1 2 3

Task Java

Your computer's hard drive is almost full. In order to make some space, you need to compile some file statistics. You want to know how many bytes of memory each file type is consuming. Each file has a name, and the part of the name after the last dot is called the file extension, which identifies what type of file it is. We distinguish four broad types of file:

- music (only extensions: `mp3`, `aac`, `flac`)
- image (only extensions: `jpg`, `bmp`, `gif`)
- movie (only extensions: `mp4`, `avi`, `mkv`)
- other (all other extensions; for example: `7z`, `txt`, `zip`)

You receive string S, containing a list of all the files on your computer (each file appears on a separate line). Each line contains a file name and the file's size in bytes, separated by a space. For example, string S could look like:

```
"my.song.mp3 11b
greatSong.flac 1000b
not3.txt 5b
video.mp4 200b
game.exe 100b
mov!e.mkv 10000b"
```

There are two music files (`my.song.mp3` and `greatSong.flac`, of size 11 and 1000 bytes respectively). There are no images files. We have two movies files (`video.mp4` and `mov!e.mkv` of size 200 and 10000 bytes). There are two files of other types (`not3.txt` and `game.exe` of size 5 and 100 bytes). In total there are 1011 bytes of music, 0 bytes of images, 10200 bytes of movies and 105 bytes of other files.

Write a function:

```
class Solution { public String solution(String S); }
```

that, given string S describing the files on disk, returns a string containing four rows, describing music, images, movies and other file types respectively. Each row should consist of a file type and the number of bytes consumed by files of that type on the disk (use format "<<type>> <<size>>b", where <<type>> is the files' type and <<size>> is the total file size of this group).

For instance, given string S as shown above, your function should return:

```
"music 1011b
images 0b
movies 10200b
other 105b"
```

as described above.

Assume that:

- string S contains at most 500 lines; there are no empty lines; each line contains exactly one space, separating the file name from its size;
- each file described in S has a non-empty file name of at most 30 characters' length; the file name includes a file extension as its final few characters, from the character after the last dot to the end of the file name; the file extension is never empty and it consists only of lower-case English letters (`a-z`) and digits (`0-9`); the file name can't consist of an extension alone;
- the file name consists only of lower-case English letters (`a-z`), upper-case English letters (`A-Z`), digits (`0-9`) and special characters `"^&'@{}[],$=!-#()%.+~_"` (without quotes);
- the size of each file described in S is a positive integer, less than or equal to 1,000,000 bytes.

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Solution Java SE 8

```
// you can also use imports, for example:
// import java.util.*;

// you can write to stdout for debugging purposes, e.g.
// System.out.println("this is a debug message");

class Solution {
    public String solution(String S) {
        // write your code in Java SE 8
    }
}
```

You will see save status here

Test Output Run Tests

Custom test cases (0 of 10) + Add