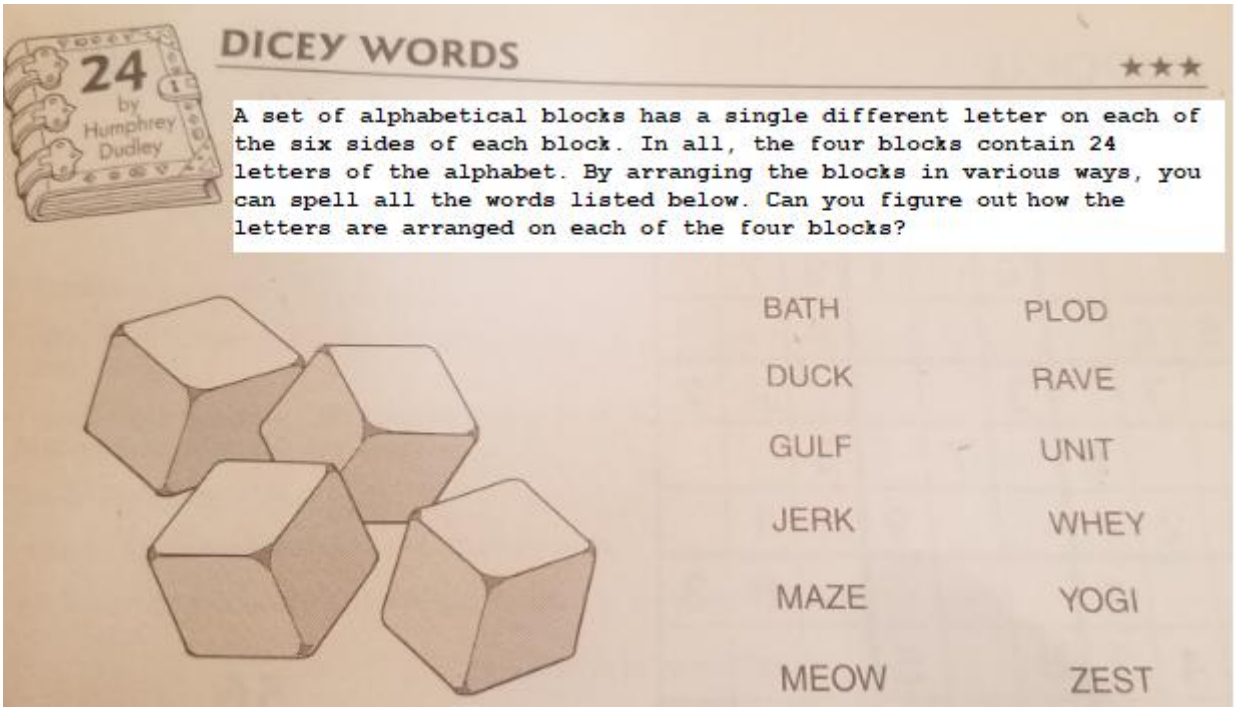
DiceyWords

First the motivation:



`Special note: All Letters will be UPPER CASE letters.`

`Special note: No die may be used more than once.`

You will implement the constructor and three methods in the `DiceyWords` class. The constructor has a `String[][]die` parameter representing four 6-sided die. `die[0]` represents one die, `die[1]` represents another die, and so on. Each die will contain 6 individual letters. No letter will appear on more than one die, and no letter will appear more than once on a given die.

The three methods are `getImpossibleWords (ArrayList<String> words)`, `missingDie(String[][] die, ArrayList<String> words)`, and `unusedLetters(ArrayList<String> words)`.

The `getImpossibleWords(ArrayList<String> words)` returns a `List<String>` containing all words in the parameter `words` that cannot be spelled with the four die passed to the constructor.

See next page for sample code.

The following code shows the results of the `getImpossibleWords` method.

The following code	Returns

`String[][] dice = { {"A", "W", "K", "U", "C", "X"},`
`{"B", "E", "S", "I", "G", "Q"},`
`{"T", "Z", "O", "R", "Y", "N"},`
`{"H", "M", "J", "D", "P", "F"}};`
`// note missing letters: "L" and "V"`
`DiceyWords dw = new DiceyWords(dice);`	

`ArrayList<String> words = new ArrayList<String>();`
`words.add("BATH");`
`words.add("MAZE");`
`words.add("MEOW");`
`words.add("WHEY");`
`words.add("JERK");`
`words.add("PYIC");`
`words.add("RASH");`
`words.add("BARD");`
`words.add("JUST");`
`words.add("MINX");`
`words.add("QUOD");`
`words.add("FRAG");`	

`dw.getImpossibleWords(words).size();`
 	
`//  every word in words can be spelled with the four die  */`
	0

`words.add(0, "FLAG");`
`// add the word FLAG which cannot be spelled with the four die`
 	
`words.add("WAKE");`
`// add the word WAKE which cannot be spelled with the four die`
 	
`words.add("MASS");`
`// add the word MASS which cannot be spelled with the four die`
`// This implies any word with duplicate letters cannot be`
`// spelled with the four die`	

`List<String> sol = dw.getImpossibleWords (words);`	
`sol.size();`	3
`sol.contains("FLAG");`	true
`sol.contains("WAKE ");`	true
`sol.contains("MASS ");`	true

The `missingDie(String[][] die, ArrayList<String> words)` has two parameters. The parameter `String[][] die` represents three 6-sided die. `die[0]` represents one die, `die[1]` represents another die, and `die[2]` represents the third die. Each die will contain 6 individual letters. No letter will appear on more than one die, and no letter will appear more than once on a given die. The `ArrayList<String> words` contains a List of words with length 4. This method is to return a `String` with length 6 containing the letters required on the missing fourth die allowing each word in `words` to be spelled. You may assume the String returned by the `missingDie` method will always contain 6 letters.

The following code shows the results of the `missingDie` method.

The following code	Returns

`String[][] d = { {"A", "W", "K", "U", "C", "X"},`
`{"B", "E", "S", "I", "G", "Q"},`
`{"H", "M", "J", "D", "P", "F"} };`	

`ArrayList<String> ws = new ArrayList<String>();`
`ws.add("BATH");`
`ws.add("MAZE");`
`ws.add("MEOW");`
`ws.add("WHEY");`
`ws.add("JERK");`
`ws.add("PYIC");`
`ws.add("RASH");`
`ws.add("BARD");`
`ws.add("JUST");`
`ws.add("MINX");`
`ws.add("QUOD");`
`ws.add("FRAG");`
`String strSol = dw.missingDie (d, ws);`	

`strSol.length()`	`6`
`"TZORYN".indexOf(strSol.substring(0, 1) >= 0);`	`true`
`"TZORYN".indexOf(strSol.substring(1, 2) >= 0);`	`true`
`"TZORYN".indexOf(strSol.substring(2, 3) >= 0);`	`true`
`"TZORYN".indexOf(strSol.substring(3, 4) >= 0);`	`true`
`"TZORYN".indexOf(strSol.substring(4, 5) >= 0);`	`true`
`"TZORYN".indexOf(strSol.substring(5, 6) >= 0);`	`true`
Yes, this implies `strSol` is a permutation of the letters: `"TZORYN"`	

The `unusedLetters(ArrayList<String> words)` has the single parameter `ArrayList<String> words` which contains a List of words with length 4. This method is to return a `String[]` containing all letters from the four die that were not needed to spell any word in `words`.

The following code shows the results of the `unusedLetters` method.

The following code	Returns

`String[][] dice = { {"A", "W", "K", "U", "C", "X"},`
`{"B", "E", "S", "I", "G", "Q"},`
`{"T", "Z", "O", "R", "Y", "N"},`
`{"H", "M", "J", "D", "P", "F"}};`
`DiceyWords dw = new DiceyWords(dice);`	

`ArrayList<String> moreWords = new ArrayList<String>();`
`moreWords.add("BATH");`
`moreWords.add("MAZE");`
`moreWords.add("MEOW");`
`moreWords.add("WHEY");`
`moreWords.add("JERK");`
`moreWords.add("PYIC");`
`moreWords.add("RASH");`
`moreWords.add("BARD");`
`moreWords.add("JUST");`
`moreWords.add("MINX");`
 	
`String letters = dw.unusedLetters(moreWords);`	

`letters.length();`	`3`
`"FGQ".indexOf(letters.substring(0,1)) >= 0;`	`true`
`"FGQ".indexOf(letters.substring(1,2)) >= 0;`	`true`
`"FGQ".indexOf(letters.substring(2,3)) >= 0;`	`true`
"FGQ" are the only letters not used by any word in `words` on the four die **Yes, this implies `letters` is a permutation of the letters: "FGQ"**	

`moreWords.add("FGQT");`
 	
`letters = dw.unusedLetters(moreWords);`	

`letters.length();`	`0`
Adding a word that requires the letters "FGQ" to `words` will require all letters on the four die be used	