```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using HybridShapeTypeLib;
using INFITF;
using KnowledgewareTypeLib;
using MECMOD;
using PARTITF;
using ProductStructureTypeLib;
using System.Runtime.InteropServices;
using System.IO;

namespace Block_erstellen
{
    public partial class Form1 : Form
    {
        //INFITF.Application CATIA;
        MECMOD.Bodies Bodies1;
        MECMOD.Part Part1;
        MECMOD.PartDocument PartDocument1;
        MECMOD.HybridBodies hybridBodies1;
        MECMOD.HybridBody hybridBody1;

        const double PI = 3.1415926535897931;
        string Dateiname = "Daten_" + DateTime.Now.ToString().Replace(@" ", @"_").Replace
    (@".", @"_").Replace(@":", @"_") + ".txt";

        public void DateiErzeugen(string Beginne)
        {
        StreamWriter writer = new StreamWriter(Dateiname, false, System.Text.Encoding.
    Default); //Erzeugt neue Datei (false)
        writer.Write(Beginne+"\r\n"); // Schreibt die Variable Text in die Datei mit
    Zeilenumbruch
        writer.Close(); // Schliesst die Datei
        }
        public Form1()
        {
            InitializeComponent();
            DateiErzeugen("Log-Datei zu dem Programm ");
        }
        private void Catia_starten()
        {
            try //Versuchen Catia zu starten
            {
                object CATIA = System.Runtime.InteropServices.Marshal.GetActiveObject(
    "Catia.Application");
            }
            catch //Falls Fehler auftritt, MsgBox mit Info und Abbruch
            {
                System.Windows.Forms.MessageBox.Show("CATIA nicht gestartet\r\nBitte
    Programm starten", "Fehler", MessageBoxButtons.OK, MessageBoxIcon.Error);
                System.Windows.Forms.Application.Exit();
            }
            try //Versuchen in Catia ein Bauteil zu erstellen
            {
                object CATIA = System.Runtime.InteropServices.Marshal.GetActiveObject(
    "Catia.Application");
                INFITF.Application catiaapp = (INFITF.Application)CATIA;
                string Test;
                Test = "Part";
                this.PartDocument1 = catiaapp.Documents.Add(ref Test) as PartDocument;
                this.Part1 = this.PartDocument1.Part;
            }
            catch //Falls Fehler auftritt, MsgBox mit Info und Abbruch
            {
                System.Windows.Forms.MessageBox.Show("Abbruch des Versuches!\r\nProgramm
    neu starten", "Fehler", MessageBoxButtons.OK, MessageBoxIcon.Error);
                System.Windows.Forms.Application.Exit();
            }
```

```
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Catia_starten(); //Rufe Prozedur Catia starten auf

            MECMOD.Sketch sketch;
            MECMOD.Body body;
            MECMOD.OriginElements OriginElements;
            MECMOD.Factory2D factory2D;
            MECMOD.GeometricElements geometricElements;
            MECMOD.Constraints constraints;
            MECMOD.Point2D point1;
            MECMOD.Point2D[] arrayPoint2D;
            MECMOD.Circle2D Kreis;
            MECMOD.Constraint[] arrayConstraint;
            INFITF.Reference[] reference;
            PARTITF.Pad Block;
            HybridShapeTypeLib.HybridShapePointCoord[] HybridShapePoint;
            HybridShapeTypeLib.HybridShapeFactory HybridShapeFactory1;
            HybridShapeTypeLib.HybridShapeDirection hybridShapeDirection1;
            HybridShapeTypeLib.HybridShapePlane3Points hybridShapePlane3Points;
            HybridShapeTypeLib.HybridShapeLinePtPt hybridLinePtPt;
            HybridShapeTypeLib.HybridShapePlaneNormal hybridShapeNormal;
            PARTITF.ShapeFactory shapeFactory1;

            arrayConstraint = new MECMOD.Constraint[100];
            arrayPoint2D = new Point2D[100];
            reference = new INFITF.Reference[100];
            HybridShapePoint = new HybridShapeTypeLib.HybridShapePointCoord[100];
            double dbl7=1;

            dbl7 = Convert.ToDouble(NUpD_Z1.Value);

            this.Bodies1 = this.Part1.Bodies;
            OriginElements=this.Part1.OriginElements;
            reference[1]=OriginElements.PlaneXY as INFITF.Reference; //Setzt die xy-Ebene ↙
    als Bezugsobject
            body = this.Part1.MainBody;
            sketch = body.Sketches.Add(reference[1]);
            sketch.SetAbsoluteAxisData(new object[] { 0, 0, 0, 1, 0, 0, 0, 1, 0 }); //Fügt↙
     ein Achsensystem ein
            this.Part1.InWorkObject = sketch;

            factory2D = sketch.OpenEdition();
                geometricElements = sketch.GeometricElements;
                constraints = sketch.Constraints;
                point1 = sketch.AbsoluteAxis.Origin;
                point1.ReportName = 1;
                arrayPoint2D[1] = factory2D.CreatePoint(0, 0);
                arrayPoint2D[1].ReportName = 2;
                Kreis = factory2D.CreateClosedCircle(0, 0, 150);
                Kreis.CenterPoint = arrayPoint2D[1];
                Kreis.ReportName = 3;
                Kreis.Construction = false;
                reference[0] = this.Part1.CreateReferenceFromObject(arrayPoint2D[1]);
                reference[1] = this.Part1.CreateReferenceFromObject(point1);
                arrayConstraint[1] = constraints.AddBiEltCst(CatConstraintType.          ↙
    catCstTypeOn,reference[0], reference[1]);
                reference[0] = this.Part1.CreateReferenceFromObject(Kreis);
                arrayConstraint[3] = constraints.AddMonoEltCst(CatConstraintType.        ↙
    catCstTypeRadius, reference[0]);
                arrayConstraint[3].Mode = CatConstraintMode.catCstModeDrivingDimension;
                arrayConstraint[3].Dimension.Value = dbl7 / 2;
            sketch.CloseEdition();
            this.Part1.Update();
            HybridShapeFactory1 = this.Part1.HybridShapeFactory as HybridShapeTypeLib.    ↙
    HybridShapeFactory;
            this.hybridBodies1 = this.Part1.HybridBodies;
            this.hybridBody1 = this.hybridBodies1.Add();

            HybridShapePoint[1] = HybridShapeFactory1.AddNewPointCoord(0, 0, 100); //Fügt ↙
    den Punkt hinzu
            reference[5] = this.Part1.CreateReferenceFromObject(HybridShapePoint[1]); //  ↙
    Erzeugt Bezug?
```

```
            HybridShapePoint[1].Compute(); //Berechnet Lage?
            this.hybridBody1.AppendHybridShape(HybridShapePoint[1]); //Erzeugt den Punkt im Space

            HybridShapePoint[2] = HybridShapeFactory1.AddNewPointCoord(0, 50, 0);
            reference[6] = this.Part1.CreateReferenceFromObject(HybridShapePoint[2]);
            HybridShapePoint[2].Compute();
            this.hybridBody1.AppendHybridShape(HybridShapePoint[2]);

            HybridShapePoint[3] = HybridShapeFactory1.AddNewPointCoord(30, 0, 0);
            reference[7] = this.Part1.CreateReferenceFromObject(HybridShapePoint[3]);
            HybridShapePoint[3].Compute();
            this.hybridBody1.AppendHybridShape(HybridShapePoint[3]);

            hybridShapeDirection1 = HybridShapeFactory1.AddNewDirectionByCoord(0, 1, 0);
            hybridShapeDirection1.Compute();
            this.hybridBody1.AppendHybridShape(hybridShapeDirection1);

            hybridShapePlane3Points = HybridShapeFactory1.AddNewPlane3Points(reference[5], reference[6], reference[7]);
            hybridShapePlane3Points.Compute();
            this.hybridBody1.AppendHybridShape(hybridShapePlane3Points);

            hybridLinePtPt = HybridShapeFactory1.AddNewLinePtPt(reference[5], reference[7]);
            hybridLinePtPt.Compute();
            this.hybridBody1.AppendHybridShape(hybridLinePtPt);
            reference[8]=this.Part1.CreateReferenceFromObject(hybridLinePtPt);

            hybridShapeNormal = HybridShapeFactory1.AddNewPlaneNormal(reference[8], reference[5]);
            hybridShapeNormal.Compute();
            this.hybridBody1.AppendHybridShape(hybridShapeNormal);

            this.Part1.InWorkObject = body;
            shapeFactory1 = this.Part1.ShapeFactory as PARTITF.ShapeFactory;

            reference[4] = this.Part1.CreateReferenceFromObject(sketch);
            //shapeFactory1.AddNewPadFromRef(reference[4], 25).DirectionOrientation = PARTITF.CatPrismOrientation.catInverseOrientation;
            Block=shapeFactory1.AddNewPad(sketch, 30);
            Block.IsSymmetric = true;

            this.Part1.Update();
        }
        private void Berechnen()
        {
            int z1, z2;
            z2 = 13;
            z1 = z2;

        }
        private void Form1_Load(object sender, EventArgs e)
        {
        }
        private void Beenden_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.Application.Exit();
        }
        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked==true)
            {
                NUpD_C.Enabled=true;
            }
            else
            {
                NUpD_C.Enabled = false;
                NUpD_C.Value = Convert.ToDecimal(0.25);
            }
        }
        private void ButBERECHNEN_Click(object sender, EventArgs e)
        {
            Berechnen();
```

```
        }
        public void Ausschreiben(string Text)
        {
            StreamWriter writer = new StreamWriter(Dateiname, true, System.Text.Encoding. ↙
    Default); //Hängt an bestehende Datei Daten an
            writer.WriteLine(Text); // Schreibt die Variable Text in die Datei
            writer.Close(); // Schliesst die Datei
        }

        private void beendenToolStripMenuItem_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.Application.Exit();
        }

    }
}
```