



WS 2017/18

Skript - Kap. 7

Autodesk Revit 2017 SDK (3. Hörsaalübung)

Prof. Dr.-Ing. Uwe Rüppel
Anna Wagner, M.Sc.

- Wiederholung 1. Blockübung
- Revit API
 - Events
 - Parameter ändern
 - Parameter anlegen
 - Objekte platzieren
- Ausgabe 2. Hausübung

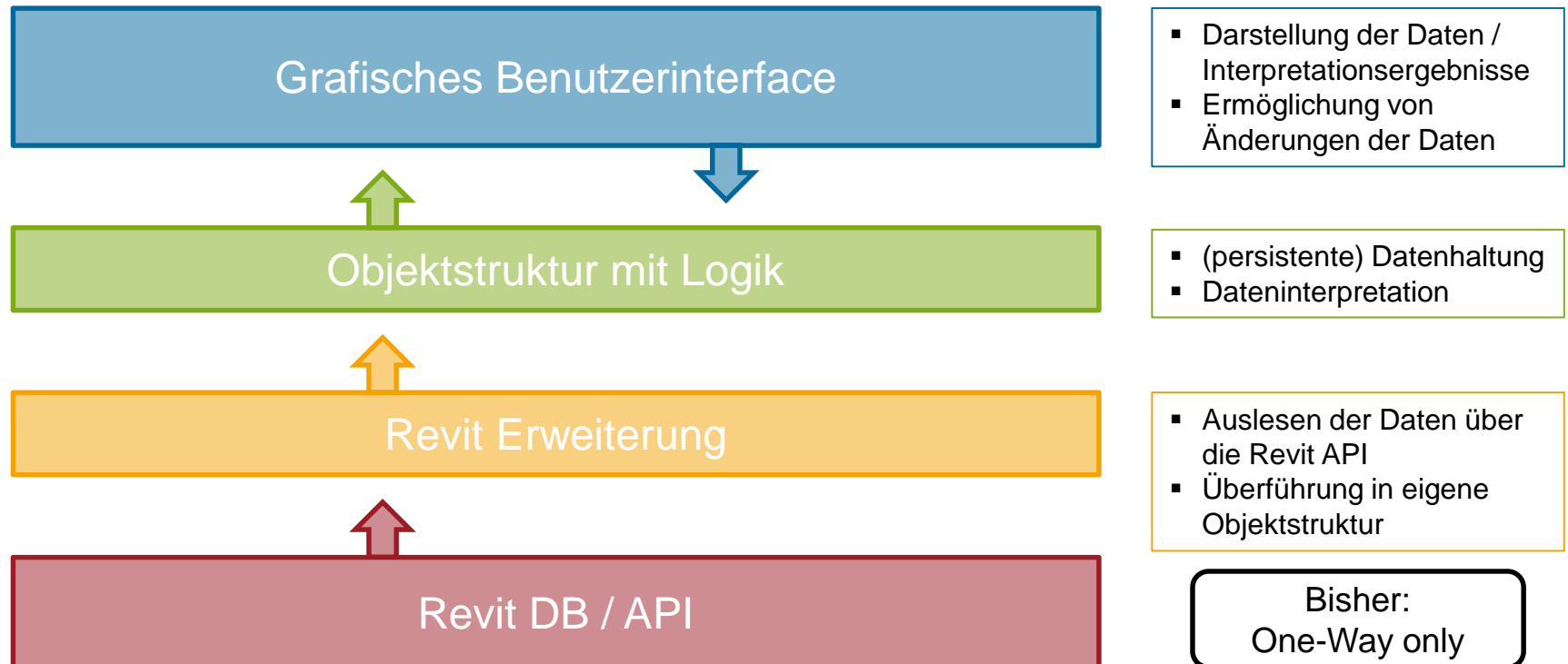


1. Blockübung

Wiederholung

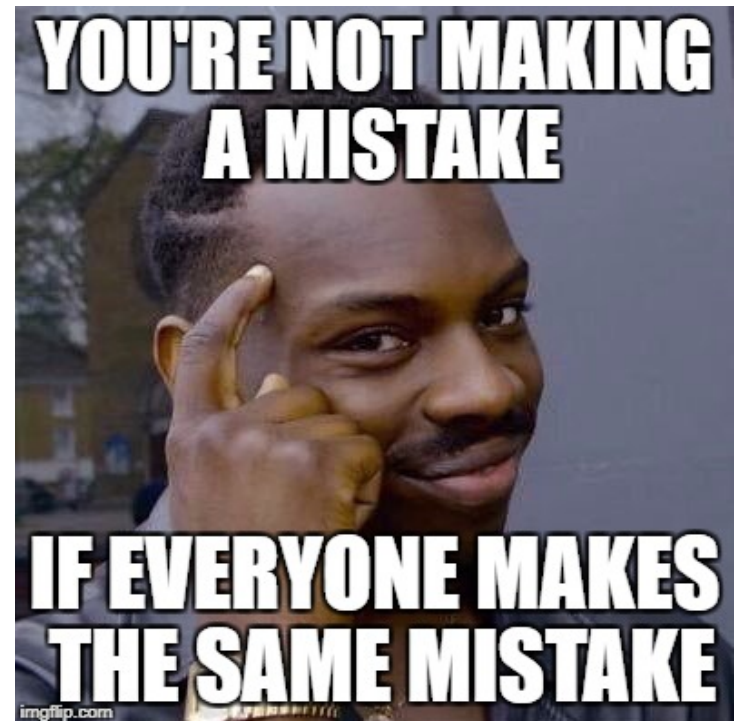


TECHNISCHE
UNIVERSITÄT
DARMSTADT



Häufig aufgetretene Probleme

- Nichteinhaltung der Namenskonvention
- Finden der richtigen Parameterbezeichnungen in Revit
- De-/Serialisierung aus in Revit gestarteter Anwendung



Wieso?

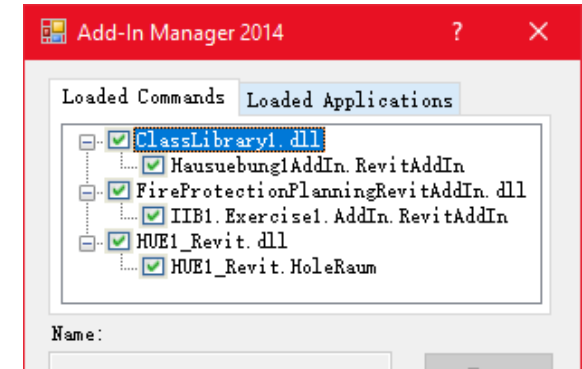
Einfachere Verwaltung mehrerer Abgaben

Projekte:

- IIB1_GruppeXX_Klassen
- IIB1_GruppeXX_GUI
- IIB1_GruppeXX_Revit

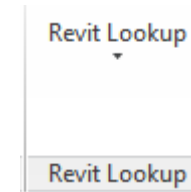
Command-Klasse / Ribbon:

- IIB1_GruppeXX



```
[Autodesk.Revit.Attributes.Transaction(Autodesk.I  
public class IIB1_GruppeXX : IExternalCommand  
{  
    private static BindingList<Raum> meineRaeume  
  
    public Result Execute(ExternalCommandData coi  
    {  
        try {  
  
            UIApplication uiApp = commandData.Api  
            UIDocument mdoc = uiApp.ActiveUIDocu  
            Util.Doc = mdoc.Document;  
  
            IList<Element> Rooms = new FilteredFl
```

Parameterbezeichnungen – Revit Lookup

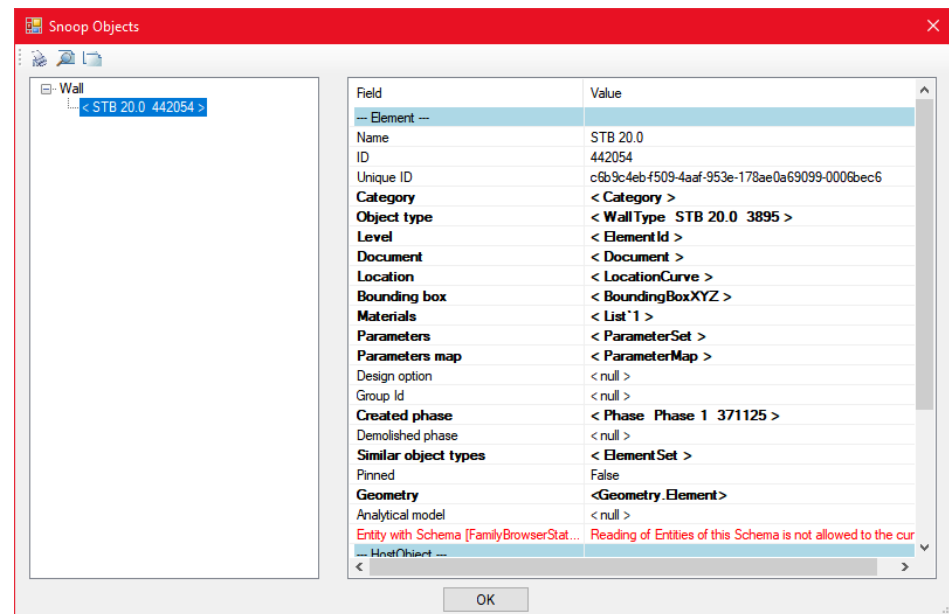


TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Teil des Revit SDK
 - AddIn-Manifest anpassen (Dateipfad zur DLL → bin → Debug)
 - AddIn-Manifest in C:/ProgramData/Autodesk/Revit/Addins/2017 kopieren

- Verschiedene Modi:
 - Snoop Current Selection
 - Snoop DB
 - Snoop Active View
 - ...

- Fett gedruckte Zeilen öffnen neues Fenster mit Feldern des ausgewählten Objekts bei Doppelklick



- Grund: Assembly (der Klassenbibliothek) kann nicht gefunden werden

- Lösung:
 - Alle Projekte in eine Projektmappe laden (nicht nur die Verweise hinzufügen
 - oder
 - Verwendung eines AssemblyResolveEventHandlers, der bei werfen des Fehlers die entsprechende Assembly übergibt

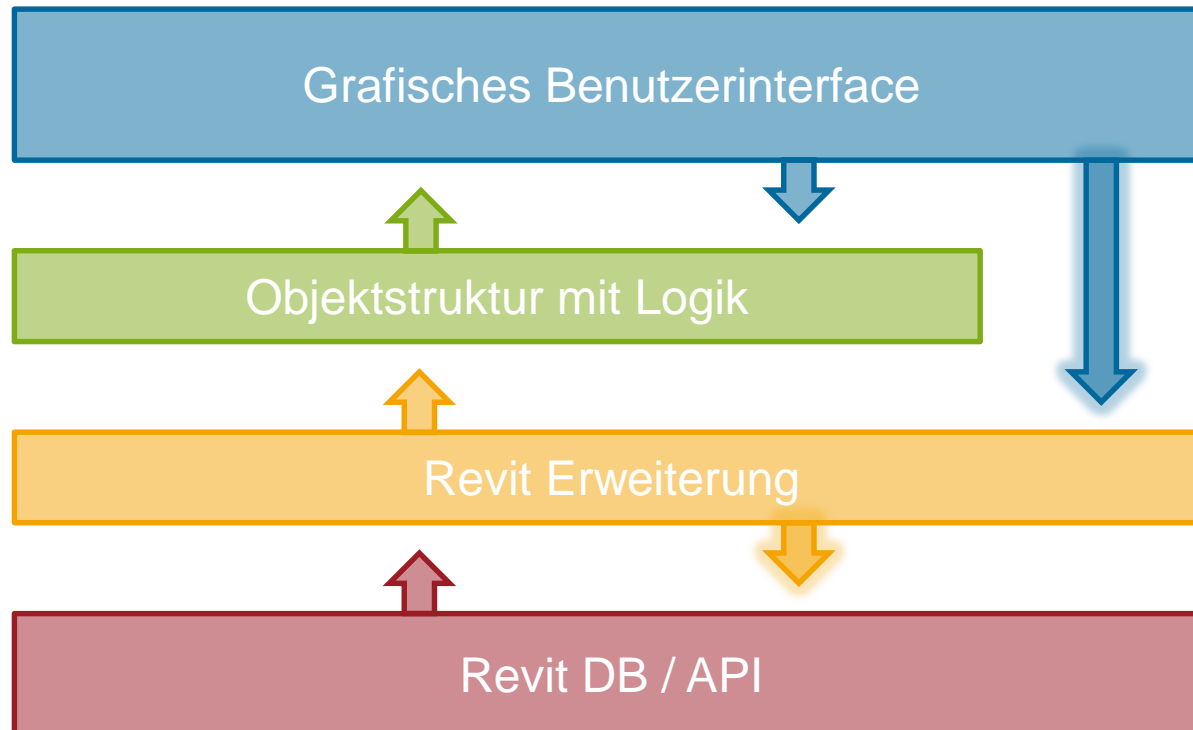
- Daten zurück in Revit schreiben
 - Änderungen in aus Revit gelesenen Objekten
- Neue Informationen in Revit übertragen
 - Ergebnisse von Berechnungen
- Objekte in Revit platzieren
 - Im Programm neu erstellte/zugewiesene Objekte
- Funktionserweiterung der Programme

Revit API



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Änderungen am Revit Document durchführen



- Übergabe der Änderungen in die Revit Erweiterung

- Veränderung der Datenbasis von Revit auf Basis der Nutzereingabe

- Ribbon/Button erstellen
- Raumbezeichnung und -nutzungsart in Revit ändern
- Bei Nichteinhaltung der geforderten Werte Lampe im Raum platzieren

Ribbon/Button erstellen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhalte:

- Typen von Revit-Erweiterungen
- AddIn Manifest Datei
- Ribbon und Button
- Auszuführende Commands

External Command:

- Können über den „External Tools“ Button aufgerufen werden
- Command wird bei Benutzereingabe ausgeführt (Execute()-Methode)

External Application:

- Können mehrere External Commands verwalten
- Können eigene Tabs, Ribbons und Buttons in der Revit GUI haben
- Application wird beim Start von Revit geladen
 - u.a. Erzeugung von Tabs, Ribbons und Buttons
- Bei Betätigung eines Buttons wird in Application bestimmter Command ausgeführt

AddIn Manifest (1)

- Muss in Ordner C:\ProgramData\Autodesk\Revit\Addins\2017 abgelegt werden
- Definiert eine Erweiterung
 - Dateipfad zur DLL und voller Klassenname (mit Namespace)
 - Bezeichnung der Anwendung (Namenskonvention !)
 - Urheber / Hersteller
 - **Eindeutige** ID für das AddIn (→ The Building Coder)
 - Es kann nicht mehrere Erweiterungen mit der selben GUID geben
- Wird bei Start von Revit ausgelesen und weiterverarbeitet
 - Commands werden „External Tools“-Button hinzugefügt
 - Applications werden ausgeführt

weitere Hilfe:

<https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2017/ENU/Revit-API/files/GUID-7577712B-B09F-4585-BE0C-FF16A5078D29-htm.html>

(Aufbau und weitere Infos zum Thema Manifest)

<http://thebuildingcoder.typepad.com/blog/2010/04/addin-manifest-and-guidize.html> (s.o. plus Anleitung zur Erstellung einer neuen GUID)

AddIn Manifest (2)

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<RevitAddIns>
```

```
<AddIn Type="Application">
```

Erweiterungsart

```
<Name>IIB 1 Demonstrator</Name>
```

```
<Assembly>[DATEIPFAD]\IIB1_Demonstrator_AddIn.dll</Assembly>
```

```
<AddInId>c47e9a59-b5dd-40f5-a443-0bde9b0957cc</AddInId>
```

Muss unique sein

```
<FullClassName>IIB1_Demonstrator_AddIn.RevitAddIn</FullClassName>
```

```
<VendorId>Anna Wagner</VendorId>
```

Namespace.Klassenname

```
<VendorDescription>Informatik im Bauwesen 1</VendorDescription>
```

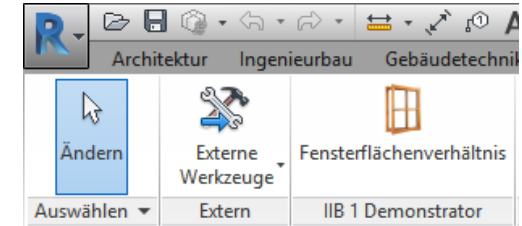
```
</AddIn>
```

```
</RevitAddIns>
```

Ribbon erstellen

Erstellen eines neuen Ribbons:

```
String panelName = "IIB 1 Demonstrator";  
RibbonPanel ribbonDemoPanel = application.CreateRibbonPanel(panelName);
```



Hinzufügen eines Buttons mit Bild:

```
PushButton myButton = (PushButton)ribbonDemoPanel.AddItem(new PushButtonData("IIB1Demo",  
    "Fensterflächenverhältnis", AddInPath, "IIB1_Demonstrator_AddIn.StartApp"));
```

Text des Buttons

namespace und Bezeichnung der Klasse, die aufgerufen werden soll

```
myButton.LargeImage = new BitmapImage(new Uri(Path.Combine(ButtonIconsFolder, "Fenster.png"),  
    UriKind.Absolute));
```

Hinzufügen eines ToolTips:

Text, der angezeigt wird

```
myButton.ToolTip = "Öffnet ein Tool, in dem das Fensterflächenverhältnis bestimmt wird.";
```

```
myButton.ToolTipImage = new BitmapImage(new Uri(Path.Combine(ButtonIconsFolder,  
    "panoramafenster.jpg"), UriKind.Absolute));
```

Bild, das angezeigt wird

Event für den Button hinzufügen

Neue Klasse vom Typ “IExternalCommand”

```
[Transaction(TransactionMode.Manual)]  
public class StartApp : IExternalCommand  
{  
    public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)  
    {  
        ...  
        RevitAddIn.thisApp.ShowForm(meineRaeume);  
        return Autodesk.Revit.UI.Result.Succeeded;  
    }  
}
```

Gleicher Name wie der beim
Button angegebene

Beim Ausführen werden zunächst alle
Räume & Fenster ausgelesen (...)

Anschließend wird in der ShowForm()-
Methode der RevitAddIn-Klasse das Form
gestartet

Hands On !



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erzeugt ein Ribbon mit Button für das Demonstrator-AddIn

- Einstiegsklasse vom Typ IExternalApplication
- AddIn-Manifest schreiben / anpassen und verschieben
- Ribbon mit Button & ToolTip erzeugen
- Neue Klasse vom Typ IExternalCommand erzeugen und mit Button verlinken
- Auslesen der Informationen in Command-Klasse / Util
- Aufrufen der Form aus Einstiegsklasse
- Funktionstest

Dauer: 30 Minuten

Parameter in Revit ändern



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhalte:

- Rechte in Revit
- Externe Events
- Transactions
- Parameter

Revit aus einem externen Programm bedienen

Problem:

Revit API erlaubt nur bedingt Zugriffe von externen Threads

- Transactions können nur aus einem validen Revit Kontext aufgerufen werden

Lösung:

- ExternalEvents – von einer externen Anwendung getriggertes Event
- Idling Event – Frequenz der Abrufe nicht kontrollierbar, kann die CPU auslasten

Thread: kleinster sequentieller Abarbeitungsstrang innerhalb eines Prozesses



- Von der RevitAPI gestellte Events
- Verwendung ähnlich normaler Events in Forms

Schritte:

1. Erstellen einer geschachtelten Klasse vom Typ `IExternalEventHandler` in `RevitAddIn.cs`
 - Vom Interface erforderte Methoden: `Execute(UIApplication app)` und `GetName()`
 - `Execute` führt gewünschte Funktionen aus
2. Erzeugen eines Events unter Verwendung des erstellten EventHandlers
3. Übergabe des Events an die aufgerufene Form
4. In der Form an gewünschter Stelle das Event mit der `Raise()` Funktion triggern

Weitere Infos:

<http://thebuildingcoder.typepad.com/blog/2015/12/external-event-and-10-year-forum-anniversary.html#3>

<http://thebuildingcoder.typepad.com/blog/2012/04/idling-enhancements-and-external-events.html>

SDK Samples - ModelessDialog

Externe Events (2) – Implementierung

Implementierung der Klasse in RevitAddIn.cs (geschachtelte Klasse):

```
public class RaumdatenUpdater : IExternalEventHandler
{
    public void Execute(UIApplication app)
    {
        Util.updateRaumDaten(demoForm.Raeume);
    }
    public string GetName()
    {
        return "RaumdatenUpdater";
    }
}
```

Ruft Util-Methode, die Änderungen
in Revit überträgt, auf

Externe Events (3) – Erzeugung Event(Handler)

Übergabe des Events:

```
public void ShowForm(BindingList<Raum> revitRaeume)
{
    if (demoForm == null || demoForm.IsDisposed)
    {
        RaumdatenUpdater updateHandler = new RaumdatenUpdater();
        ExternalEvent updateEvent = ExternalEvent.Create(updateHandler);
        demoForm = new FormMain(updateEvent, revitRaeume);
        demoForm.Show();
    }
}
```

Erzeugt Event und übergibt dieses
dem Form im Konstruktor

Externe Events (4) – Aufrufen Event

In FormMain:

```
public FormMain(ExternalEvent update, BindingList<Raum> _raeume)
{
    this.ex_updateEvent = update;
    InitializeComponent();
    this.raeume = _raeume;
    fuelleListe();
}
```

Speichern des Events als globale Variable

```
private void buttonUpdate_Click(object sender, EventArgs e)
{
    ex_updateEvent.Raise();
}
```

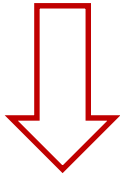
Triggert das Event in der RevitAddIn-Klasse

- Veränderungen am aktiven Revit-Dokument können über Transactions vorgenommen werden (Zur Erinnerung: Revit-Modell = Datenbank)
- Eigenschaften / Methoden:
 - `Start()` → Startet Transaction-Kontext
 - `Commit()` → Beendet Transaction und übergibt alle Änderungen an das Dokument
 - `Rollback()` → Beendet Transaction und ignoriert alle Änderungen des Dokuments
 - `GetStatus()` → liefert aktuellen Status der Transaction (Uninitialized, Started, RolledBack, Committed, Pending)

Transactions – Hinweise

```
Transaction trans = new Transaction(doc);  
trans.Start(„MyTransaction“);  
// Mache Veränderungen am Document  
trans.Commit();
```

besser



```
using (Transaction trans = new Transaction(doc))  
{  
    if (trans.Start("MyTransaction") == TransactionStatus.Started)  
    {  
        // Mache Veränderungen am Document  
        trans.Commit();  
    }  
}
```

ggf.



```
if (TransactionStatus.Committed != trans.Commit())  
{  
    TaskDialog.Show("Failure", "Transaction could not be committed");  
}
```

- Transactions immer so nah wie möglich an der Stelle im Code platzieren, wo die Veränderungen am Document vorgenommen werden!
- Transactions immer in **using**-Blöcken verwenden!

Properties in Revit ändern

Verwendete Methode: `Parameter.Set("Wert")`

```
room.Number = r.RaumNummer;
```

```
if (zugehörigeNutzungsart(r) != "")
```

```
    room.GetParameters(nutzungsart).First().Set(zugehörigeNutzungsart(r));
```

mit

Room room: betrachteter Raum – Durch RevitId von r aus Revit
Dokument auslesen

zugehörigeNutzungsart: Methode, die Raumklasse entsprechende
Nutzungsart nach DIN 277-2 zurückgibt

nutzungsart: Globale, finale Variable mit Bezeichnung des Parameters

→ *Für alle Räume des Gebäudes durchführen!*

- Nicht alle Parameter sind editierbar
- Fehler: „The parameter is read-only“
- Ursachen
 - Parameter setzt sich aus anderen Parametern zusammen
 - z.B. Raumname = Raumschlüssel + Raumnummer
 - Parameter wurde als nicht-editierbar bestimmt
- Einsehen über LookUp:
 - Parameterliste auswählen
 - Gewünschten Parameter auswählen

Field	Value
-- APIObject --	
Is read-only	True
-- Parameter --	

Hands On !



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erstellt ein Event und Util-Methoden, um die Änderungen in Revit zu übernehmen

- In RevitAddIn.cs:
 - ExternalEvent „RaumdatenUpdater“ implementieren
 - Aufrufen der Util-Methode „updateRaumDaten(BindingList<Raum>)“ in Execute() Methode
 - Event in Methode ShowForm() deklarieren & initialisieren
 - Event an Form übergeben
- In FormMain.cs:
 - Übergebenes Event als globale Variable speichern
 - Bei Betätigung des „buttonUpdate“ Event raisen
- In Util.cs:
 - Parameter in Revit überschreiben
 - Raumnummer
 - Raumnutzungsart
 - Über alle Räume iterieren
- Funktionstest

Dauer: 45 Minuten

Platzieren von Objekten in Revit



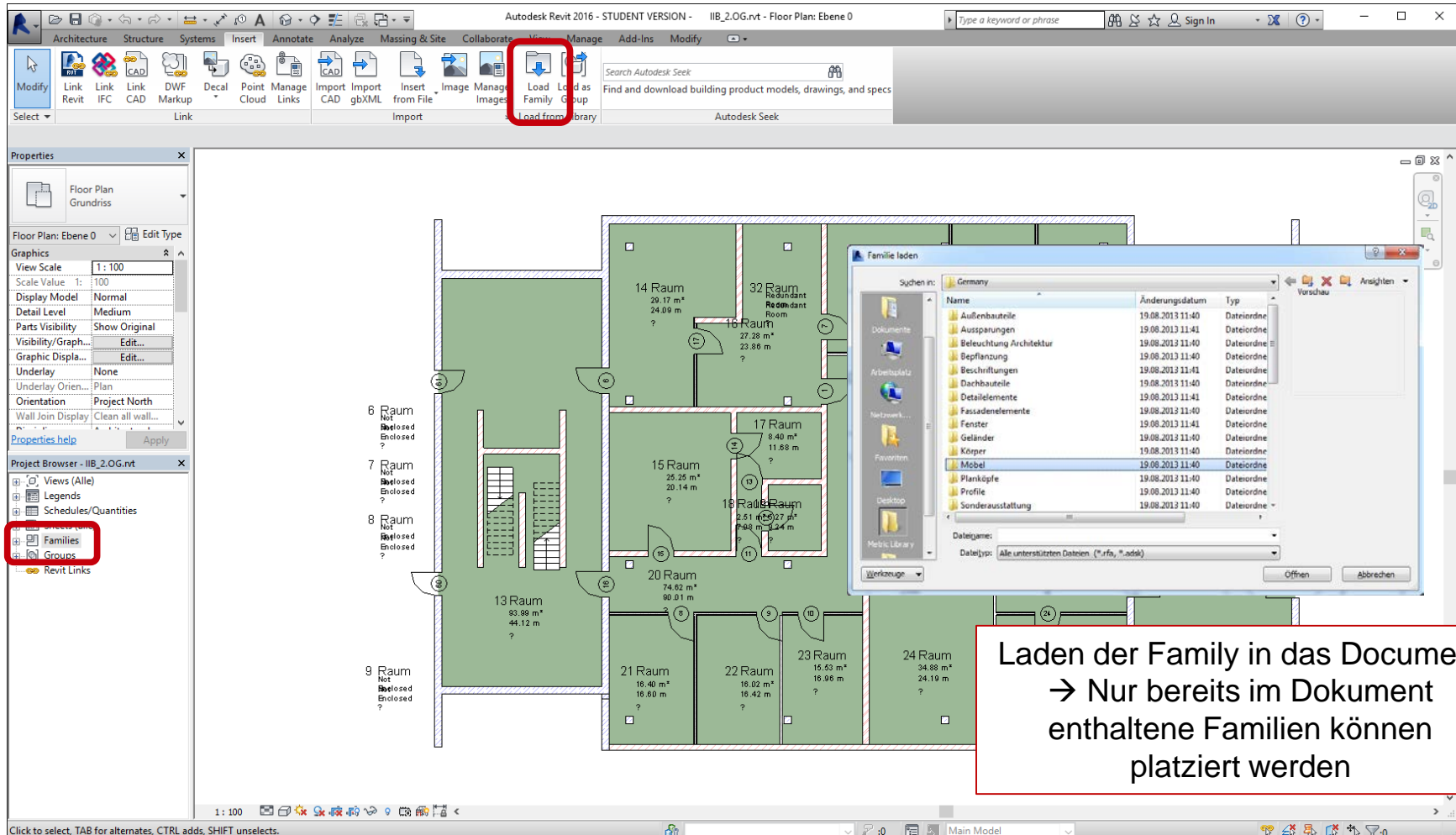
TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhalte:

- Revit Families
- Laden von Families (UI & API)
- Platzieren von Families

- Eine **Family** ist eine Gruppe/Kategorie von Elementen
 - Gemeinsame Eigenschaften (Parameter) und ähnliche grafische Repräsentation
- Verschiedene Elemente einer Familie können verschiedene Werte für ihre Parameter haben, man spricht deshalb von Family Types
- Family Types werden in der API durch die Klasse **FamilySymbol** repräsentiert
- Fügt man ein Element einer bestimmten Family und eines bestimmten Family Types (d.h. ein FamilySymbol) zum Document hinzu, so spricht man von einer **FamilyInstance**
 - Jede Family Instance hat ein Set von Eigenschaften, welche unabhängig von den Eigenschaften des Family Types geändert werden können. Derartige Änderungen betreffen ausschließlich dieses eine Element.
 - Änderungen an Eigenschaften des Family Types hingegen haben Einfluss auf alle Elemente dieses Typs im gesamten Projekt.

Load Family (UI)



Autodesk Revit 2016 - STUDENT VERSION - IIB_2.OG.rvt - Floor Plan: Ebene 0

Architecture Structure Systems Insert Annotate Analyze Massing & Site Collaborate Manage Add-Ins Modify

Search Autodesk Seek
Find and download building product models, drawings, and specs
Autodesk Seek

Properties

Floor Plan Grundriss

Floor Plan: Ebene 0 Edit Type

Graphics

View Scale 1:100

Scale Value 1:100

Display Model Normal

Detail Level Medium

Parts Visibility Show Original

Visibility/Graph... Edit...

Graphic Displa... Edit...

Underlay None

Underlay Orien... Plan

Orientation Project North

Wall Join Display Clean all wall...

Properties help Apply

Project Browser - IIB_2.OG.rvt

Views (Alle)

Legends

Schedules/Quantities

Families

Groups

Revit Links

Famile laden

Suchen in: Germany

Name	Änderungsdatum	Typ
Außenbauteile	19.08.2013 11:40	Dateiordne
Aussparungen	19.08.2013 11:41	Dateiordne
Beleuchtung Architektur	19.08.2013 11:40	Dateiordne
Bepflanzung	19.08.2013 11:40	Dateiordne
Beschriftungen	19.08.2013 11:41	Dateiordne
Dachbauteile	19.08.2013 11:40	Dateiordne
Detailelemente	19.08.2013 11:41	Dateiordne
Fassadenelemente	19.08.2013 11:40	Dateiordne
Fenster	19.08.2013 11:41	Dateiordne
Geländer	19.08.2013 11:40	Dateiordne
Körper	19.08.2013 11:40	Dateiordne
Möbel	19.08.2013 11:40	Dateiordne
Planköpfe	19.08.2013 11:40	Dateiordne
Profile	19.08.2013 11:40	Dateiordne
Sonderausstattung	19.08.2013 11:40	Dateiordne

14 Raum 29.17 m² 24.09 m ?

16 Raum 27.28 m² 23.86 m ?

17 Raum 8.40 m² 11.68 m ?

15 Raum 25.25 m² 20.14 m ?

18 Raum 2.51 m² 2.27 m ?

20 Raum 74.62 m² 90.01 m ?

13 Raum 93.99 m² 44.12 m ?

21 Raum 16.40 m² 16.60 m ?

22 Raum 16.02 m² 16.42 m ?

23 Raum 15.53 m² 16.96 m ?

24 Raum 34.88 m² 24.19 m ?

8 Raum Not Reflosed Enclosed ?

7 Raum Not Reflosed Enclosed ?

9 Raum Not Reflosed Enclosed ?

32 Raum Redundant Room

1:100

Click to select, TAB for alternates, CTRL adds, SHIFT unselects.

Main Model

Laden der Family in das Document
→ Nur bereits im Dokument
enthaltene Familien können
platziert werden

Revit API Grundlagen – Load Family (API)

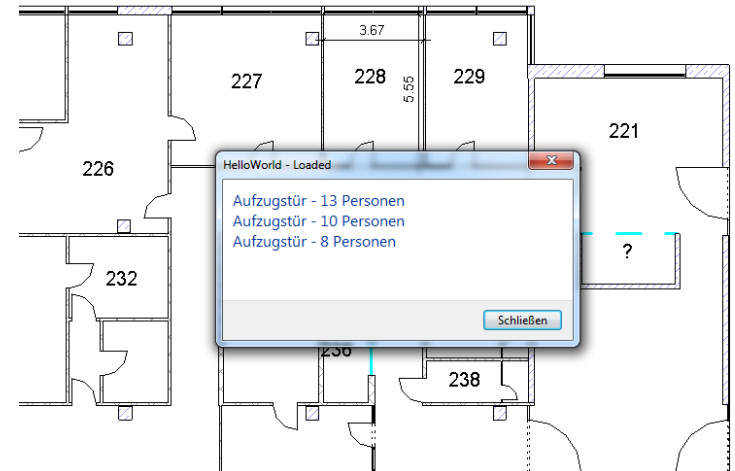
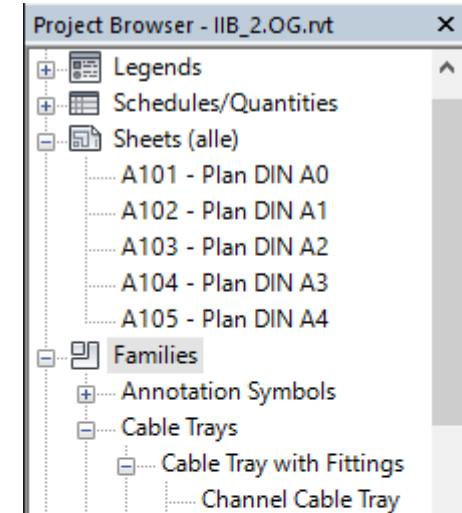


TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
public void loadFamilyExample(Document doc){
    string fileName = @"C:\ProgramData\Autodesk\RVT 2016\Libraries\" +
        "Germany\Sonderausstattung\Rolltreppen und Aufzüge\Aufzugstür.rfa";
    Family family = null;
    using (Transaction t = new Transaction(doc)) {
        if (t.Start("LoadFamily") == TransactionStatus.Started) {
            // try to load family
            if (!doc.LoadFamily(fileName, out family)) {
                throw new Exception("Unable to load " + fileName);
            }
            t.Commit();
        }
    }

    // loop through family symbols
    FamilySymbolSet familySymbolsId= family.GetFamilySymbolIds();
    string symbolNames = "";
    foreach (ElementId symbolId in familySymbolsId) {
        symbolNames += family.Name + " - " + ((FamilySymbol)
            family.Document.GetElement(symbolId)).Name + "\n";
    }
    TaskDialog.Show("Loaded", symbolNames);
}
```

Laden der Family in das Document
via API & Dateipfad zur .rfa-Datei



FamilySymbol auswählen

Filterung aller Elemente des Dokumentes nach

- BuiltInCategory
- Klasse(FamilySymbol)
- Name

```
private static FamilySymbol GetFamilySymbolByName(BuiltInCategory bic, string name)
{
    return new FilteredElementCollector(doc).OfCategory(bic)
        .OfClass(typeof(FamilySymbol)).FirstOrDefault<Element>
        (e => e.Name.Equals(name)) as FamilySymbol;
}
```

Im Demonstrator – Lampe:

BuiltInCategory: OST_LightingFixtures

Name: " F - 1850 x 450 "

Platzierung der Lampen ermitteln

Abhängig von Platzierung der Raumbeschriftung

```
Room rr = doc.GetElement(r.RevitId) as Room;  
XYZ locR = ((LocationPoint) rr.Location).Point;
```

Erzeugung & Platzierung neue FamilyInstance

```
FamilyInstance fi = doc.Create.NewFamilyInstance(locR,  
    GetFamilySymbolByName(BuiltInCategory.OST_LightingFixtures,  
        "F - 1850 x 450") , StructuralType.NonStructural);
```

Hands On !



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erstellt ein Event und Util-Methoden, um Lampen in allen Räumen, die den Grenzwert nicht einhalten, zu platzieren

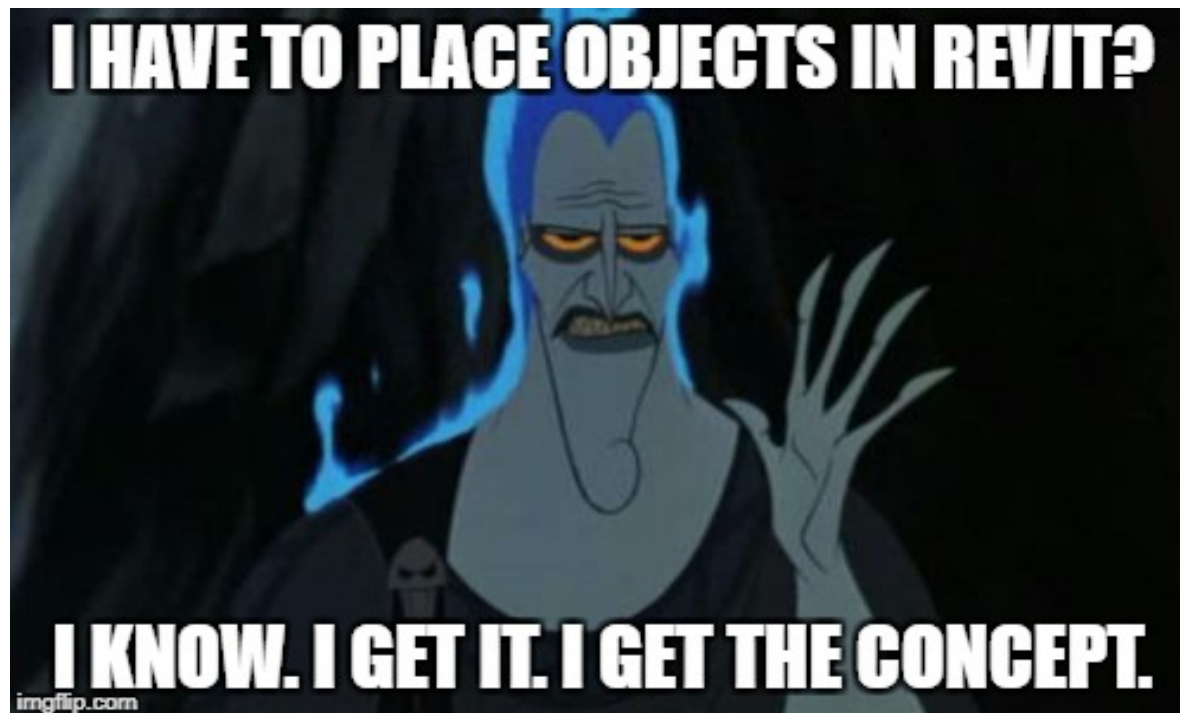
- In RevitAddIn.cs:
 - ExternalEvent „LampenPlatzierer“ implementieren
 - Aufrufen der Util-Methode „platziereLampen(BindingList<Raum>)“ in Execute() Methode
 - Event in Methode ShowForm() deklarieren & initialisieren
 - Event an Form übergeben
- In FormMain.cs:
 - Übergebenes Event als globale Variable speichern
 - Bei Betätigung des „buttonLampen“ Event raisen
- In Util.cs:
 - Familie "F - 1850 x 450" laden
 - Vorher einmalig händisch ins Dokument laden (UI)
 - Bei Nichteinhaltung des Grenzwertes im Raum platzieren
 - Über alle Räume iterieren
- Funktionstest

Dauer: 30 Minuten

Weitere nützliche Funktionen

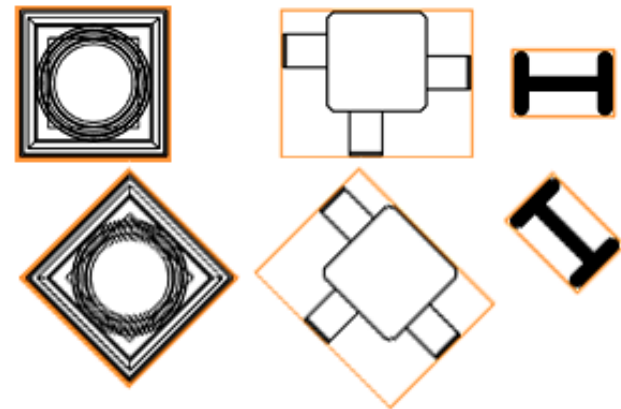
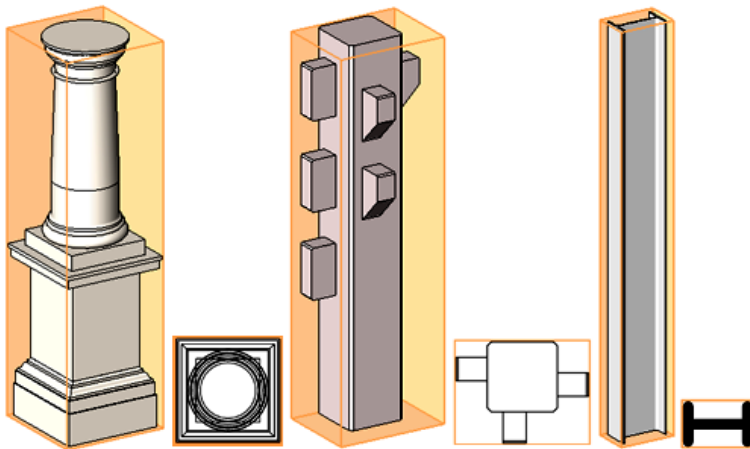


TECHNISCHE
UNIVERSITÄT
DARMSTADT



- Unsichtbarer 3D-Hüllkörper, der die minimalen und maximalen Ausdehnungen eines Körpers beschreibt

```
BoundingBoxXYZ b = element.get_BoundingBox(doc.ActiveView);  
XYZ max = b.Max;  
XYZ min = b.Min;
```



- Dient bei Benutzerauswahl als Filter
 - Erlaubt Auswahl der definierten Elemente

```
public class RaumFilter : ISelectionFilter
{
    Document doc = null;
    public RaumFilter(Document _doc)
    {
        doc = _doc;
    }

    bool ISelectionFilter.AllowElement(Element elem)
    {
        return elem is Room;
    }

    bool ISelectionFilter.AllowReference(Reference reference, XYZ position)
    {
        return true;
    }
}
```

Neue Parameter erstellen

- Neue Parameter in .txt Datei definieren
- Zum Start von Revit einlesen und anlegen
- Ausführlicher Blogeintrag und öffentliches Beispielprojekt von Jeremy Tammik (The Building Coder):
 - Building Coder Projekt:
<https://github.com/jeremytammik/PopulateMaterialProperty/blob/master/PopulateMaterialProperty/ExportParameters.cs>
 - Building Coder Blog-Eintrag:
<http://thebuildingcoder.typepad.com/blog/2009/06/model-group-shared-parameter.html>

CopyElement

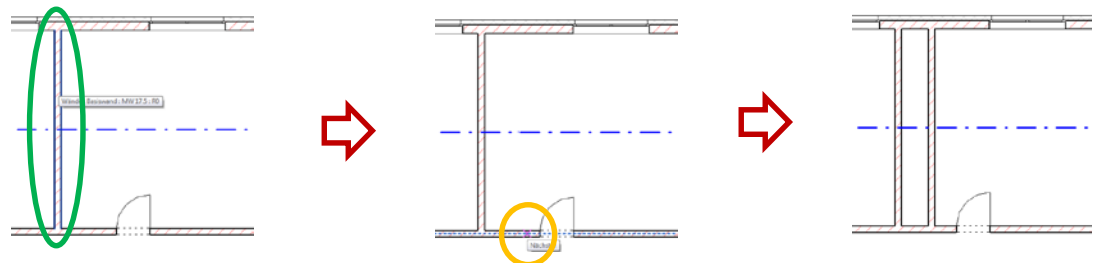
```
public void copyWallExample(ExternalCommandData revit)
{
    UIDocument uidoc = revit.Application.ActiveUIDocument;
    Document doc = uidoc.Document;
    Selection sel = uidoc.Selection;

    1 Reference picked = sel.PickObject(ObjectType.Element, new ElementSelectionFilter<Wall>(), "Pick a wall.");
    Wall wall = doc.GetElement(picked) as Wall;

    2 XYZ point = revit.Application.ActiveUIDocument.Selection.PickPoint("Pick a point to place wall.");

    using (Transaction trans = new Transaction(doc))
    {
        if (trans.Start("CopyWall") == TransactionStatus.Started)
        {
            ElementTransformUtils.CopyElement(doc, wall.Id, point);
            trans.Commit();
        }
    }
}
```

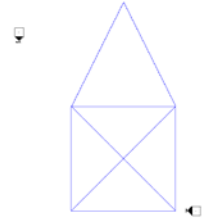
Beispiel: Kopieren einer
ausgewählten Wand an
einen gewünschten Punkt



Auf eine Ebene zeichnen

```
public void drawOnPlane(Document doc)
{
    using (Transaction trans = new Transaction(doc))
    {
        if (trans.Start(„Draw“) == TransactionStatus.Started)
        {
            Plane planeXY = doc.Application.Create.NewPlane(new XYZ(100, 0, 0), new XYZ(0, 100, 0), XYZ.Zero);
            SketchPlane sketchPlane = SketchPlane.Create(doc, planeXY);

            ModelCurveArray lines = new ModelCurveArray();
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(25, 0, 0), new XYZ(25, 50, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(25, 50, 0), new XYZ(75, 50, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(75, 50, 0), new XYZ(75, 0, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(75, 0, 0), new XYZ(25, 0, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(25, 0, 0), new XYZ(75, 50, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(25, 50, 0), new XYZ(75, 0, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(25, 50, 0), new XYZ(50, 100, 0)), sketchPlane));
            lines.Append(doc.Create.NewModelCurve(Line.CreateBound(new XYZ(75, 50, 0), new XYZ(50, 100, 0)), sketchPlane));
            trans.Commit();
        }
    }
}
```



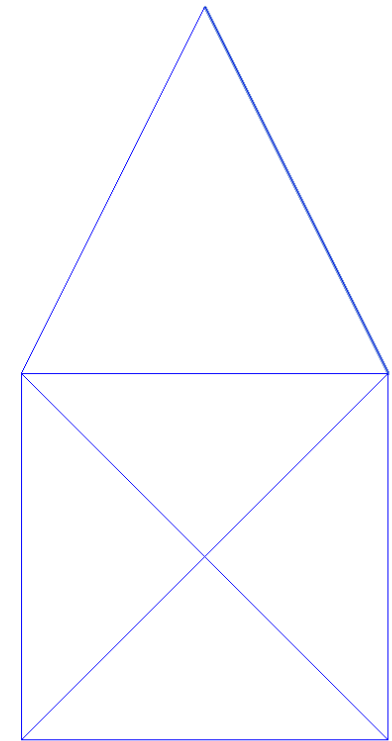
```
public void textNoteExample(Document doc, string text)
{
    using (Transaction trans = new Transaction(doc))
    {
        if (trans.Start(„Texting“) == TransactionStatus.Started)
        {
            TextNote textNote = doc.Create.NewTextNote(
                doc.ActiveView,
                new XYZ(-10,0,0),
                new XYZ(0,0,0),
                new XYZ(0,0,1),
                2.0,
                TextAlignFlags.TEF_ALIGN_CENTER,
                text
            );
            trans.Commit();
        }
    }
}

/* ... */
textNoteExample(doc, "Das ist das Haus vom Nikolaus :P");
```

Bei Änderungen am aktuellen Dokument sind immer Transactions erforderlich!



Das ist das Haus vom Nikolaus :P



Hilfreiche Foren / Internetadressen

- MSDN Library ([https://msdn.microsoft.com/de-de/library/w0x726c2\(v=vs.110\).aspx](https://msdn.microsoft.com/de-de/library/w0x726c2(v=vs.110).aspx))
- Stackoverflow (<http://stackoverflow.com/>)
 - Beachtet den grünen Haken!
- Jeremy Tammik's Blog (<http://thebuildingcoder.typepad.com/>)



October 27, 2016

AI, Edit and Continue



I am still in Munich supporting the one-week Forge accelerator workshop, returning back to Switzerland by train tonight. For ecological reasons, I prefer to avoid flying whenever I

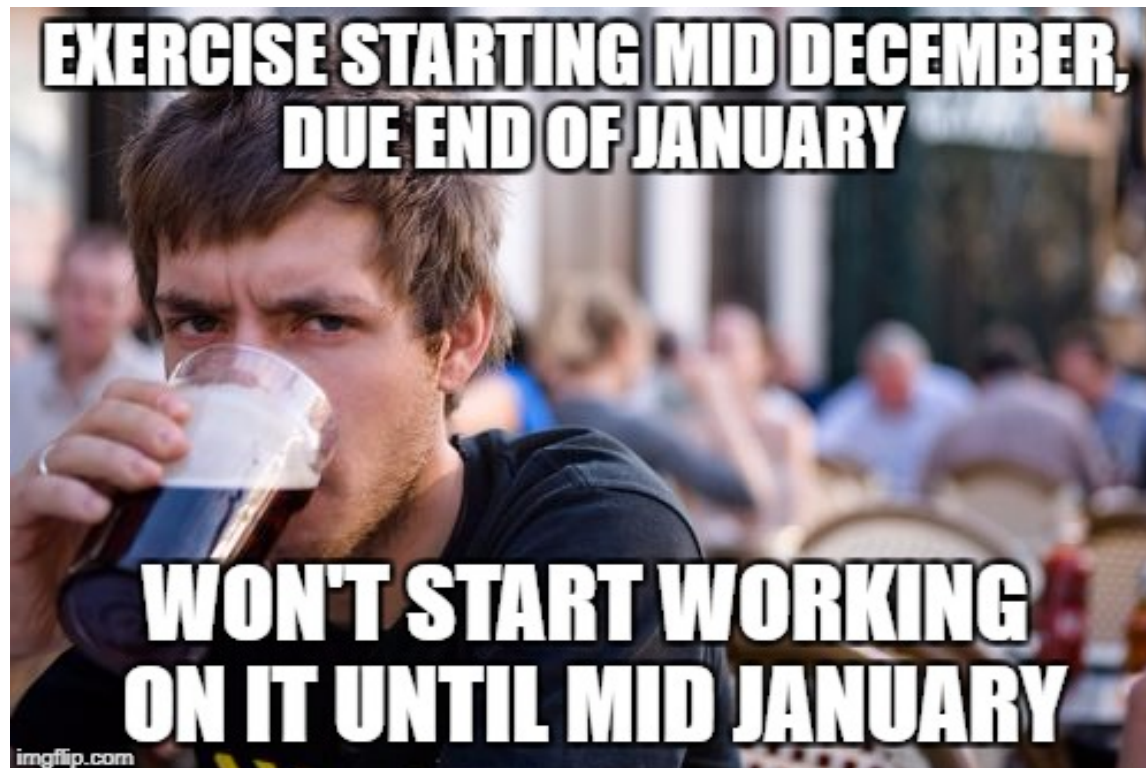


Hausübung 2



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorstellung der Aufgabenstellungen



Aufgabe 1 + 2

Aufbauend auf Programm der 1. Blockübung

Übertragung der Änderungen in das Revit Dokument

- Ändern von Parametern
- Hinzufügen neuer Parameter
- Platzieren von Familien

Aufgabe 3

Korrektur / Bewertung von UML Diagrammen

- Diagramme einer anderen Aufgabenvariante
- Zuteilung von Diagrammen bis 22.12.2017 in Moodle
- Verwendung von Formular (ab 22.12.2017 in Moodle zur Verfügung gestellt)
- Positive wie auch Negative Kritik

Viel Erfolg !

- Abgabe: 21.01.2018, 23:55 Uhr, Moodle
- Betreuungstermine (L5|01 – 222) :
 - Fr., 12.01.2018 10:00 – 17:00 Uhr
 - Di., 16.01.2018 10:00 – 15:00 Uhr
 - Do., 18.01.2018 13:00 – 18:00 Uhr
 - Fr., 19.01.2018 10:00 – 17:00 Uhr
- Kolloquien:
 - KW 6

