

```
package genericity;

/**
 * @convention length > 0
 * @correspondence s = length <br />
 *                  c = color
 */
public class LeadedPencil implements Pencil {

    /**
     * Default value used to initialize length if
     * constructor is not called with a positive value.
     */
    private static final int DEFAULT_LENGTH = 10;

    /**
     * Color of the lead in this pencil. The lead can always be
     * replaced with different color.
     */
    private Colors color;

    /**
     * The length of the pencil. This length is always positive.
     */
    private int length;

    /**
     * @param color the initial lead color
     * @param length the initial length of the pencil
     * @ensures this.color = color <br />
     * length > 0 ==> this.length = length <br />
     * length <= 0 ==> this.length = DEFAULT_LENGTH
     */
    public LeadedPencil(Colors color, int length) {
        this.color = color;
        this.length = DEFAULT_LENGTH;
        if (length > 0) {
            this.length = length;
        }
    }

    /**
     * @see Pencil#toString()
     */
    public String toString() {
        return (this.length + ": " + this.color);
    }

    /**
     * @param newColor is a valid color
     * @alters this.color
     * @ensures this.color = newColor
     * @see genericity.Pencil#setColor(genericity.Colors)
     */
    public void setColor(Colors newColor) {
        this.color = newColor;
    }

    /**
     * @param remove amount by which pencil will be shortened
     * @requires remove >= 0
     * @alters this.length
     * @ensures remove < #length ==> length + remove = #length
     * @see genericity.Pencil#sharpen(int)
     */
}
```

```
    */  
    public void sharpen(int remove) {  
        assert (remove >= 0);  
        if (remove < this.length) {  
            this.length -= remove;  
        }  
    }  
}
```