CS5800 Homework 07

Subham Panda

017314921

GitHub link

Q1:

Character.java

```java
package Q1;
import java.io.Serializable;


public class Character implements Serializable {
    private char character;
    private CharacterProperties properties;

    public Character(char character, CharacterProperties properties) {
        this.character = character;
        this.properties = properties;
    }

    public char getCharacter() {
        return character;
    }

    public CharacterProperties getProperties() {
        return properties;
    }
}
```

CharacterProperties.java

```java
package Q1;

import java.io.Serializable;

public class CharacterProperties implements Serializable {
    private String font;
    private String color;
    private int size;

    public CharacterProperties(String font, String color, int size) {
        this.font = font;
        this.color = color;
        this.size = size;
    }
    public String getFont() {
        return font;
    }

    public String getColor() {
        return color;
    }

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

Document.java

```
package Q1;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class Document {
    private List<Character> characters = new ArrayList<>();

    public void addCharacter(char character, CharacterProperties properties)
{
        characters.add(new Character(character, properties));
    }

    public void save(String filename) throws IOException {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filename))) {
            for (Character character : characters) {
                writer.write("Character: " + character.getCharacter() +
                        ", Font: " + character.getProperties().getFont() +
                        ", Color: " + character.getProperties().getColor() +
                        ", Size: " + character.getProperties().getSize() +
"\n");
            }
        }
    }

    public static Document load(String filename) throws IOException {
        Document doc = new Document();
        try (BufferedReader reader = new BufferedReader(new
FileReader(filename))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split(",");
                char character = parts[0].split(":")[1].trim().charAt(0);
                String font = parts[1].split(":")[1].trim();
                String color = parts[2].split(":")[1].trim();
                int size = Integer.parseInt(parts[3].split(":")[1].trim());
                doc.addCharacter(character, new CharacterProperties(font,
color, size));
            }
        }
        return doc;
    }
    public List<Character> getCharacters() {
        return characters;
    }
}
```

Driver.java

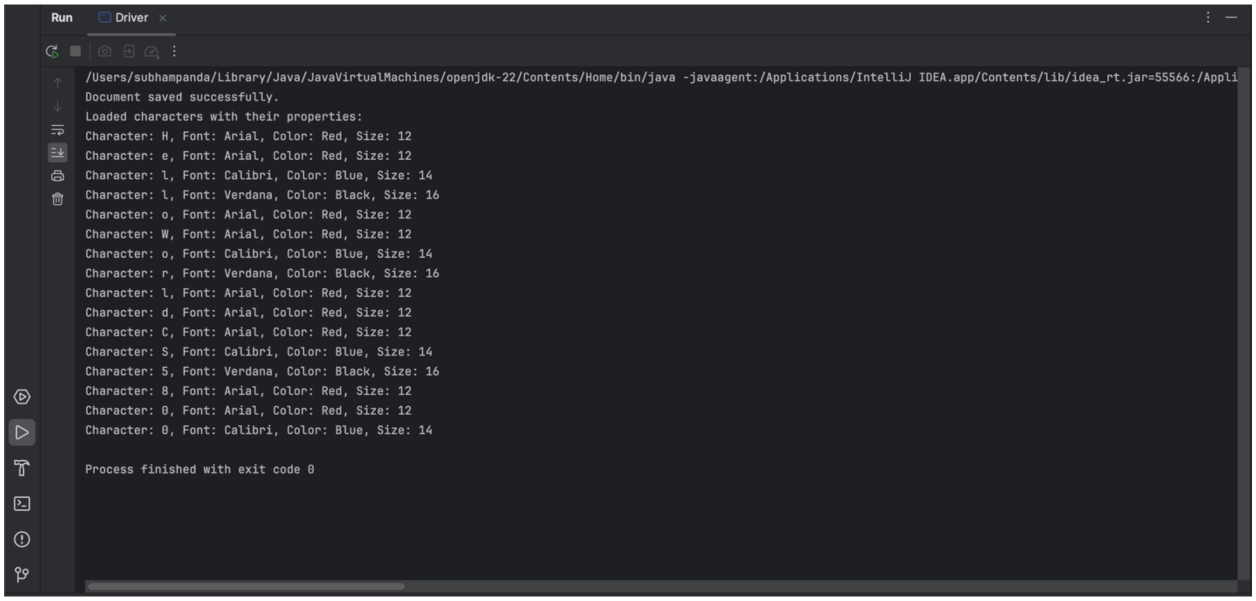
```
package Q1;

import java.io.IOException;

public class Driver {
    public static void main(String[] args) {
        Document document = new Document();
        document.addCharacter('H',
FlyweightFactory.getCharacterProperties("Arial", "Red", 12));
        document.addCharacter('e',
FlyweightFactory.getCharacterProperties("Arial", "Red", 12));
        document.addCharacter('l',
FlyweightFactory.getCharacterProperties("Calibri", "Blue", 14));
        document.addCharacter('l',
FlyweightFactory.getCharacterProperties("Verdana", "Black", 16));
        document.addCharacter('o',
FlyweightFactory.getCharacterProperties("Arial", "Red", 12));
        document.addCharacter('W',
FlyweightFactory.getCharacterProperties("Arial", "Red", 12));
        document.addCharacter('o',
FlyweightFactory.getCharacterProperties("Calibri", "Blue", 14));
        document.addCharacter('r',
FlyweightFactory.getCharacterProperties("Verdana", "Black", 16));
        document.addCharacter('l',
FlyweightFactory.getCharacterProperties("Arial", "Red", 12));
        document.addCharacter('d',
FlyweightFactory.getCharacterProperties("Arial", "Red", 12));
        document.addCharacter('C',
FlyweightFactory.getCharacterProperties("Arial", "Red", 12));
        document.addCharacter('S',
FlyweightFactory.getCharacterProperties("Calibri", "Blue", 14));
        document.addCharacter('5',
FlyweightFactory.getCharacterProperties("Verdana", "Black", 16));
        document.addCharacter('8',
FlyweightFactory.getCharacterProperties("Arial", "Red", 12));
        document.addCharacter('0',
FlyweightFactory.getCharacterProperties("Arial", "Red", 12));
        document.addCharacter('0',
FlyweightFactory.getCharacterProperties("Calibri", "Blue", 14));

        try {
            document.save("example_document.txt");
            System.out.println("Document saved successfully.");

            // Load the saved document
            Document loadedDocument = Document.load("example_document.txt");

            // Print loaded characters with their properties
            System.out.println("Loaded characters with their properties:");
```

```java
            for (Character character : loadedDocument.getCharacters()) {
                System.out.println("Character: " + character.getCharacter() +
                        ", Font: " + character.getProperties().getFont() +
                        ", Color: " + character.getProperties().getColor() +
                        ", Size: " + character.getProperties().getSize());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

FlyweightFactory.java

```java
package Q1;

import java.util.HashMap;

public class FlyweightFactory {
    private static HashMap<String, CharacterProperties>
characterPropertiesCache = new HashMap<>();

    public static CharacterProperties getCharacterProperties(String font,
String color, int size) {
        String key = font + "_" + color + "_" + size;
        if (!characterPropertiesCache.containsKey(key)) {
            characterPropertiesCache.put(key, new CharacterProperties(font,
color, size));
        }
        return characterPropertiesCache.get(key);
    }
}
```

Output:



```
/Users/subhampanda/Library/Java/JavaVirtualMachines/openjdk-22/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=55566:/Appli
Document saved successfully.
Loaded characters with their properties:
Character: H, Font: Arial, Color: Red, Size: 12
Character: e, Font: Arial, Color: Red, Size: 12
Character: l, Font: Calibri, Color: Blue, Size: 14
Character: l, Font: Verdana, Color: Black, Size: 16
Character: o, Font: Arial, Color: Red, Size: 12
Character: W, Font: Arial, Color: Red, Size: 12
Character: o, Font: Calibri, Color: Blue, Size: 14
Character: r, Font: Verdana, Color: Black, Size: 16
Character: l, Font: Arial, Color: Red, Size: 12
Character: d, Font: Arial, Color: Red, Size: 12
Character: C, Font: Arial, Color: Red, Size: 12
Character: S, Font: Calibri, Color: Blue, Size: 14
Character: 5, Font: Verdana, Color: Black, Size: 16
Character: 8, Font: Arial, Color: Red, Size: 12
Character: 0, Font: Arial, Color: Red, Size: 12
Character: 0, Font: Calibri, Color: Blue, Size: 14

Process finished with exit code 0
```

Tests:

CharacterPropertiesTest.java

```java
package Q1.tests;

import Q1.Character;
import Q1.CharacterProperties;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CharacterTest {
    @Test
    public void testGetCharacter() {
        CharacterProperties properties = new CharacterProperties("Arial",
"Red", 12);
        Character character = new Character('A', properties);
        assertEquals('A', character.getCharacter());
    }

    @Test
    public void testGetProperties() {
        CharacterProperties properties = new CharacterProperties("Arial",
"Red", 12);
        Character character = new Character('A', properties);
        assertEquals(properties, character.getProperties());
    }
}
```

CharacterTest.java

```java
package Q1.tests;

import Q1.Character;
import Q1.CharacterProperties;
import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CharacterTest {
    @Test
    public void testGetCharacter() {
        CharacterProperties properties = new CharacterProperties("Arial",
"Red", 12);
        Character character = new Character('A', properties);
        assertEquals('A', character.getCharacter());
    }

    @Test
    public void testGetProperties() {
        CharacterProperties properties = new CharacterProperties("Arial",
"Red", 12);
        Character character = new Character('A', properties);
        assertEquals(properties, character.getProperties());
    }
}
```

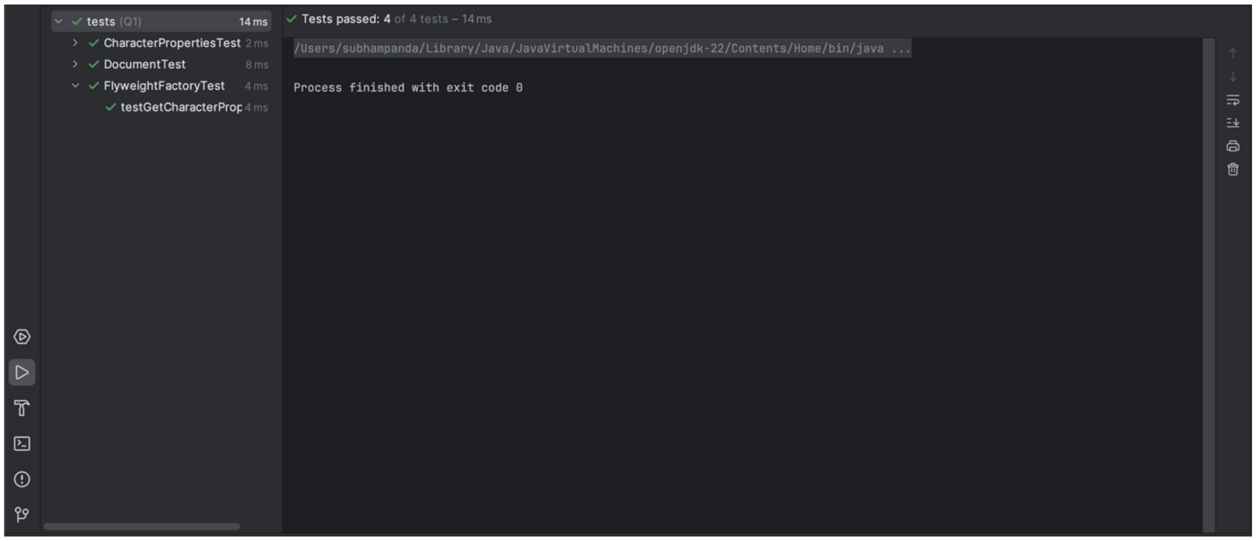
DocumentTest.java

```
package Q1.tests;

import Q1.Character;
import Q1.CharacterProperties;
import Q1.Document;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

import java.io.*;
import java.util.List;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;


public class DocumentTest {
    private static final String TEST_FILE = "test_document.txt";
    private Document document;

    @Before
    public void setUp() {
        document = new Document();
        document.addCharacter('A', new CharacterProperties("Arial", "Black",
12));
        document.addCharacter('B', new CharacterProperties("Times New Roman",
"Red", 14));
    }

    @Test
    public void testAddCharacter() {
        List<Character> characters = document.getCharacters();
        assertEquals(2, characters.size());
        assertEquals('A', characters.get(0).getCharacter());
        assertEquals("Arial", characters.get(0).getProperties().getFont());
        assertEquals(12, characters.get(0).getProperties().getSize());
        assertEquals('B', characters.get(1).getCharacter());
        assertEquals("Times New Roman",
characters.get(1).getProperties().getFont());
        assertEquals(14, characters.get(1).getProperties().getSize());
    }

    @Test
    public void testSaveAndLoad() throws IOException {
        document.save(TEST_FILE);
        Document loadedDoc = Document.load(TEST_FILE);
        List<Character> characters = loadedDoc.getCharacters();
        assertEquals(2, characters.size());
        assertEquals('A', characters.get(0).getCharacter());
        assertEquals("Arial", characters.get(0).getProperties().getFont());
        assertEquals(12, characters.get(0).getProperties().getSize());
        assertEquals('B', characters.get(1).getCharacter());
        assertEquals("Times New Roman",
characters.get(1).getProperties().getFont());
        assertEquals(14, characters.get(1).getProperties().getSize());
    }

    @After
    public void tearDown() {
        File file = new File(TEST_FILE);
        if (file.exists()) {
            assertTrue(file.delete());
        }
    }
}
```

FlyweightFactoryTest.java

```
package Q1.tests;

import Q1.CharacterProperties;
import Q1.FlyweightFactory;
import org.junit.Test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertSame;

public class FlyweightFactoryTest {
    @Test
    public void testGetCharacterProperties() {
        CharacterProperties properties1 =
FlyweightFactory.getCharacterProperties("Arial", "Red", 12);
        CharacterProperties properties2 =
FlyweightFactory.getCharacterProperties("Arial", "Red", 12);
        assertEquals(properties1, properties2);
        assertSame(properties1, properties2);
    }
}
```

Output:



tests (Q1) 14 ms
CharacterPropertiesTest 2 ms
DocumentTest 8 ms
FlyweightFactoryTest 4 ms
testGetCharacterProp 4 ms

```
Tests passed: 4 of 4 tests – 14 ms

/Users/subhampanda/Library/Java/JavaVirtualMachines/openjdk-22/Contents/Home/bin/java ...

Process finished with exit code 0
```

Q2:

```
RealSongService.java

package Q2;

import java.util.ArrayList;
import java.util.List;

public class RealSongService implements SongService {
    private List<Song> songs;

    public RealSongService() {
        // Initialize songs
        songs = new ArrayList<>();
        songs.add(new Song(1, "Tell me why", "Backstreet Boys", "Album 1",
180));
        songs.add(new Song(2, "Kaise Hua", "Sachet Tandon", "Kabir Singh",
200));
        songs.add(new Song(3, "Paper Cut", "Linkin Park", "Album 1", 220));
        songs.add(new Song(4, "Bekhayali", "Sachet Tandon", "Kabir Singh",
190));
        songs.add(new Song(5, "Kohinoor", "Sachet Tandon", "Album 1", 210));
    }

    @Override
    public Song searchById(Integer songID) {
        try {
            // Simulate delay
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        for (Song song : songs) {
            if (song.getSongID().equals(songID)) {
                return song;
            }
        }
        return null; // Return null if songID is not found
    }

    @Override
    public List<Song> searchByTitle(String title) {
        try {
            // Simulate delay
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        List<Song> result = new ArrayList<>();
        for (Song song : songs) {
            if (song.getTitle().equalsIgnoreCase(title)) {
                result.add(song);
            }
        }
        return result;
    }
```

```java
    @Override
    public List<Song> searchByAlbum(String album) {
        try {
            // Simulate delay
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        List<Song> result = new ArrayList<>();
        for (Song song : songs) {
            if (song.getAlbum().equalsIgnoreCase(album)) {
                result.add(song);
            }
        }
        return result;
    }
}
```

Song.java

```java
package Q2;

public class Song {
    private Integer songID;
    private String title;
    private String artist;
    private String album;
    private int duration;

    public Song(Integer songID, String title, String artist, String album,
int duration) {
        this.songID = songID;
        this.title = title;
        this.artist = artist;
        this.album = album;
        this.duration = duration;
    }

    // Getters for song metadata
    public Integer getSongID() {
        return songID;
    }

    public String getTitle() {
        return title;
    }

    public String getArtist() {
        return artist;
    }

    public String getAlbum() {
        return album;
    }
    public int getDuration() {
        return duration;
    }
}
```

SongService.java

```java
package Q2;

import java.util.List;

public interface SongService {
    Song searchById(Integer songID);
    List<Song> searchByTitle(String title);
    List<Song> searchByAlbum(String album);
}
```

SongServiceProxy.java

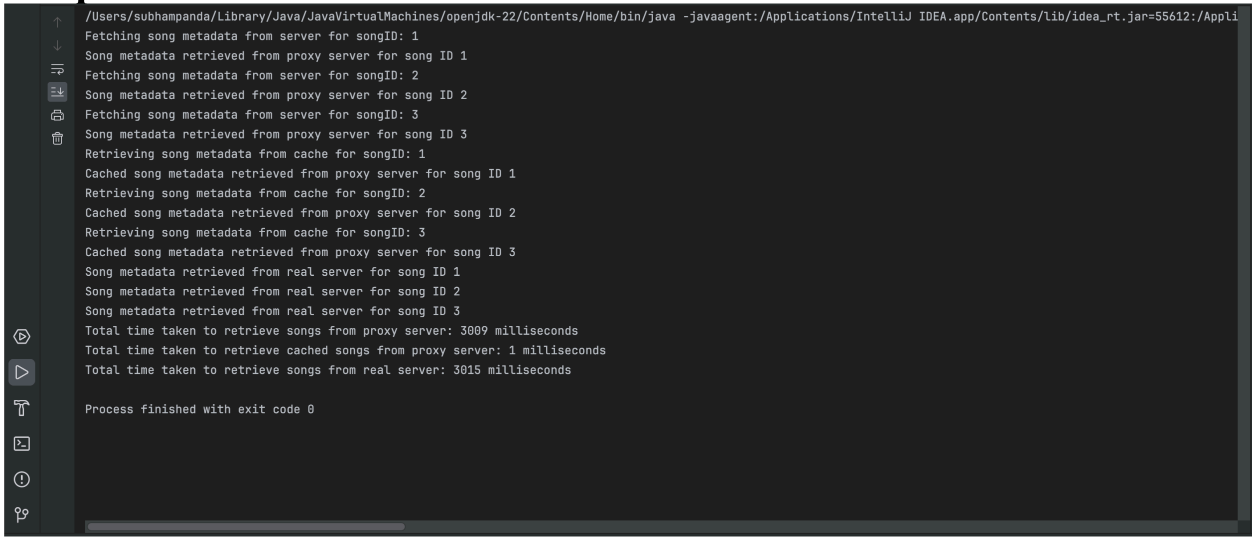
```java
package Q2;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class SongServiceProxy implements SongService {
    private SongService songService;
    private Map<Integer, Song> songCache;

    public SongServiceProxy(SongService songService) {
        this.songService = songService;
        this.songCache = new HashMap<>();
    }

    @Override
    public Song searchById(Integer songID) {
        if (songCache.containsKey(songID)) {
            System.out.println("Retrieving song metadata from cache for 
songID: " + songID);
            return songCache.get(songID);
        } else {
            System.out.println("Fetching song metadata from server for 
songID: " + songID);
            Song song = songService.searchById(songID);
            songCache.put(songID, song);
            return song;
        }
    }

    @Override
    public List<Song> searchByTitle(String title) {
        return songService.searchByTitle(title);
    }

    @Override
    public List<Song> searchByAlbum(String album) {
        return songService.searchByAlbum(album);
    }
}
```

Driver.java

```
package Q2;

import java.util.Arrays;
import java.util.List;

public class Driver {
    public static void main(String[] args) {
        // Simulating a music streaming application
        SongService realSongService = new RealSongService();
        SongService songServiceProxy = new SongServiceProxy(realSongService);

        // Song IDs to search for
        List<Integer> songIds = Arrays.asList(1, 2, 3);

        // Search for songs using the proxy server
        long startTimeProxyServer = System.currentTimeMillis();
        for (Integer songId : songIds) {
            Song songFromProxyServer = songServiceProxy.searchById(songId);
// This will fetch the song from the real server and cache it
            System.out.println("Song metadata retrieved from proxy server for
song ID " + songId);
        }
        long endTimeProxyServer = System.currentTimeMillis();
        long timeTakenProxyServer = endTimeProxyServer -
startTimeProxyServer;

        // Search for the same songs again using the proxy server to
demonstrate caching
        startTimeProxyServer = System.currentTimeMillis();
        for (Integer songId : songIds) {
            Song cachedSongFromProxyServer =
songServiceProxy.searchById(songId); // This should retrieve the song from
the cache
            System.out.println("Cached song metadata retrieved from proxy
server for song ID " + songId);
        }
        long cachedTimeTakenProxyServer = System.currentTimeMillis() -
startTimeProxyServer;

        // Search for songs using the real server
        long startTimeRealServer = System.currentTimeMillis();
        for (Integer songId : songIds) {
            Song songFromRealServer = realSongService.searchById(songId);
            System.out.println("Song metadata retrieved from real server for
song ID " + songId);
        }
        long endTimeRealServer = System.currentTimeMillis();
        long timeTakenRealServer = endTimeRealServer - startTimeRealServer;

        // Display results
```

```java
        System.out.println("Total time taken to retrieve songs from proxy
server: " + timeTakenProxyServer + " milliseconds");
        System.out.println("Total time taken to retrieve cached songs from
proxy server: " + cachedTimeTakenProxyServer + " milliseconds");
        System.out.println("Total time taken to retrieve songs from real
server: " + timeTakenRealServer + " milliseconds");
    }
}
```

Output:



```
/Users/subhampanda/Library/Java/JavaVirtualMachines/openjdk-22/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=55612:/Appli
Fetching song metadata from server for songID: 1
Song metadata retrieved from proxy server for song ID 1
Fetching song metadata from server for songID: 2
Song metadata retrieved from proxy server for song ID 2
Fetching song metadata from server for songID: 3
Song metadata retrieved from proxy server for song ID 3
Retrieving song metadata from cache for songID: 1
Cached song metadata retrieved from proxy server for song ID 1
Retrieving song metadata from cache for songID: 2
Cached song metadata retrieved from proxy server for song ID 2
Retrieving song metadata from cache for songID: 3
Cached song metadata retrieved from proxy server for song ID 3
Song metadata retrieved from real server for song ID 1
Song metadata retrieved from real server for song ID 2
Song metadata retrieved from real server for song ID 3
Total time taken to retrieve songs from proxy server: 3009 milliseconds
Total time taken to retrieve cached songs from proxy server: 1 milliseconds
Total time taken to retrieve songs from real server: 3015 milliseconds

Process finished with exit code 0
```

Tests:

RealSongService.java

```java
package Q2.tests;

import Q2.RealSongService;
import Q2.Song;
import org.junit.Test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;

public class RealSongServiceTest {

    @Test
    public void testSearchById() {
        RealSongService realSongService = new RealSongService();
        Song song = realSongService.searchById(1);
        assertNotNull(song);
        assertEquals("Tell me why", song.getTitle());
        assertEquals("Backstreet Boys", song.getArtist());
        assertEquals("Album 1", song.getAlbum());
        assertEquals(180, song.getDuration());
    }
}
```

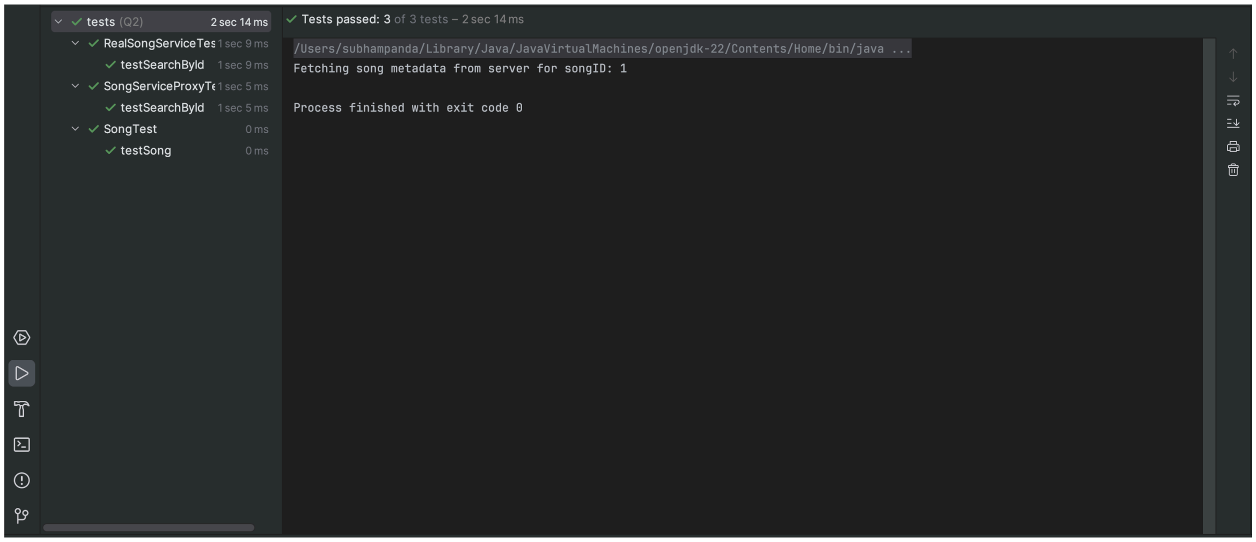
SongServiceProxyTest.java

```java
package Q2.tests;

import org.junit.Test;
import Q2.RealSongService;
import Q2.SongService;
import Q2.SongServiceProxy;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;

public class SongServiceProxyTest {

    @Test
    public void testSearchById() {
        SongService realSongService = new RealSongService();
        SongService songServiceProxy = new SongServiceProxy(realSongService);

        long startTime = System.currentTimeMillis();
        assertNotNull(songServiceProxy.searchById(1));
        long endTime = System.currentTimeMillis();
        long timeTakenProxyServer = endTime - startTime;

        // Assert that the proxy server responds faster than the real server
        assertEquals(1000, timeTakenProxyServer, 200);
    }
}
```

SongTest.java

```java
package Q2.tests;

import org.junit.Test;
import Q2.Song;

import static org.junit.Assert.assertEquals;

public class SongTest {

    @Test
    public void testSong() {
        Song song = new Song(1, "Song 1", "Artist 1", "Album 1", 180);
        assertEquals(1, (int) song.getSongID());
        assertEquals("Song 1", song.getTitle());
        assertEquals("Artist 1", song.getArtist());
        assertEquals("Album 1", song.getAlbum());
        assertEquals(180, song.getDuration());
    }
}
```

Output:


tests (Q2) 2 sec 14 ms
RealSongServiceTes 1 sec 9 ms
testSearchById 1 sec 9 ms
SongServiceProxyTe 1 sec 5 ms
testSearchById 1 sec 5 ms
SongTest 0 ms
testSong 0 ms
Tests passed: 3 of 3 tests – 2 sec 14 ms

```
/Users/subhampanda/Library/Java/JavaVirtualMachines/openjdk-22/Contents/Home/bin/java ...
Fetching song metadata from server for songID: 1

Process finished with exit code 0
```