

BasicGeoName

<code>*</code> <code>org.ahoyahoy</code>	<code>getAdmin3Code</code> <code>String</code>	<code>getAncestryKey</code> <code>String</code>	<code>getFeatureCode</code> <code>String</code>	<code>getLongitude</code> <code>String</code>	<code>getParentId</code> <code>none</code>	<code>getPopulation</code> <code>none</code>	<code>getRefName</code> <code>none</code>	<code>getRefParentCode</code> <code>none</code>
<code>buildAncestryKey</code> <code>String</code>	<code>getAdmin4Code</code> <code>String</code>	<code>getAsciiName</code> <code>String</code>	<code>getGazetteerRecord</code> <code>String</code>	<code>getModificationDate</code> <code>String</code>	<code>getRefParentCode</code> <code>none</code>	<code>isAncestorOf</code> <code>none</code>	<code>isAncestryResolved</code> <code>none</code>	<code>isDescendantOf</code> <code>none</code>
<code>equals</code> <code>String</code>	<code>getAdminLevel</code> <code>String</code>	<code>getGeoFeatureCode</code> <code>String</code>	<code>getRefParentCode</code> <code>String</code>	<code>getName</code> <code>String</code>	<code>getTimezone</code> <code>none</code>	<code>getRefParentCode</code> <code>none</code>	<code>getRefParentCode</code> <code>org.ahoyahoy</code>	<code>parseGeoName</code> <code>none</code>
<code>getAdmin1Code</code> <code>String</code>	<code>getRefParentCode</code> <code>String</code>	<code>getElevation</code> <code>String</code>	<code>getGeoNameId</code> <code>String</code>	<code>getParent</code> <code>String</code>	<code>hashCode</code> <code>none</code>	<code>getRefParentCode</code> <code>none</code>	<code>setParent</code> <code>org.ahoyahoy</code>	<code>toString</code> <code>none</code>
<code>getAdmin2Code</code> <code>String</code>	<code>getAlternateNames</code> <code>String</code>	<code>getFeatureClass</code> <code>String</code>	<code>getLatitude</code> <code>String</code>	<code>getParentAncestryKey</code> <code>String</code>				

The screenshot displays two Java test classes, `BasicGeoNameTest` and `LazyAncestryGeoNameTest`, within the `FeatureCodeBuilder` project. The `BasicGeoNameTest` class is on the left, and the `LazyAncestryGeoNameTest` class is on the right. Both classes are part of the `org.mesothorpes.featurecodebuilder` package. The `BasicGeoNameTest` class includes methods for testing basic geo-name functionality, such as `testAllAttributes`, `testParseTerritory`, and `testParseExceptions`. The `LazyAncestryGeoNameTest` class includes methods for testing lazy ancestry geo-name functionality, such as `testAllAttributes`, `testParseTerritory`, and `testParseExceptions`. The code is written in Java and uses the `JUnit` testing framework.

The diagram illustrates the interfaces for the LuceneGazetteerTest and QueryBuilder classes. Each interface is represented by a grid of colored boxes, where each box contains a method name and a small input field.

LuceneGazetteerTest Interface

The LuceneGazetteerTest interface consists of 16 methods arranged in a 4x4 grid:

- setUp**: A method that takes a single string input.
- testBorderCases**: A method that takes a single string input.
- testFilterDups**: A method that takes a single string input.
- testGetFullGeoName**: A method that takes a single string input.
- testLoadAncestry**: A method that takes a single string input.
- testNonExistentIndex**: A method that takes a single string input.
- testFullLoadAncestry**: A method that takes a single string input.
- testFuzzyMode**: A method that takes a single string input.
- testFullGeoName**: A method that takes a single string input.
- testGeoName**: A method that takes a single string input.
- testGeoName**: A method that takes a single string input.
- testGeoName**: A method that takes a single string input.
- testGeoName**: A method that takes a single string input.
- testGeoName**: A method that takes a single string input.
- testGeoName**: A method that takes a single string input.

QueryBuilder Interface

The QueryBuilder interface consists of 16 methods arranged in a 4x4 grid:

- addAdminCodes**: A method that takes a single string input.
- addCountryCodes**: A method that takes a single string input.
- addFeatureCodes**: A method that takes a single string input.
- addParentIds**: A method that takes a single string input.
- ancestryMode**: A method that takes a single string input.
- build**: A method that takes a single string input.
- filterDups**: A method that takes a single string input.
- fuzzyMode**: A method that takes a single string input.
- includeHistorical**: A method that takes a single string input.
- clearFeatureCodes**: A method that takes a single string input.
- location**: A method that takes a single string input.
- removeAdminCodes**: A method that takes a single string input.
- removeFeatureCodes**: A method that takes a single string input.
- clearParentIds**: A method that takes a single string input.
- maxResults**: A method that takes a single string input.
- removeCityCodes**: A method that takes a single string input.
- removeCountryCodes**: A method that takes a single string input.
- featureCodes**: A method that takes a single string input.
- parentIds**: A method that takes a single string input.
- toString**: A method that takes a single string input.

The screenshot shows a Java Swing window titled "com.novetta.clavin.resolver". The window is divided into two main panels. The top panel, titled "ClavinLocationResolverHeuristicsTest", contains a grid of 10 test cases, each with a "setUp" button and a "test" button. The bottom panel, titled "ClavinLocationResolverTest", contains a grid of 10 test cases, each with a "setUp" button and a "test" button. To the right of the bottom panel is a "ResolvedLocation" panel with buttons for "getGeoName", "getGazetteer", "pickBestCandidates", "generateAllCombos", "isDemonym", and "resolvedLocations". Below this is a "LuceneLocationResolver" panel with a "resolvedLocations" button. The bottom right corner features a "ResolvedLocationTest" panel with a "test" button.

```

classDiagram
    class MatchedLocation {
        *
        getMatchCount()
        getMatches()
        equals()
        getMultipartMatch()
        isFullySpecified()
        getMatch()
        hashCode()
        toString()
    }
    class ResolvedMultipartLocation {
        *
        getCountry()
        getState()
        equals()
        hashCode()
        toString()
        getCity()
    }
    class DefaultScorer {
        *
        getMatchCount()
        getScore()
        hashCode()
        toString()
    }
    class MultipartLocationResolverTest {
        *
        setTestClass()
        verifyCity()
        verifyLocation()
        parameters()
    }
    class MultipartLocationResolver {
        *
        getMatchCount()
        getScore()
        hashCode()
        toString()
    }
    class SearchResult {
        *
        getMatchCount()
        getScore()
        hashCode()
        toString()
    }
    class Scorer {
        *
    }

    MatchedLocation --> ResolvedMultipartLocation : getMatchCount, getMatches, equals, getMultipartMatch, isFullySpecified
    MatchedLocation --> DefaultScorer : getMatch, hashCode, toString
    ResolvedMultipartLocation --> MultipartLocationResolverTest : getCountry, getState, equals, hashCode, toString, getCity
    ResolvedMultipartLocation --> DefaultScorer : getCountry, getState, equals, hashCode, toString, getCity
    DefaultScorer --> MultipartLocationResolver : getMatchCount, getScore, hashCode, toString
    MultipartLocationResolver --> SearchResult : getMatchCount, getScore, hashCode, toString
    SearchResult --> Scorer : getMatchCount, getScore, hashCode, toString
  
```

The screenshot shows the 'com.novetta.clavin.util' package in the IntelliJ IDE. The package contains the following classes and their methods/attributes:

- DamerauLevenshteinTest** (Green icon):
 - Methods: `testDistance1`, `testDistance2`
- DamerauLevenshtein** (Green icon):
 - Methods: `distance`, `distance1`, `distance2`, `distance3`
- Null** (Green icon):
 - Methods: `hashCode`, `equals`, `toString`
- InfiniteCharArray** (Green icon):
 - Methods: `get`, `removeAt`
- ListUtilsTest** (Blue icon):
 - Methods: `testShuffle`, `testSort`
- ListUtils** (Blue icon):
 - Methods: `shuffleList`
- TextUtils** (Blue icon):
 - Methods: `reverseString`
- GeoParser** (Yellow icon):
 - Methods: `parse`
- GeoParserFactory** (Yellow icon):
 - Methods: `getParser`
- WorkflowDemo** (Grey icon):
 - Methods: `getWorkflow`
- ClaimException** (Blue icon):
 - Methods: `getMessage`
- AltTextSuite** (Blue icon):
 - Methods: `testAltText`