//r1

[TestClass]

public class UnitTestPoint

{

[TestMethod]

[ExpectedException(typeof(ArgumentOutOfRangeException))]

public void TestMethodCtor\_ThrowsArgumentOutOfRangeException()

{

Point p;

double a = -1000.0;

double b = -1000.0;

p = new Point(a, b);

}

}

public class Point

{

public Point(double lat, double longit)

{

}

}

//g1

using System;

namespace GeographicLibrary

{

public class Point

{

public Point(double lat, double longit)

{

throw new ArgumentOutOfRangeException();

}

}

}

//r2

[TestMethod]

public void TestMethodCtor\_CteatePoint()

{

Point p;

double a = 0.0;

double b = 0.0;

p = new Point(a, b);

Assert.IsNotNull(p);

}

//g2

double lat;

double longit;

public Point(double lat, double longit)

{

if (Math.Abs(lat) > 90 || Math.Abs(longit) > 180)

{

throw new ArgumentOutOfRangeException();

}

this.lat = lat;

this.longit = longit;

}

//ref2

using System;

namespace GeographicLibrary

{

public class Point

{

double latitude;

double longitude;

public double Longitude

{

get => longitude;

set

{

if (Math.Abs(value) > 180)

{

throw new ArgumentOutOfRangeException();

}

longitude = value;

}

}

public double Latitude

{

get => latitude;

set

{

if (Math.Abs(value) > 90)

{

throw new ArgumentOutOfRangeException();

}

latitude = value;

}

}

public Point(double latitude, double longitude)

{

if (Math.Abs(latitude) > 90 || Math.Abs(longitude) > 180)

{

throw new ArgumentOutOfRangeException();

}

this.Latitude = latitude;

this.Longitude = longitude;

}

}

}

//r3

using System;

using GeographicLibrary;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace GeographicLibraryTest

{

[TestClass]

public class UnitTestPoint

{

[TestMethod]

[DataRow(-1000, 1000)]

[DataRow(1000, 1000)]

[DataRow(91, 179)]

[DataRow(-91, 179)]

[DataRow(89, 181)]

[DataRow(-89, -181)]

[ExpectedException(typeof(ArgumentOutOfRangeException))]

public void TestMethodCtor\_ThrowsArgumentOutOfRangeException(double a, double b)

{

Point p;

p = new Point(a, b);

}

[TestMethod]

[DataRow(89, 179)]

[DataRow(-89.999999999999999999999999999999, 179)]

[DataRow(89, 179.9999999999999999999999999)]

[DataRow(-89, -179.9999999999999999999999999)]

[DataRow(0.0, 179.999999)]

[DataRow(-89.53465489347548957894, 179.48902378947238472389479823)]

[DataRow(89, 178)]

[DataRow(-89.6048923842389948923753452347, 0)]

public void TestMethodCtor\_CteatePoint(double a, double b)

{

Point p;

p = new Point(a, b);

Assert.IsNotNull(p);

}

}

}

//r4

using GeographicLibrary;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System.Collections.Generic;

namespace GeographicLibraryTest

{

[TestClass]

class AzimutTests

{

[TestMethod]

[DynamicData(nameof(GetPoints), DynamicDataSourceType.Method)]

public void Test\_CreateArc\_CreatedArcNotNull(Point a, Point b)

{

Arc arc = new Arc(a, b);

Assert.IsNotNull(arc);

}

public static IEnumerable<List<Point>> GetPoints()

{

yield return new List<Point>() { new Point(0.0, 180.0), new Point(0.0, 90.0) };

yield return new List<Point>() { new Point(0.0, -90.0), new Point(0.0, 90.0) };

}

}

}

//g4

namespace GeographicLibrary

{

public class Arc

{

public Arc(Point p1, Point p2)

{

}

}

}

//ref4

namespace GeographicLibrary

{

public class Arc

{

Point p1;

Point p2;

public Arc(Point p1, Point p2)

{

this.p1 = p1;

this.p2 = p2;

}

}

}