

[illegible]

com.convetta.clavin.gazetteer.query

```
graph TD
    Root[com.convetta.clavin.gazetteer.query] --> LuceneGazetteer[LuceneGazetteer]
    Root --> LuceneGazetteerTest[LuceneGazetteerTest]
    Root --> QueryBuilder[QueryBuilder]
    Root --> UniqueFuzzyScoringRewrite[UniqueFuzzyScoringRewrite]
    Root --> GazetteerQuery[GazetteerQuery]
    Root --> Gazetteer[Gazetteer]

    LuceneGazetteer --> executeQuery[executeQuery]
    LuceneGazetteer --> getGeoName[getGeoName]

    LuceneGazetteerTest --> setUp[setUp]
    LuceneGazetteerTest --> testBorderCases[testBorderCases]
    LuceneGazetteerTest --> testFilterDups[testFilterDups]
    LuceneGazetteerTest --> testFuzzyMode[testFuzzyMode]
    LuceneGazetteerTest --> testGetGeoName[testGetGeoName]
    LuceneGazetteerTest --> testLoadAncestry[testLoadAncestry]
    LuceneGazetteerTest --> testNonAncestryIndex[testNonAncestryIndex]
    LuceneGazetteerTest --> testResolveAncestry[testResolveAncestry]
    LuceneGazetteerTest --> testSanitizeQueryText[testSanitizeQueryText]
    LuceneGazetteerTest --> testSanitizeLocations[testSanitizeLocations]
    LuceneGazetteerTest --> testSanitizeInput[testSanitizeInput]

    QueryBuilder --> build[build]
    QueryBuilder --> filterDups[filterDups]
    QueryBuilder --> fuzzyMode[fuzzyMode]
    QueryBuilder --> location[location]
    QueryBuilder --> maxResults[maxResults]
    QueryBuilder --> parents[parents]
    QueryBuilder --> removeParents[removeParents]
    QueryBuilder --> toString[toString]
    QueryBuilder --> toString2[toString]

    UniqueFuzzyScoringRewrite --> addClause[addClause]

    GazetteerQuery --> getGeoName1[getGeoName]
    GazetteerQuery --> getGeoName2[getGeoName]

    Gazetteer --> getGeoName3[getGeoName]
    Gazetteer --> getGeoName4[getGeoName]
```

The image displays 12 word cloud diagrams, each representing the vocabulary of a specific Stanford NLP Java API class. The words are arranged in a grid-like fashion, with colors indicating grammatical categories. The diagrams are:

- IndexDirectoryBuilder**: Contains words like `main`, `indexGeoName`, `buildIndex`, and `getOptions`.
- BinarySimilarity**: Contains words like `testTF`.
- WhitespaceLowerCaseAnalyzer**: Contains words like `createComponents`.
- BinarySimilarityTest**: Contains words like `WhitespaceLowerCaseTokenizer`.
- idfExplain**: Contains words like `idf` and `tf`.
- WhitespaceLowerCaseTokenizer**: Contains words like `isTokenChar` and `normalize`.

[illegible]

com.novetta.clavin.resolver

The diagram illustrates the dependency structure of the `com.novetta.clavin.resolver` project, organized into three main sections:

- ClavinLocationResolverHeuristicsTest** (Orange background):
 - `.setUp` depends on `slf4j`, `api`, and `logging`.
 - `testBorderCases` depends on `junit` and `testing`.
 - `testHeuristicsDC` depends on `junit` and `testing`.
 - `testHeuristicsEngland` depends on `junit` and `testing`.
 - `testHeuristicsIllinois` depends on `junit` and `testing`.
 - `testHeuristicsNorthAfrica` depends on `testing` and `junit`.
 - `testHeuristicsOntario` depends on `testing` and `junit`.
 - `testHeuristicsUK` depends on `testing` and `junit`.
 - `testHeuristicsUSA` depends on `testing` and `junit`.
 - `testHeuristics` depends on `junit` and `testing`.
- ClavinLocationResolverTest** (Orange background):
 - `.setUp` depends on `slf4j`, `logging`, and `api`.
 - `testBorderCases` depends on `junit` and `testing`.
 - `testIsDemonym` depends on `junit` and `testing`.
 - `testResolveLocations` depends on `junit` and `testing`.
 - `testSanitizeInput` depends on `testing` and `junit`.
 - `testFuzzyMatching` depends on `testing` and `junit`.
 - `testHeuristicsLoad_Lazy` depends on `junit` and `testing`.
 - `testHeuristicsLoad_Manual` depends on `testing` and `junit`.
 - `testHeuristicsLoad_Default` depends on `junit` and `testing`.
- LuceneLocationResolver** (Orange background):
 - `.setUp` depends on `lucene`, `slf4j`, `logging`, `api`, `apache`, and `index`.
 - `resolveLocations` depends on `api`, `index`, `lucene`, `logging`, `apache`, and `slf4j`.
- ClavinLocationResolver** (Green background):
 - `.equals` depends on `hashCode`.
 - `getHeuristics` depends on `getLocations`.
 - `getHeuristicsName` depends on `hashCode`.
 - `hashCode` depends on `hashCode`.
 - `LocationResolver` depends on `ResolvedLocationTest`.
 - `ResolvedLocationTest` depends on `testEquals`.

The diagram illustrates the evolution of a Java project structure, showing the progression from a single package to a complex, modular architecture.

Initial State: A single package, `com.novetta.clavin`, is shown at the top.

Left Branch (Extractor):

- The package `com.novetta.clavin` is split into `extractor` and `com.novetta.clavin.util`.
- The `extractor` package is further divided into `extractLocationNames` and `extractLocationOccurances`.
- The `extractLocationNames` package is further divided into `ApacheExtractor` and `TextBody`.
- The `extractLocationOccurances` package is further divided into `AdaptNlpExtractorTest` and `LocationOccuranceTest`.

Right Branch (Utils):

- The package `com.novetta.clavin.util` is further divided into `DamerauLevenshteinTest`, `ListUtilsTest`, `InfiniteCharArray`, `Null`, `TextUtilsTest`, and `ListUtils`.

Final State: The project structure is fully developed, showing a complex, modular architecture with multiple packages and classes.