Bongard

First the motivation:

In his book, Bongard presented “zadachnik” — a set of 100 problems, which are now known as “Bongard problems.” The problems were meant to be solved by a computer program. No such program exists, as far as we know. But people are pretty good at solving Bongard problems and usually have fun doing so. A “Bongard problem” presents 12 small pictures that include geometric shapes. The pictures are split into two groups of six; the task is to figure out what makes the pictures in the left group different from pictures in the right group. Figure 02-1 shows the first three very easy problems from Bongard’s “zadachnik”: (1) empty and non-empty; (2) big and small; (3) white and black.

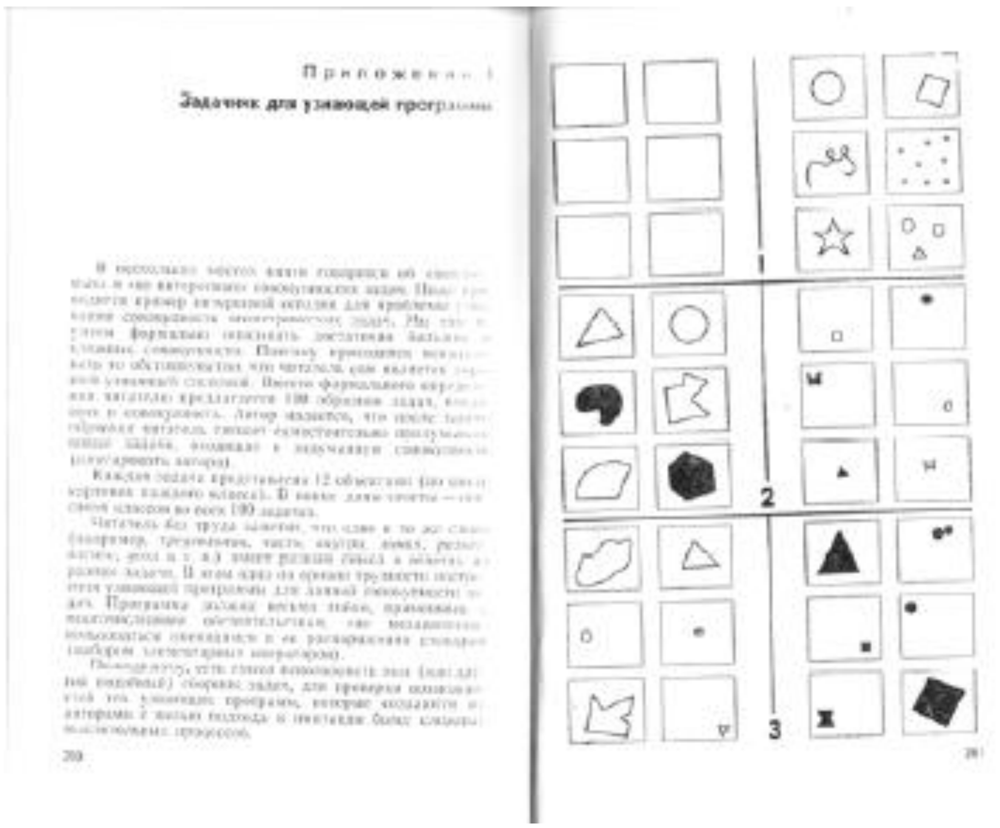

Figure 02-1

Special note: All Strings will contain only UPPER CASE letters

You will implement the constructor and eight methods in the `Bongard` class. The constructor has two `String[]` parameters. One `String[]` contains the `String`s in the left group, the other `String[]` contains the `String`s in the right group. The eight methods are `getNumTallLetters(String word)`, `getNumVowels(String word)`, `getNumVowelsWithY(String word)`, `endsWith(String word)`, `startsWith(String word)`, `getPlusMinus(String word)`, `getLength(String word)`, and `whichBox(String word)`.

The `getNumTallLetters(String word)` is a static method that returns the number of tall letters contained in the `String` parameter word. Tall letters are: BDFHKLT
The following code shows the results of the `getNumTallLetters` method.

The following code	Returns
`Bongard.getNumTallLetters("BIOETHIC");`	3
`Bongard.getNumTallLetters("BALLET");`	4

The `getNumVowels(String word)` is a static method that returns the number of vowels in the parameter `word`. Only the letters A, E, I, O and U are consider vowels in this method.

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

The following code	Returns
`Bongard.getNumVowels("GULLY")`	1
`Bongard.getNumVowels("ASTRONOMY")`	3

The `getNumVowelsWithY(String word)` is a static method that returns the number of vowels in the parameter `word`. In addition to the letters A, E, I, O and U being vowels, the letter Y is also to be counted as a vowel.

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

The following code	Returns
`Bongard.getNumVowelsWithY("GULLY")`	2
`Bongard.getNumVowelsWithY("ASTRONOMY")`	4

The `endsWith(String word)` is a static method that returns the last three letters of the parameter `word`. If `word.length() < 3`, return the entire string `word`.

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

The following code	Returns
`Bongard.endsWith("COURT")`	“URT”
`Bongard.endsWith("AT")`	“AT”

The `startsWith(String word)` is a static method that returns the first three letters of the parameter `word`. If `word.length() < 3`, return the entire string `word`.

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

The following code	Returns
`Bongard.startsWith("COMET")`	“COM”
`Bongard.startsWith("AT")`	“AT”

The `getPlusMinus(String word)` is a static method which compares the number of vowels (A, E, I, O, and U) and the number of consonants in the parameter `word`. (Note that the letter Y counts as a consonant.) This method returns:

- A negative number if the parameter `word` contains **more** consonants than vowels.
- A positive number if the parameter `word` contains **fewer** consonants than vowels.
- Zero if the parameter `word` contains the same number of consonants and vowels.

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

The following code	Returns
`Bongard.getPlusMinus("PROGRAMMER") < 0;`	true
`Bongard.getPlusMinus("EMAIL") > 0;`	true
`Bongard.getPlusMinus("FUTURE") == 0;`	true

The `getLength(String word)` is a static method that returns the length (number of letters) in the parameter `word`. Remember, all Strings will contain only UPPER CASE letters.

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

The following code	Returns
`Bongard.getLength("FUTURE");`	6
`Bongard.getLength("AT");`	2

The final method to complete is the `whichBox(String word)` method which returns a String. This method looks at the two `String[]` passed to the **constructor**. All the Strings in the `leftWords` will have the same return value for a subset of the previous eight methods. And all the Strings in the `rightWords` will have the same return value for a subset of the previous eight methods.

In the following example, the left words `“BATH”` and `“THAT”` have the following 5 similar properties:

- `getNumTallLetters(“BATH”) == getNumTallLetters(“THAT”)`
- `getNumVowels(“BATH”) == getNumVowels(“THAT”)`
- `getNumVowelsWithY(“BATH”) == getNumVowelsWithY(“THAT”)`
- `getPlusMinus(“BATH”) < 0 &&  getPlusMinus(“THAT”) < 0`
- `getLength(“BATH”) == getLength(“THAT”)`

Note: Any word added to the left words must have the same value for these five properties. The value of the remaining three properties does NOT matter.
Since the word `“HALT”` has the same five properties, the word `“HALT”` belongs in the left box and `whichBox` should return “LEFT”.

In the same example, the right words `“ARID”` and `“HAIR”` have the following 5 similar properties:

- `getNumVowels(“ARID”) == getNumVowels(“HAIR”)`
- `getNumVowelsWithY(“ARID”) == getNumVowelsWithY(“HAIR”)`
- `getPlusMinus(“BATH”) == getPlusMinus(“THAT”) == 0`
- `getLength(“BATH”) == getLength(“THAT”)`

Note: Any word added to the right words must have the same value for these four properties. The value of the remaining four properties does NOT matter.
Since the word `“LOVE”` has the same four properties, the word `“LOVE”` belongs in the right box and `whichBox` should return “RIGHT”.

Since the word `“ANGEL”` does not have the same five properties as the left words and does not have the same four properties as the right words, the word `“ANGEL”` belongs in neither the left words nor the right words and `whichBox` should return “`NEITHER`”.

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

The following code	Returns

`String[] left1 = {"BATH", "THAT"};`
`String[] right1 = {"ARID", "HAIR"};`
`Bongard bon1 = new Bongard(left1, right1);`	

`Bon1.whichBox("HALT");`	`"LEFT"`
`Bon1.whichBox("LOVE");`	`"RIGHT"`
`Bon1.whichBox("ANGEL")`	`"NEITHER"`

One more example of the method `WhichBox(String word)` follows on next page:

In the following example, the left words "LEAVE", "ALONE", "HOUSE", "ABOVE", and "MEDIA" have the same value for length (five), number of Vowels (three) and tall letters (one). Therefore, any word with five letters, three vowels and one tall letter belongs with the left words. "RADIO" has length five, three vowels and one tall letter and belongs with the left words.

In the same example, the right words "FAMILY", "GALAXY", "HOCKEY", "MONKEY", "MONEY", "GOODY" have the same value for number of Vowels (two) and number of Vowels including Y (three). Therefore, any word WITH two vowels and three vowels if you count Y's belong with the right words. "NOWAY" has two vowels and three vowels if you count Y and belongs with the right words.

Since the word "ANIMAL" does not have the same properties as the left words and does not have the same properties as the right words, the word "ANIMAL" belongs in neither the left words nor the right words.

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

The following code	Returns

`String[] left2 = {"LEAVE", "ALONE", "HOUSE", "ABOVE", "MEDIA"};`
 	
`String[] right2 = {"FAMILY", "GALAXY", "HOCKEY", "MONKEY", "MONEY", "GOODY"};`
 	
`Bongard bon2 = new Bongard(left2, right2);`	

`Bon1.whichBox("RADIO");`	"LEFT"
`Bon1.whichBox("NOWAY");`	"RIGHT"
`Bon1.whichBox("ANIMAL")`	"NEITHER"