse integer to avoid Interger Object creation for reducing GC

# Small Memory Footprint

* 1. Zero-memory allocation when processing message

|  |
| --- |
|  |

* 1. Specialized Integer Parser to avoid Integer Object Creation

|  |
| --- |
|  |

* 1. Custom Map to avoid creation of Integer and String Object to reduce memory footprint , using primitive array as replacement.
     1. **Position tracking**: Stores (start,end) indexes instead of copies
     2. **Array reuse**: FixMap reuses position arrays
     3. **Lazy copying**: Only creates strings/arrays when requested
     4. **Primitive arrays**: int[] instead of Integer[]
     5. **No boxing**: Avoids Integer object creation

|  |
| --- |
|  |

# Thread Safety

* 1. **Stateless parsing:** all state is method-local on stack

|  |
| --- |
|  |

* 1. **Immutable Result:** accessibility of put method is set to private

|  |
| --- |
|  |

* 1. **Safe publication**: Arrays are fully constructed before publishing
  2. **No synchronization needed**: Lock-free design

# Test Coverage

* 1. **Following Unit Test has been implemented**

|  |
| --- |
|  |

# Benchmark

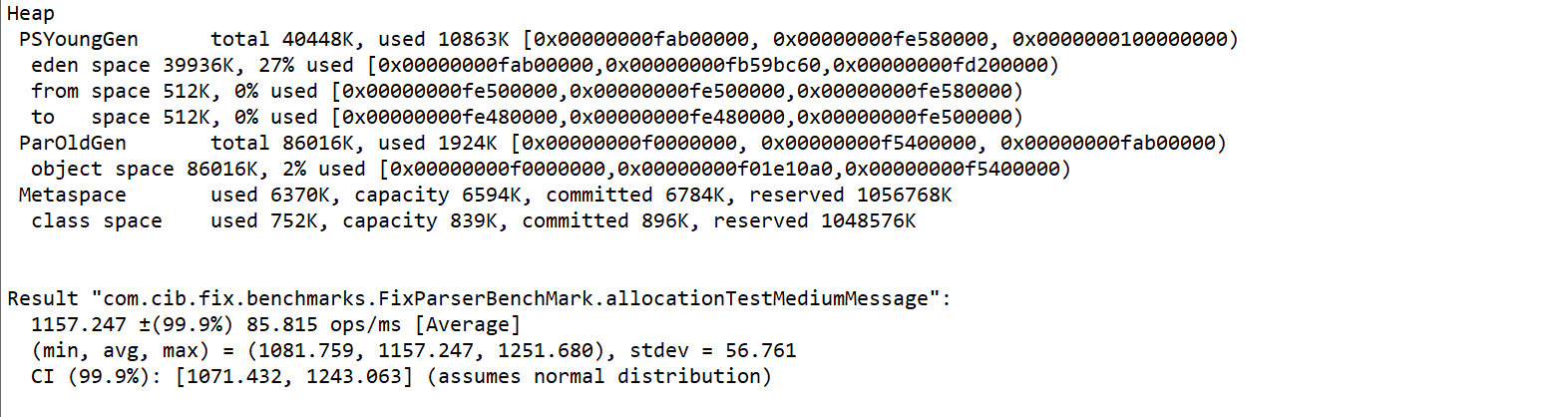
FixMessageParser is benchmark by using standing JMH and compare against FixMessageBenchMarkParser which implement using standing Map and String.split operation

* 1. Efficiency
     1. **allocationMediumMessage**: FixMessageParser is 2.6x more efficient than normal Parser
     2. **throughputMemoryTradeOff**: FixMessageParser is 3x more efficient than normal Parser

|  |
| --- |
|  |

* 1. Memory Footprint: FixMessage use less memory in yongGen, OldGen and MetaSpace when parsing medium message

FixMessageParser



FixMessageBenchMarkParser

