Homework 7, Structural Patterns

Marvin Sevilla

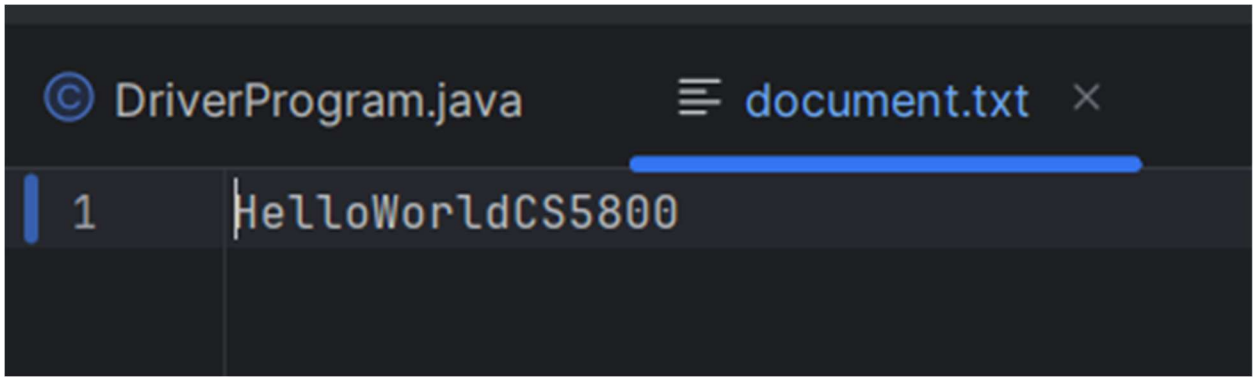
CS 4800.01, Software Engineering

GitHub Link: https://github.com/LoloMarty/2024-/tree/main/CS4800/Homework7

```
HelloWorldCS5800

Char: H
Font: Arial
Color: Red
Size: 12

Char: e
Font: Calibri
Color: Blue
Size: 14

Char: l
Font: Verdana
Color: Black
Size: 16

Char: l
Font: Roboto
Color: White
Size: 12

Char: o
Font: Arial
Color: Red
Size: 12

Char: W
Font: Arial
Color: Red
Size: 12

Char: o
Font: Calibri
Color: Blue
Size: 14
```

```
Char: r
Font: Verdana
Color: Black
Size: 16

Char: l
Font: Roboto
Color: White
Size: 12

Char: d
Font: Arial
Color: Red
Size: 12

Char: C
Font: Arial
Color: Red
Size: 12

Char: S
Font: Calibri
Color: Blue
Size: 14

Char: 5
Font: Verdana
Color: Black
Size: 16

Char: 8
Font: Roboto
Color: White
Size: 12

Char: 0
Font: Arial
Color: Red
Size: 12
```

```
Char: 0
Font: Arial
Color: Red
Size: 12


Process finished with exit code 0
```



Part 1 Source Code

```java
import org.junit.Test;

import static org.junit.Assert.*;

public class CharacterAttributesFactoryTest {

    @Test
    public void getCharacterProperties() {
        String font = "Arial";
        String color = "black";
        int size = 12;

        CharacterAttributes attributes1 = CharacterAttributesFactory.getCharacterProperties(font, color,
        CharacterAttributes attributes2 = CharacterAttributesFactory.getCharacterProperties(font, color,

        assertNotNull(attributes1);
        assertNotNull(attributes2);
        assertEquals(attributes1, attributes2);
    }
}
```

```java
import org.junit.Test;

import static org.junit.Assert.*;

public class CharacterAttributesTest {

    @Test
    public void getFont() {
        String expectedFont = "Arial";
        String color = "Black";
        int size = 12;
        CharacterAttributes attributes = new CharacterAttributes(expectedFont, color, size);

        String actualFont = attributes.getFont();

        assertEquals(expectedFont, actualFont);
    }

    @Test
    public void getColor() {
        CharacterAttributes attributes = new CharacterAttributes("Arial", "Red", 12);

        String expectedColor = "Red";
        String actualColor = attributes.getColor();
        assertEquals(expectedColor, actualColor);
    }

    @Test
    public void getSize() {
        CharacterAttributes attributes = new CharacterAttributes("Arial", "Black", 12);

        int expectedSize = 12;
        int actualSize = attributes.getSize();
        assertEquals(expectedSize, actualSize);
    }
}
```

```
interface CharacterInterface {
        void apply();
}
```

```java
import org.junit.Test;

import static org.junit.Assert.*;

public class CharacterTest {

    @Test
    public void setAttributes() {
        String initialFont = "Arial";
        String initialColor = "Red";
        int initialSize = 12;
        CharacterAttributes initialAttributes = new CharacterAttributes(initialFont, initialColor, initi
        Character character = new Character("A", initialFont, initialColor, initialSize);

        String newFont = "Calibri";
        String newColor = "Blue";
        int newSize = 14;
        CharacterAttributes newAttributes = new CharacterAttributes(newFont, newColor, newSize);

        character.setAttributes(newAttributes);

        assertEquals(newFont, character.getAttributes().getFont());
        assertEquals(newColor, character.getAttributes().getColor());
        assertEquals(newSize, character.getAttributes().getSize());
    }
}
```

```java
import java.util.ArrayList;
import java.io.FileWriter;
import java.io.IOException;
import java.io.FileReader;
import java.io.BufferedReader;

public class CharString {
    ArrayList<Character> string;
    String fileName;

    public CharString() {
        string = new ArrayList<Character>();
        fileName = "document.txt"; // Name of the file to write to
    }

    public void save(String givenCharacter, String givenFont, String givenColor, int givenSize) {
        string.add(new Character(givenCharacter, givenFont, givenColor, givenSize));

        try {
            FileWriter writer = new FileWriter(fileName);
            writer.write(this.buildString());
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public String buildString() {
        String builtString = "";
        for (Character character : this.string) {
            builtString += character.getHeldCharacter();
        }

        return builtString;
    }

    public void load() {
        try {
            FileReader reader = new FileReader(fileName);
            BufferedReader bufferedReader = new BufferedReader(reader);

            String line;
            while ((line = bufferedReader.readLine()) != null) {
                System.out.println("\n" + line + "\n");
            }

            bufferedReader.close();
        } catch (IOException e) {
            System.out.println("An error occurred while reading the file.");
            e.printStackTrace();
        }

        for (Character character : this.string) {
```

```
54              System.out.printf("Char: %s\n", character.getHeldCharacter());
55              character.printCharacaterAttributes();
56              System.out.println("\n");
57          }
58      }
59  }
```

```java
import org.junit.Test;
import static org.junit.Assert.*;
import java.io.File;

public class CharStringTest {

    @Test
    public void save() {
        CharString charString = new CharString();

        charString.save("A", "Arial", "Black", 12);

        File file = new File("document.txt");
        assertTrue(file.exists());

        String expectedContent = "A";
        assertEquals(expectedContent, charString.buildString());
    }

    @Test
    public void buildString() {
        CharString charString = new CharString();
        charString.save("A", "Arial", "Black", 12);
        charString.save("B", "Times New Roman", "Red", 14);

        assertEquals("AB", charString.buildString());
    }
}
```

```java
public class Disk {
    static CharString document;

    public static CharString getDocument() {
        if (document == null) {
            document = new CharString();
        }

        return document;
    }
}
```

```java
import org.junit.Test;

import static org.junit.Assert.*;

public class DiskTest {

    @Test
    public void getDocument() {
        CharString expected = Disk.getDocument();
        CharString actual = Disk.getDocument();

        assertNotNull(actual);
        assertEquals(expected, actual);
    }
}
```

```java
public class DriverProgram {
    public static void main(String[] args) {
        CharString document = Disk.getDocument();

        document.save("H", "Arial", "Red", 12);
        document.save("e", "Calibri", "Blue", 14);
        document.save("l", "Verdana", "Black", 16);
        document.save("l", "Roboto", "White", 12);
        document.save("o", "Arial", "Red", 12);
        document.save("W", "Arial", "Red", 12);
        document.save("o", "Calibri", "Blue", 14);
        document.save("r", "Verdana", "Black", 16);
        document.save("l", "Roboto", "White", 12);
        document.save("d", "Arial", "Red", 12);
        document.save("C", "Arial", "Red", 12);
        document.save("S", "Calibri", "Blue", 14);
        document.save("5", "Verdana", "Black", 16);
        document.save("8", "Roboto", "White", 12);
        document.save("0", "Arial", "Red", 12);
        document.save("0", "Arial", "Red", 12);

        document.load();

    }
}
```

```
1  public class Character {
2      String heldCharacter;
3      CharacterAttributes attributes;
4
5      public Character(String givenCharacter, String givenFont, String givenColor, int givenSize) {
6          this.heldCharacter = givenCharacter;
7
8          CharacterAttributes fetchedAttributes = CharacterAttributesFactory.getCharacterProperties(givenF
9                  givenColor, givenSize);
10
11         if (fetchedAttributes != null) {
12             this.attributes = fetchedAttributes;
13         } else {
14             this.attributes = new CharacterAttributes(givenFont, givenColor, givenSize);
15         }
16     }
17
18     public void printCharacaterAttributes() {
19         this.attributes.apply();
20     }
21
22     public CharacterAttributes getAttributes() {
23         return attributes;
24     }
25
26     public void setAttributes(CharacterAttributes attributes) {
27         this.attributes = attributes;
28     }
29
30     public String getHeldCharacter() {
31         return heldCharacter;
32     }
33 }
```

```java
public class CharacterAttributes implements CharacterInterface {
    private final String font;
    private final String color;
    private final int size;

    public CharacterAttributes(String givenFont, String givenColor, int givenSize) {
        this.font = givenFont;
        this.color = givenColor;
        this.size = givenSize;
    }

    public void apply() {
        System.out.printf("Font: %s\nColor: %s\nSize: %d", this.font, this.color, this.size);
    }

    public String getFont() {
        return font;
    }

    public String getColor() {
        return color;
    }

    public int getSize() {
        return size;
    }
}
```

```
public class DriverProgram {
    public static void main(String[] args) {
        CharString document = Disk.getDocument();

        document.save("H", "Arial", "Red", 12);
        document.save("e", "Calibri", "Blue", 14);
        document.save("l", "Verdana", "Black", 16);
        document.save("l", "Roboto", "White", 12);
        document.save("o", "Arial", "Red", 12);
        document.save("W", "Arial", "Red", 12);
        document.save("o", "Calibri", "Blue", 14);
        document.save("r", "Verdana", "Black", 16);
        document.save("l", "Roboto", "White", 12);
        document.save("d", "Arial", "Red", 12);
        document.save("C", "Arial", "Red", 12);
        document.save("S", "Calibri", "Blue", 14);
        document.save("5", "Verdana", "Black", 16);
        document.save("8", "Roboto", "White", 12);
        document.save("0", "Arial", "Red", 12);
        document.save("0", "Arial", "Red", 12);

        document.loadimport java.util.HashMap;
import java.util.Map;

public class CharacterAttributesFactory {
    private static Map<String, CharacterAttributes> propertiesMap = new HashMap<>();

    public static CharacterAttributes getCharacterProperties(String font, String color, int size) {
        String key = font + color + size;
        if (!propertiesMap.containsKey(key)) {
            propertiesMap.put(key, new CharacterAttributes(font, color, size));
        }
        return propertiesMap.get(key);
    }

}
();

    }
}
```