```
//------------------------------------------------------------------------------
//File:   PNGChunk.cs
//Desc:   This program defines a class PNGChunk which holds the information
//         for a single chunk of PNG image file.
//------------------------------------------------------------------------------


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

class PNGChunk
{
    private uint chunkLength;
    private string chunkType;
    private byte[] chunkData;
    private uint crc;

    public PNGChunk()
    {
        chunkLength = 0u;
        chunkType = "";
        chunkData = new byte[] { };
        crc = 0u;

    }

    public uint ChunkLength
    {
        get
        {
            return chunkLength;
        }
        set
        {
            chunkLength = value;
        }
    }

    public string ChunkType
    {
        get
        {
            return chunkType;
        }
        set
        {
            chunkType = value;
        }
    }

    public byte[] ChunkData
    {
        get
        {
            return chunkData;
        }
        set
        {
            chunkData = value;
        }
    }

    public uint Crc
    {
        get
        {
            return crc;
```

```
        }
        set
        {
            crc = value;
        }
    }

    //Extracts the metadata in the ChunkData and returns it as a string
    public string ExtractMData()
    {
        string keyWrdsValues = "";

        foreach (byte b in ChunkData)
        {
            if ((char)b == '\0')
            {
                keyWrdsValues += ":";
            }
            else
            {
                keyWrdsValues += (char)b;
            }
        }
        return keyWrdsValues;
    }
}
```

```
//-----------------------------------------------------------------------------
--------------
//File:   PNGFile.cs
//Desc:   This program defines a class PNGFile, which contains a method Load() t
o load
//         each chunk of a PNG image file seperately to a list of chunks.
//-----------------------------------------------------------------------------
---------------


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;


class PNGFile
{
    private List<PNGChunk> pngChunks;

    public PNGFile()
    {
        pngChunks = new List<PNGChunk> { };
    }

    public List<PNGChunk> PngChunks
    {
        get
        {
            return pngChunks;
        }
        set
        {
            pngChunks = value;
        }
    }

    //Loads the chunks of the PNG file from the 'filename' in the parameter and
returns PNGFile object
    public static PNGFile Load(string fileName)
    {
        PNGFile pngChunkCol = new PNGFile();

        using (PNGFileReader pngReader = new PNGFileReader(fileName))
        {

            PNGChunk pChunk = pngReader.ReadChunk(); ;
            while (true)
            {
                if (pChunk.ChunkType == "IEND")
                {
                    pngChunkCol.PngChunks.Add(pChunk);
                    break;
                }
                else
                {
                    pngChunkCol.PngChunks.Add(pChunk);
                    pChunk = pngReader.ReadChunk();
                }
            }
        }
        return pngChunkCol;
    }

}
```

```
//-----------------------------------------------------------------------------
//File:   PNGFileReader.cs
//Desc:   This program defines a class PNGFileReader that returns a single chunk
//          of a PNG image file.
//-----------------------------------------------------------------------------


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;


class PNGFileReader: IDisposable
{
    private BinaryReader fileReader;

    //Constructor that initializes a BinaryReader obj from the
    //in the 'pngFile' parameter, validating that the file is truly a PNG file
    public PNGFileReader(string pngFile)
    {
        fileReader = new BinaryReader(File.Open(pngFile, FileMode.Open));

        byte[] pngSignature = fileReader.ReadBytes(8);
        ValidateSignature(pngSignature);
    }

    public PNGChunk ReadChunk()
    {
        PNGChunk chunk = new PNGChunk();

        byte[] lengthArray = new byte[4];
        byte[] typeArray = new byte[4];
        byte[] dataArray = new byte[] { };
        byte[] crcArray = new byte[4];

        //Read length and assign it to the ChunkLength
        lengthArray = fileReader.ReadBytes(4);
        chunk.ChunkLength = ReadLength(lengthArray);

        //Read the chunk type and assign it to the ChunkType
        typeArray = fileReader.ReadBytes(4);
        chunk.ChunkType = ReadType(typeArray);

        //Read the chunk data and assign it to the ChunkData
        chunk.ChunkData = fileReader.ReadBytes((int)chunk.ChunkLength);


        //Read the crc and assign it the Crc
        crcArray = fileReader.ReadBytes(4);
        chunk.Crc = ReadLength(crcArray);       //uses same code as the length b
ecause it is a 32-bit integer.

        return chunk;
    }

    //Reads a byte array from the param and returns an integer that contains the
 length of the chunk data
    private static uint ReadLength(byte[] chunkLength)
    {
        uint length = 0u;

        length = (uint)((chunkLength[0] << 24) + ((uint)chunkLength[1] << 16) +
((uint)chunkLength[2] << 8) + ((uint)chunkLength[3] << 0));

        return length;
    }
    //Reads a byte array and returns the string containing the chunk type
    private static string ReadType(byte[] chunkType)
    {
        string chunkTyp = "";

        for (int i = 0; i < 4; i++)
        {
            chunkTyp += (char)chunkType[i];
        }

        return chunkTyp;
    }

    //Checks to see if the first 8 bytes of the 'signature' are those of a PNG f
ile
    private static void ValidateSignature(byte[] signature)
    {
        List<int> sequence = new List<int> { 137, 80, 78, 71, 13, 10, 26, 10 };
        for (int i = 0; i < signature.Length; i++)
        {
            if (sequence[i] != signature[i])
            {
                throw new ArgumentException("\nThis is not a valid png file.");
            }
        }
    }

    //Implement the Dispose method of IDisposable and closes the BinaryReader
    public void Dispose()
    {
        fileReader.Close();
    }
}
```

```
//---------------------------------------------------------------------------------
//File:   PNGFileTest.cs
//Desc:   This program contains tests for the Load() method of the PNGFile class
.
//---------------------------------------------------------------------------------


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using NUnit.Framework;


[TestFixture]
public class PNGFileTest
{
  [Test]
    public void Load_ValidPNG_ChunksLoadedCorrectly()
    {
        string fileName = System.IO.Path.Combine(TestContext.CurrentContext.Test
Directory, "ice.png");
        string[] chunkNames = new string[] {"IHDR", "gAMA", "bKGD", "IDAT", "ID
AT", "IDAT", "tEXt", "tEXt", "tEXt", "IEND"};

        PNGFile pngChunkCol = PNGFile.Load(fileName);

        Assert.True(pngChunkCol.PngChunks.Count == 10);
        for (int i = 0; i < pngChunkCol.PngChunks.Count; i++)
        {
            Assert.True(pngChunkCol.PngChunks[i].ChunkType == chunkNames[i]);
        }
    }

  [Test]
    public void ValidateSignature_InvalidPngFile_DoesNotLoad()
    {
        string fileName = System.IO.Path.Combine(TestContext.CurrentContext.Test
Directory, "shovel.jpeg");
        try
        {
            PNGFile pngChunkCol = PNGFile.Load(fileName);
            Assert.Fail();
        }
        catch (ArgumentException)
        {
        }
    }
}
```