



```

21 public class NGramMap { 12 usages
106 > public TimeSeries weightHistory(String word) { return weightHistory(word, MIN_YEAR, MAX_YEAR); }
109
110 /**
111  * Provides the summed relative frequency per year of all words in WORDS between STARTYEAR and
112  * ENDYEAR, inclusive of both ends. If a word does not exist in this time frame, ignore it
113  * rather than throwing an exception.
114  */
115 @ public TimeSeries summedWeightHistory(Collection<String> words, 1 usage
116                                     int startYear, int endYear) {
117     TimeSeries holder = new TimeSeries();
118     for (String word: words) {
119         if (map.containsKey(word)) {
120             TimeSeries x = weightHistory(word, startYear, endYear);
121             holder = holder.plus(x);
122         }
123     }
124     return holder;
125 }
126
127
128 > /** Returns the summed relative frequency per year of all words in WORDS. If a word does not ...*/
132 public TimeSeries summedWeightHistory(Collection<String> words) { no usages
133     return summedWeightHistory(words, MIN_YEAR, MAX_YEAR);
134 }
135 }

```