

[illegible]

The treemap visualization displays the package structure of the LuceneGazetteer project. The packages are color-coded: orange for main classes, green for test classes, and yellow for utility classes. The treemap is organized into three main sections: LuceneGazetteer, LuceneGazetteerTest, and QueryBuilder. Each section contains sub-packages and their respective classes, with the size of each rectangle representing the relative size of the package.

LuceneGazetteer

- executeQuery
 - lucene
 - index
- apache
 - logging
 - full-text
 - slf4j
 - indexing
- api
- getGeoName
 - api
 - apache
 - full-text
 - logging
 - indexing
 - slf4j
 - lucene
 - index
- resolveParents
 - slf4j
 - full-text
 - logging
 - apache
 - lucene
 - api
 - index
- getClosestLocations
 - logging
 - slf4j
 - apache
 - api
 - index
 - lucene
- buildFilters
 - apache
 - full-text
 - indexing
 - lucene
 - index
- sanitizeQueryText
 - apache
 - lucene
 - index
 - loadAncestry

LuceneGazetteerTest

- setUp
 - lucene
 - index
- testBorderCases
 - lucene
 - index
- testFilterDups
 - lucene
 - index
- testGeoName
 - lucene
 - index
- testLoadAncestry
 - lucene
 - index
- testResolveAncestry
 - lucene
 - index
- testResolveLocations
 - lucene
 - index
- testSanitizedInput
 - lucene
 - index

QueryBuilder

- build
 - lucene
 - index
 - indexing
 - full-text
 - apache
- getTopLevelBuilder
 - apache
 - index
 - full-text
 - indexing
 - lucene
 - getMaxSize
- filterDups
 - lucene
 - index
- location
 - lucene
 - index
- parents
 - lucene
 - index
- toString
 - lucene
 - index

com.novetta.claavin.index

IndexDirectoryBuilder

```

classDiagram
    class IndexDirectoryBuilder {
        .
        sf4j
        api
        full-text
        apache
        index
        lucene
        indexing
        logging
    }
    class main {
        command-line
        parser
        sf4j
        cli
        io
        logging
        api
    }
    class indexGeoName {
        indexing
        apache
        lucene
        index
        full-text
    }
    class buildIndex {
        logging
        indexing
        index
        apache
        full-text
        sf4j
        lucene
        api
    }
    class getOptions {
        command-line
        lucene
        apache
        parser
        cli
    }
    class indexGeoName {
        <del>checkDescendantsResolved</del>
        logging
        api
        sf4j
    }
    class loadAlternateNames {
        sf4j
        api
    }
    class printHelp {
        parser
        command-line
        cli
    }
    class resolveAncestry {
        api
        logging
        sf4j
    }
    class logUnresolved {
        api
        logging
    }
    class resolveUnresolved {
        sf4j
        api
        logging
    }
    IndexDirectoryBuilder --> main
    IndexDirectoryBuilder --> indexGeoName
    IndexDirectoryBuilder --> buildIndex
    IndexDirectoryBuilder --> getOptions
    IndexDirectoryBuilder --> indexGeoName
    IndexDirectoryBuilder --> loadAlternateNames
    IndexDirectoryBuilder --> printHelp
    IndexDirectoryBuilder --> resolveAncestry
    IndexDirectoryBuilder --> logUnresolved
    IndexDirectoryBuilder --> resolveUnresolved
  
```

BinarySimilarity

```

classDiagram
    class BinarySimilarity {
        .
        lucene
        index
        apache
        full-text
        indexing
    }
    class idfExplain {
        index
        apache
        full-text
        indexing
        lucene
    }
    class idf {
        <del>idf</del>
        tf
    }
    BinarySimilarity --> idfExplain
    BinarySimilarity --> idf
  
```

WhitespaceLowerCaseAnalyzer

```

classDiagram
    class WhitespaceLowerCaseAnalyzer {
        .
        lucene
        full-text
        apache
        index
    }
    class createComponents {
        full-text
        indexing
        lucene
        apache
    }
    WhitespaceLowerCaseAnalyzer --> createComponents
  
```

BinarySimilarityTest

```

classDiagram
    class BinarySimilarityTest {
        testTF
    }
    class WhitespaceLowerCaseTokenizer {
        <del>tokens</del>
        aTokenChar
        normalize
    }
    BinarySimilarityTest --> WhitespaceLowerCaseTokenizer
  
```

The image displays 12 screenshots of Java code snippets, each representing a different class or method. The snippets are arranged in a grid-like fashion, with each snippet showing a different set of code blocks and comments. The snippets are:

- MultipartLocationResolver**: Includes methods like `setUpClass`, `parameters`, `verifyCity`, and `verifyLocation`.
- MatchedLocation**: Includes methods like `equals`, `getBatch`, `getBatchSize`, `getBatchType`, `getBatchValue`, and `toString`.
- AdaptNlpEx**: Includes methods like `apache`, `client`, `logging`, `json`, `http`, and `slf4j`.
- MultipartLocationResolver**: Includes methods like `findCandidates` and `resolveLocation`.
- MultipartLocationName**: Includes methods like `equals`, `getCity`, `getState`, `getCountry`, and `hashCode`.
- ResolvedMultipartLocation**: Includes methods like `equals`, `getCity`, `getState`, `getCountry`, and `hashCode`.
- MultiLevelMultipartLocationResolverTest**: Includes methods like `setUpClass` and `parameters`.
- DefaultScorer**: Includes a method like `score`.
- SearchResult**: Includes a method like `Scorer`.
- LocationOccurrence**: Includes methods like `equals`, `getCity`, `getState`, `getCountry`, and `hashCode`.
- ApacheExtractor**: Includes a method like `testNlp`.
- Entity**: Includes methods like `getCity`, `getState`, `getCountry`, and `hashCode`.

The diagram illustrates the layout of a Java IDE workspace, showing two main tabs: **com.novetta.clavin.resolver** and **ClavinLocationResolverTest**.

com.novetta.clavin.resolver contains several sub-tabs:

- ClavinLocationResolverHeuristicsTest**: Includes a package view with `.*`, `slf4j`, `api`, and `heuristic`. The `setUp` method is visible, and `testBorderCases` is listed. The `testHeuristicsDC`, `testHeuristicsEngland`, and `testHeuristicsIllinois` methods are shown, each with `unit` and `testing` parameters.
- LuceneLocationResolver**: Includes a package view with `.*`, `lucene`, `slf4j`, `logging`, `apache`, `index`, and `api`. The `resolveLocations` method is visible, and `testBorderCases` is listed. The `testHeuristicsDC`, `testHeuristicsEngland`, and `testHeuristicsIllinois` methods are shown, each with `unit` and `testing` parameters.
- ResolvedLocation**: Includes a package view with `.*`, `getHashCode`, `getMatchedName`, `hashCode`, `equals`, `toString`, and `testEquals`. The `testBorderCases` method is listed.

ClavinLocationResolverTest contains several sub-tabs:

- ClavinLocationResolver**: Includes a package view with `.*`, `slf4j`, `heuristic`, `api`, and `testing`. The `setUp` method is visible, and `testBorderCases` is listed. The `testHeuristicsDC`, `testHeuristicsEngland`, and `testHeuristicsIllinois` methods are shown, each with `unit` and `testing` parameters.
- ResolvedLocation**: Includes a package view with `.*`, `getHashCode`, `getMatchedName`, `hashCode`, `equals`, `toString`, and `testEquals`. The `testBorderCases` method is listed.
- ResolvedLocationTest**: Includes a package view with `.*`, `getHashCode`, `getMatchedName`, `hashCode`, `equals`, `toString`, and `testEquals`. The `testBorderCases` method is listed.

The diagram shows the flow of execution from the `Test` button to the `ClavinLocationResolverTest` tab, then to the `ClavinLocationResolver` tab, and finally to the `ResolvedLocation` tab. The `testBorderCases` method is highlighted in the `ResolvedLocation` tab, indicating it is the current test being executed.

The image displays 15 abstract visualizations of Java code snippets, each with a title and a grid of colored blocks representing code elements. The visualizations are arranged in a grid-like fashion, with some having multiple rows and columns. The colors used for the blocks are primarily green, yellow, orange, and blue. The titles of the visualizations are: AdaptnlpExtractor, extractLocationNames, Entity, ApacheExtractor, LocationOccurrence, TextBody, com.novetta.clavin, GeoParser, AllTestsSuite, GeoParserFactoryTest, AdaptnlpExtractorTest, extractLocationNames, DamerauLevenshtein, ListUtilsTest, InfiniteCharArray, TextUtilsTest, and ListUtils. Each visualization shows a different arrangement of code elements, such as class names, method names, and variables, represented by the colored blocks.