

PES University
Department of Computer Science & Engineering

UE21CS352B : Object Oriented Analysis & Design using Java Lab

Week 8 : Java Multithreading & Serialization

NAME : NAGAVENI L G

SRN : PES2UG21CS315

SEC :6F

(a) Problem Statement: Word Frequency Counter using Multithreading in Java

You're tasked with creating a program that counts the frequency of words in a given text file. To make the process more efficient, you've to use multithreading to parallelize the counting process.

The program should contain the following:

1. Read a text file containing words.
2. Split the text into multiple chunks, each of which will be processed by a separate thread.
3. Use multithreading to count the frequency of words in each chunk concurrently.
4. Merge the word frequency counts from all threads to get the final word frequency count.
5. Output the word frequency count to the console or write it to another file.

Your program should consist of the following components:

1. A `WordFrequencyCounter` class with methods to read the text file, split it into chunks, and merge the word frequency counts.
2. A `WordFrequencyTask` class representing a single counting task. Each instance of `WordFrequencyTask` should be responsible for counting the frequency of words in a specific chunk of the text.
3. Use synchronization mechanisms such as locks or other thread-safe data structures to ensure that concurrent access to the word frequency counts is properly handled.

Your goal is to create a word frequency counter that can efficiently count the frequency of words in a large text file using multithreading.

Example output:

If your sample.txt looks like this:

[illegible]

This would be the required output:

```
This: 8
is: 32 an:
92
example: 4
```

Code:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.*;

class WordFrequencyTask implements Callable<Map<String, Integer>> {
    private final List<String> words;

    public WordFrequencyTask(List<String> words) {
        this.words = words;
    }

    @Override
    public Map<String, Integer> call() {
        Map<String, Integer> wordFrequency = new HashMap<>();

        for (String word : words) {
            wordFrequency.put(word, wordFrequency.getOrDefault(word, 0) + 1);
        }
    }
}
```

```

    }

    return wordFrequency;
}
}

class WordFrequencyCounter {
    private static final int THREAD_POOL_SIZE = 4;
    private final ExecutorService executorService;
    private final Map<String, Integer> globalWordFrequency;

    public WordFrequencyCounter() {
        this.executorService = Executors.newFixedThreadPool(THREAD_POOL_SIZE);
        this.globalWordFrequency = new ConcurrentHashMap<>();
    }

    public void processFile(String filePath) throws IOException,
        InterruptedException, ExecutionException {
        List<String> allWords = readWordsFromFile(filePath);
        List<List<String>> wordChunks = splitIntoChunks(allWords);

        List<Future<Map<String, Integer>>> futures = new ArrayList<>();

        for (List<String> chunk : wordChunks) {
            WordFrequencyTask task = new WordFrequencyTask(chunk);
            futures.add(executorService.submit(task));
        }

        for (Future<Map<String, Integer>> future : futures) {
            mergeWordFrequency(future.get());
        }

        executorService.shutdown();
        outputWordFrequency();
    }

    private List<String> readWordsFromFile(String filePath) throws IOException {
        List<String> words = new ArrayList<>();

        try (BufferedReader reader = new BufferedReader(new
            FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] lineWords = line.split("\\s+");
            }
        }
    }
}

```

```

        words.addAll(Arrays.asList(lineWords));
    }
}

return words;
}

private List<List<String>> splitIntoChunks(List<String> words) {
    int chunkSize = words.size() / THREAD_POOL_SIZE;
    List<List<String>> chunks = new ArrayList<>();

    for (int i = 0; i < words.size(); i += chunkSize) {
        int end = Math.min(i + chunkSize, words.size());
        chunks.add(words.subList(i, end));
    }

    return chunks;
}

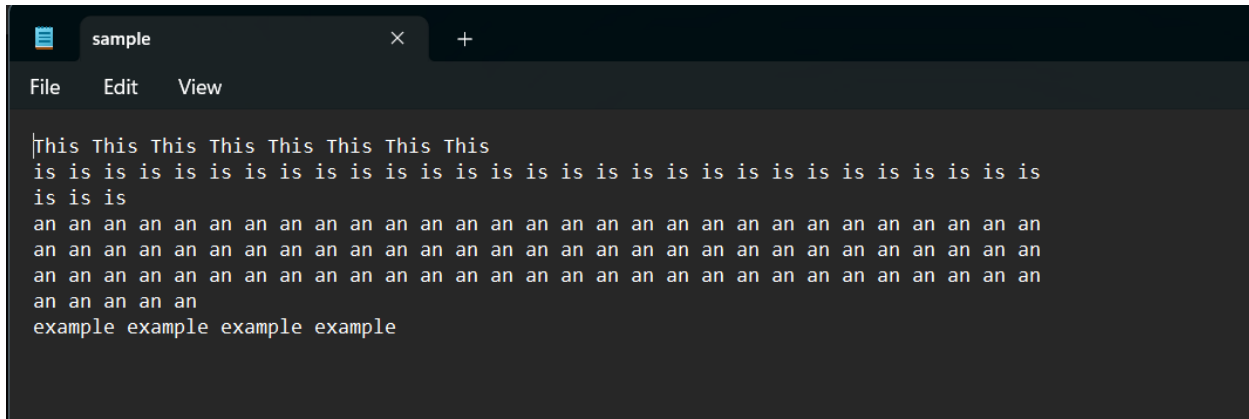
private void mergeWordFrequency(Map<String, Integer> localWordFrequency) {
    localWordFrequency.forEach((word, count) ->
globalWordFrequency.merge(word, count, Integer::sum));
}

private void outputWordFrequency() {
    globalWordFrequency.forEach((word, count) -> System.out.println(word +
": " + count));
}
}

public class PES2UG21CS315_Q1 {
    public static void main(String[] args) {
        WordFrequencyCounter wordFrequencyCounter = new WordFrequencyCounter();
        try {
            wordFrequencyCounter.processFile("C:\\Users\\Praka\\OneDrive\\Documents\\JAVA\\sample.txt");
        } catch (IOException | InterruptedException | ExecutionException e) {
            e.printStackTrace();
        }
    }
}

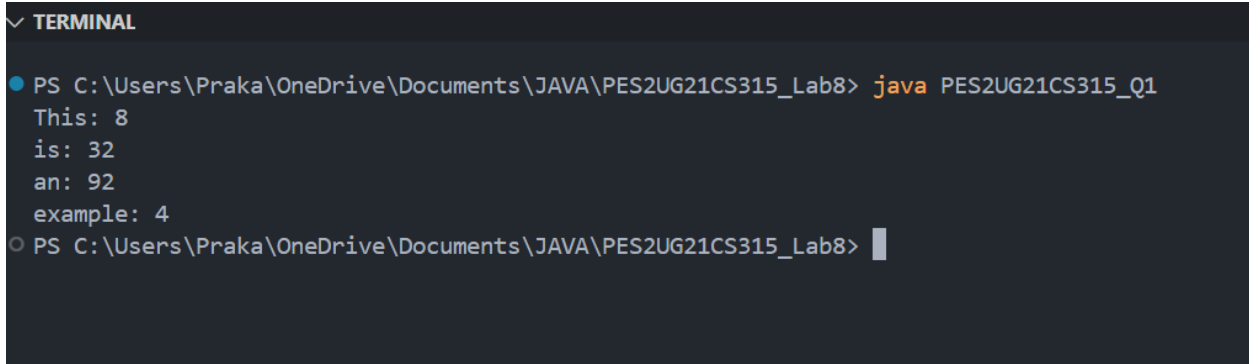
```

Sample.txt



```
File Edit View
This This This This This This This This
is is is is is is is is is is is is is is is is is is is is is is is is is is is is is is
is is is
an an an an an an an an an an an an an an an an an an an an an an an an an an an an an an
an an an an an an an an an an an an an an an an an an an an an an an an an an an an an an
an an an an an an an an an an an an an an an an an an an an an an an an an an an an an an
an an an an
example example example example
```

Output:



```
✓ TERMINAL
● PS C:\Users\Praka\OneDrive\Documents\JAVA\PES2UG21CS315_Lab8> java PES2UG21CS315_Q1
This: 8
is: 32
an: 92
example: 4
○ PS C:\Users\Praka\OneDrive\Documents\JAVA\PES2UG21CS315_Lab8> █
```

(b) Problem Statement: Music Library Management System with Serialization in Java

You're tasked with creating a Java program to implement a music library management system. The system should allow users to add, remove, update, and view music tracks in the library. Additionally, the program should support serialization and deserialization of the music library data to save and load the library to/from a file.

Write a Java program with the following requirements:

1. Create a `MusicTrack` class to represent music tracks in the library. Each track should have attributes such as title, artist, album, duration, and any other relevant information.
2. Implement a `MusicLibraryManager` class that manages the music library. The `MusicLibraryManager` class should allow users to add music tracks, remove music tracks, update track details, and view the music tracks in the library.

3. Include methods in the `MusicLibraryManager` class to serialize the music library data to a file when the program exits, and deserialize the music library data to restore it when the program starts.
4. You can add the values for each track and its details within the program itself.

Your program should consist of the following components:

1. A `MusicTrack` class with attributes and methods to represent a music track.
2. A `MusicLibraryManager` class with methods to manage the music library, including serialization and deserialization methods.
3. Implement the necessary serialization and deserialization methods using Java's `ObjectOutputStream` and `ObjectInputStream` classes.
4. Provide proper exception handling for serialization and deserialization operations.

Your goal is to create a music library management system that allows users to efficiently manage music tracks, supports serialization for persistence, and ensures that the music library data is saved and loaded reliably across different sessions of running the program.

Example Output:

```
Music Tracks: Title: Future Perfect, Artist: The Durutti Column, Album: Fidelity,
Duration: 314 seconds Title: Chamber of Reflection, Artist: Mac DeMarco, Album:
Salad Days, Duration: 231 seconds Music library serialized successfully.
```

```
Music library deserialized successfully. Deserialized Music Tracks: Title: Future
Perfect, Artist: The Durutti Column, Album: Fidelity, Duration: 314 seconds Title:
Chamber of Reflection, Artist: Mac DeMarco, Album: Salad Days, Duration: 231
seconds
```

Code:

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

class MusicTrack implements Serializable {
    private String title;
    private String artist;
    private String album;
    private double duration;
```

```

    public MusicTrack(String title, String artist, String album, double
duration) {
        this.title = title;
        this.artist = artist;
        this.album = album;
        this.duration = duration;
    }

    public String getTitle() {
        return title;
    }

    // Other getters and setters...

    @Override
    public String toString() {
        return "Title: " + title + ", Artist: " + artist + ", Album: " + album +
", Duration: " + duration + " minutes";
    }
}

class MusicLibraryManager {
    private List<MusicTrack> musicLibrary;

    public MusicLibraryManager() {
        this.musicLibrary = new ArrayList<>();
        // Add some default tracks for testing
        musicLibrary.add(new MusicTrack("Neverland:Freefall", "TXT",
"Chapter:Temptation", 4.5));
        musicLibrary.add(new MusicTrack("Fever", "Enhypen", "Fever", 3.2));
        musicLibrary.add(new MusicTrack("Deja Vu", "Ateez", "Zero:Fever", 3.2));
    }

    public void addMusicTrack(MusicTrack track) {
        musicLibrary.add(track);
    }

    public void removeMusicTrack(String title) {
        musicLibrary.removeIf(track -> track.getTitle().equals(title));
    }

    public void updateMusicTrack(String title, MusicTrack updatedTrack) {
        for (int i = 0; i < musicLibrary.size(); i++) {

```

```

        if (musicLibrary.get(i).getTitle().equals(title)) {
            musicLibrary.set(i, updatedTrack);
            break;
        }
    }
}

public void viewMusicLibrary() {
    musicLibrary.forEach(System.out::println);
}

public void serializeMusicLibrary(String filename) {
    try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(filename))) {
        oos.writeObject(musicLibrary);
        System.out.println("Music library serialized successfully.");
    } catch (IOException e) {
        System.err.println("Error during serialization: " + e.getMessage());
    }
}

@SuppressWarnings("unchecked")
public void deserializeMusicLibrary(String filename) {
    try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(filename))) {
        musicLibrary = (List<MusicTrack>) ois.readObject();
        System.out.println("Music library deserialized successfully.");
    } catch (IOException | ClassNotFoundException e) {
        System.err.println("Error during deserialization: " +
e.getMessage());
    }
}
}

public class PES2UG21CS315_Q2{
    public static void main(String[] args) {
        MusicLibraryManager libraryManager = new MusicLibraryManager();

        // Adding a new track
        MusicTrack newTrack = new MusicTrack("House of Cards", "BTS", "HYYN",
5.0);
        libraryManager.addMusicTrack(newTrack);

        // Viewing the music library

```



```

        System.out.println("Music Library:");
        libraryManager.viewMusicLibrary();

        // Serializing the music library to a file
        libraryManager.serializeMusicLibrary("musicLibrary.ser");

        // Removing a track
        libraryManager.removeMusicTrack("House of Cards");

        // Updating a track
        MusicTrack updatedTrack = new MusicTrack("Fever", "Enhypen", "Fever:1",
3.5);
        libraryManager.updateMusicTrack("Fever", updatedTrack);

        // Viewing the updated music library
        System.out.println("\nUpdated Music Library:");
        libraryManager.viewMusicLibrary();

        // Deserializing the music library from the file
        libraryManager.deserializeMusicLibrary("musicLibrary.ser");

        // Viewing the music library after deserialization
        System.out.println("\nMusic Library After Deserialization:");
        libraryManager.viewMusicLibrary();
    }
}

```

Output:

▼ **TERMINAL**

```
PS C:\Users\Praka\OneDrive\Documents\JAVA\PES2UG21CS315_Lab8> java PES2UG21CS315_Q2
```

```
Music Library:
```

```
Title: Neverland:Freefall, Artist: TXT, Album: Chapter:Temptation, Duration: 4.5 minutes
```

```
Title: Fever, Artist: Enhypen, Album: Fever, Duration: 3.2 minutes
```

```
Title: Deja Vu, Artist: Ateez, Album: Zero:Fever, Duration: 3.2 minutes
```

```
Title: House of Cards, Artist: BTS, Album: HYYN, Duration: 5.0 minutes
```

```
Music library serialized successfully.
```

```
Updated Music Library:
```

```
Title: Neverland:Freefall, Artist: TXT, Album: Chapter:Temptation, Duration: 4.5 minutes
```

```
Title: Fever, Artist: Enhypen, Album: Fever:1, Duration: 3.5 minutes
```

```
Title: Deja Vu, Artist: Ateez, Album: Zero:Fever, Duration: 3.2 minutes
```

```
Music library deserialized successfully.
```

```
Music Library After Deserialization:
```

```
Title: Neverland:Freefall, Artist: TXT, Album: Chapter:Temptation, Duration: 4.5 minutes
```

```
Title: Fever, Artist: Enhypen, Album: Fever, Duration: 3.2 minutes
```

```
Title: Deja Vu, Artist: Ateez, Album: Zero:Fever, Duration: 3.2 minutes
```

```
Title: House of Cards, Artist: BTS, Album: HYYN, Duration: 5.0 minutes
```

```
○ PS C:\Users\Praka\OneDrive\Documents\JAVA\PES2UG21CS315_Lab8> █
```