Problem: Red, White and Blue

Level: Medium

Note: This solution uses Enums and Exceptions. While optional, it always looks good if you use them in an interview.

You're given a list of Marbles. Each marble can have one of 3 colors (Red, White or Blue). Arrange the marbles in order of the colors (Red -> White -> Blue).

Colors are represented as follows:

0 - Red
1 - White
2 - Blue

Questions to Clarify:
Q. What to do if the input has an unknown color?
A. Throw an Exception

Q. Do I need to move around the existing elements of the array, or can I count the elements (for example, 4 zeros) and then fill them up in a new array?
A. Yes, you need to move elements around. A marble is represented as an integer in this problem, but in reality there might be more information about a marble in each element.

Solution:
This is very similar to the Dutch National Flag problem. You will use the same solution, with 2 pointers - one tracking *low_boundary* and one tracking *high_boundary*.
Only white marbles will be left in the middle.

Pseudocode:

```
(Note: Never write pseudocode in an actual interview. Unless you're writing a
few lines quickly to plan out your solution. Your actual solution should be in
a real language and use good syntax.)

low_boundary = 0, high_boundary = a.length - 1
i = 0

while i <= high_boundary
     if a[i].color is RED:
         add a[i] to low boundary and expand boundary by 1
         i++
     else if a[i].color is BLUE:
         add a[i] to high boundary and expand boundary by 1
         don't increment i because the current i needs to be processed
                as it came from ahead
```

```
    else if a[i].color is WHITE:
        i++
    else: // unknown color
        throw exception
```

Test Cases:
Edge Cases: empty array, null array, invalid color
Base Cases: single element, two elements
Regular Cases: list has element with - all 3 colors, only 2 colors, only 1 color

Time Complexity: O(n)

Space Complexity: O(1)

```
public static void redWhiteBlue(Marble[] a) {
    if (a == null)
        return;

    int low_boundary = 0, high_boundary = a.length - 1;
    int i = 0;

    while (i <= high_boundary) {
        Color color = a[i].getColor();
        if (color == Color.RED) {
            Utils.swap(a, i, low_boundary);
            low_boundary++;
            i++;
        } else if (color == Color.BLUE) {
            Utils.swap(a, i, high_boundary);
            high_boundary--;
        } else if (color == Color.WHITE){
            i++;
        } else {
            throw new IllegalArgumentException("Unknown Color: " + color);
        }
    }
}

/***** Helper Code *****/
/* Ask the interviewer if you need to implement this. */

public static enum Color {
    RED(0),
    WHITE(1),
    BLUE(2);

    private final int colorId;
```

```
    Color(int colorId) {
        this.colorId = colorId;
    }

    public int getValue() {
        return colorId;
    }
}

public static class Marble {
    Color color;
    int data;

    public Marble(Color color) {
        super();
        this.color = color;
        this.data = 0;
    }

    public Marble(Color color, int data) {
        super();
        this.color = color;
        this.data = data;
    }

    public Color getColor() {
        return color;
    }

    public int getData() {
        return data;
    }

    public String toString() {
        return color.toString() + ": " + data;
    }
}
```