Phase 1 Branch Assessment: `cn_cram_3_1_write`

Summary

The branch implements the core Phase 1 goals — all codec wiring is correct and the write path produces valid CRAM 3.1 files. There are several items that need addressing before merge.

What's Done and Working

- Encoder throw removed from `CRAMEncoderV3_1`
- `DEFAULT_CRAM_VERSION` set to 3.1
- `ReadsResolver` filter override removed
- All RANS4x8 references in `CompressionHeaderEncodingMap` replaced with RANSNx16
- `RN_ReadName` uses Name Tokeniser
- `getBestExternalCompressor()` uses RANSNx16 (affects tag compression)
- `NameTokenisationEncode.tryCompress()` tuned with narrowed flag sets and CAT fallback
- Write tests exist in both `CRAM31Tests` and `HtsCRAMCodec31Test`
- Test data covers WGS, small slices, and unmapped reads

Blockers

1. TODO flagged for removal in `CRAMContainerStreamWriter`

There is a deprecated `writeHeader(SAMFileHeader)` overload with an explicit comment: *"remove before merging this branch."* This must be dealt with.

2. Samtools test guards missing

`CRAM31Tests.testCRAM31SamtoolsFidelity` and `HtsCRAMCodec31Test.testCRAMDecoder` throw `RuntimeException` (not TestNG `SkipException`) when samtools isn't installed. This will fail CI builds rather than skip gracefully. The correct pattern (used elsewhere in the codebase, e.g. `CRAMAllEncodingStrategiesTest`) is:

```
if (SamtoolsTestUtils.isSamtoolsAvailable()) {
    // ... do samtools work ...
} else {
    throw new SkipException("samtools is not installed, skipping test");
}
```

3. No way to request CRAM 3.0 output

Changing `DEFAULT_CRAM_VERSION` globally with no opt-out is a backward-incompatible change. Any downstream tool using `SAMFileWriterFactory.makeCRAMWriter()` will now produce 3.1 with no escape hatch. None of the writer constructors (`CRAMFileWriter`, `SAMFileWriterFactory`, `CRAMEncoderOptions`) accept a version parameter. This needs either:

- A deliberate decision to accept this as a version bump (with release notes and migration guidance), or
- A version parameter added to the writer API before merge

4. "Temp fix" for `preSorted=false`

`HtsCRAMCodec31Test.testRoundTripCRAM31` works around a bug by setting `.setPreSorted(true)`. The commit is labelled *"Temp fix for preSorted=false default."* The root cause of why `preSorted=false` fails or behaves unexpectedly should be investigated before merge.

Minor Issues

5. Stale Javadoc

`CompressionHeaderEncodingMap` line 72 still says *"Relies heavily on GZIP and RANS"* — should say RANSNx16.

6. Cross-codec constant in `NameTokenisationEncode`

The arith CAT fallback path in `tryCompress()` uses `RANSNx16Params.CAT_FLAG_MASK` as the parameter to `RangeEncode`. `RangeParams.CAT_FLAG_MASK` would be the correct constant. The values are likely identical (both `0x20`) but it's fragile and confusing.

7. No htsjdk-writes → samtools-reads interop test

The tests verify samtools→htsjdk and htsjdk→htsjdk round-trips, but don't verify that samtools can read htsjdk-written 3.1 files. Adding the reverse direction (htsjdk writes, samtools reads back) would strengthen confidence in interoperability.

8. Some GZIP data series could benefit from RANSNx16

Data series like `BS_BaseSubstitutionCode`, `MQ_MappingQualityScore`, `FN_NumberOfReadFeatures`, and `FP_FeaturePosition` are still GZIP-compressed. This is acceptable for a basic "normal" profile but leaves compression ratio on the table compared to what samtools achieves. Not a correctness issue.