3. Gyakorlat: OpenAmeos

Simon Balázs, BME IIT, 2013.

1 Feladat

Készítsünk kódgenerátort OpenAmeos-ban, amely egyszerűsített Java kódot készít UML osztálydiagram alapján!

Lépések:

1. Olvassuk el a 2. fejezetet, amely az OpenAmeos használatát mutatja be!
2. Készítsük el a 3. fejezetben leírt kódgenerátort!

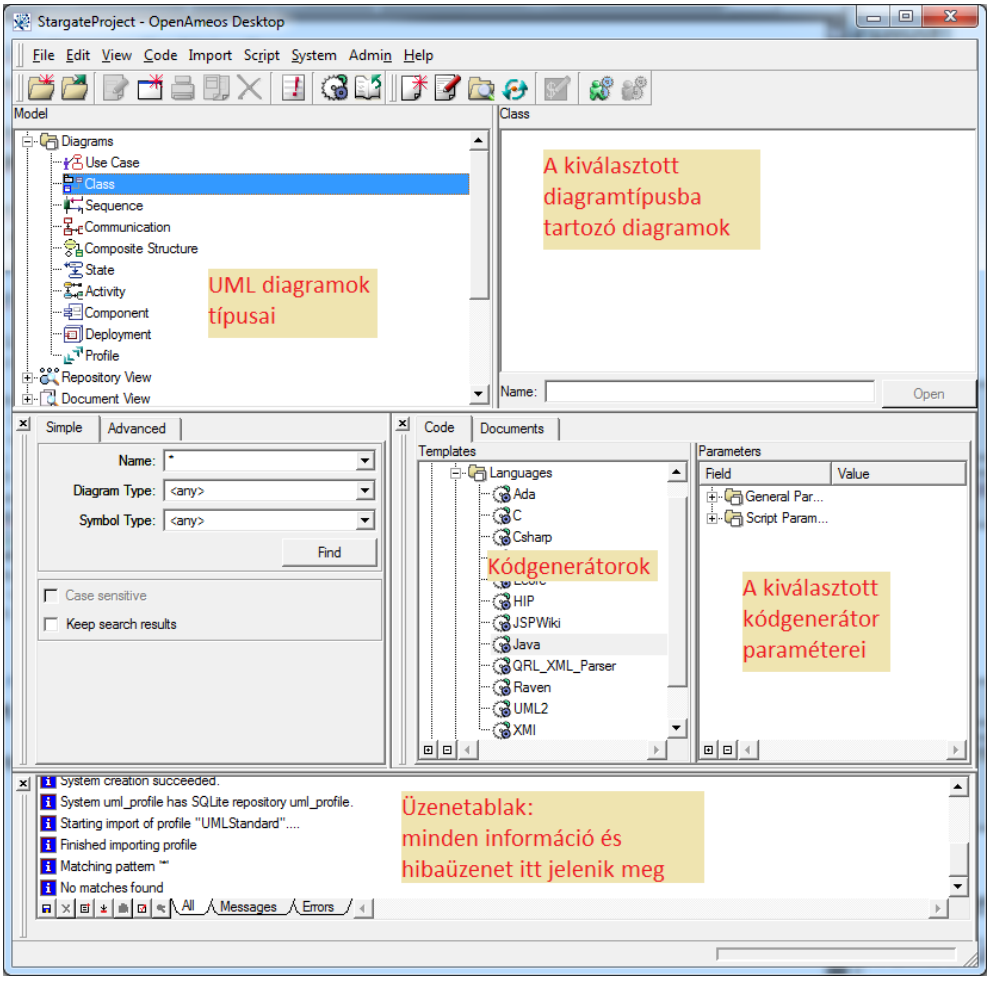
2 Ismerkedés az OpenAmeos-szal

2.1 Az OpenAmeos elindítása

Az OpenAmeos hivatalos honlapja a következő: http://www.openameos.org/

Az eszköz elindítása után a **File/New system…** menüpontban hozzunk létre egy új projektet **StargateProject** néven, adatbáziskezelőnek pedig az **SQLite**-ot válasszuk! A megjelenő **Import Profile** ablakban válasszuk az **UMLStandard**-ot!

A főablak a következőképpen épül fel:



Az OpenAmeos több fejlesztő munkáját egyszerre tudja támogatni, azonban a zárolások néha beragadnak. Kapcsoljuk ki a zárolási lehetőségeket! Ehhez az **Admin/Ameos Utility** menüpontban válasszuk ki a **Locking Subsystem**-et! A megjelenő ablakban adjuk ki a következő parancsokat és figyeljük meg, hogy az alábbi válaszokat kapjuk-e:

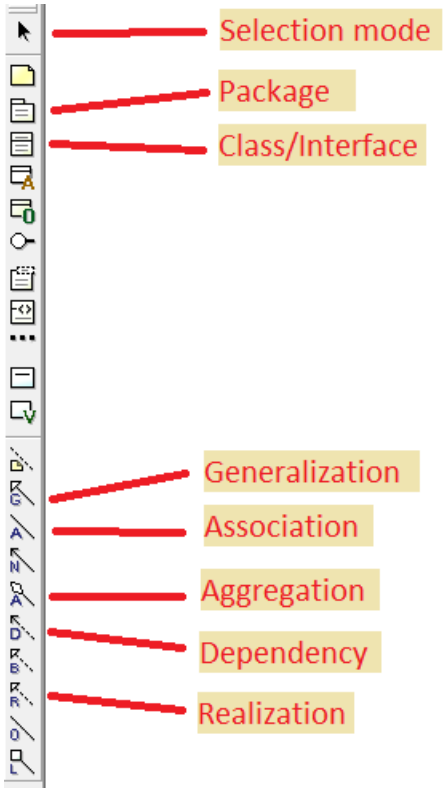
```
Ameos-UTILITY
> Interactive 1
> lock
lock-> status
```

```
Executing: Locking Status
Locking is enabled
Enabled by: xy
on 09/09/09 15:46:24
Users allowed to disable:
User xy@COMPUTER
Execution Complete.
lock-> disable
lock-> status
Locking is disabled
lock-> quit
-> quit
```

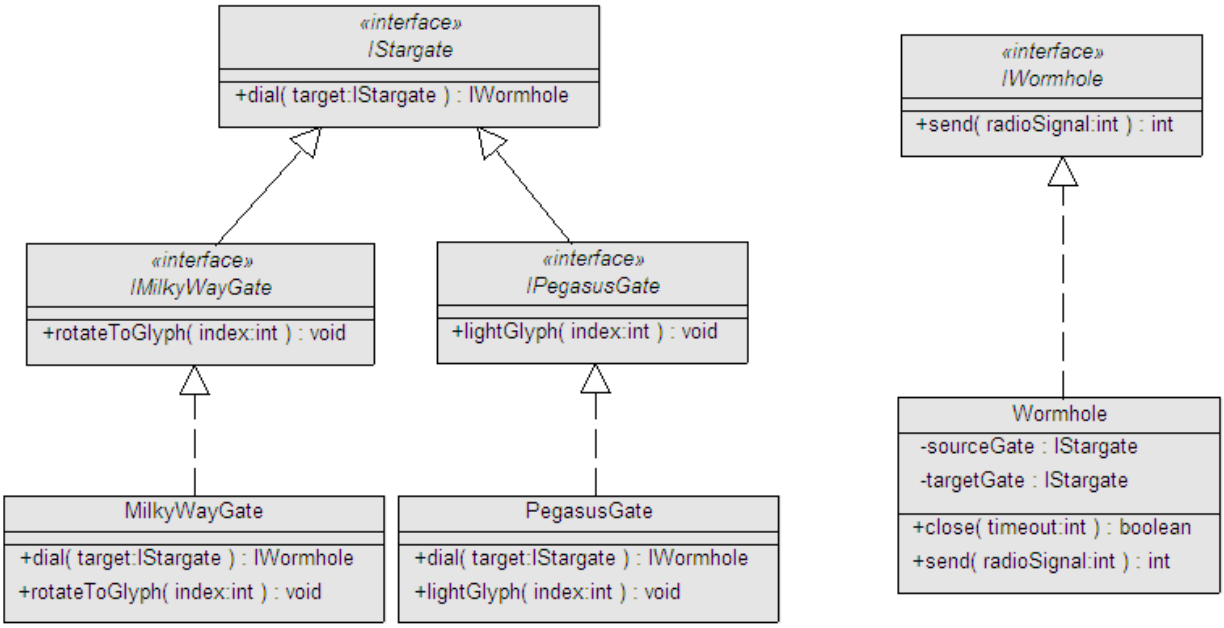
2.2 UML osztálydiagram rajzolása

Hozzunk létre egy új osztálydiagramot: kattintsunk jobbgombbal a **Diagrams / Class** elemen, majd válasszuk a **New…** menüpontot!

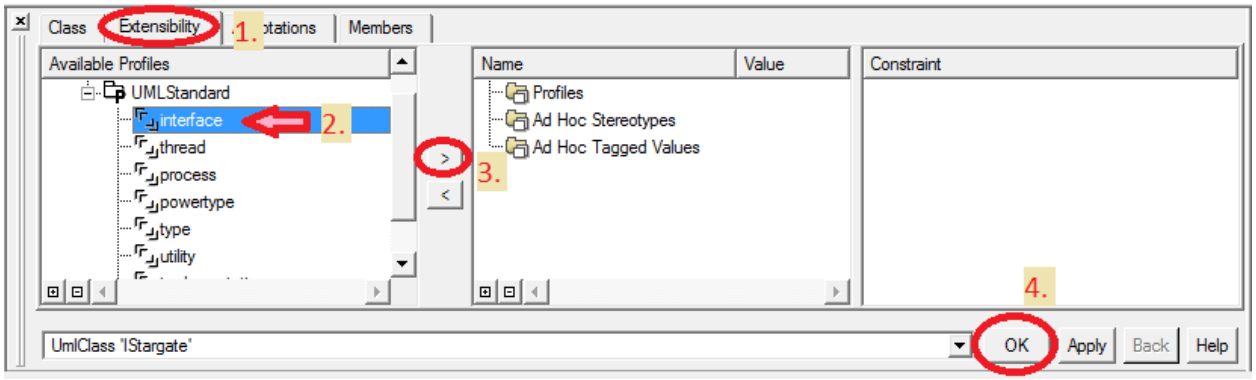
A fontosabb diagramelemek ikonjai:



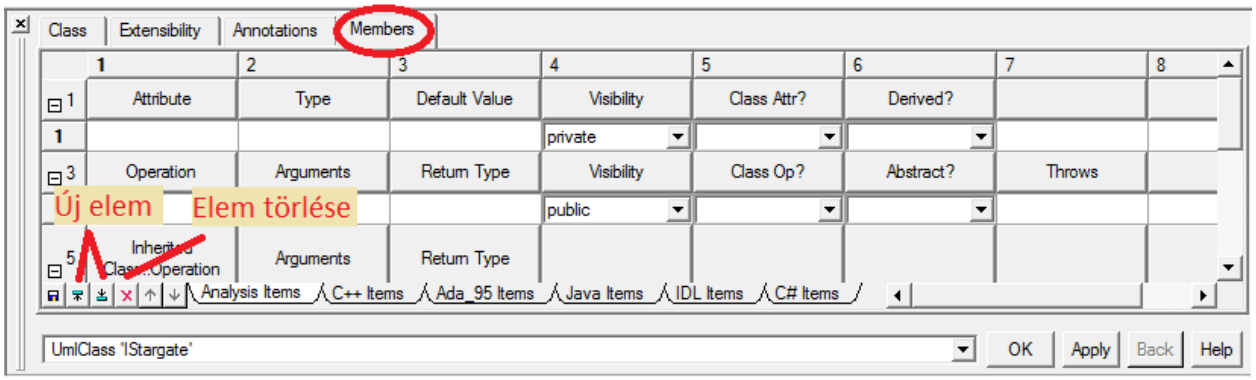
Rajzoljuk meg és mentsük el tetszőleges néven az alábbi osztálydiagramot:



Egy osztályra vagy interfészre duplán kattintva szerkeszthetjük az adatait. Osztályból úgy lehet interfészt készíteni, ha **<<interface>>** sztereotípiát rendelünk hozzá. Ezt az **Extensibility** fülön tehetjük meg:

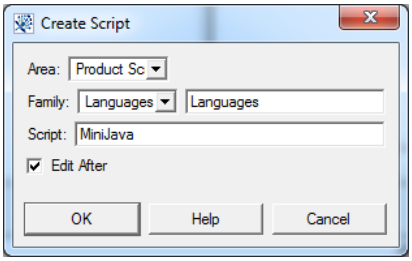


A **Members** fülön attribútumokat és metódusokat adhatunk az osztályokhoz és interfészekhez:



2.3 Kódgenerátor készítése

Kattintsunk jobbgombbal a főablak kódgenerátorokat tartalmazó részében a **Product Scripts > Languages** elemen, majd válasszuk a **Create Script…** menüpontot! A megjelenő ablakban a **Script** sorba írjuk be a választott kódgenerátor nevét, pl.:



A notepad-ben megnyílik a generátor kódja:

```
// Copyright (c) 2007 by Aonix
// Registered 2007 by ScopeSET GmbH
// Modified 2007-2009 by ScopeSET GmbH

/*
This script file is used as template when creating a new script
*/

proc main()
end proc
```

Ezt cseréljük le a következőre:

```
/*{
{ Category
  { Label "Java" }
  { Variable
      { Name ProjectName }
      { Type text }
      { DefaultValue "${system}" }
      { EditorType text }
  }
}
}*/

USES std;

proc main()
    init();
    if([srcdir] == "")
        info = "There is no output directory specified!\n";
        return;
    end if
    if([ProjectName] == "")
        info = "There is no project name specified!\n";
        return;
    end if

    mkdir([srcdir] "/" [ProjectName]);
    output([srcdir] "/" [ProjectName] "/" [ProjectName] ".java");
    out = genClasses();
end proc
```

```
template genClasses()
  [loop(Instances->MClass)]
public class [MClass.name]
{
    …
}
  [end loop]
end template

…

// ==================
// Utility functions:
// ==================

// Upcases the first character of a word.
tcl_proc string_capitalize(str)
    set fc [string index $str 0];
    set fc [string toupper $fc];
    set rest [string range $str 1 end];
    set str [append fc $rest];
    return $str;
end proc

// Lowercases the first character of a word.
tcl_proc string_decapitalize(str)
    set fc [string index $str 0];
    set fc [string tolower $fc];
    set rest [string range $str 1 end];
    set str [append fc $rest];
    return $str;
end proc

/* Generates a marked section in the output. The content of this section can
be changed by the user in the generated file. These changes will not be
overwritten in subsequent cycles of code generation */
template mergeOut(uniqueId,desc)
//#ACD# M([uniqueId]) [desc]
//user defined code to be added here ...
//#end ACD#
end template

/* this procedure is called once by main (defined in cpp_main.tdl) before
   any other procedure or template. It sets up the merge mechanism for
   incremental code generation (delMergeFile and appendMergeFile).
   The current language for type mapping is defined (setLanguage) */
proc init()
    delMergeFile();
    appendMergeFile("merge_config_JAVA.txt");
    setLanguage("Java");
end proc
```

Mentsük el a generátort! Az OpenAmeos főablakában a középen lévő **Code** nevű tabon a **Product Scripts > Languages > [generárot neve]**-n jobb gombbal kattintva a **Run Script** menüponttal indíthatjuk el a generátort. Ha a generátor kódjában hiba van, akkor ezt az OpenAmeos a főablak alján található üzenetnaplóban jelzi. Ha a generátor kódja hibátlan, a script sikeresen lefut és a létrehozott projekt **src_files** könyvtárában megtalálható az elkészült kimenet.

A vastagbetűs részek helyett írjuk meg a saját kódgenerátorunkat! Ha készen vagyunk, mentsük el és futtassuk le a generátort!

3 Egyszerűsített Java kódgenerátor készítése

Írjunk egy generátort **MiniJava** néven, amely a fenti osztálydiagramból az alábbi kódot állítja elő a **[ProjectName].java** fájlban:

```
public class [ProjectName]
{
    public static void main(String[] args)
    {
    }
}

interface IMilkyWayGate extends IStargate
{
    void rotateToGlyph(int index);
}

interface IPegasusGate extends IStargate
{
    void lightGlyph(int index);
}

interface IWormhole
{
    int send(int radioSignal);
}

interface IStargate
{
    IWormhole dial(IStargate target);
}

class Wormhole implements IWormhole
{
    private IStargate sourceGate;
    private IStargate targetGate;

    public IStargate getSourceGate() {
        return this.sourceGate;
    }

    public void setSourceGate(IStargate sourceGate) {
        this.sourceGate = sourceGate;
    }

    public IStargate getTargetGate() {
        return this.targetGate;
    }

    public void setTargetGate(IStargate targetGate) {
        this.targetGate = targetGate;
    }

    public boolean close(int timeout)
    {
        boolean result = false;
        //#ACD# M(UID_740BD4B7-5FBC-45c5-A0AF-B1E5F894B25D)
        //user defined code to be added here ...
        //#end ACD#
        return result;
    }
```

```
    public int send(int radioSignal)
    {
        int result = 0;
        //#ACD# M(UID_740BD4B7-5FBC-45c5-A0AF-B1E5F894B25E)
        //user defined code to be added here ...
        //#end ACD#
        return result;
    }
}

class PegasusGate implements IPegasusGate
{
    public IWormhole dial(IStargate target)
    {
        IWormhole result = null;
        //#ACD# M(UID_740BD4B7-5FBC-45c5-A0AF-B1E5F894B25F)
        //user defined code to be added here ...
        //#end ACD#
        return result;
    }

    public void lightGlyph(int index)
    {
        //#ACD# M(UID_740BD4B7-5FBC-45c5-A0AF-B1E5F894B200)
        //user defined code to be added here ...
        //#end ACD#
    }
}

class MilkyWayGate implements IMilkyWayGate
{
    public IWormhole dial(IStargate target)
    {
        IWormhole result = null;
        //#ACD# M(UID_740BD4B7-5FBC-45c5-A0AF-B1E5F894B201)
        //user defined code to be added here ...
        //#end ACD#
        return result;
    }

    public void rotateToGlyph(int index)
    {
        //#ACD# M(UID_740BD4B7-5FBC-45c5-A0AF-B1E5F894B202)
        //user defined code to be added here ...
        //#end ACD#
    }
}
```

Az alábbi loop-konstrukciókat célszerű felhasználni:

- Instances->MInterface
- Instances->MNormalClass
- MInterface->Implement
- MInterface->SuperClass
- MNormalClass->Implement
- MNormalClass->SuperClass
- MInterface->MOperation
- MNormalClass->MAttribute
- MNormalClass->MOperation
- MOperation->OpPara

Ezek mellett használhatjuk a metaosztályok alábbi metaattribútumait:

- MInterface.name
- MNormalClass.name
- Implement.name
- MAttribute.type
- MAttribute.name
- MOperation.name
- MOperation.returnType
- OpPara.type
- OpPara.name