# Introduction to a Simple OCL-like Imperative Language – SOIL

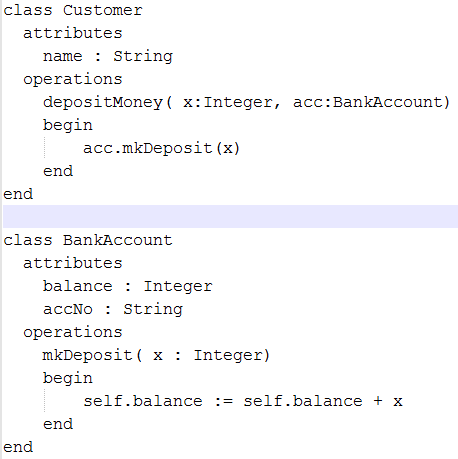
SOIL is a simple and unspectacular but complete imperative language that can be used to operationally specify UML models (i. e., to program or animate UML models).

SOIL is rather lightweight and does not aim to compete against general purpose languages such as Java or C#.

## Add SOIL code to USE Spec

Here we provide a SOIL implementation of the operations/methods depositMoney() and mkDeposit() in the Customer and BankAccount classes.

Use Notepad++ to open **bank.use** which you created in last lab session. Then modify the Customer and BankAccount classes as shown next.

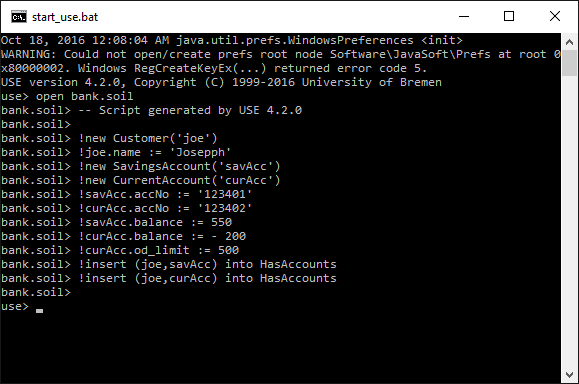


## Load new Spec

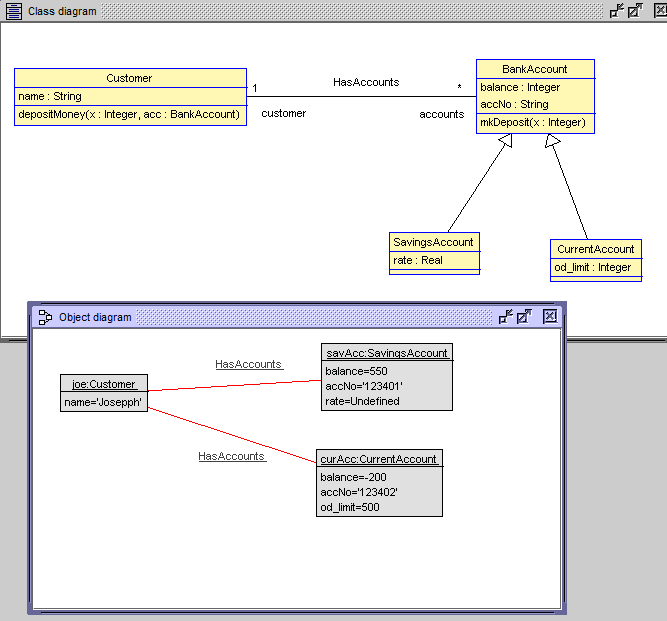
Start USE and load **bank.use**.

At the command window, type: open bank.use as show below.

This loads the object commands used and saved previously for creating and modifying objects.



Then open a class diagram view and an object diagram view and load the corresponding layout files saved in last lab session. USE GUI should look something like:

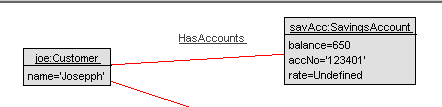


## Execute SOIL Operations

At the command window try



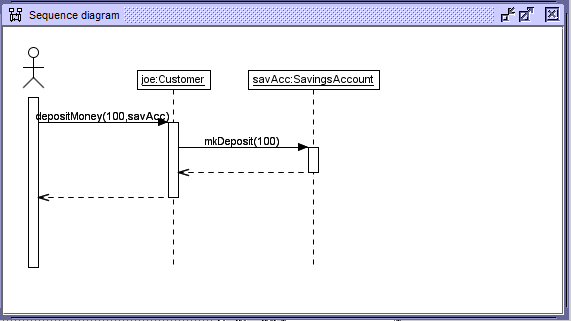
Look at your object diagram. Has Joe’s savings account balance been updated? Should now be:

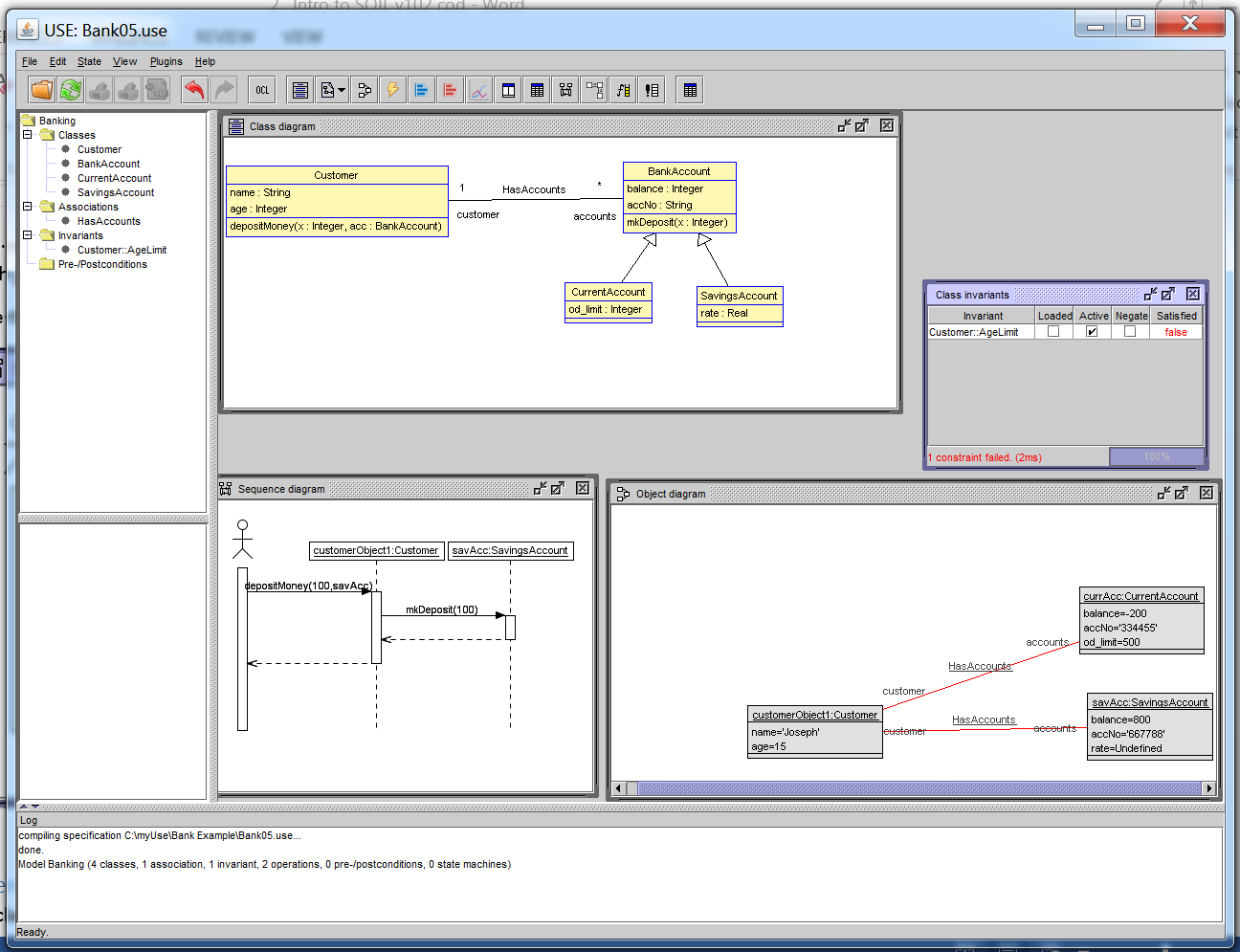


## Create a Sequence Diagram

We now create a sequence diagram that shows the object interactions that we cause to be executed with joe.depositMoney(100, savAcc).

Use the menu **View | Create view | Sequence diagram**. Should get:





## Exercise

Declare, implement and test operations

* withdrawMoney( m : Integer, a : BankAccount)
* mkWithdrawal(m : Integer)

Open a sequence diagram view. Any differences?

# Constraints

The bank\*.use file can be modified to add constraints.

context Customer

-- customers must all be over 18

inv AgeLimit:

self.age >=18

The bank00\*.soil file can be modified to set the value for the age to say 15

Run the soil file in the command window by typing :

Use> open bank00\*.soil

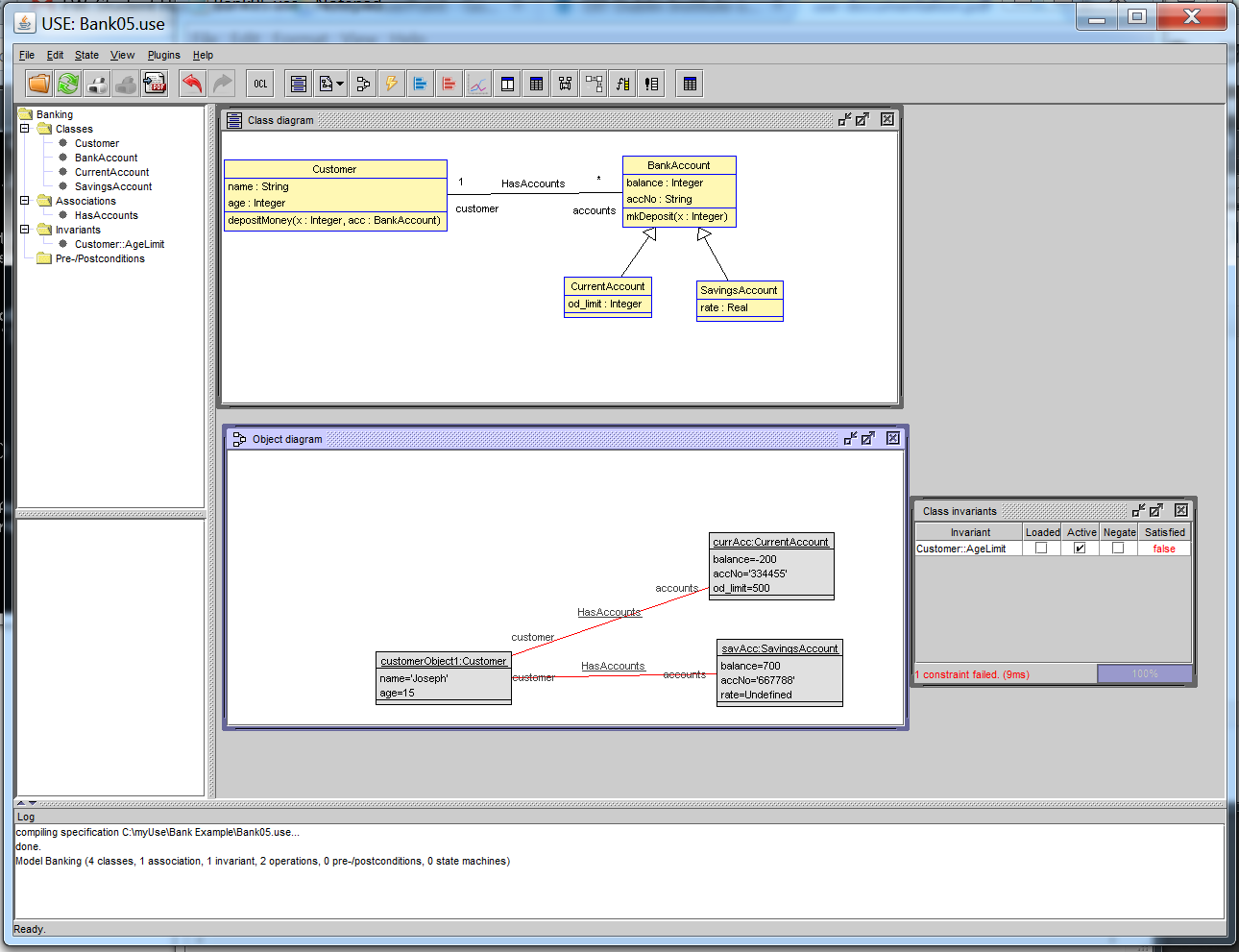
Select the button “Create object diagram view”

Right click and choose “load layout”

Use a previously created layout for the objects

Select the button “create class Invariant view”

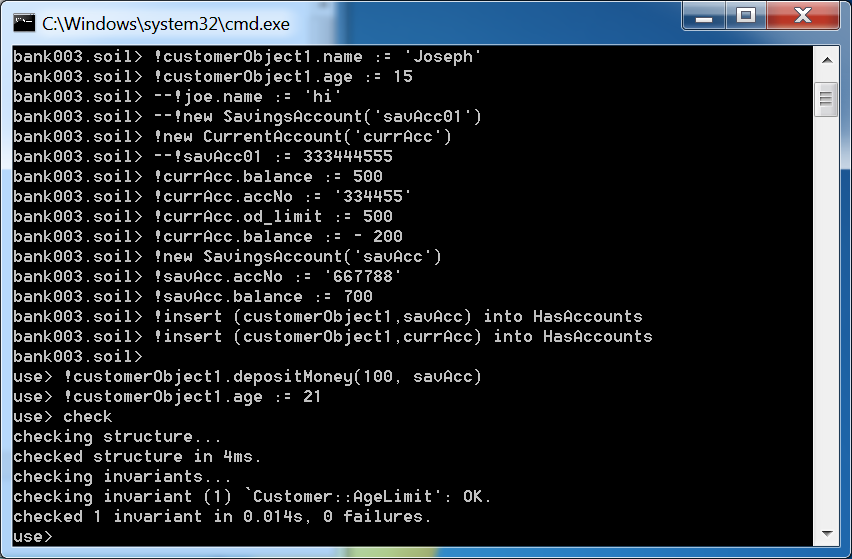
This shows the constraint rule for AgeLimit has failed



Object values can be updated in the console window

The age can be changed as below.

A check reveals the constraint passes.



The constraint is now valid so the invariant becomes True

