Bmod: a DSL for simulation of evacuation plans using Xtext

Schoofs Ebert 20161650

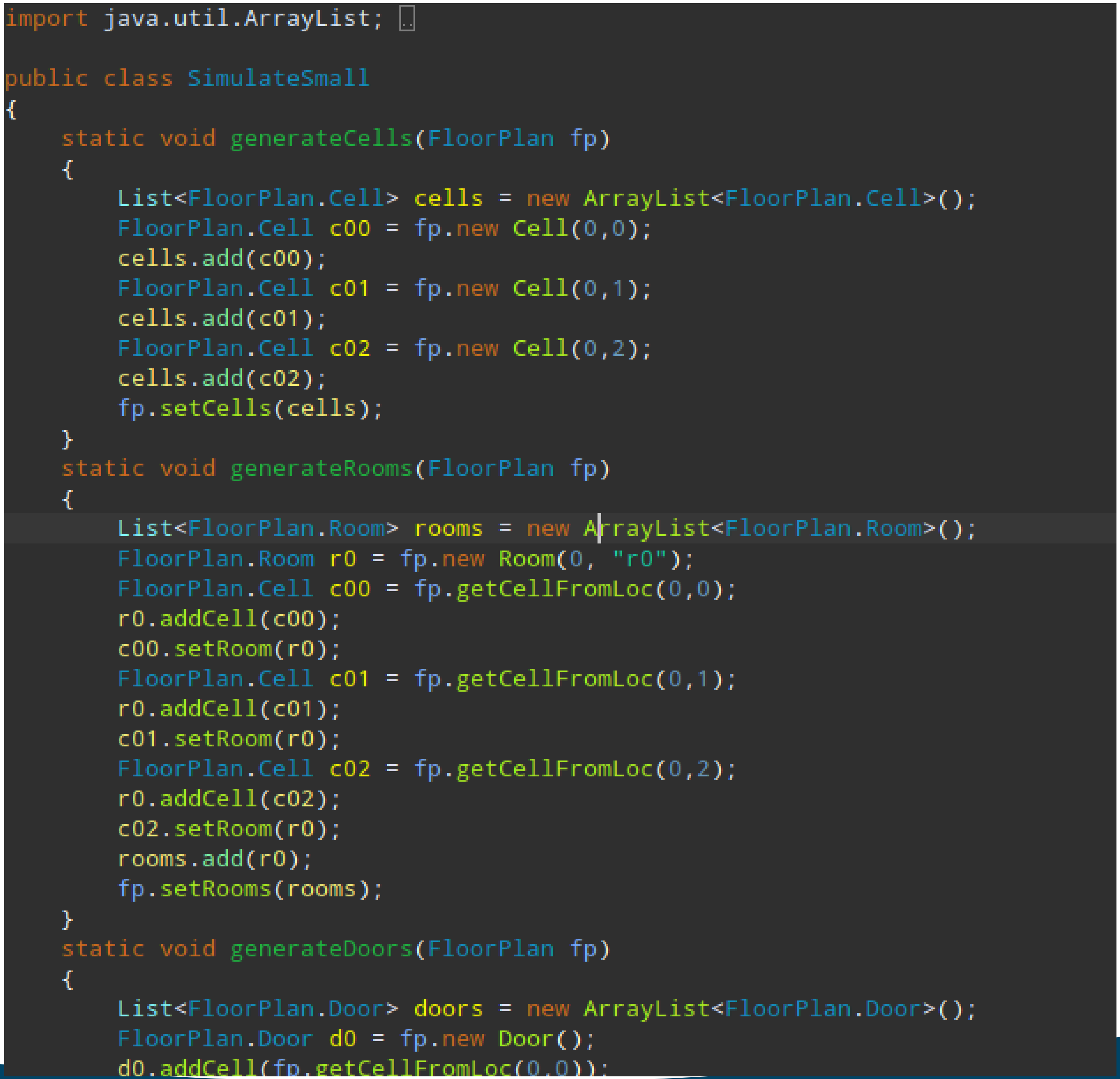
09/01/2020

University of Antwerp

Syntax

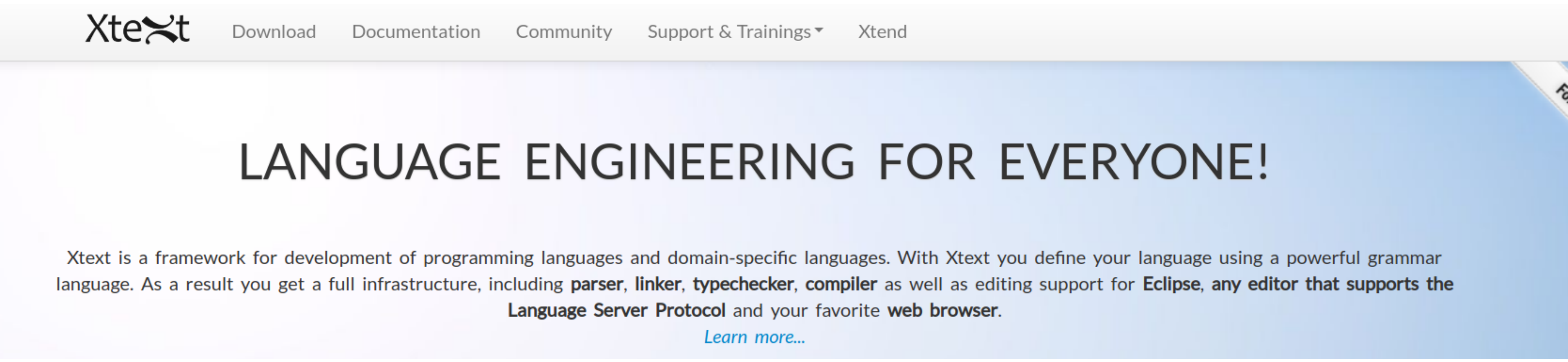
```
FloorPlan Small
{
      Cell c00
      {
            x : 0
            y : 0
      }
      Cell c01
      {
            x : 0
            y : 1
      }
      Cell c02
      {
            x : 0
            y : 2
      }
      Room r0
      {
            cells : [c00, c01, c02]
      }
      Door d0
      {
            cell : c00
      }
      Person Jef
      {
            action : experienced
            perception : smeller
            cell : c01
      }
      Fire
      {
            cells : [c02]
      }
}
```

Translation to Java



```
import java.util.ArrayList; 

public class SimulateSmall
{
    static void generateCells(FloorPlan fp)
    {
        List<FloorPlan.Cell> cells = new ArrayList<FloorPlan.Cell>();
        FloorPlan.Cell c00 = fp.new Cell(0,0);
        cells.add(c00);
        FloorPlan.Cell c01 = fp.new Cell(0,1);
        cells.add(c01);
        FloorPlan.Cell c02 = fp.new Cell(0,2);
        cells.add(c02);
        fp.setCells(cells);
    }
    static void generateRooms(FloorPlan fp)
    {
        List<FloorPlan.Room> rooms = new ArrayList<FloorPlan.Room>();
        FloorPlan.Room r0 = fp.new Room(0, "r0");
        FloorPlan.Cell c00 = fp.getCellFromLoc(0,0);
        r0.addCell(c00);
        c00.setRoom(r0);
        FloorPlan.Cell c01 = fp.getCellFromLoc(0,1);
        r0.addCell(c01);
        c01.setRoom(r0);
        FloorPlan.Cell c02 = fp.getCellFromLoc(0,2);
        r0.addCell(c02);
        c02.setRoom(r0);
        rooms.add(r0);
        fp.setRooms(rooms);
    }
    static void generateDoors(FloorPlan fp)
    {
        List<FloorPlan.Door> doors = new ArrayList<FloorPlan.Door>();
        FloorPlan.Door d0 = fp.new Door();
        d0.addCell(fp.getCellFromLoc(0,0));
```

Xtext

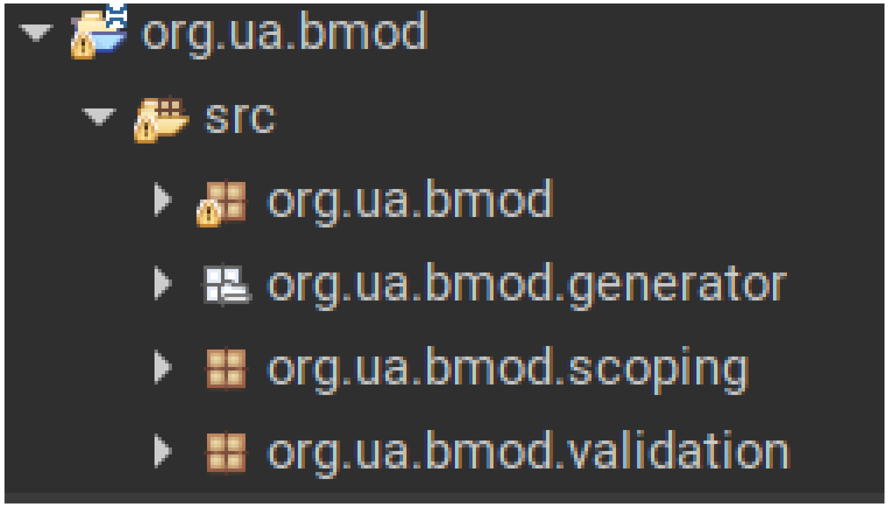


LANGUAGE ENGINEERING FOR EVERYONE!

Xtext is a framework for development of programming languages and domain-specific languages. With Xtext you define your language using a powerful grammar language. As a result you get a full infrastructure, including **parser**, **linker**, **typechecker**, **compiler** as well as editing support for **Eclipse**, **any editor that supports the Language Server Protocol** and your favorite **web browser**.

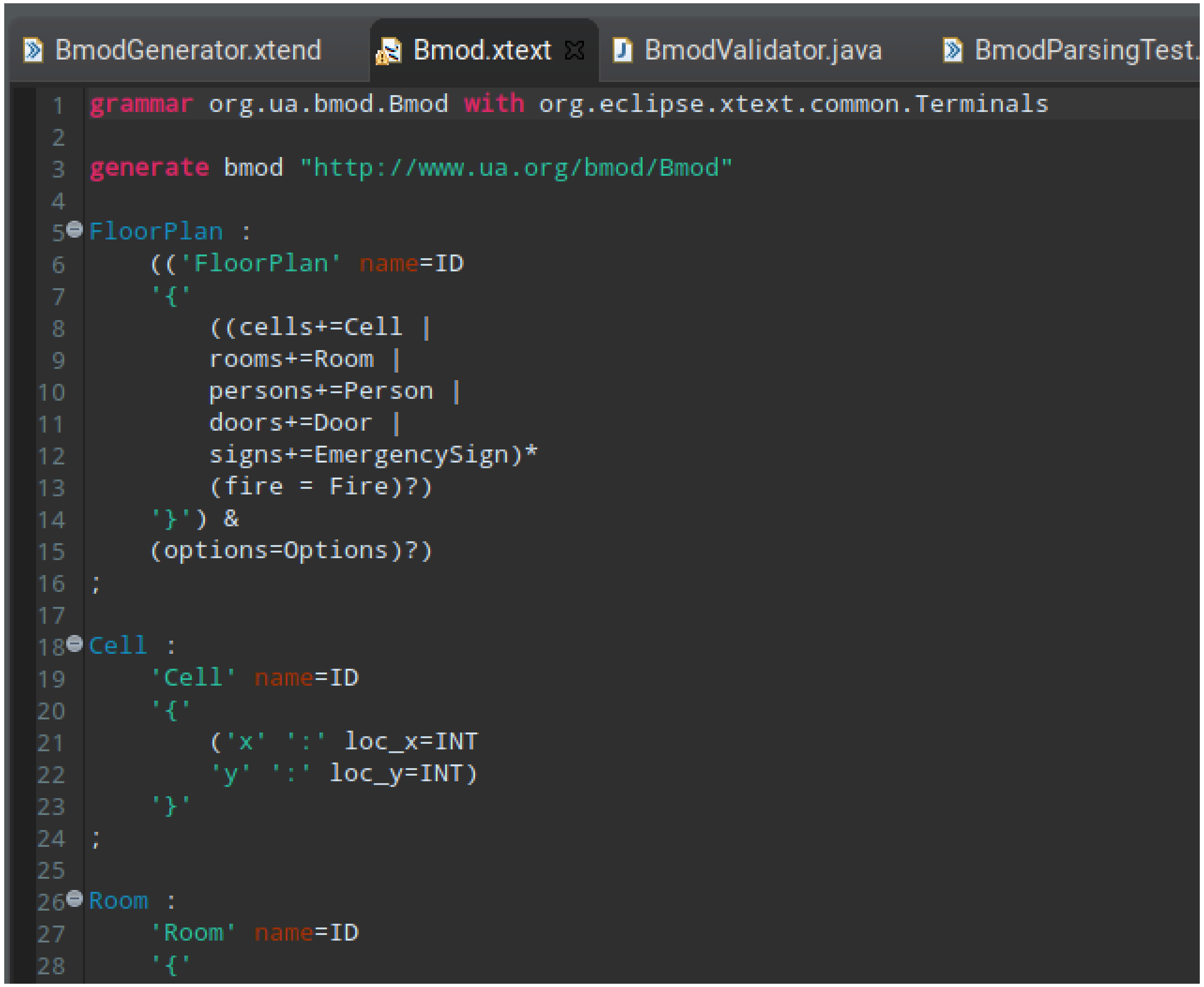
Learn more...

Xtext



```
▾ org.ua.bmod
    ▾ src
        ▸ org.ua.bmod
        ▸ org.ua.bmod.generator
        ▸ org.ua.bmod.scoping
        ▸ org.ua.bmod.validation
```

Grammar Language

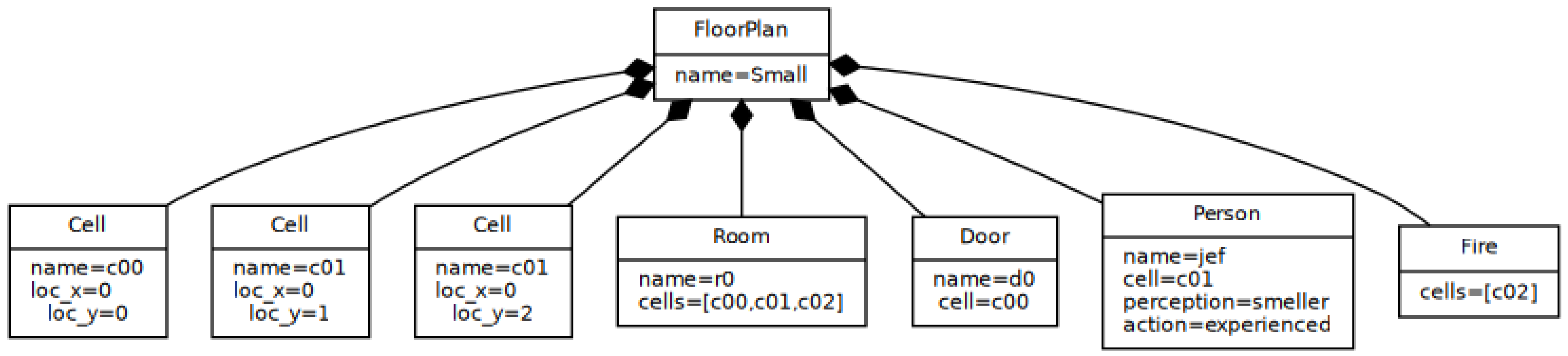


```
grammar org.ua.bmod.Bmod with org.eclipse.xtext.common.Terminals

generate bmod "http://www.ua.org/bmod/Bmod"

FloorPlan :
    (('FloorPlan' name=ID
    '{'
        ((cells+=Cell |
        rooms+=Room |
        persons+=Person |
        doors+=Door |
        signs+=EmergencySign)*
        (fire = Fire)?)
    '}') &
    (options=Options)?)
;

Cell :
    'Cell' name=ID
    '{'
        ('x' ':' loc_x=INT
        'y' ':' loc_y=INT)
    '}'
;

Room :
    'Room' name=ID
    '{'
```

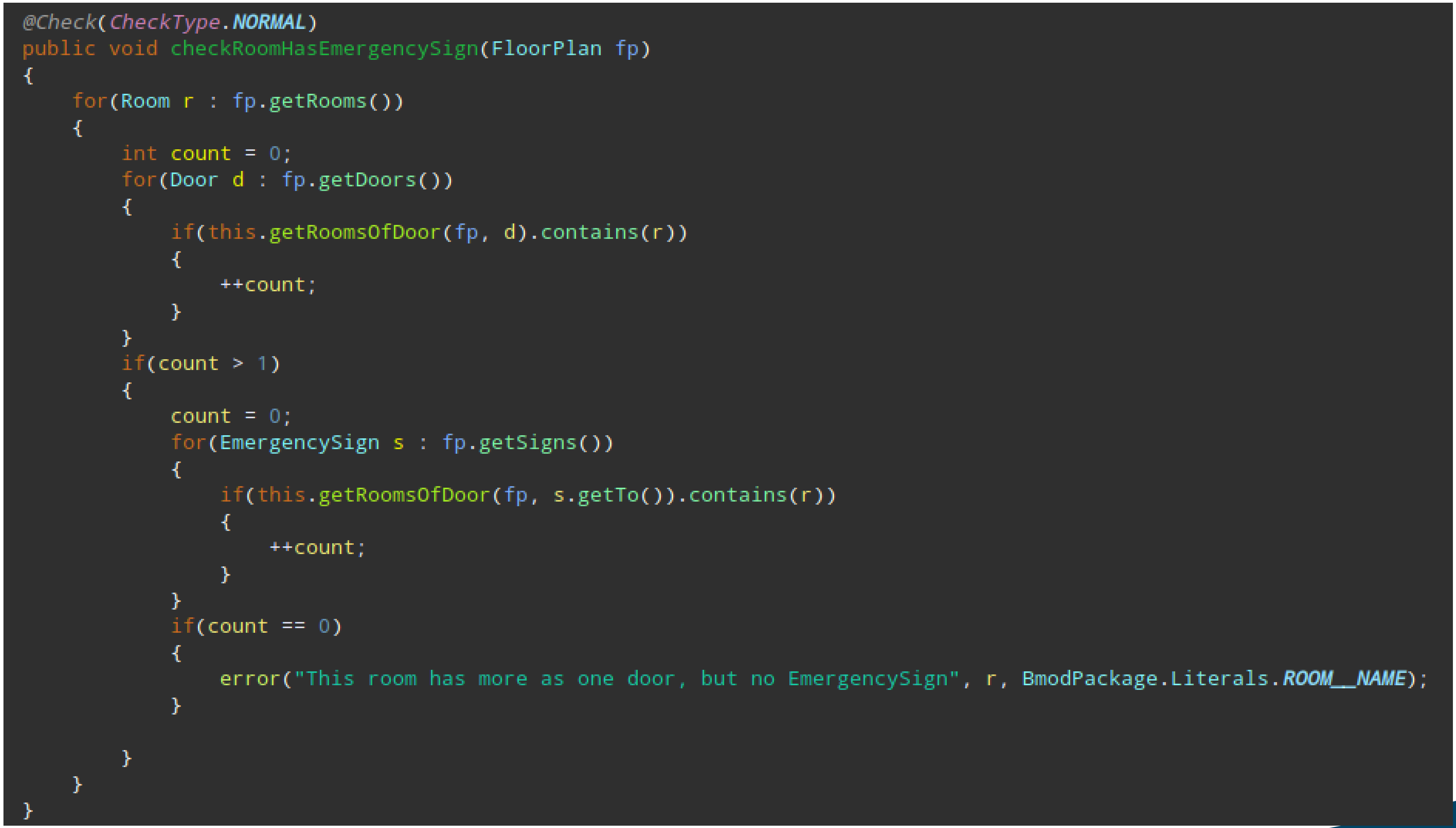
Model

```
FloorPlan Small
{
	Cell c00
	{
		x : 0
		y : 0
	}
	Cell c01
	{
		x : 0
		y : 1
	}
	Cell c02
	{
		x : 0
		y : 2
	}
	Room r0
	{
		cells : [c00, c01, c02]
	}
	Door d0
	{
		cell : c00
	}
	Person Jef
	{
		action : experienced
		perception : smeller
		cell : c01
	}
	Fire
	{
		cells : [c02]
	}
}
```

Model

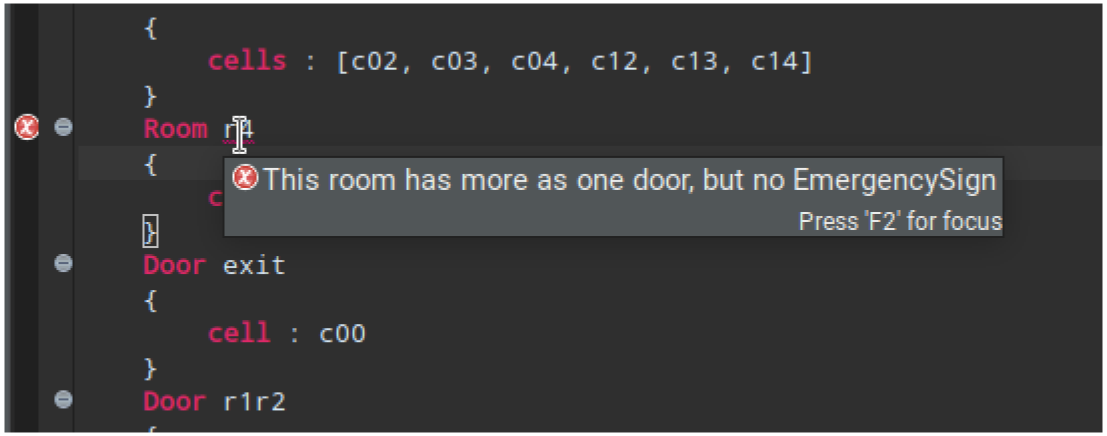

FloorPlan
name=Small
Cell
name=c00
loc_x=0
loc_y=0
Cell
name=c01
loc_x=0
loc_y=1
Cell
name=c01
loc_x=0
loc_y=2
Room
name=r0
cells=[c00,c01,c02]
Door
name=d0
cell=c00
Person
name=jef
cell=c01
perception=smeller
action=experienced
Fire
cells=[c02]

Custom Validation



```
@Check(CheckType.NORMAL)
public void checkRoomHasEmergencySign(FloorPlan fp)
{
    for(Room r : fp.getRooms())
    {
        int count = 0;
        for(Door d : fp.getDoors())
        {
            if(this.getRoomsOfDoor(fp, d).contains(r))
            {
                ++count;
            }
        }
        if(count > 1)
        {
            count = 0;
            for(EmergencySign s : fp.getSigns())
            {
                if(this.getRoomsOfDoor(fp, s.getTo()).contains(r))
                {
                    ++count;
                }
            }
            if(count == 0)
            {
                error("This room has more as one door, but no EmergencySign", r, BmodPackage.Literals.ROOM__NAME);
            }

        }
    }
}
```

Custom Validation

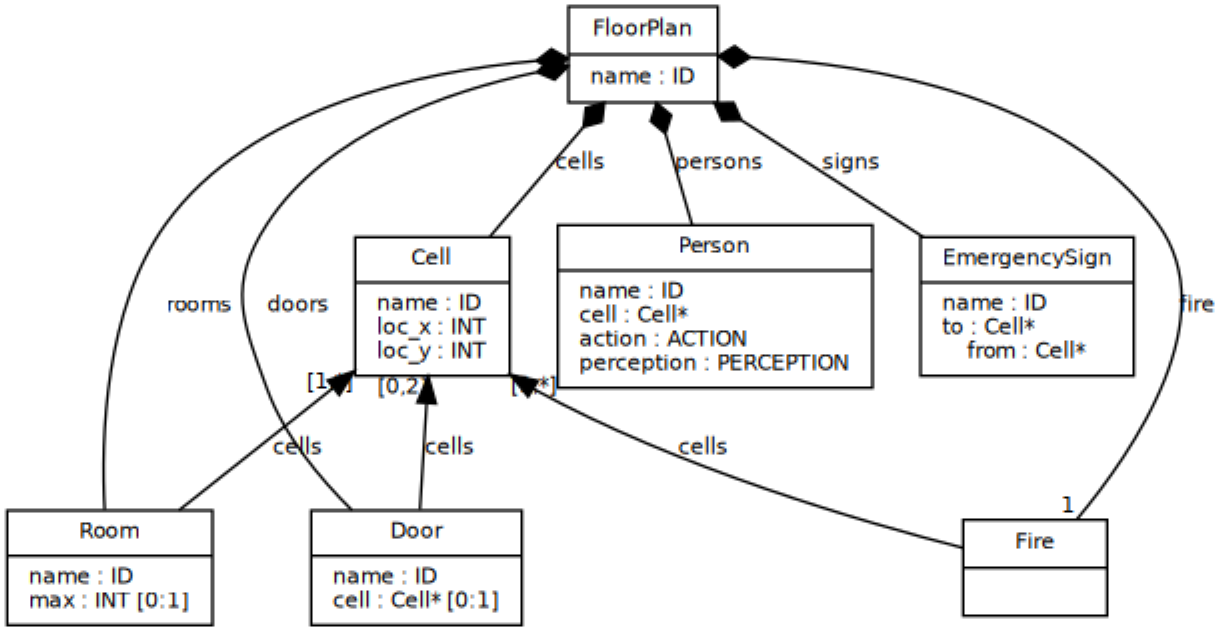


```
{
    cells : [c02, c03, c04, c12, c13, c14]
}
Room r4
{
    c
}
Door exit
{
    cell : c00
}
Door r1r2
```

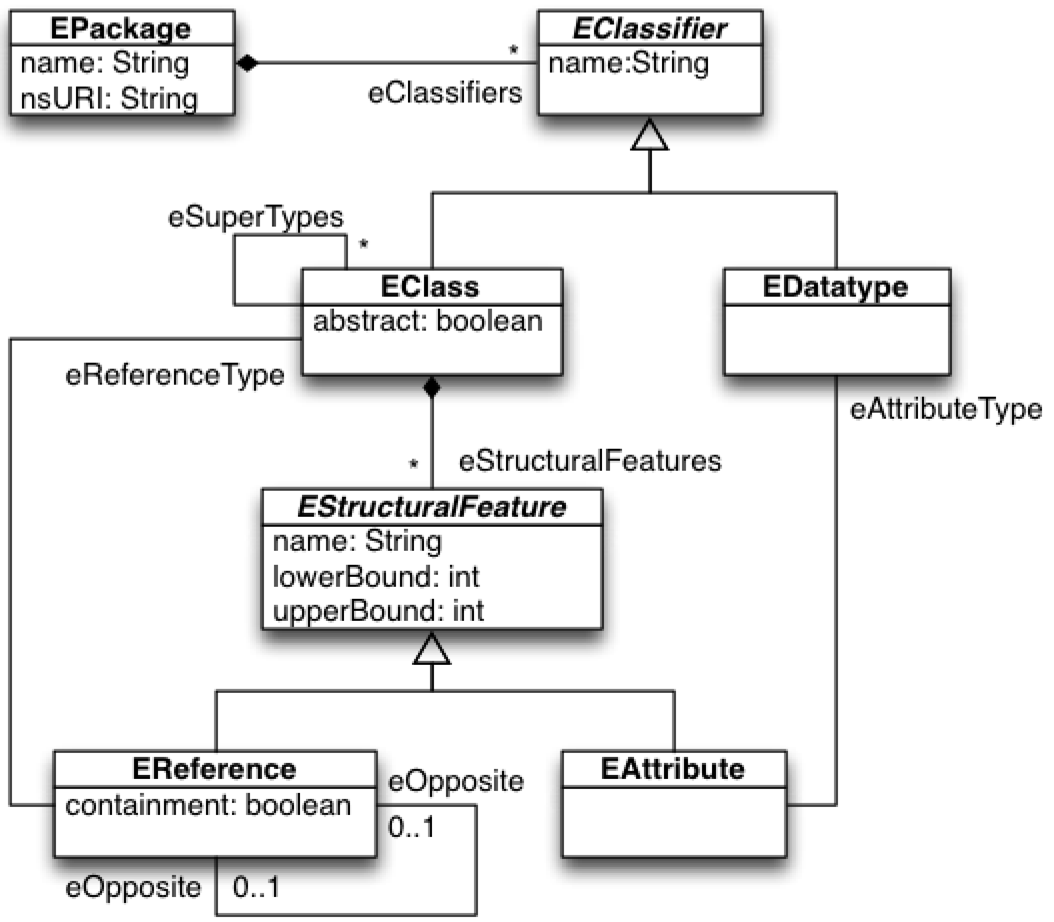
This room has more as one door, but no EmergencySign

Press 'F2' for focus

Meta Model


FloorPlan
name : ID
cells
persons
signs
rooms
doors
fire
Cell
name : ID
loc_x : INT
loc_y : INT
Person
name : ID
cell : Cell*
action : ACTION
perception : PERCEPTION
EmergencySign
name : ID
to : Cell*
from : Cell*
[1,*]
[0,2]
[1,*]
cells
cells
cells
1
Room
name : ID
max : INT [0:1]
Door
name : ID
cell : Cell* [0:1]
Fire

ECore


EPackage
name: String
nsURI: String
eClassifiers
*
EClassifier
name:String
eSuperTypes
*
EClass
abstract: boolean
EDatatype
eReferenceType
eAttributeType
*
eStructuralFeatures
EStructuralFeature
name: String
lowerBound: int
upperBound: int
EReference
containment: boolean
eOpposite
0..1
EAttribute
eOpposite
0..1

Code Generation

```
class BmodGenerator extends AbstractGenerator {

    override void doGenerate(Resource resource, IFileSystemAccess2 fsa, IGeneratorContext context)
    {
        fsa.generateFile("FloorPlan.java", declareFloorPlan());
        var fp = resource.allContents.head as FloorPlan;
        fsa.generateFile("Simulate"+fp.getName()+".java", toJava(fp));
    }
```

```
protected def generateRooms(FloorPlan fp)
'''
    static void generateRooms(FloorPlan fp)
    {
        List<FloorPlan.Room> rooms = new ArrayList<FloorPlan.Room>();
        «FOR room : fp.rooms»
            FloorPlan.Room «room.name» = fp.new Room(«room.max», "«room.name»");
            «FOR cell : room.cells»
                FloorPlan.Cell «cell.name» = fp.getCellFromLoc(«cell.loc_x»,«cell.loc_y»);
                «room.name».addCell(«cell.name»);
                «cell.name».setRoom(«room.name»);
            «ENDFOR»
            rooms.add(«room.name»);
        «ENDFOR»
        fp.setRooms(rooms);
    }
'''
```

But even more powerfull

- The Grammar Language
- Configuration
- Language Implementation
  - Code Generation
  - Validation
  - Linking
  - Scoping
  - Value Converter
  - Serialization
  - Formatting
  - Character Encoding
  - Unit Testing
- Integration with Java
- Typical Language Configurations
- Integration with EMF
- Eclipse Support
- Web Editor Support
- Continuous Integration

- Eclipse Support
  - Label Provider
  - Content Assist
  - Quick Fixes
  - Template Proposals
  - Outline View
  - Hyperlinking
  - Syntax Coloring
  - Rename Refactoring
  - Project Wizard
  - File Wizard
  - Code Mining

DEMO

University
of Antwerp