Aufgabe 3

a) **List:**

- Man kann jeweils eine Liste für Benutzernamen und E-Mail-Adressen definieren, wobei die Paare in beiden Listen den gleichen Index haben. Problem dabei wäre, alles was man in der einen Liste ändert, muss auch in der anderen Liste geändert werden um die Paare nicht zu vertauschen.
- Deswegen könnte man nur eine Liste erstellen, bei der Benutzername und E-Mail-Adressen in einem String zusammengesetzt (und durch ein Zeichen getrennt) der Liste hinzugefügt werden können. Anstelle des Strings wäre auch möglich eine eigene Klasse zu definieren, die jeweils Variablen für Name und E-Mail-Adresse enthält, oder für jedes Paar eine Liste oder ein Array (mit jeweils 2 Elementen) die dann in der Liste sind.
- Ein Problem wäre auch, dass man extra kontrollieren müsste, dass kein Benutzername mehr als einmal der Liste hinzugefügt wird.

-> Mithilfe einer oder mehrerer Listen könnte man das realisieren, allerdings ist es ziemlich unpraktisch.

Set:

- Im Gegensatz zur Liste kann man nicht zwei Sets verwenden, da man Benutzernamen und E-Mail-Adressen nicht in Relation setzen kann, weil die Elemente ungeordnet sind.
- Man hätte zwar die Möglichkeit, wie bei der Liste in einem Set alle Paare in Form von einem zusammengesetzten String, mithilfe einer Klasse oder einem Array Paare zu bilden. Jedoch bietet das Set keine Möglichkeit ein Element anhand eines Index oder des Wertes zu erhalten, sondern man hat nur einen Iterator (dabei ist nicht gewährleistet, dass die Elemente jedes Mal in der gleichen Reihenfolge sind)

-> Ein Set wäre sehr ungeeignet für diese Anwendung.

Map:

- Die Map bietet die Möglichkeit Wertepaare zu speichern, die dann wie in einer Liste geordnet sind und einen Index haben. So bräuchte man nur eine Map und könnte alle Paare darin speichern.
- Dabei hat sie auch noch den Vorteil, dass die Paare als Key-Value-Paar gespeichert werden, d.h. man benötigt nur den Key (in diesem Fall den Benutzernamen) um den dazugehörigen Wert zu bekommen.
- Mit einer Map ist außerdem automatisch gewährleistet, dass jeder Key nur einmal hinzugefügt werden darf.

-> Die Map ist optimal geeignet für diese Anwendung.

c) Die dynamischen Datenstrukturen bieten einfache Funktionen an, um Elemente hinzuzufügen, auszuwählen, zu entfernen (ohne dass Lücken entstehen wie beim Array). Zusätzlich dazu gibt es komplexere Datenstrukturen (wie z.B. die Map), die noch mehr Funktionen (wie z.B. die Speicherung als Wertepaar) bieten. Besonderer Vorteil gegenüber Arrays ist, dass sie sehr einfach erweiterbar sind. Bei einem Array müsste man ein komplett neues Array erstellen, wenn man dieses verlängern will.