Teil XII

Innere Klassen

Datenstruktur Trie

Trie und TrieNode public

```java
public class Trie {

  private final TrieNode root = new TrieNode();

  public Trie() { }

  public void add(
    String word
  ) {
    root.add(word);
  }

  public int getPrefixCount(
      String prefix
  ) {
    return root.getPrefixCount(prefix);
  }

  public int getWordCount(
    String word
  ) {
      return root.getWordCount(word);
  }
}
```

```java
public class TrieNode {

  ...

  public TrieNode() { }

  public void add(
    String word
  ) {
    ...
  }

  public int getPrefixCount(
    String prefix
  ) {
    ...
  }

  public int getWordCount(
    String word
  ) {
    ...
  }
}
```

Datenstruktur Trie

`TrieNode`: package local

```java
public class Trie {

  private final TrieNode root = new TrieNode();

  public Trie() { }

  public void add(
    String word
  ) {
    root.add(word);
  }

  public int getPrefixCount(
      String prefix
  ) {
    return root.getPrefixCount(prefix);
  }

  public int getWordCount(
    String word
  ) {
      return root.getWordCount(word);
  }
}
```

```java
class TrieNode {

  ...

  TrieNode() { }

  void add(
    String word
  ) {
    ...
  }

  int getPrefixCount(
    String prefix
  ) {
    ...
  }

  int getWordCount(
    String word
  ) {
    ...
  }
}
```

Datenstruktur Trie

`TrieNode` als lokale Klasse innerhalb der Klasse `Trie`

```
1   public class Trie {
2
3     private final TrieNode root = new TrieNode();
4
5     public Trie() { }
6
7     public void add(
8       String word
9     ) {
10      root.add(word);
11    }
12
13    public int getPrefixCount(
14      String prefix
15    ) {
16      return root.getPrefixCount(prefix);
17    }
18
19    public int getWordCount(
20      String word
21    ) {
22        return root.getWordCount(word);
23    }
```

```
1     private class TrieNode {
2
3       ...
4
5       private TrieNode() { }
6
7       private void add(
8         String word
9       ) {
10        ...
11      }
12
13      private int getPrefixCount(
14        String prefix
15      ) {
16        ...
17      }
18
19      private int getWordCount(
20        String word
21      ) {
22        ...
23      }
24    }
25  }
```

Datenstruktur Trie

`Trie` als Interface, `TreeTrie` als Implementierung

```
1   public interface Trie {
2
3     public void add(
4       String word
5     );
6
7     public int getPrefixCount(
8       String prefix
9     );
10
11    public int getWordCount(
12      String word
13    );
14  }
```

```
1   public class TreeTrie
2     implements Trie {
3
4     private final TrieNode root = new TrieNode();
5
6     public TreeTrie() { }
7
8     public void add( ... ) { ... }
9
10    public int getPrefixCount( ... ) { ... }
11
12    public int getWordCount( ... ) { ... }
13
14    private class TrieNode {
15
16      ...
17
18      private TrieNode() { }
19
20      private void add( ... ) { ... }
21      private int getPrefixCount( ... ) { ... }
22      private int getWordCount( ... ) { ... }
23    }
24  }
```

Datenstruktur Trie

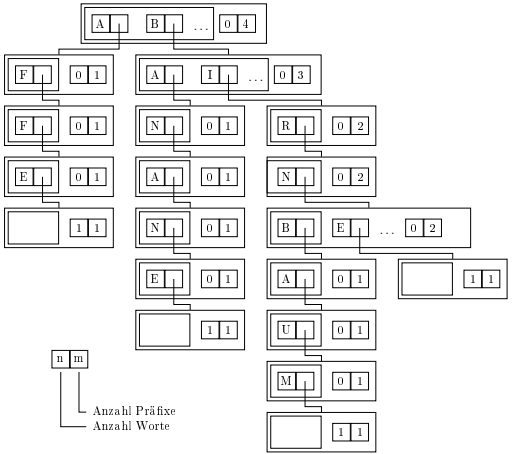
`Trie` als Erweiterung von `Collection`

```
1   public interface Trie
2     extends Collection{
3
4     public void add(
5       String word
6     );
7
8     public int getPrefixCount(
9       String prefix
10    );
11
12    public int getWordCount(
13      String word
14    );
15  }
```

```
1   public class TreeTrie
2     implements Trie {
3
4     private final TrieNode root = new TrieNode();
5     public TreeTrie() { }
6     public void add( ... ) { ... }
7     public int getPrefixCount( ... ) { ... }
8     public int getWordCount( ... ) { ... }
9
10    public void clear() {
11      root.clear();
12    }
13
14    public boolean isEmpty() {
15      return root.isEmpty();
16    }
17
18    public int size() {
19      return root.size();
20    }
21
22    private class TrieNode { ... }
23  }
```

Datenstruktur Trie

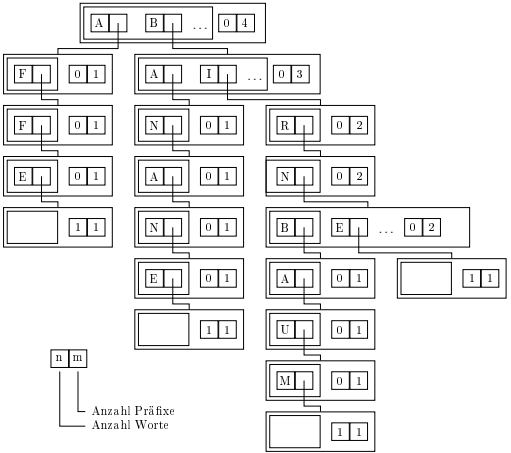
`TrieNode` als Erweiterung von `Collection`




```
1   public class TreeTrie
2    implements Trie {
3
4    private class TrieNode {
5
6     public void clear() {
7      for (Map.Entry<Character, TrieNode>
          successor : successors.entrySet()) {
8       successor.getValue().clear();
9      }
10     successors.clear();
11     prefixCount = 0;
12     wordCount = 0;
13    }
14   }
15  }
```

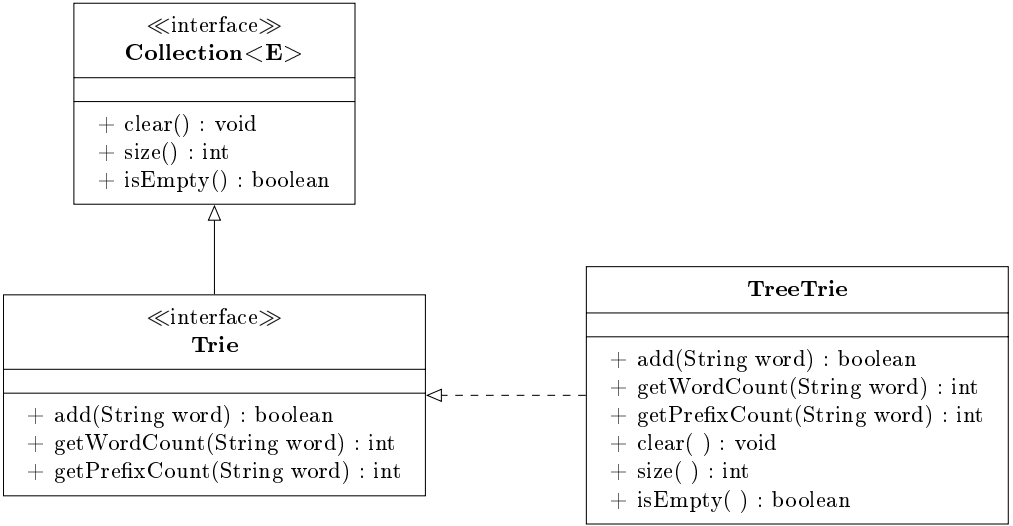
Datenstruktur Trie

TrieNode als Erweiterung von Collection




```
1   public class TreeTrie
2    implements Trie {
3
4    private class TrieNode {
5
6     public boolean isEmpty() {
7      return (wordCount == 0);
8     }
9
10    public int size() {
11     int numberOfElements = 0;
12      for (Map.Entry<Character, TrieNode>
         successor : successors.entrySet()) {
13       numberOfElements += successor.
         getValue().size();
14      }
15      numberOfElements += wordCount;
16
17      return numberOfElements;
18    }
19   }
20  }
```

Collections: Allgemeine Funktionalität



Datenstruktur Trie

Fehlende Methoden des Interface `Collection`

```
1       @Override
2       public Iterator<String> iterator() {
3           ...
4       }
5
6       @Override
7       public boolean addAll(Collection<?
         extends String> c) {
8           ...
9       }
10
11      @Override
12      public boolean remove(Object o) {
13          ...
14      }
15
16      @Override
17      public boolean removeAll(Collection<?>
          c) {
18          ...
19      }
```

```
1       @Override
2       public Object[] toArray() {
3           ...
4       }
5
6       @Override
7       public <T> T[] toArray(T[] a) {
8           ...
9       }
10
11      @Override
12      public boolean contains(Object o) {
13          ...
14      }
15
16      @Override
17      public boolean containsAll(Collection
         <?> c) {
18          ...
19      }
20
21      @Override
22      public boolean retainAll(Collection<?>
          c) {
23          ...
24      }
```