Tests/PointPairsTest.md

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

using Vache.Models;


namespace Vache.Tests;

[TestClass]
public class PointTest
{
    [TestMethod]
    public void PointTranslate1()
    {
        Point2  pnt = new(1, 2);
        Vector2 vec = new(2, 3);
        Point2  res = new(3, 5);

        Assert.AreEqual(res, pnt.Translate(vec));
    }

    [TestMethod]
    public void PointTranslate2()
    {
        Point2  pnt = new(4.5, -2);
        Vector2 vec = new(-3, 0);
        Point2  res = new(1.5, -2);

        Assert.AreEqual(res, pnt.Translate(vec));
    }

    [TestMethod]
    public void PointParseSuccess()
    {
        Assert.IsTrue(Point2.TryParse("(1, 2)", out Point2? point));
        Assert.IsNotNull(point);
    }

    [TestMethod]
    public void PointParseFail1()
    {
        Assert.IsFalse(Point2.TryParse("(1, 2", out Point2? point));
        Assert.IsNull(point);
    }

    [TestMethod]
    public void PointParseFail2()
    {
        Assert.IsFalse(Point2.TryParse("(1 2)", out Point2? point));
        Assert.IsNull(point);
    }

    [TestMethod]
    public void PointParseFail3()
    {
        Assert.IsFalse(Point2.TryParse("(A, 2)", out Point2? point));
        Assert.IsNull(point);
    }

    [TestMethod]
    public void PointString1()
    {
        Point2 pnt = new(1, 2);

        Assert.AreEqual("(1, 2)", pnt.ToString());
    }

    [TestMethod]
    public void PointString2()
    {
        Point2 pnt = new(-1, 2.5);

        Assert.AreEqual("(-1, 2.5)", pnt.ToString());
```

```
    }

    [TestMethod]
    public void PointEqual1()
    {
        Point2 pnt1 = new(1, 2),
               pnt2 = new(1, 2);

        Assert.IsTrue(pnt1 == pnt2);
    }

    [TestMethod]
    public void PointEqual2()
    {
        Point2 pnt1 = new(1, 2),
               pnt2 = new(-1, 2);

        Assert.IsTrue(pnt1 != pnt2);
    }

    [TestMethod]
    public void PointEqual3()
    {
        Point2? vec1 = new(1, 2),
                vec2 = null;

        Assert.IsFalse(vec1 == vec2);
    }

    [TestMethod]
    public void PointEqual4()
    {
        Point2? vec1 = new(1, 2),
                vec2 = null;

        Assert.IsTrue(vec1 != vec2);
    }

    [TestMethod]
    public void PointEqual5()
    {
        Point2? vec1 = null,
                vec2 = null;

        Assert.IsTrue(vec1 == vec2);
    }

    [TestMethod]
    public void PointEqual6()
    {
        Point2? vec1 = null,
                vec2 = null;

        Assert.IsFalse(vec1 != vec2);
    }

    [TestMethod]
    public void PointEqual7()
    {
        Point2  pnt = new(1, 2);
        Vector2 vec = new(1, 2);

        // ReSharper disable once SuspiciousTypeConversion.Global
        Assert.IsFalse(pnt.Equals(vec));
    }

    [TestMethod]
    public void PointHash()
    {
        Point2 pnt = new(1, 2),
               res = new(1, 2);

        Assert.IsTrue(pnt.GetHashCode() == res.GetHashCode());
    }
}
```

Tests/VectorTest

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

using Vache.Models;

namespace Vache.Tests;

[TestClass]
public class VectorTest
{
    [TestMethod]
    public void VectorParseSuccess()
    {
        Assert.IsTrue(Vector2.TryParse("{1, -1}", out Vector2? vec));
        Assert.IsNotNull(vec);

        Assert.AreEqual(1, vec.X);
        Assert.AreEqual(-1, vec.Y);
    }

    [TestMethod]
    public void VectorParseFail1()
    {
        Assert.IsFalse(Vector2.TryParse("{1, -1", out Vector2? vec));
        Assert.IsNull(vec);
    }

    [TestMethod]
    public void VectorParseFail2()
    {
        Assert.IsFalse(Vector2.TryParse("{1 -1}", out Vector2? vec));
        Assert.IsNull(vec);
    }

    [TestMethod]
    public void VectorParseFail3()
    {
        Assert.IsFalse(Vector2.TryParse("{A, -1}", out Vector2? vec));
        Assert.IsNull(vec);
    }

    [TestMethod]
    public void VectorString1()
    {
        Vector2 vec = new(1, 2);

        Assert.AreEqual("{1, 2}", vec.ToString());
    }

    [TestMethod]
    public void VectorString2()
    {
        Vector2 vec = new(-1, 2.5);

        Assert.AreEqual("{-1, 2.5}", vec.ToString());
    }

    [TestMethod]
    public void VectorAdd1()
    {
        Vector2 vec1 = new(1, 1),
                vec2 = new(2, 3),
                res  = new(3, 4);

        Assert.AreEqual(res, vec1 + vec2);
    }

    [TestMethod]
    public void VectorAdd2()
    {
        Vector2 vec1 = new(0, 5),
                vec2 = new(-3, -7),
                res  = new(-3, -2);

        Assert.AreEqual(res, vec1 + vec2);
```

```
    }

    [TestMethod]
    public void VectorSubtract1()
    {
        Vector2 vec1 = new(2, 3),
                vec2 = new(1, 1),
                res  = new(1, 2);

        Assert.AreEqual(res, vec1 - vec2);
    }

    [TestMethod]
    public void VectorSubtract2()
    {
        Vector2 vec1 = new(-3, 2),
                vec2 = new(-6, 4),
                res  = new(3, -2);

        Assert.AreEqual(res, vec1 - vec2);
    }

    [TestMethod]
    public void VectorNegate()
    {
        Vector2 vec = new(-3, 2),
                res = new(3, -2);

        Assert.AreEqual(res, -vec);
    }

    [TestMethod]
    public void VectorMultiply1()
    {
        Vector2 vec = new(1, 2),
                res = new(2, 4);

        Assert.AreEqual(res, vec * 2);
    }

    [TestMethod]
    public void VectorMultiply2()
    {
        Vector2 vec = new(1, -2),
                res = new(-3, 6);

        Assert.AreEqual(res, vec * -3);
    }

    [TestMethod]
    public void VectorDivide1()
    {
        Vector2 vec = new(1, 2),
                res = new(0.5, 1);

        Assert.AreEqual(res, vec / 2);
    }

    [TestMethod]
    public void VectorDivide2()
    {
        Vector2 vec = new(1, -2),
                res = new(-0.25, 0.5);

        Assert.AreEqual(res, vec / -4);
    }

    [TestMethod]
    public void VectorEqual1()
    {
        Vector2 vec1 = new(1, 2),
                vec2 = new(1, 2);

        Assert.IsTrue(vec1 == vec2);
    }

    [TestMethod]
    public void VectorEqual2()
    {
        Vector2 vec1 = new(1, 2),
```

```
                vec2 = new(-1, 2);

        Assert.IsTrue(vec1 != vec2);
    }

    [TestMethod]
    public void VectorEqual3()
    {
        Vector2? vec1 = new(1, 2),
                 vec2 = null;

        Assert.IsFalse(vec1 == vec2);
    }

    [TestMethod]
    public void VectorEqual4()
    {
        Vector2? vec1 = new(1, 2),
                 vec2 = null;

        Assert.IsTrue(vec1 != vec2);
    }

    [TestMethod]
    public void VectorEqual5()
    {
        Vector2? vec1 = null,
                 vec2 = null;

        Assert.IsTrue(vec1 == vec2);
    }

    [TestMethod]
    public void VectorEqual6()
    {
        Vector2? vec1 = null,
                 vec2 = null;

        Assert.IsFalse(vec1 != vec2);
    }

    [TestMethod]
    public void VectorEqual7()
    {
        Vector2 vec = new(1, 2);
        Point2  pnt = new(1, 2);

        // ReSharper disable once SuspiciousTypeConversion.Global
        Assert.IsFalse(vec.Equals(pnt));
    }

    [TestMethod]
    public void VectorHash()
    {
        Vector2 vec = new(1, 2),
                res = new(1, 2);

        Assert.IsTrue(vec.GetHashCode() == res.GetHashCode());
    }
}
```

Tests/PolygonTest

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

using Vache.Models;


namespace Vache.Tests;

[TestClass]
public class PolygonTest
{
```

```
    [TestMethod]
    public void PolygonFull1()
    {
        var polygon = new Polygon2(new Point2[] { new(-1, 1), new(-1, -1), new(1, -1), new(1, 1) });

        double area      = polygon.Area;
        Point2 cog       = polygon.CenterOfGravity;
        bool   inPolygon = polygon.IsPointInside(cog);

        Assert.AreEqual(4, area, Program.TOLERANCE);
        Assert.AreEqual(0, cog.X, Program.TOLERANCE);
        Assert.AreEqual(0, cog.Y, Program.TOLERANCE);
        Assert.IsTrue(inPolygon);
    }

    [TestMethod]
    public void PolygonFull2()
    {
        var polygon = new Polygon2(new Point2[] { new(-16.6, -20), new(-12, -18), new(-11, -16), new(-15,
-15) });

        double area      = polygon.Area;
        Point2 cog       = polygon.CenterOfGravity;
        bool   inPolygon = polygon.IsPointInside(cog);

        Assert.AreEqual(14.4, area, Program.TOLERANCE);
        Assert.AreEqual(-13.95, cog.X, Program.TOLERANCE);
        Assert.AreEqual(-17.25, cog.Y, Program.TOLERANCE);
        Assert.IsTrue(inPolygon);
    }

    [TestMethod]
    public void PolygonFull3()
    {
        var polygon = new Polygon2(new Point2[] { new(-1, -1), new(2, 3), new(5, -1), new(2, 2) });

        double area      = polygon.Area;
        Point2 cog       = polygon.CenterOfGravity;
        bool   inPolygon = polygon.IsPointInside(cog);

        Assert.AreEqual(-3, area, Program.TOLERANCE);
        Assert.AreEqual(2, cog.X, Program.TOLERANCE);
        Assert.AreEqual(1.333, cog.Y, Program.TOLERANCE);
        Assert.IsFalse(inPolygon);
    }

    [TestMethod]
    public void PolygonFull4()
    {
        var polygon = new Polygon2(new Point2[] { new(-1, -1), new(-1, -2), new(2, -5), new(4, 1), new(2,
-4) });

        double area      = polygon.Area;
        Point2 cog       = polygon.CenterOfGravity;
        bool   inPolygon = polygon.IsPointInside(cog);

        Assert.AreEqual(4, area, Program.TOLERANCE);
        Assert.AreEqual(1.04, cog.X, Program.TOLERANCE);
        Assert.AreEqual(-2.91, cog.Y, Program.TOLERANCE);
        Assert.IsFalse(inPolygon);
    }

    [TestMethod]
    public void PolygonZeroPoints()
    {
        Assert.ThrowsException<ArgumentException>(() => new Polygon2(Array.Empty<Point2>()));
    }

    [TestMethod]
    public void PolygonOnePoint()
    {
        Assert.ThrowsException<ArgumentException>(() => new Polygon2(new Point2[] { new(1, 1) }));
    }

    [TestMethod]
    public void PolygonTwoPoints()
    {
        Assert.ThrowsException<ArgumentException>(() => new Polygon2(new Point2[] { new(1, 1), new(2, 2)
}));
    }
```

```
    [TestMethod]
    public void PolygonNonDistinctPoints()
    {
        Assert.ThrowsException<ArgumentException>(() => new Polygon2(new Point2[] { new(1, 1), new(1, 1),
new(1, 1) }));
    }

    [TestMethod]
    public void PolygonParseSuccess()
    {
        Assert.IsTrue(Polygon2.TryParse("(-1, 1), (-1, -1), (1, -1), (1, 1)", out Polygon2? polygon));
        Assert.IsNotNull(polygon);

        double area      = polygon.Area;
        Point2 cog       = polygon.CenterOfGravity;
        bool   inPolygon = polygon.IsPointInside(cog);

        Assert.AreEqual(4, area, Program.TOLERANCE);
        Assert.AreEqual(0, cog.X, Program.TOLERANCE);
        Assert.AreEqual(0, cog.Y, Program.TOLERANCE);
        Assert.IsTrue(inPolygon);
    }

    [TestMethod]
    public void PolygonParseFail1()
    {
        Assert.IsFalse(Polygon2.TryParse("(-1), (-1, -1), (1, -1), (1, 1)", out Polygon2? polygon));
        Assert.IsNull(polygon);
    }

    [TestMethod]
    public void PolygonParseFail2()
    {
        Assert.IsFalse(Polygon2.TryParse("(-1, 1), (-1 -1), (1, -1), (1, 1)", out Polygon2? polygon));
        Assert.IsNull(polygon);
    }

    [TestMethod]
    public void PolygonParseFail3()
    {
        Assert.IsFalse(Polygon2.TryParse("(-1, 1), (-1, -1), (b, -1), (1, 1)", out Polygon2? polygon));
        Assert.IsNull(polygon);
    }

    [TestMethod]
    public void PolygonParseFail4()
    {
        Assert.IsFalse(Polygon2.TryParse("(-1, 1), (-1, -1)", out Polygon2? polygon));
        Assert.IsNull(polygon);
    }

    [TestMethod]
    public void PolygonString1()
    {
        Polygon2 polygon = new(new Point2[] { new(1, 1), new(-1, 1), new(-1, -1) });

        Assert.AreEqual("(1, 1), (-1, 1), (-1, -1)", polygon.ToString());
    }

    [TestMethod]
    public void PolygonString2()
    {
        Polygon2 polygon = new(new Point2[] { new(0, -6), new(2.5, 8.4), new(-1.2, 99) });

        Assert.AreEqual("(0, -6), (2.5, 8.4), (-1.2, 99)", polygon.ToString());
    }

    [TestMethod]
    public void PolygonEquals1()
    {
        Polygon2 poly1 = new(new Point2[] { new(1, 1), new(-1, 1), new(-1, -1) }),
                 poly2 = new(new Point2[] { new(1, 1), new(-1, 1), new(-1, -1) });

        Assert.IsTrue(poly1 == poly2);
    }

    [TestMethod]
    public void PolygonEquals2()
    {
```

```
        Polygon2 poly1 = new(new Point2[] { new(1, 1), new(-1, 1), new(-1, -1) }),
                 poly2 = new(new Point2[] { new(1, 2), new(-1, 1), new(-1, -1) });

        Assert.IsTrue(poly1 != poly2);
    }

    [TestMethod]
    public void PolygonEquals3()
    {
        Polygon2? poly1 = new(new Point2[] { new(1, 1), new(-1, 1), new(-1, -1) }),
                  poly2 = null;

        Assert.IsFalse(poly1 == poly2);
    }

    [TestMethod]
    public void PolygonEquals4()
    {
        Polygon2? poly1 = new(new Point2[] { new(1, 1), new(-1, 1), new(-1, -1) }),
                  poly2 = null;

        Assert.IsTrue(poly1 != poly2);
    }

    [TestMethod]
    public void PolygonEquals5()
    {
        Polygon2? poly1 = null,
                  poly2 = null;

        Assert.IsTrue(poly1 == poly2);
    }

    [TestMethod]
    public void PolygonEquals6()
    {
        Polygon2? poly1 = null,
                  poly2 = null;

        Assert.IsFalse(poly1 != poly2);
    }

    [TestMethod]
    public void PolygonEquals7()
    {
        Polygon2 poly  = new(new Point2[] { new(1, 1), new(-1, 1), new(-1, -1) });
        Point2[] array = { new(1, 1), new(-1, 1), new(-1, -1) };

        Assert.IsFalse(poly.Equals(array));
    }

    [TestMethod]
    public void PolygonHash()
    {
        Polygon2 poly1 = new(new Point2[] { new(1, 1), new(-1, 1), new(-1, -1) }),
                 poly2 = new(new Point2[] { new(1, 1), new(-1, 1), new(-1, -1) });

        Assert.IsTrue(poly1.GetHashCode() == poly2.GetHashCode());
    }
}
```

Tests/RegexTest

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

using Vache.Utils;


namespace Vache.Tests;

[TestClass]
public class RegexTest
{
```

```
    [TestMethod]
    public void NumberRegex1()
    {
        Assert.IsTrue(Consts.NumberRe.IsMatch("42"));
    }

    [TestMethod]
    public void NumberRegex2()
    {
        Assert.IsTrue(Consts.NumberRe.IsMatch("69.420"));
    }

    [TestMethod]
    public void NumberRegex3()
    {
        Assert.IsTrue(Consts.NumberRe.IsMatch(".666"));
    }

    [TestMethod]
    public void NumberRegex4()
    {
        Assert.IsTrue(Consts.NumberRe.IsMatch("0."));
    }

    [TestMethod]
    public void NumberRegex5()
    {
        Assert.IsTrue(Consts.NumberRe.IsMatch("1e1"));
    }

    [TestMethod]
    public void NumberRegex6()
    {
        Assert.IsTrue(Consts.NumberRe.IsMatch("-18"));
    }

    [TestMethod]
    public void NumberRegex7()
    {
        Assert.IsTrue(Consts.NumberRe.IsMatch("-1e-1"));
    }

    [TestMethod]
    public void NumberRegex8()
    {
        Assert.IsTrue(Consts.NumberRe.IsMatch("+4"));
    }

    [TestMethod]
    public void NumberRegex9()
    {
        Assert.IsFalse(Consts.NumberRe.IsMatch("A"));
    }

    [TestMethod]
    public void NumberRegex10()
    {
        Assert.IsFalse(Consts.NumberRe.IsMatch("."));
    }

    [TestMethod]
    public void NumberRegex11()
    {
        Assert.IsFalse(Consts.NumberRe.IsMatch("e"));
    }
}
```

Tests/PairsTest

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

using Vache.Utils;
```

```
namespace Vache.Tests;

[TestClass]
public class PairsTest
{
    [TestMethod]
    public void PairsTest1()
    {
        int[]        testArray   = { 1, 2, 3, 4, 5 };
        (int, int)[] resultArray = { (1, 2), (2, 3), (3, 4), (4, 5), };

        (int, int)[] cycledArray = testArray.Pairs().ToArray();

        for (var i = 0; i < resultArray.Length; i++)
            Assert.AreEqual(resultArray[i], cycledArray[i]);
    }

    [TestMethod]
    public void PairsTest2()
    {
        int[] testArray =
        {
            7, 2, 1, 8, 1,
            2, 0,
        };
        (int, int)[] resultArray =
        {
            (7, 2), (2, 1), (1, 8), (8, 1), (1, 2),
            (2, 0), (0, 7),
        };

        (int, int)[] cycledArray = testArray.Pairs(true).ToArray();

        for (var i = 0; i < cycledArray.Length; i++)
            Assert.AreEqual(resultArray[i], cycledArray[i]);
    }

    [TestMethod]
    public void PairsTestEmpty()
    {
        int[]        testArray   = Array.Empty<int>();
        (int, int)[] resultArray = Array.Empty<(int, int)>();

        (int, int)[] cycledArray = testArray.Pairs().ToArray();

        Assert.AreEqual(resultArray, cycledArray);
    }

    [TestMethod]
    public void PairsTestCycleOne()
    {
        int[]        testArray   = { 0 };
        (int, int)[] resultArray = { (0, 0) };

        (int, int)[] cycledArray = testArray.Pairs(true).ToArray();

        for (var i = 0; i < cycledArray.Length; i++)
            Assert.AreEqual(resultArray[i], cycledArray[i]);
    }
}
```