Super Mario Brothers Sprite

Scenario

Once again, you find yourself working in the video game industry. You've managed to find yourself working on media, specifically, you need to create classes to handle drawing a character on the screen. One big problem, though, is that there are a lot of different ways the character can appear! The character starts out short, but when he gets a mushroom, becomes tall. Then, the character may also get a flower, and change into a white and red costume. Finally, regardless of size and costume, if the character gets a star, then he flashes for a short while.

Problem

A quick look at the combinations shows that, if you aren't careful, you'll be creating about 12 subclasses using inheritance for this game, and with the game designer's love of costumes, this could easily explode in future games. Luckily, you notice that each possible character is really just some transform of the base, short character: either the character gets tall, or changes outfit colors, or flashes. In addition, different costumes can add behavior to the character. The flower allows the character to throw fireballs, and the star makes the character invincible.

Your task is to efficiently create a set of classes which provide the various different image combinations, implement a *draw()* method, and provide the *throwFireball()* method if a flower is obtained, or *beInvincible()* method if a star is obtained. For practical purposes, you do not need to actually draw images, it is sufficient to output a description of the character to the console. The messages should be:

- Short: "It'sa me!"
- Tall: "I'm tall!"
- Flower: "And I can throw fireballs!"
- Star: "Wow! I'm invincible!"

For example, if the character is tall and has the flower, then *draw* should result in "It'sa me! I'm tall! And I can throw fireballs!" to be printed to the screen. A simulation class is provided, which will invoke a *giveMushroom()*, *giveFlower()* and *giveStar()* method. Your object should return some new, enhanced object when these methods are called.

Deliverables

1. Identify the design pattern you used to solve this problem, and the participants (i.e., the roles each class takes).
2. An implementation in a language of your choice.
3. A class diagram of your solution (including existing classes), so future developers can easily see how to work with your solution.