Mustertestangabe: Scrabble

Scrabble ist ein Brettspiel, bei dem ein Spieler aus einer Menge zufällig gezogener Buchstaben Wörter bilden muss. In diesem Beispiel soll eine Menge von Buchstaben zunächst eingelesen werden. Danach folgen ein oder mehrere Befehle. Unter anderem kann geprüft werden ob ein angegebenes Wort (String) aus der Menge von Buchstaben gebildet werden kann.

Aufgabenstellung

Die zu implementierenden Teile des Programms umfassen das Verarbeiten der Eingabe in der Methode `main` sowie fünf weitere Methoden. Alle zu implementierenden Teile gehören zur ausführbaren Klasse `Scrabble`.

Einlesen der Befehle

In der `main`-Methode der Klasse `Scrabble` wird von der Standardeingabe (`System.in`) mittels Scanner eingelesen. Zunächst soll ein String eingelesen werden, der die Buchstaben für das Scrabble Spiel enthält.

Der String wird durch die bereits vorgegebenen Befehle eingelesen und in ein Array von Zeichen (Typ: `char[]`) umgewandelt. Danach erwartet das Programm einen oder mehrere der folgenden Befehle:

- `print` gibt das Array in einer entsprechenden Formatierung aus. Dazu wird die Methode `Arrays.toString(char[])` benutzt. Der entsprechende Programmteil ist bereits vorgegeben.
- `occurrences` ***`c`*** gibt die Anzahl der Vorkommnisse des Zeichens ***`c`*** im eingelesenen Array aus (***`c`*** ist Parameter des Befehls und kann jeder beliebige Buchstabe sein).
- `distinct` ***`w`*** gibt einen String aus, der alle Zeichen enthält, die auch in ***`w`*** vorkommen, jedoch keine Zeichen mehrfach enthält. Beispiel: `distinct Mississippi` führt zur Ausgabe `Misp`. Die Reihenfolge der Zeichen in der Ausgabe soll jeweils durch das erste Auftreten in `w` bestimmt sein.
- `buildword` ***`w`*** gibt aus, ob das als Parameter angegebene Wort ***`w`*** aus den im eingelesenen Array gespeicherten Zeichen gebildet werden kann (true) oder nicht (false)
- `moveleft` ***`c`*** bewegt alle Vorkommnisse des Zeichens ***`c`*** im eingelesenen Array ganz nach links. Die Reihenfolge aller übrigen Zeichen rechts davon ist beliebig. Beispielsweise kann aus der Zeichenfolge `'b' 'e' 'w' 'e' 'g'` durch den Befehl `moveleft e` die Zeichenfolge `'e' 'e' 'w' 'b' 'g'` werden.

Das Programm läuft solange gültige Befehle eingelesen werden können. *Sie können davon ausgehen, dass alle Eingaben korrekt sind. Sie müssen keine Fehlerbehandlung implementieren.*

Ausgabe

Die Befehle `occurrences`, `distinct`, `buildword` erzeugen unmittelbar eine Ausgabe. Dagegen erzeugt `moveleft` keine Ausgabe sondern verändert das Array. Die Wirkung dieses Befehls kann danach mit `print` überprüft werden.

Gefragte Methoden

Der Befehl `print` benutzt die Methode ***`Arrays.toString(char[])`***, um das Array auszugeben. Die letzten 3 angeführten Befehle entsprechen den zu implementierenden Methoden `occurrences` (2 Methoden), `buildWord` und `moveLeft` der Klasse `Scrabble`.

`public static int occurrences(char[] charSet, char c)`
gibt die Häufigkeit des angegebenen Zeichens `c` in der Zeichenfolge `charSet` zurück.

`public static int occurrences(String word, char c)`
gibt die Häufigkeit des angegebenen Zeichens `c` im String `word` zurück.

`public static String distinct(String word)`
liefert einen String, der alle Zeichen enthält, die auch in `word` vorkommen, jedoch keine Zeichen mehrfach enthält. Die Reihenfolge der Zeichen im Rückgabestring soll durch das erste Auftreten in `word` von links nach rechts bestimmt sein.

`public static boolean buildWord(String word, char[] from)`
liefert einen `boolean` Wert der angibt, ob das als Parameter angegebene Wort `word` aus den im Array `from` gespeicherten Zeichen gebildet werden kann (true) oder nicht (false). Hinweis: Vermeiden Sie duplizierten Code. Die Methode `buildWord` kann unter Verwendung der oben angeführten Methoden implementiert werden.

`public static void moveLeft(char[] charSet, char c)`
bewegt alle Vorkommnisse des Zeichens `c` im Array `charSet` ganz nach links. Die Reihenfolge aller anderen Zeichen im Array ist danach beliebig und kann daher von den vorgegebenen Ausgabedateien abweichen.

Implementierungsreihenfolge

Folgende Vorgehensweise ist bei der Umsetzung der Aufgabe empfehlenswert:

- Beginnen Sie mit der Implementierung der beiden Methoden `occurrences`
- Implementieren Sie dann die Methode `distinct` und `moveLeft`
- Implementieren Sie zuletzt die Methode `buildWord`.
- Vervollständigen Sie jeweils parallel dazu die `main`-Methode, um die fertiggestellten Methoden testen zu können.

Testen

Kompilieren Sie früh und oft und versuchen Sie Fehler sofort zu beheben. Testen Sie nach jedem Kompilieren Ihre Implementierung. Zum Testen Ihres Programms stehen Ihnen drei Testdateien (`spec.i1`, `spec.i2`, `spec.i3`) zur Verfügung. Um eine Datei als Eingabe zu verwenden, gehen Sie folgendermaßen vor: `java Scrabble < ../io/spec.i1`

Testfragen

- Betrachten Sie die folgende Methodendeklaration

  ```
  public static void foo (char[] charSet, char c) { … }
  ```

  und folgende Anweisungssequenz, die `foo` aufruft:

  ```
  char[] x = new char[] {'a', 'b', 'c'};
  char y = 'b';
  foo(x, y);
  ```

  Welche generellen Aussagen lassen sich über den Inhalt der Variable `x` (Größe und Inhalt des von ihr referenzierten Arrays) nach der Ausführung dieser Sequenz treffen und welche über den Wert von `y`, ohne die Implementierung von `foo` zu kennen? Warum kann man diese Aussagen machen?

- Was ist ein formaler Parameter und was ist ein aktueller Parameterwert? Erklären Sie die Begriffe anhand des obigen Beispiels mit der Methode `foo`.

Testfälle

	INPUT	OUTPUT
spec.$1	test occurrences e occurrences t occurrences u print	1 2 0 [t, e, s, t]
spec.$2	beweg print buildword weg buildword ewig	[b, e, w, e, g] true false
spec.$3	test distinct Mississippi	Misp
spec.$4	beweg moveleft e print	[e, e, w, b, g]*

* Die drei Zeichen ganz rechts können auch eine andere Reihenfolge haben.