Introduction to USE

UML Specification Environment

USE

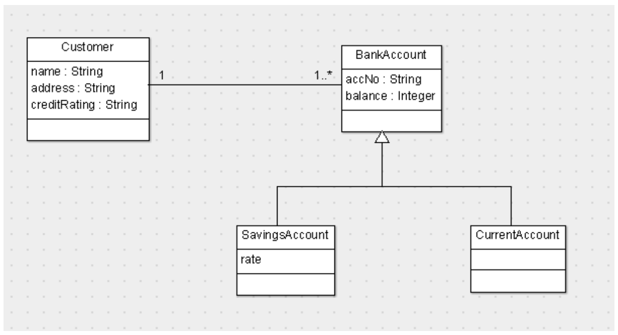
- From University of Bremen
- Download from http://useocl.sourceforge.net/w/index.php/Main_Page
- Need Java installed to run it

USE

- USE is a system for the specification of information systems
- based on a subset of the Unified Modeling Language (UML)
- has a high level implementation or action language - SOIL
- expressions written in the Object Constraint Language (OCL) can also be used and tested
- with OCL it provides an introduction to Design by Contract

Difference between UML & USE

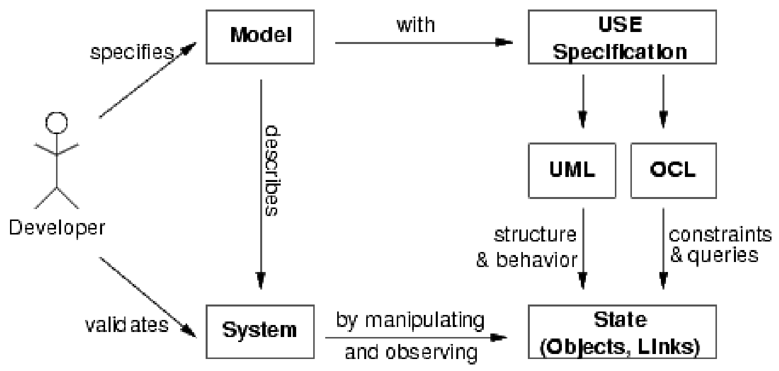
- With UML you can have class diagrams like



- But it is difficult to test them – can’t just compile and execute!

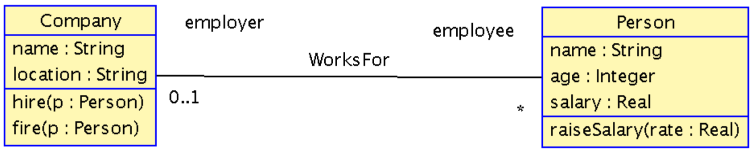
Difference between UML & USE

- USE allows you to animate or explore a UML class diagram by creating some test objects and getting them to interact
- Has a special high level abstract language called SOIL
  - Simple OCL-based Imperative Programming Language
  - Much simpler than testing your model in Java or C#



UML Class Diagram

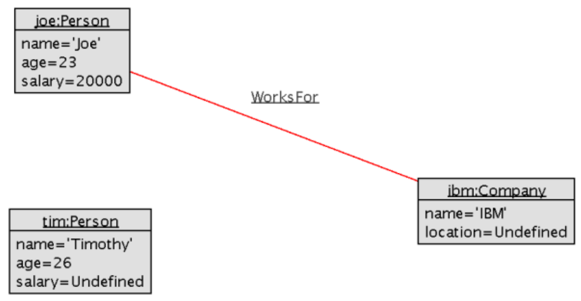
- Note that association has a name and so do both its end, called role names



USE Specification

- Code for Employee.use
- Sample objects Employee.soil

Object Diagram

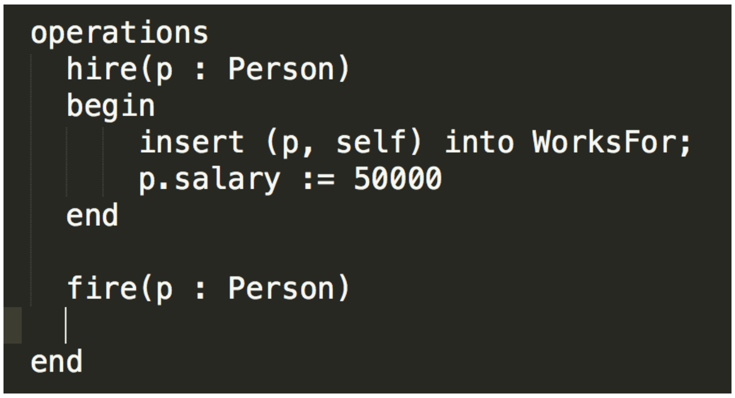


SOIL Code

```
!insert (joe,ibm) into WorksFor
!joe.salary := 20000
```

- Can put this code into the hire() operation

SOIL Operation Implementation



```
operations
  hire(p : Person)
  begin
      insert (p, self) into WorksFor;
      p.salary := 50000
  end

  fire(p : Person)
  
end
```

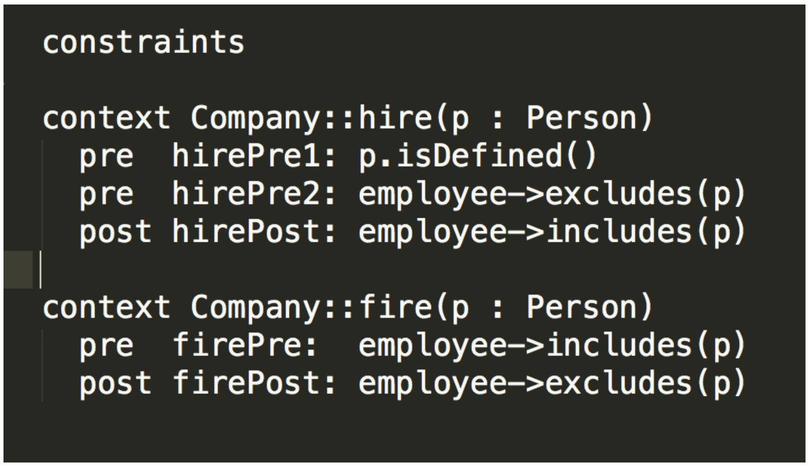
Executing an Operation

- Can create a corresponding sequence diagram

Design by Contract

- Proposed by Bertrand Meyer who invented the OO language Eiffel
- Consists of preconditions and postconditions
  - Preconditions – must be true before an operation is executed
  - Postconditions – must be true after operation has finished executing
- Together they supply the **contract** for class operation/method, sometimes referred to as constraints
- Object Constraint Language (OCL) allows one to describe these or Spec# in Visual Studio

OCL Constraints



```
constraints

context Company::hire(p : Person)
  pre  hirePre1: p.isDefined()
  pre  hirePre2: employee->excludes(p)
  post hirePost: employee->includes(p)

context Company::fire(p : Person)
  pre  firePre:  employee->includes(p)
  post firePost: employee->excludes(p)
```

Can combine USE Code & OCL

- See Employee1.use