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:

```
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
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
```

This would be the required output:

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

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
```