

```

using System;
using System.IO;
using System.Text;
using System.Collections;

namespace SB54_CSAI.Genres
{
    #region MThreadSingleton_genres CLASS

    /// <summary>
    /// MTSingleton class implements the thread-safe version of the Singleton
    /// design pattern. This class is designed to be used to provide the singleton
    /// across threads in a multi-threaded application.
    /// This class reads from an embedded resource text file which contains all possible
    /// MP3 ID3 Tag genres. As the entries are read from the file they are added
    /// the underlying database for access via other user controls within the ReMP3
    application.
    /// See the <see cref="SB54_CSAI.DatabaseAccess">DatabaseAccess class for a
    /// description of the interactions between these 2 classes.
    /// NOTE : This class has been marked as sealed to prevent it from being inherited from
    .
    /// </summary>
    public sealed class MThreadSingleton_genres
    {
        #region Instance fields
        //Instance fields

        //The volatile keyword indicates that a field can be modified in
        //the program by something such as the operating system, the hardware,
        //or a concurrently executing thread.
        //The system always reads the current value of a volatile object
        //at the point it is requested, even if the previous instruction
        //asked for a value from the same object.
        private static volatile MThreadSingleton_genres instance = null;
        private static object syncRoot = new object();
        private ArrayList genres = new ArrayList();

        #endregion
        #region Private Constructor
        /// <summary>
        /// Constructor, make the default constructor private, so that no can directly
        create it.
        /// Simply calls readGenresFromFile method
        /// </summary>
        private MThreadSingleton_genres()
        {
            readGenresFromFile();
        }
        #endregion
        #region Private Methods

        /// <summary>
        /// Obtains a file stream to the embedded resource text file "Singleton.Genres.txt",
        and
        /// then opens a stream and reads the entries of the text file into an internal
        /// ArrayList
        /// </summary>
        private void readGenresFromFile()
        {
            //Initilaise Stream and StreamReader objects
            System.IO.Stream fs = null;
            System.IO.StreamReader sr = null;

            try
            {
                //get a Stream for this Assembly manifest embedded resource file
                fs = this.GetType().Assembly.GetManifestResourceStream("Singleton.Genres.txt");
                //check that Stream exists
                if (fs != null)
                {
                    //create a new StreamReader for the input Stream
                    sr = new StreamReader(fs);
                    //Peek through the file , and read contents into internal ArrayList
                }
            }
        }
    }
}

```

```

        while (sr.Peek() >= 0)
        {
            genres.Add(sr.ReadLine());
        }
    }
}
catch(Exception ex)
{
    //print any error message that occurred during file access operation
    Console.WriteLine("Problem with reading file (Genres.txt) in
MThreadSingleton_genres");
    Console.WriteLine(ex.Message);
}
finally
{
    //release any held resources
    if (fs !=null) { fs.Close(); }
    if (sr !=null) { sr.Close(); }
}
}
#endregion
#region Public Methods/Properties

/// <summary>
/// getGenreFromID
/// </summary>
/// <param name="GenreID">int which represents a Genre number</param>
/// <returns>string which is the string which represents the input Genre number</
returns>
public string getGenreFromID(int GenreID)
{
    //check to see if genre number is within range, if it is get internal
    //genre text equivalent
    if (GenreID >= 0 && GenreID < genres.Count - 1)
    {
        return (string)genres[GenreID];
    }
    //not in range so return ""
    else
    {
        return "";
    }
}

/// <summary>
/// getGenres, retrieves a ArrayList of Genres read from an internal
/// resource text file
/// </summary>
/// <returns>ArrayList which can be used to to enumerate the list of
/// internal genres</returns>
public ArrayList getGenres()
{
    //return the genres ArrayList for use by other classes
    return genres;
}

/// <summary>
/// public property that can get the single instance of the
/// MThreadSingleton_genres object. This method uses an object lock
/// to provide synchroniztion between threads, such that it will provide
/// a thread safe MThreadSingleton_genres singleton object
/// </summary>
/// <returns>MThreadSingleton_genres which is the single instance of the
/// MThreadSingleton_genres object</returns>
public static MThreadSingleton_genres Instance
{
    //get the instance
    get
    {
        // only create a new instance if one doesn't already exist.
        if (instance == null)
        {
            // use this lock to ensure that only one thread is accessing

```

```
        // this block of code at once.
        lock(syncRoot)
        {
            //check to see if instance exists, if not create a new one
            if (instance == null)
                instance = new MThreadSingleton_genres();
        }
        // return instance where it was just created or already existed.
        return instance;
    }
}

#endregion
}

#endregion
}
```