```
using System;
using System.IO;
using System.Text;

using Dotnet.Commons.IO;

using NUnit.Framework;

namespace Dotnet.Commons.IO.Test
{

    [NUnit.Framework.TestFixture]
    public class FileUtilsTest
    {

        private const String TEST_DUMMY_FILE = "Dotnet.Commons.IO.TestFile.txt";

        private FileInfo fiTestFile = null;
        private string tempdir;

        [NUnit.Framework.SetUp]
        public void Init()
        {
            FileStream fs = null;

            // create a file in the temp directory
            tempdir = System.IO.Path.GetTempPath();

            fiTestFile = new FileInfo(tempdir + Path.DirectorySeparatorChar +
    TEST_DUMMY_FILE);

            //check to see if file exists, if not create one
            if (!fiTestFile.Exists)
            {
                string someText = " The quick brown fox jumps over the lazy dog.";

                Byte[] byteArray =  Encoding.ASCII.GetBytes(someText);

                fs = fiTestFile.OpenWrite();
                fs.Write(byteArray, 0, byteArray.Length);
                fs.Close();

                fiTestFile.Refresh();

            }


        }

        [TearDown]
        public void Cleanup()
        {
            if (fiTestFile.Exists)
                fiTestFile.Delete();
            fiTestFile.Refresh();
        }

        [Test]
        public void TestCopyFile()
        {
            Console.WriteLine("Test FileUtils.CopyFile");

            string destPath = tempdir + "Dotnet.Commons.IO.Test" + Path.
    DirectorySeparatorChar;
            string destFile = destPath + "CopyToFilename.txt";

            FileInfo fiDestFile = new FileInfo(destFile);

            if (fiDestFile.Exists)
            {
                fiDestFile.Delete();
                fiDestFile.Refresh();
            }
            Assert.IsFalse(fiDestFile.Exists);
```

```
            Console.WriteLine("Test 1. Copy {0} as {1}", fiTestFile.FullName, fiDestFile.
    FullName);
            Console.WriteLine("        * Assert {0} does not exist : {1}", fiDestFile.
    FullName, !fiDestFile.Exists);
            FileUtils.CopyFile(fiTestFile, fiDestFile);
            Console.WriteLine("        * Assert {0} exists after copy : {1}", fiDestFile.
    FullName, fiDestFile.Exists);

            Assert.IsTrue(fiDestFile.Exists);

        }

        [Test]
        public void TestTouch()
        {
            Console.WriteLine("Test FileUtils.Touch");

            DateTime fileLastAccessTime = fiTestFile.LastAccessTime;

            FileUtils.Touch(fiTestFile);
            System.Threading.Thread.Sleep(200);
            Console.WriteLine("Test 1. Assert file LastAccessTime is modified : {0}",
    fiTestFile.LastAccessTime > fileLastAccessTime);
            Assert.IsTrue(fiTestFile.LastAccessTime > fileLastAccessTime);
            Console.WriteLine();

            Console.WriteLine("Test 2. Test a new file is created by Touch");
            string dummyFileName = tempdir + Path.DirectorySeparatorChar + "Dotnet.Commons
    .IO.TestFile_tmp" +  DateTime.Now.Millisecond + ".txt";
            Console.WriteLine("        * Attempting to touch the file '{0}'",
    dummyFileName);
            FileInfo someNewFile = new FileInfo(dummyFileName);

            Console.WriteLine("        * Assert new file '{0}' does not exist: {1}",
    someNewFile.FullName, !someNewFile.Exists);
            Assert.IsFalse(someNewFile.Exists);
            FileUtils.Touch(someNewFile);

            Console.WriteLine("        * Assert new file '{0}' has been created by Touch()
    : {1}", someNewFile.FullName, File.Exists(dummyFileName));
            Assert.IsTrue(File.Exists(dummyFileName));

            if (File.Exists(dummyFileName))
                someNewFile.Delete();

            Console.WriteLine(" -- Test Completed -- ");
            Console.WriteLine();

        }


        [Test]
        public void TestFileCompare()
        {
            Console.WriteLine("Test FileUtils.Compare");

            FileInfo fileInfo1 = new FileInfo("TestData\\App.ico");
            FileInfo fileInfo2 = new FileInfo("TestData\\App1.ico");
            FileInfo fileInfo3 = new FileInfo("TestData\\App2.ico");

            Console.WriteLine("File '{0}' exists? {1}", fileInfo1.FullName, fileInfo1.
    Exists);
            Assert.IsTrue(fileInfo1.Exists);

            if (fileInfo1.IsReadOnly)
                fileInfo1.Attributes = FileAttributes.Normal;

            Console.WriteLine("File '{0}' exists? {1}", fileInfo2.FullName, fileInfo2.
    Exists);
            Assert.IsTrue(fileInfo2.Exists);

            if (fileInfo2.IsReadOnly)
                fileInfo2.Attributes = FileAttributes.Normal;
```

```
            Console.WriteLine("File '{0}' exists? {1}", fileInfo3.FullName, fileInfo3.
    Exists);
            Assert.IsTrue(fileInfo3.Exists);


            if (fileInfo3.IsReadOnly)
                fileInfo3.Attributes = FileAttributes.Normal;

            bool same = FileUtils.Compare(fileInfo1, fileInfo2);
            Console.WriteLine("Are the file same (expect yes)? {0}", same);
            Assert.IsFalse(same);

            same = FileUtils.Compare(fileInfo3, fileInfo2);
            Console.WriteLine("Are the file same (expect false)? {0}", same);
            Assert.IsFalse(same);

            FileInfo fileInfoTxt1 = new FileInfo("TestData\\TextFile1.txt");
            FileInfo fileInfoTxt2 = new FileInfo("TestData\\TextFile2.txt");
            same = FileUtils.Compare(fileInfoTxt1, fileInfoTxt2);
            Assert.IsFalse(same);
        }

        [Test]
        public void TestGetDirectoryFileInfo()
        {
            Console.WriteLine("Test FileUtils.GetDirectoryFileInfo");

            string[] fileArray = FileUtils.GetDirectoryFileInfo(Path.GetTempPath());
            Assert.IsTrue(fileArray.Length > 0);

            Console.WriteLine("Files in the directory {0} are:", System.IO.Path.
    GetTempPath());
            foreach (string file in fileArray)
            {
                Console.WriteLine(file);
            }

            Console.WriteLine();
            Console.WriteLine();

            fileArray = FileUtils.GetDirectoryFileInfo(System.Environment.CurrentDirectory
    , '|');
            Assert.IsTrue(fileArray.Length > 0);

            Console.WriteLine("Files in the directory {0} are:", System.Environment.
    CurrentDirectory);
            foreach (string file in fileArray)
            {
                Console.WriteLine(file);
            }
        }

        [Test]
        public void TestIsNewer()
        {
            Console.WriteLine("Test FileUtils.IsNewer");

            FileInfo app1 = new FileInfo("TestData\\App.ico");
            FileInfo app2 = new FileInfo("TestData\\App1.ico");

            Assert.IsFalse(FileUtils.IsNewer(app1, DateTime.Now));
            Assert.IsTrue(FileUtils.IsNewer(app2, app1));
        }

        [Test]
        public void ShouldReturnRelativePath()
        {
            Console.WriteLine("Test FileUtils.GetRelativePath");
            Console.WriteLine("Test ShouldReturnRelativePath");

            string rootPath = @"C:\rootPath";
            string fullPath = Path.Combine(rootPath, "AFile.txt");
            Assert.AreEqual("AFile.txt", FileUtils.GetRelativePath(rootPath, fullPath));
```

```
            rootPath = @"C:";
            fullPath = @"C:\ADir";
            Assert.AreEqual("ADir", FileUtils.GetRelativePath(rootPath, fullPath));

            rootPath = @"C:\rootPath";
            fullPath = @"C:\rootPathSimiliar\AFile.txt";
            Assert.IsNull(FileUtils.GetRelativePath(rootPath, fullPath));
        }

        [Test]
        public void ShouldReturnNullRelativePathIfInputsAreNull()
        {
            Console.WriteLine("Test ShouldReturnNullRelativePathIfInputsAreNull");

            Assert.IsNull(FileUtils.GetRelativePath(null, null));
            Assert.IsNull(FileUtils.GetRelativePath(null, "SomeValidFullPath"));
            Assert.IsNull(FileUtils.GetRelativePath("SomeRootPath", null));
        }

        [Test]
        public void ShouldReturnEmptyRelativePathIfRootPathAndFullPathAreEqual()
        {
            Console.WriteLine("ShouldReturnEmptyRelativePathIfRootPathAndFullPathAreEqual"↙
);
            string path = @"C:\rootPath";
            Assert.AreEqual(string.Empty, FileUtils.GetRelativePath(path, path));
        }

        [Test]
        public void TestReadBinaryFile()
        {
            Console.WriteLine("Test FileUtils.ReadBinaryFile");

            FileInfo fi = new FileInfo(AppDomain.CurrentDomain.BaseDirectory + "\\TestData↙
\\App.ico");
            byte[] content = FileUtils.ReadBinaryFile(fi.FullName);
            Assert.AreEqual(fi.Length, content.Length);
        }

        [Test]
        public void TestSaveBinaryFile()
        {
            Console.WriteLine("Test FileUtils.Save");

            FileInfo fi = new FileInfo(AppDomain.CurrentDomain.BaseDirectory + "\\TestData↙
\\App.ico");
            byte[] content = FileUtils.ReadBinaryFile(fi.FullName);

            FileUtils.Save(tempdir + fi.Name, content);
            FileInfo fiSaved = new FileInfo(tempdir + fi.Name);

            Assert.IsTrue(fiSaved.Exists);

            byte[] savedContent = FileUtils.ReadBinaryFile(fi.FullName);
            Assert.AreEqual(content, savedContent);

            Assert.IsTrue(FileUtils.Compare(fi, fiSaved));

            FileUtils.Remove(fiSaved.FullName);
            fiSaved.Refresh();
            Assert.IsFalse(fiSaved.Exists);
        }

        [Test]
        public void TestRemoveAllFiles()
        {

            Console.WriteLine("Test FileUtils.RemoveAll");
            System.Collections.ArrayList tempfileList = new System.Collections.ArrayList()↙
;

            string tempfile_template = this.GetType().Assembly.GetName().Name + "_temp_";
```

```
            for(int i=0; i< 5; i++)
            {
                string filename = tempfile_template + i.ToString() + ".temp";
                TempFileUtils.CreateTempTextFile(tempdir, filename);
                tempfileList.Add(filename);
            }

            FileInfo[] fiArray = FileUtils.GetFilesMatchWildCard(tempdir + "\\" +
    tempfile_template +  "*.temp");

            Assert.AreEqual(tempfileList.Count, fiArray.Length);

            foreach (FileInfo fi in fiArray)
            {
                Assert.IsTrue(tempfileList.Contains(fi.Name));
            }

            FileUtils.RemoveAll(tempdir + "\\" + tempfile_template + "*.temp");

            Console.WriteLine("Test FileUtils.GetFilesMatchWildCard");

            fiArray = FileUtils.GetFilesMatchWildCard(tempdir + "\\" + tempfile_template +
     "*.temp");

            Assert.AreEqual(0, fiArray.Length);
        }


        public static void Execute()
        {
            System.Console.WriteLine();
            System.Console.WriteLine("**************************************************
    ****");
            System.Console.WriteLine(" Start of FileUtilsTest ");
            System.Console.WriteLine("**************************************************
    ****");

            FileUtilsTest test = new FileUtilsTest();

            test.Init();
            test.TestCopyFile();
            test.TestTouch();
            test.TestIsNewer();
            test.TestFileCompare();
            test.TestReadBinaryFile();
            test.TestSaveBinaryFile();
            test.TestRemoveAllFiles();
            test.ShouldReturnEmptyRelativePathIfRootPathAndFullPathAreEqual();
            test.ShouldReturnNullRelativePathIfInputsAreNull();
            test.ShouldReturnRelativePath();
            test.Cleanup();

            System.Console.WriteLine();
            System.Console.WriteLine("**************************************************
    ****");
            System.Console.WriteLine(" End of FileUtilsTest ");
            System.Console.WriteLine("**************************************************
    ****");

        }
    }
}
```