Rational Software

Payroll System Class Design Solution

Version 2004

Revision History

Date	Issue	Description	Author
09/01/2000	2000	Generation for beta	Shawn Siemers
10/03/2000	2000	Final release	Shawn Siemers
01/14/2003	2003	Final Release	Alex Kutsick
05/20/2004	2004	Generation for beta	Alex Kutsick

Table of Contents

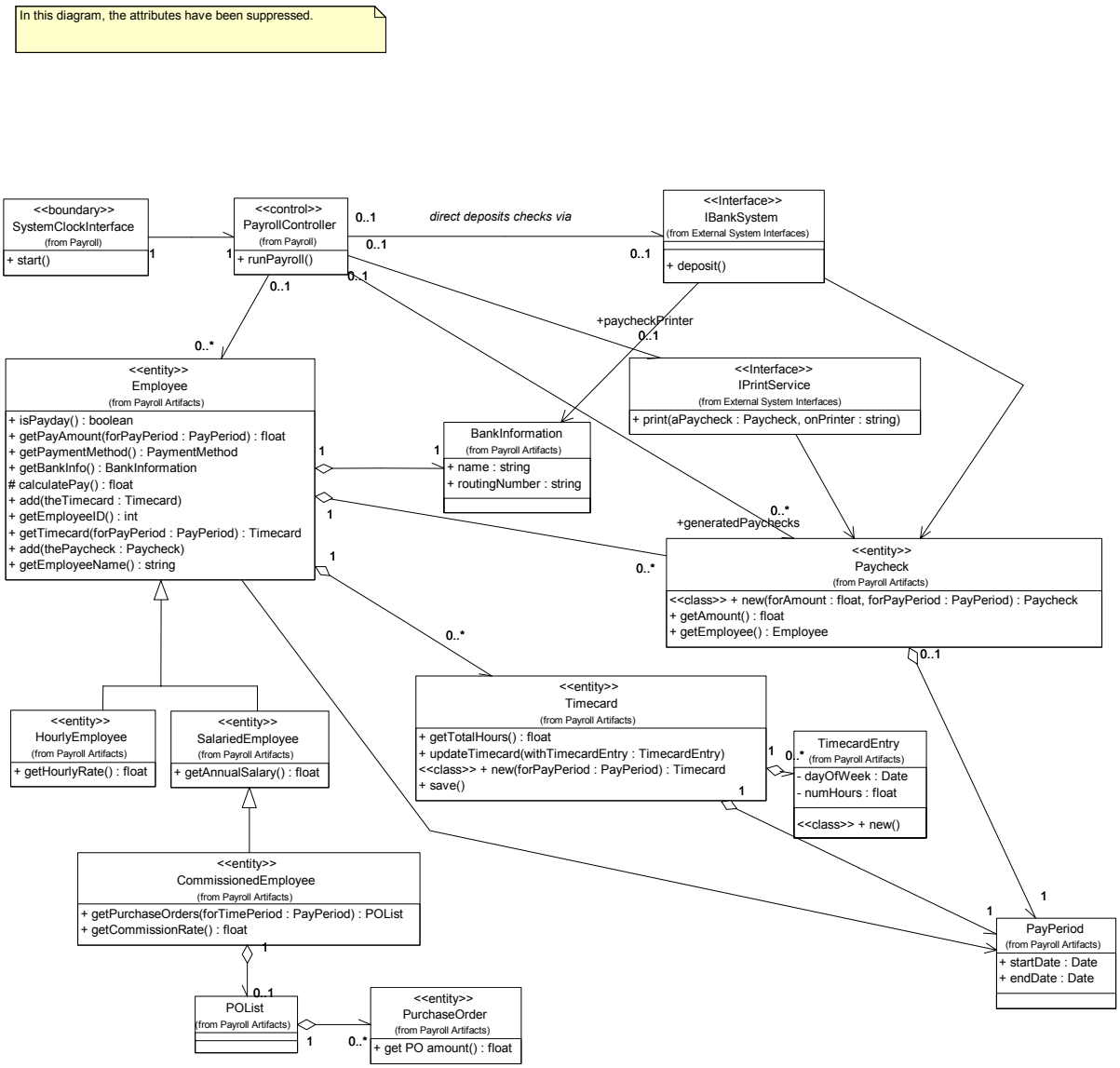
Payroll System Class Design Solution

1. Exercise: Class Design

1.1 Exercise: Define Operations

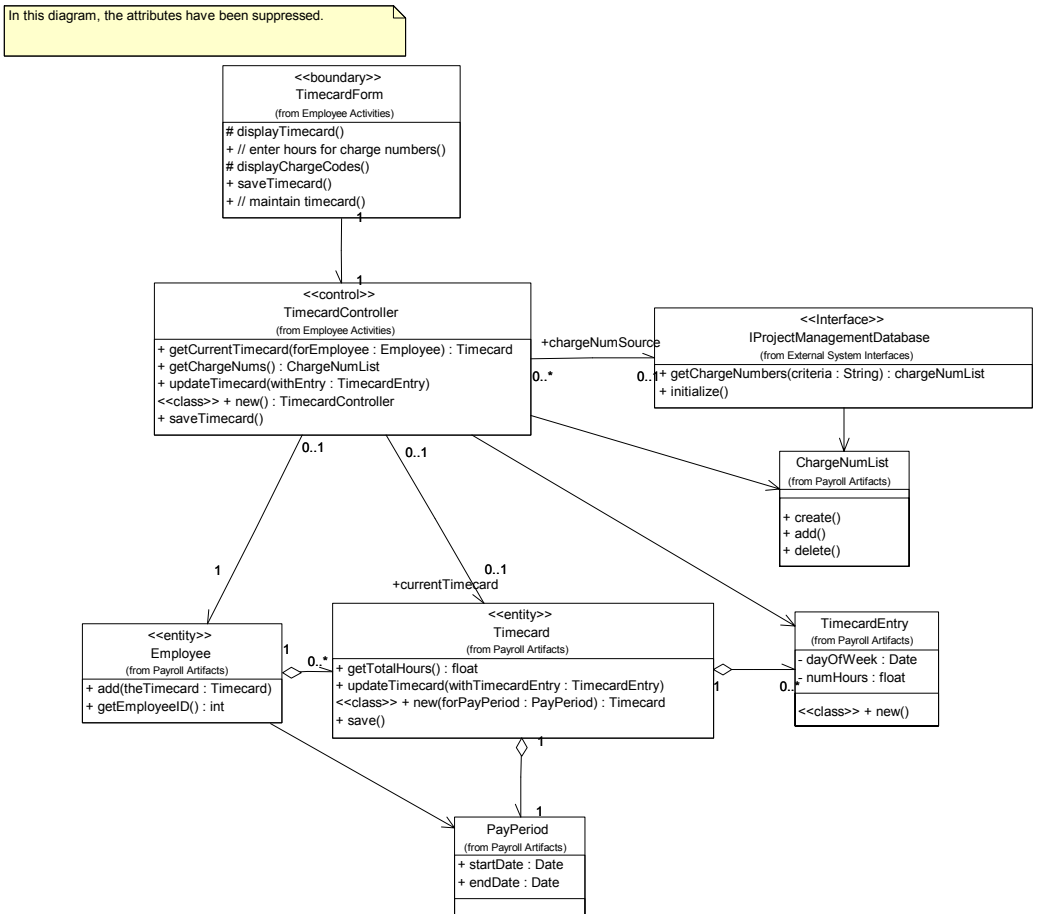
Note: Some operations on the forms were not "designed" as such detailed design is better performed as a part of user interface design, which is considered out of scope of this course.

Use-Case Realization - Run Payroll
Run Payroll - VOPC (with ss interface; ops only)



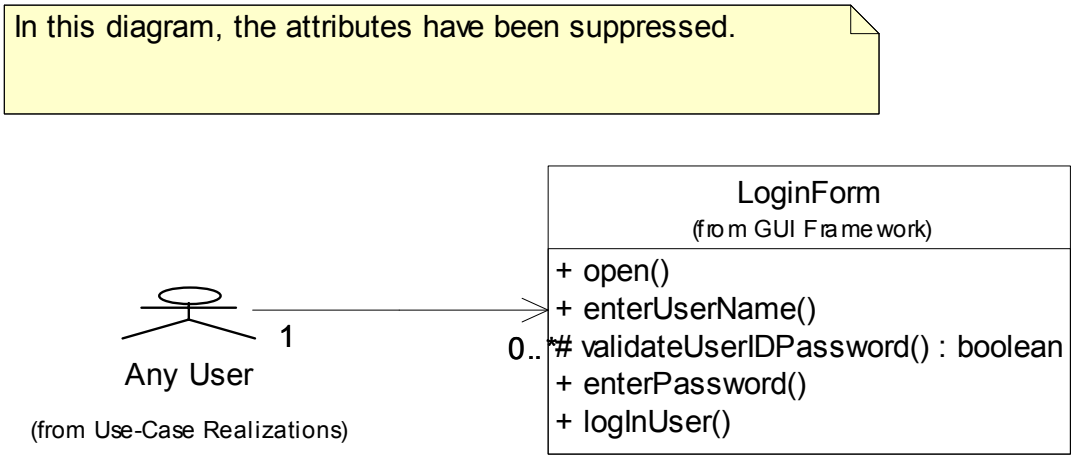
Use-Case Realization - Maintain Timecard

Maintain Timecard - VOPC (with ss interface; ops only)



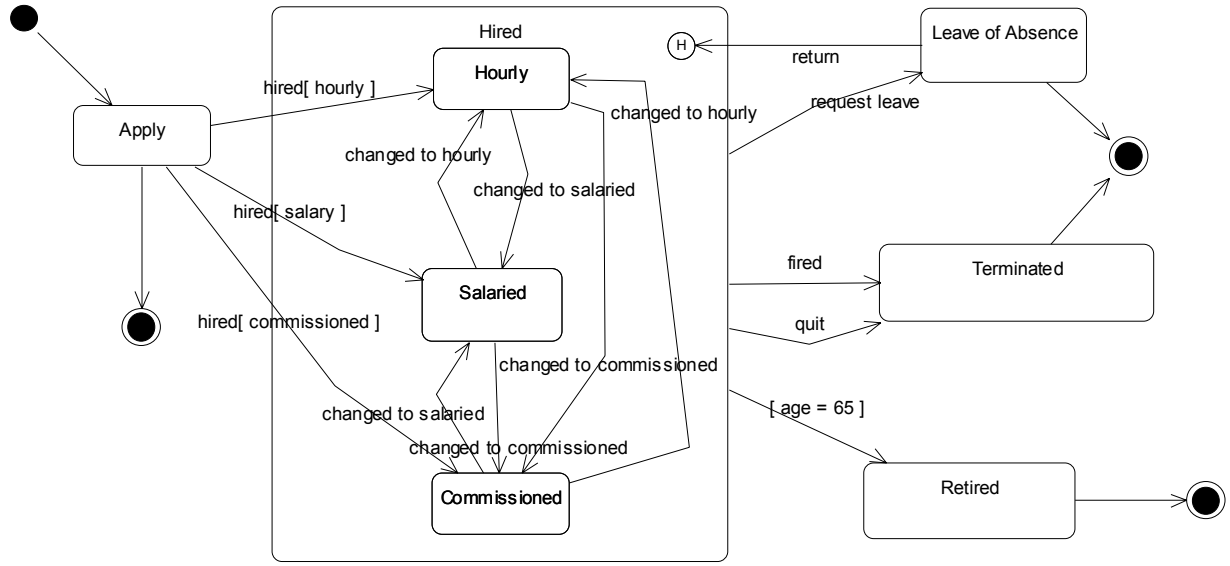
Use-Case Realization - Login

Login - VOPC (ops only)



1.2 Exercise: Define States

Employee

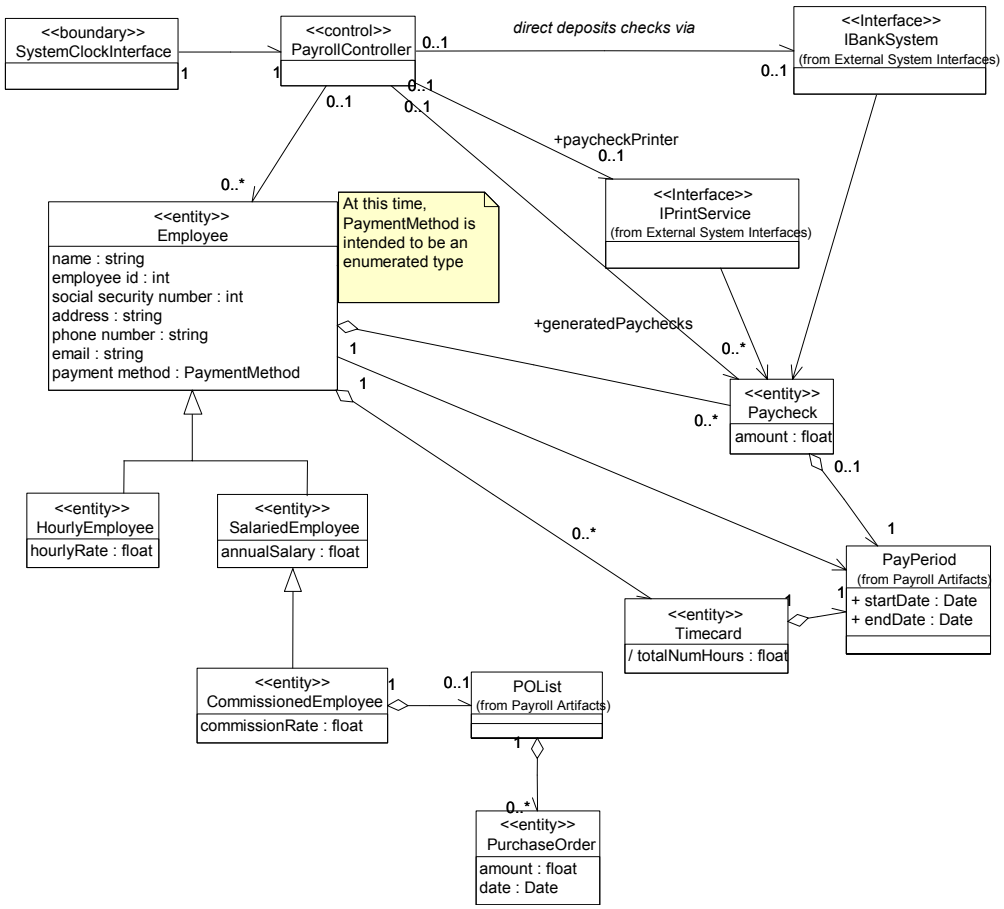


1.3 Exercise: Define Attributes

Use-Case Realization - Run Payroll

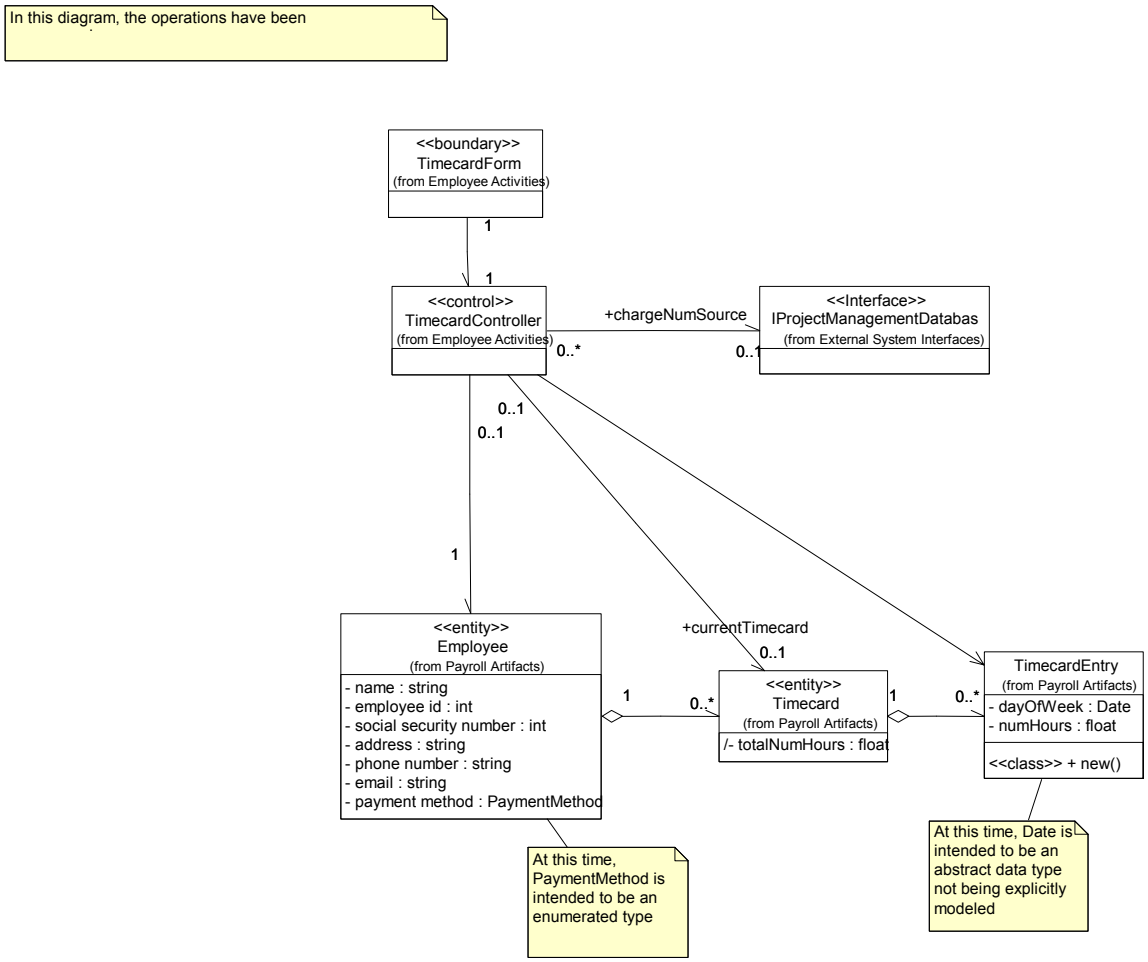
Run Payroll - VOPC (with ss interface; attr only)

In this diagram, the operations have been



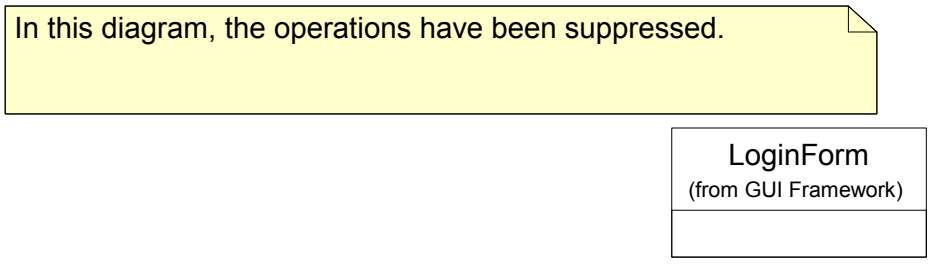
Use-Case Realization - Maintain Timecard

Maintain Timecard - VOPC (with ss interface; attr only)



Use-Case Realization - Login

Login - VOPC (attr only)

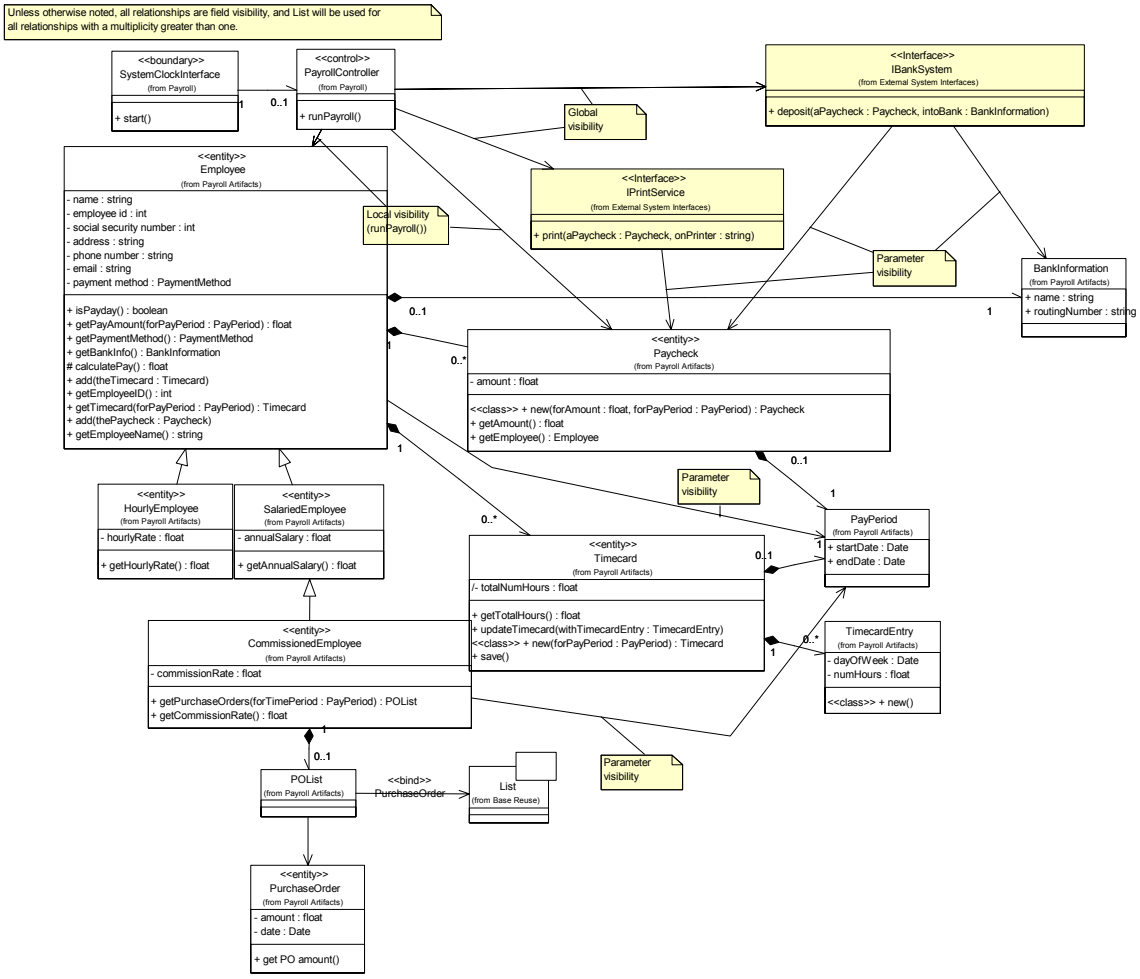


1.4 Exercise: Define Dependencies and Associations

1.4.1 Use-Case Realization - Run Payroll

1.4.1.1 Run Payroll (with ss interface)

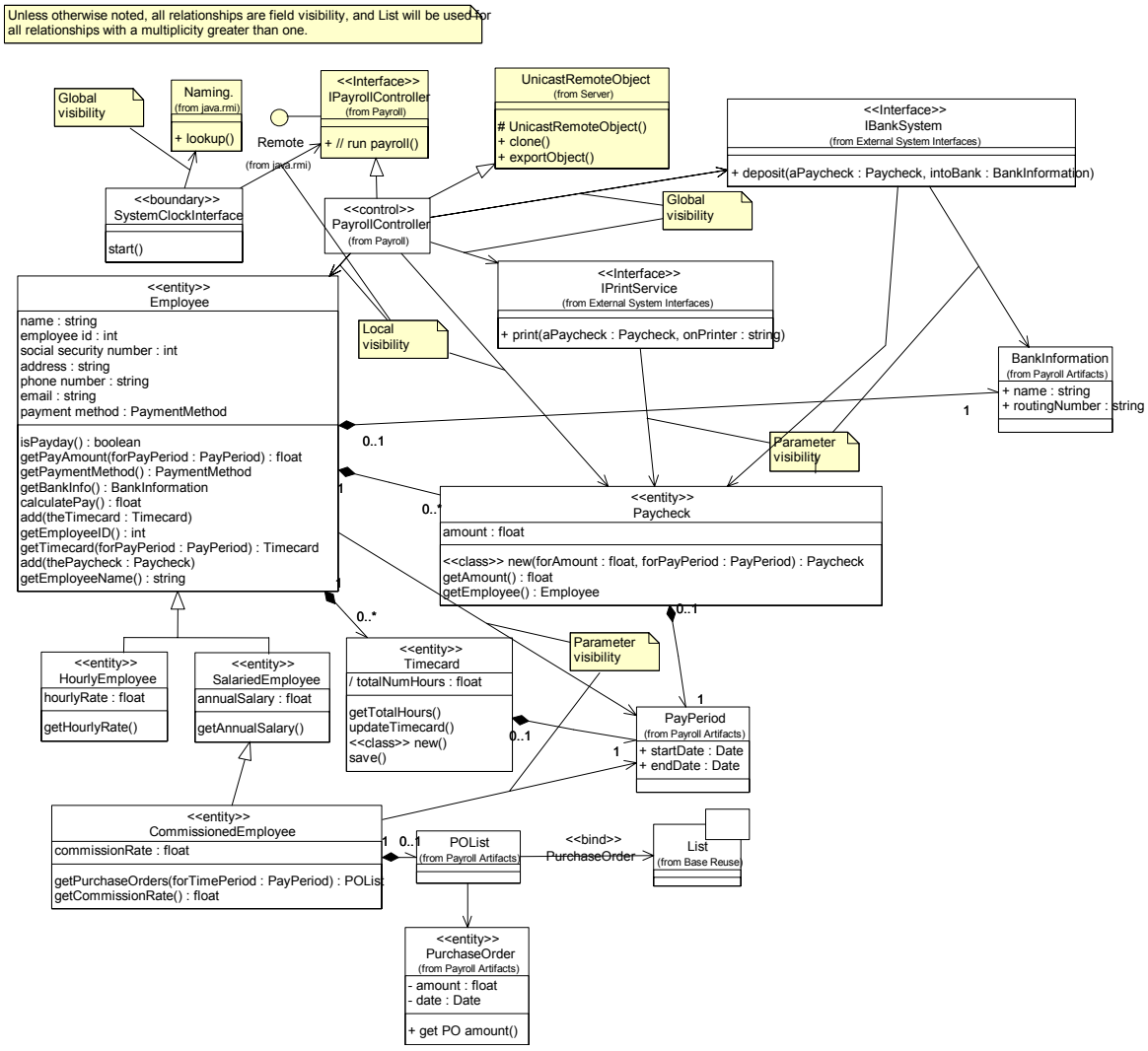
Run Payroll - VOPC (with ss interface)



1.4.1.2 Run Payroll (with Security)

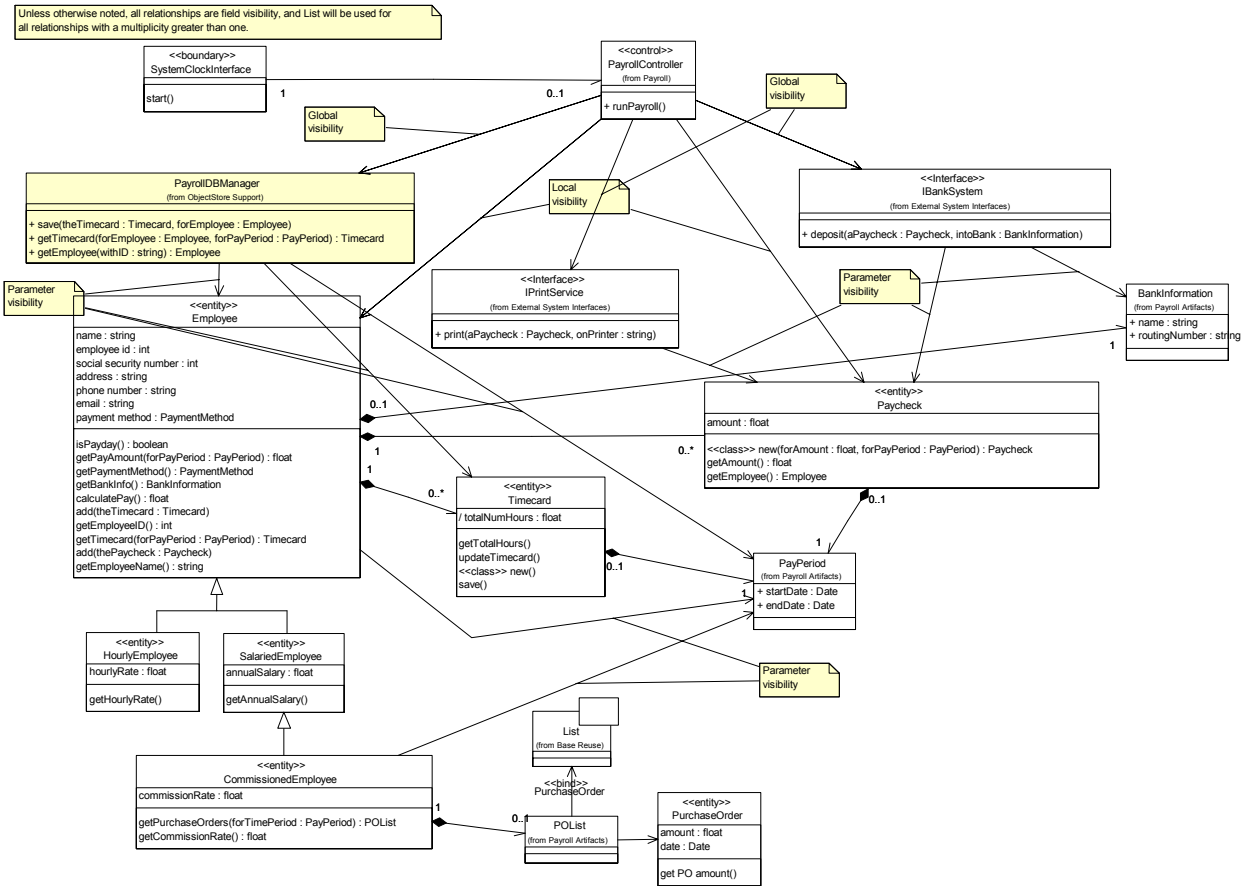
1.4.1.3 Run Payroll (with Distribution)

Run Payroll - VOPC (with Distribution)



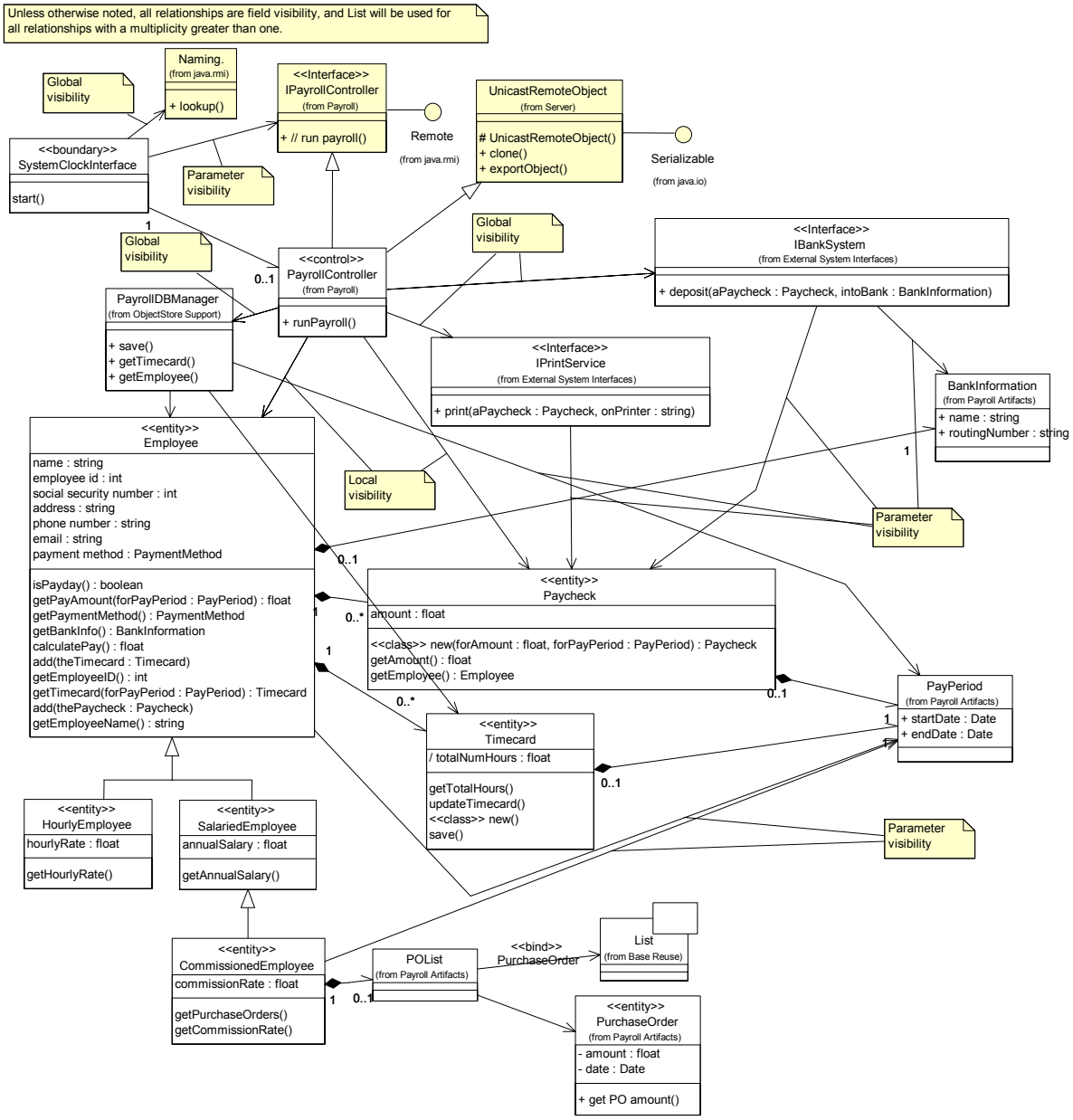
1.4.1.4 Run Payroll (with OODBMS Persistency)

Run Payroll - VOPC (with OODBMS Persistency)



1.4.1.5 Run Payroll (with everything)

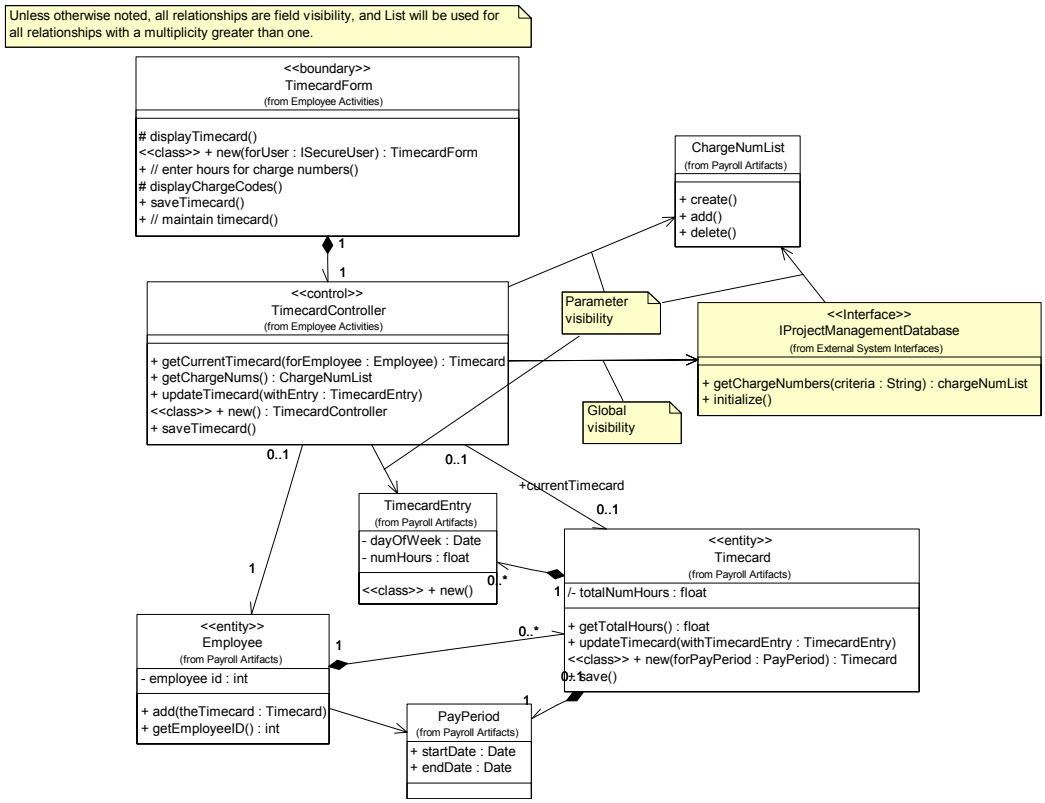
Run Payroll - VOPC (with everything)



1.4.2 Use-Case Realization - Maintain Timecard

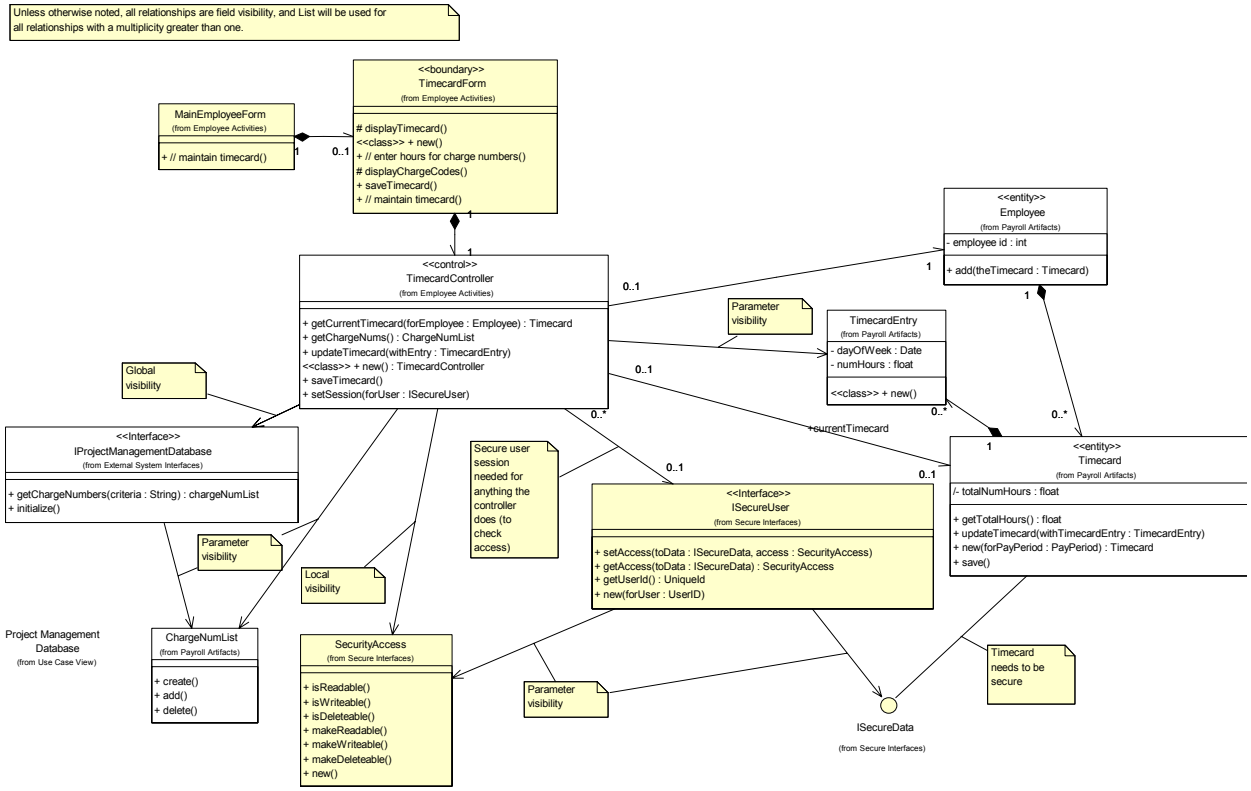
1.4.2.1 Maintain Timecard (with ss interface)

Maintain Timecard - VOPC (with ss interface)



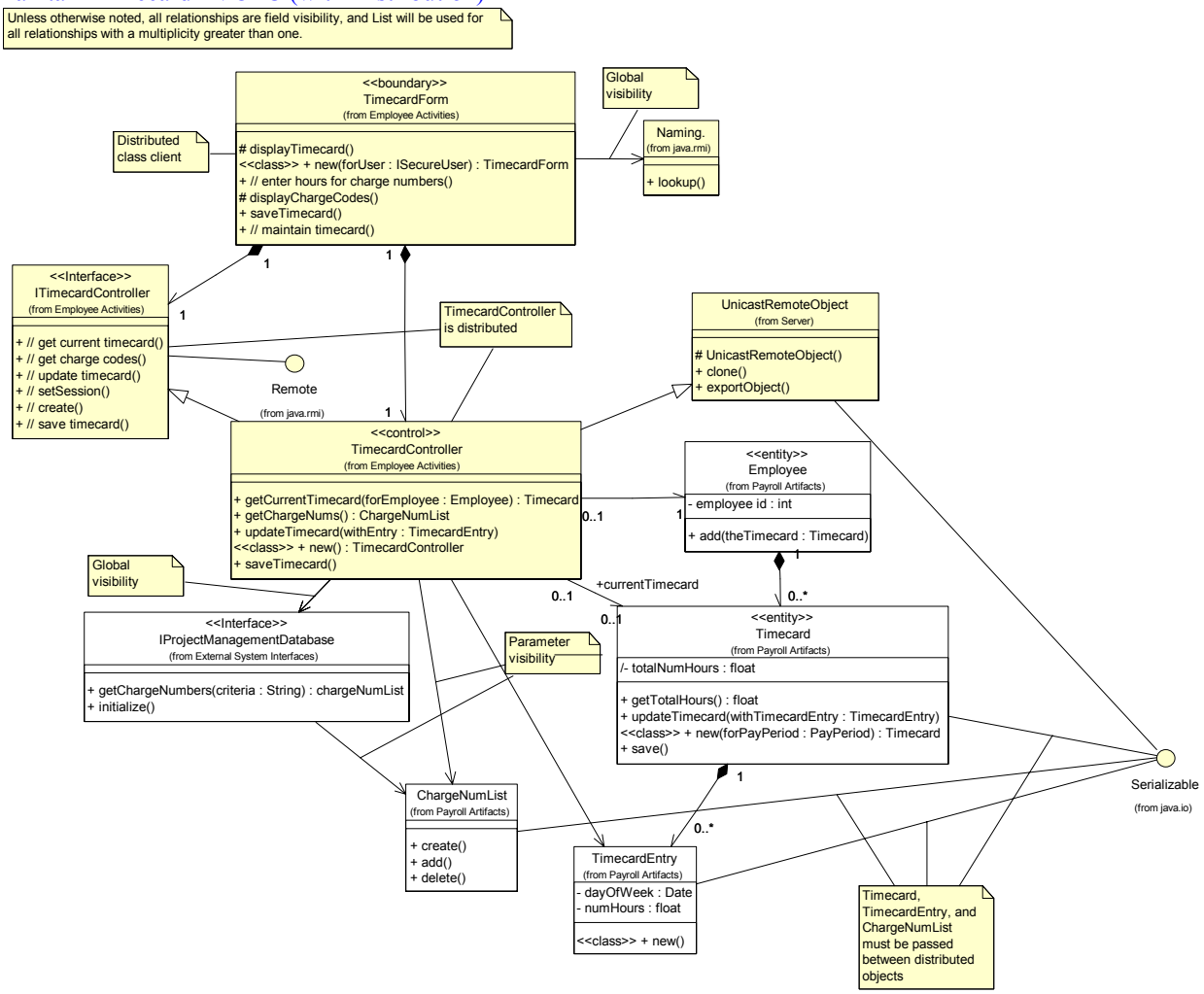
1.4.2.2 Maintain Timecard (with Security)

Maintain Timecard - VOPC (with Security)



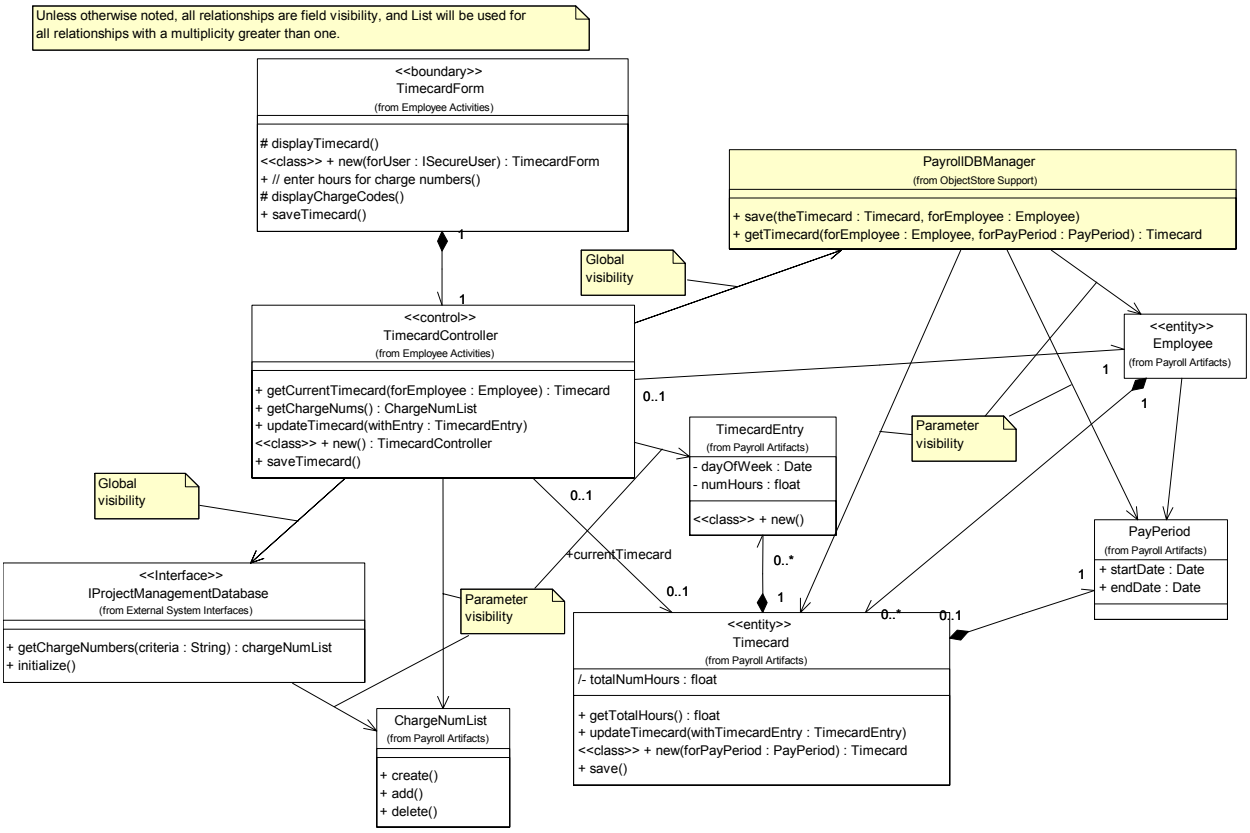
1.4.2.3 Maintain Timecard (with Distribution)

Maintain Timecard - VOPC (with Distribution)



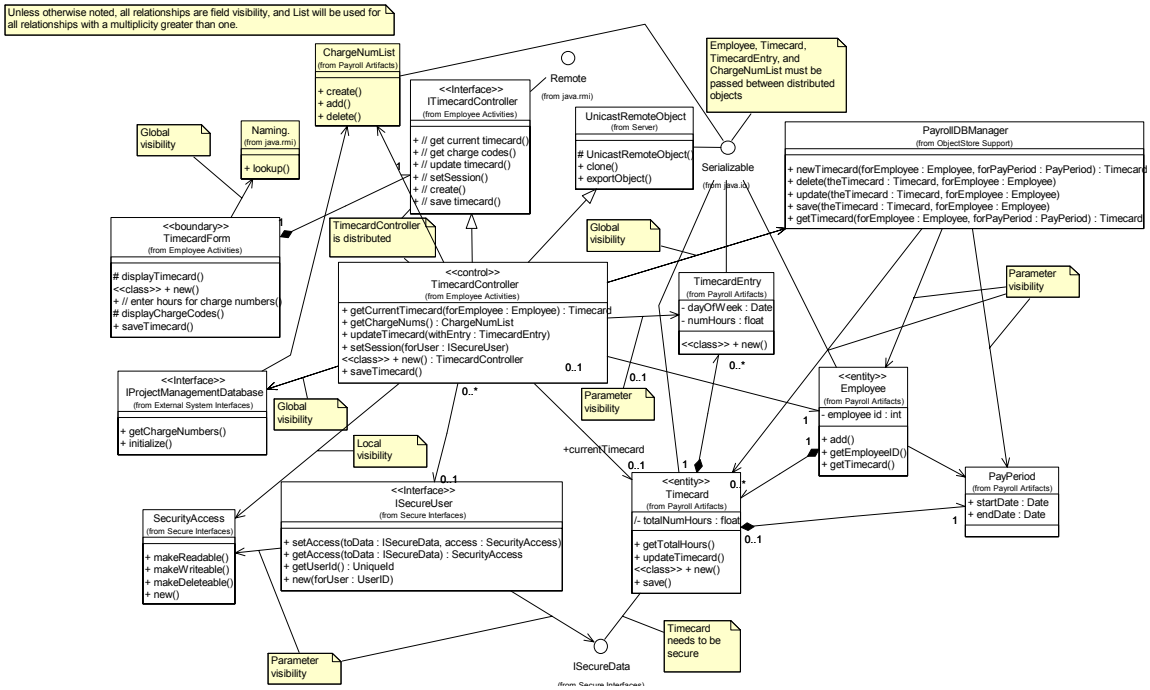
1.4.2.4 Maintain Timecard (with OODBMS Persistence)

Maintain Timecard - VOPC (with OODBMS Persistency)



1.4.2.5 Maintain Timecard (with everything)

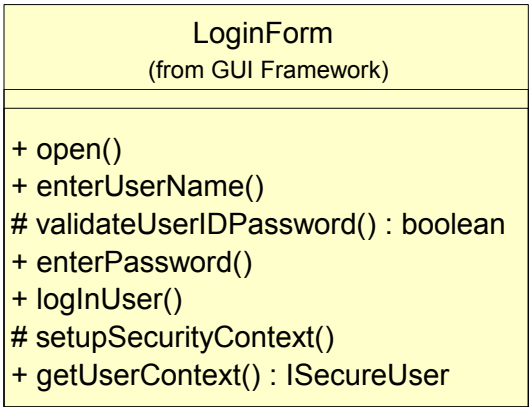
Maintain Timecard - VOPC (with everything)



1.4.3 Use-Case Realization - Login

1.4.3.1 Login

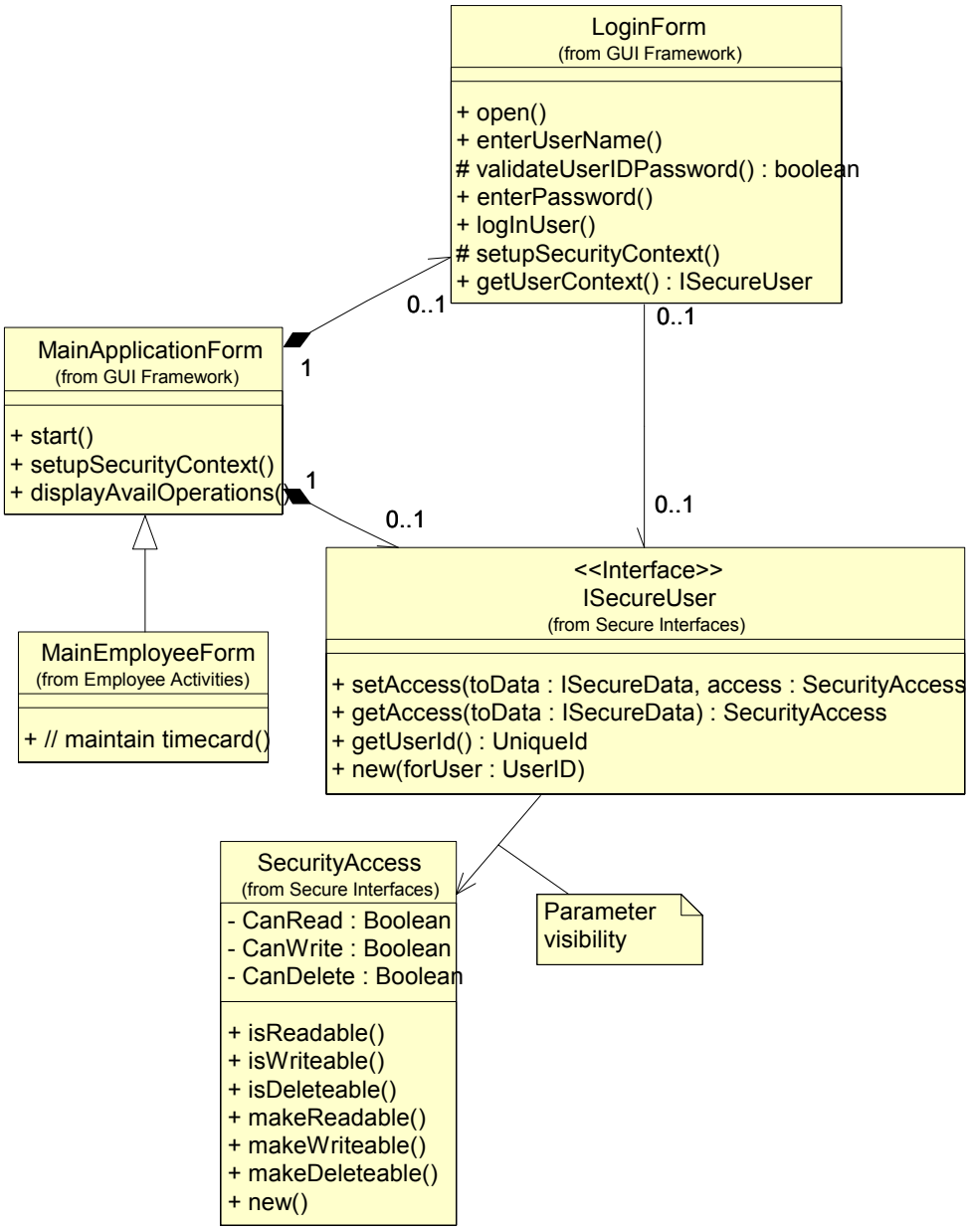
Login - VOPC



1.4.3.2 Login (with Security)

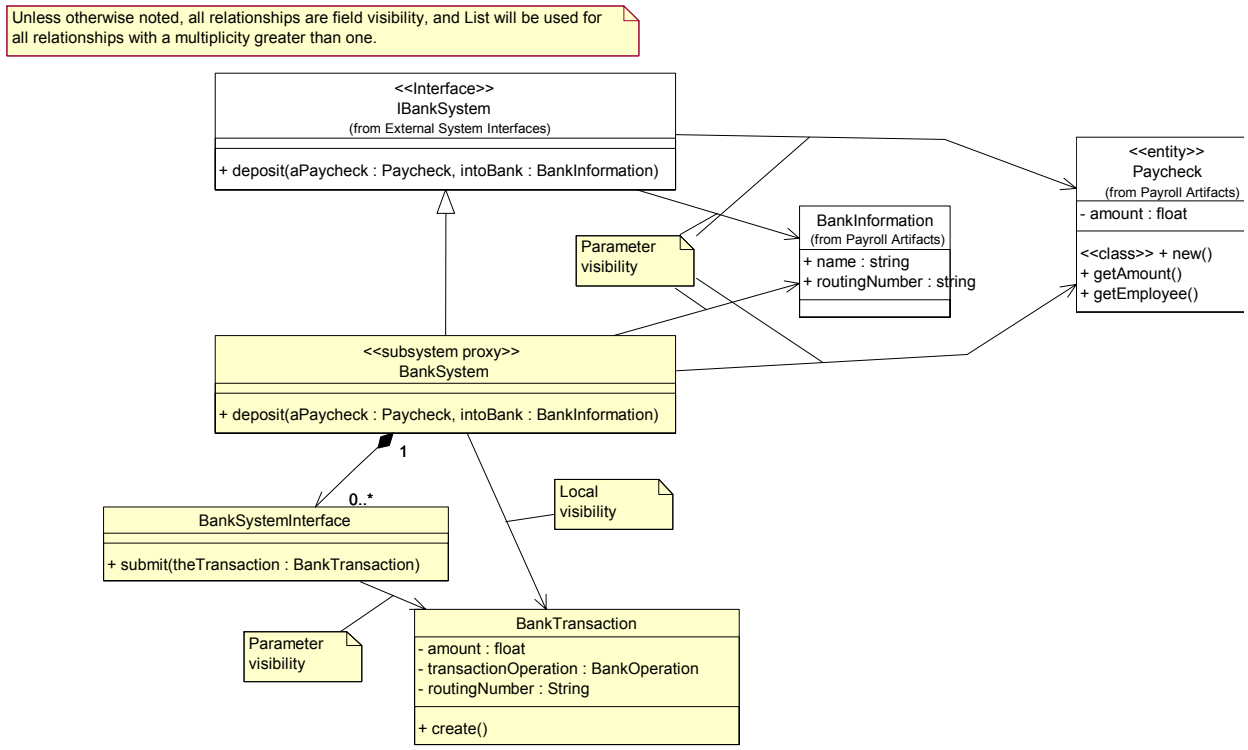
Login - VOPC (with Security)

Unless otherwise noted, all relationships are field visibility, and List will be used for all relationships with a multiplicity greater than one.



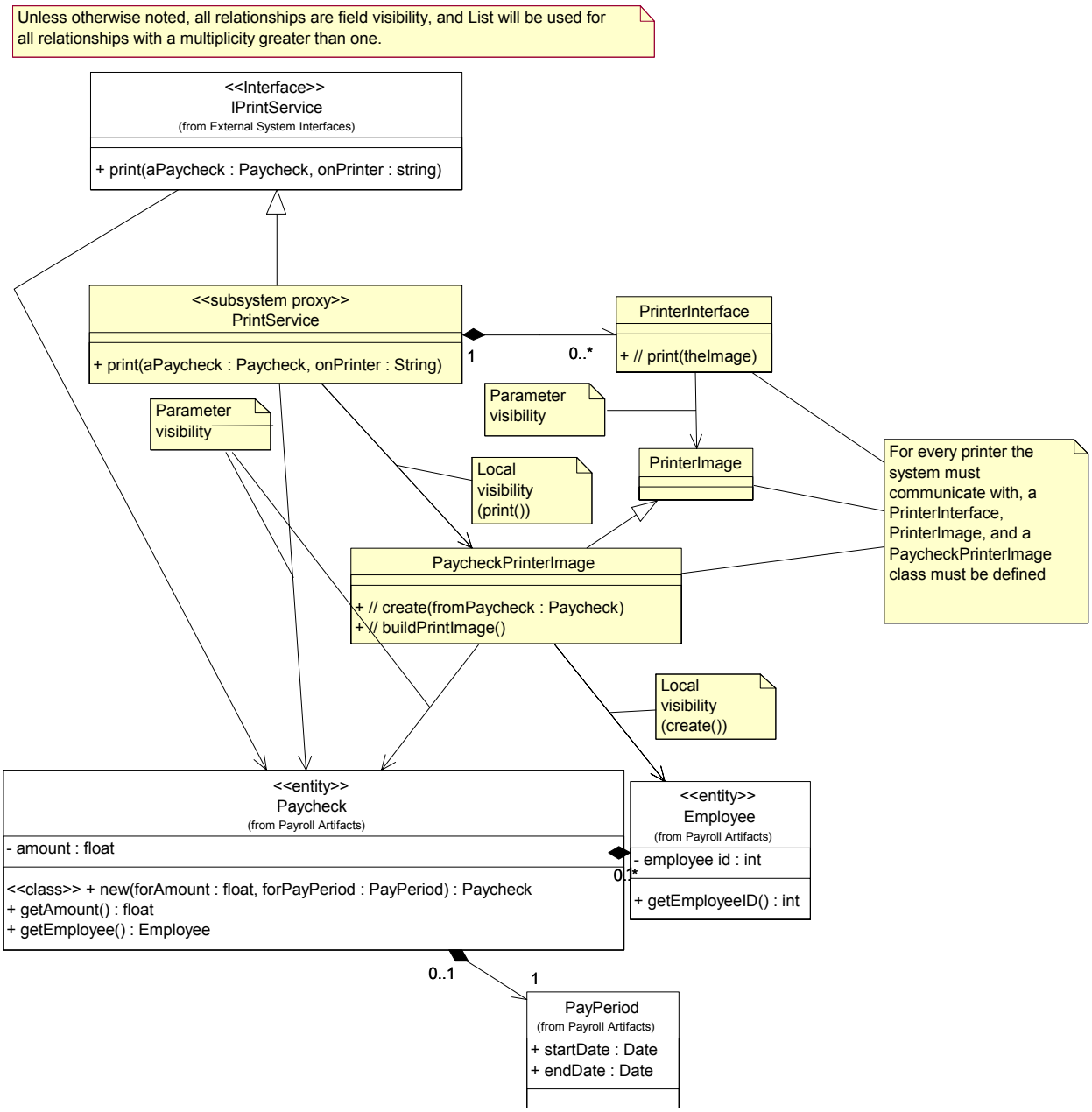
1.4.4 BankSystem

Main



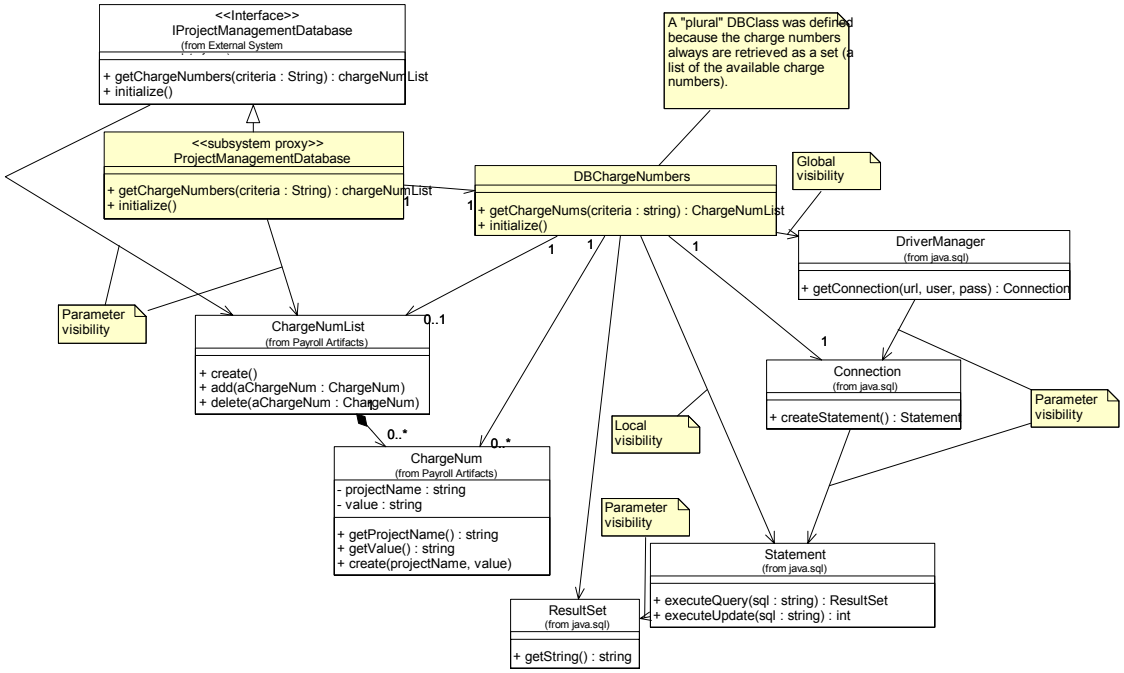
1.4.5 PrintService

Main



1.4.6 ProjectManagementDatabase

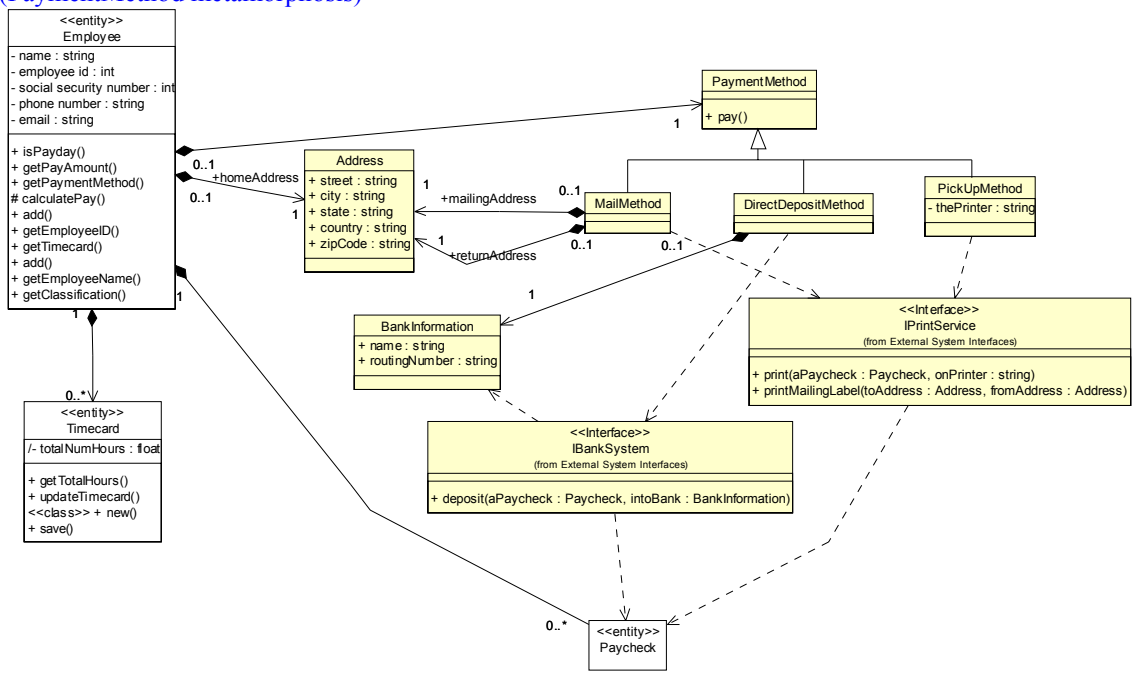
Main



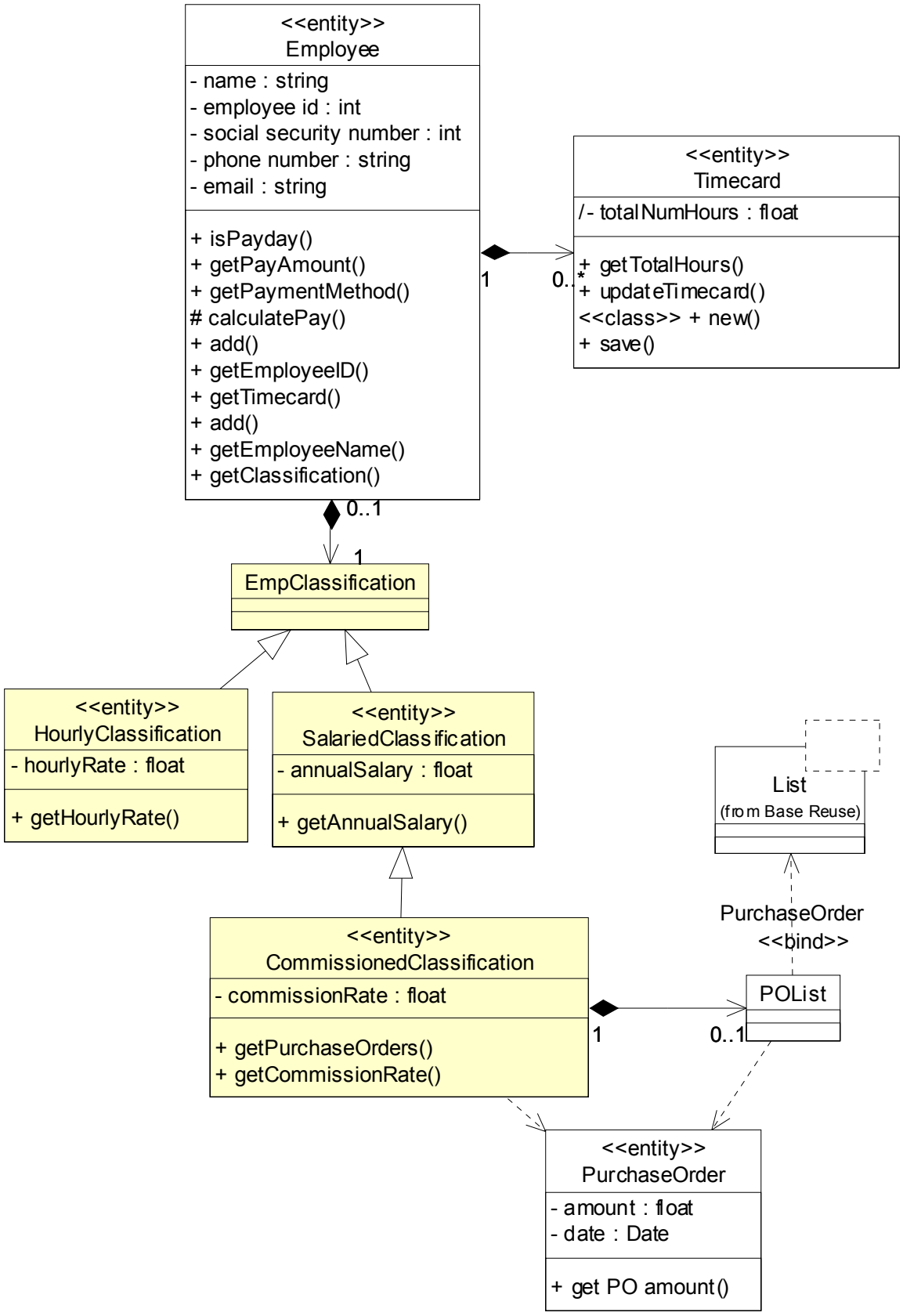
1.5 Exercise: Define Generalizations

Metamorphosis was applied to the Payroll System model. The employee payment method (pick-up, mail, or direct deposit) and the employee classification (hourly, salaried, or commission) were both abstracted out into individual classes and supporting subclasses.

Employee (PaymentMethod metamorphosis)



Employee (EmpClassification metamorphosis)



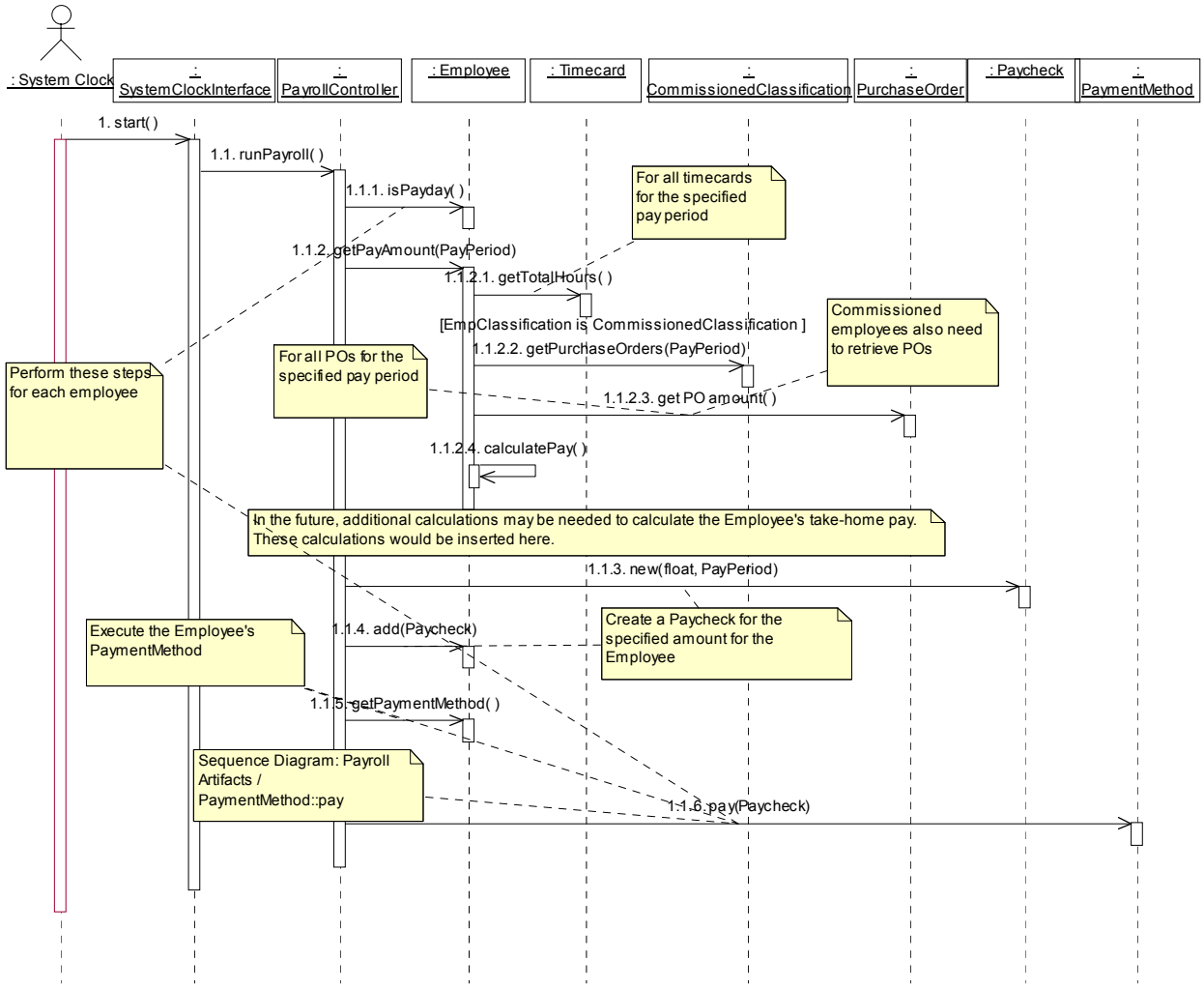
This required certain changes to the design, the most notable of which is the introduction of a dependency from the Payroll Artifacts package to the External System Interfaces package. While this may seem strange, it is required in order to allow the new payment method classes to execute themselves (i.e. to access the external systems).
This is an example where a design trade-off was made – smarter, more encapsulated payment method classes at the expense of non-circular package dependencies. Of course, the packages and their contents could be adjusted to eliminate the cycles (pull the class definitions needed by the External System Interfaces and place them in their own package that the External System Interfaces and Payroll Artifacts package are dependent on. This was not done in this example.

The use of "smarter" payment method classes simplifies the Run Payroll use-case realization as the PayrollController no longer needs to know how to execute the different payment methods.

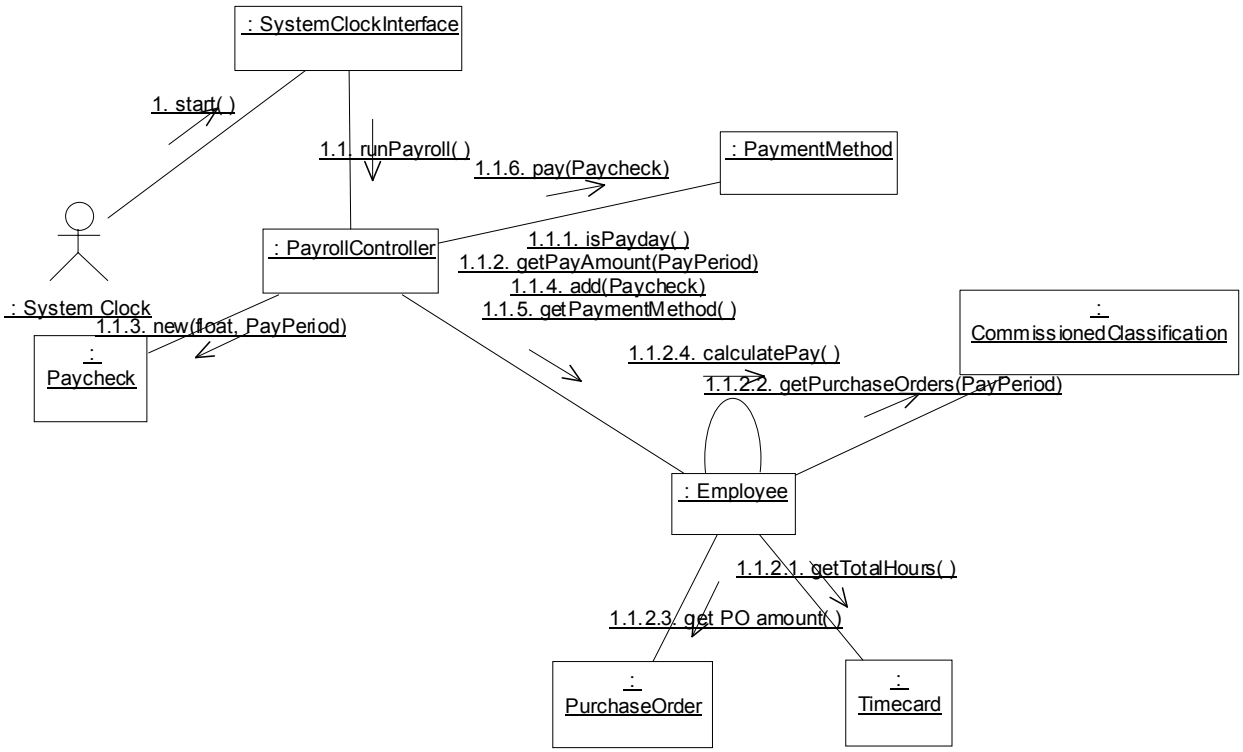
The use of the EmployeeClassification class means that, depending on the type of Employee that is being paid, different pieces of information are extracted.

These changes are reflected in the following updated version of the Run Payroll use-case realization diagrams, as well as some additional interaction diagrams that model the collaborations required for the PaymentMethod::pay() operation:

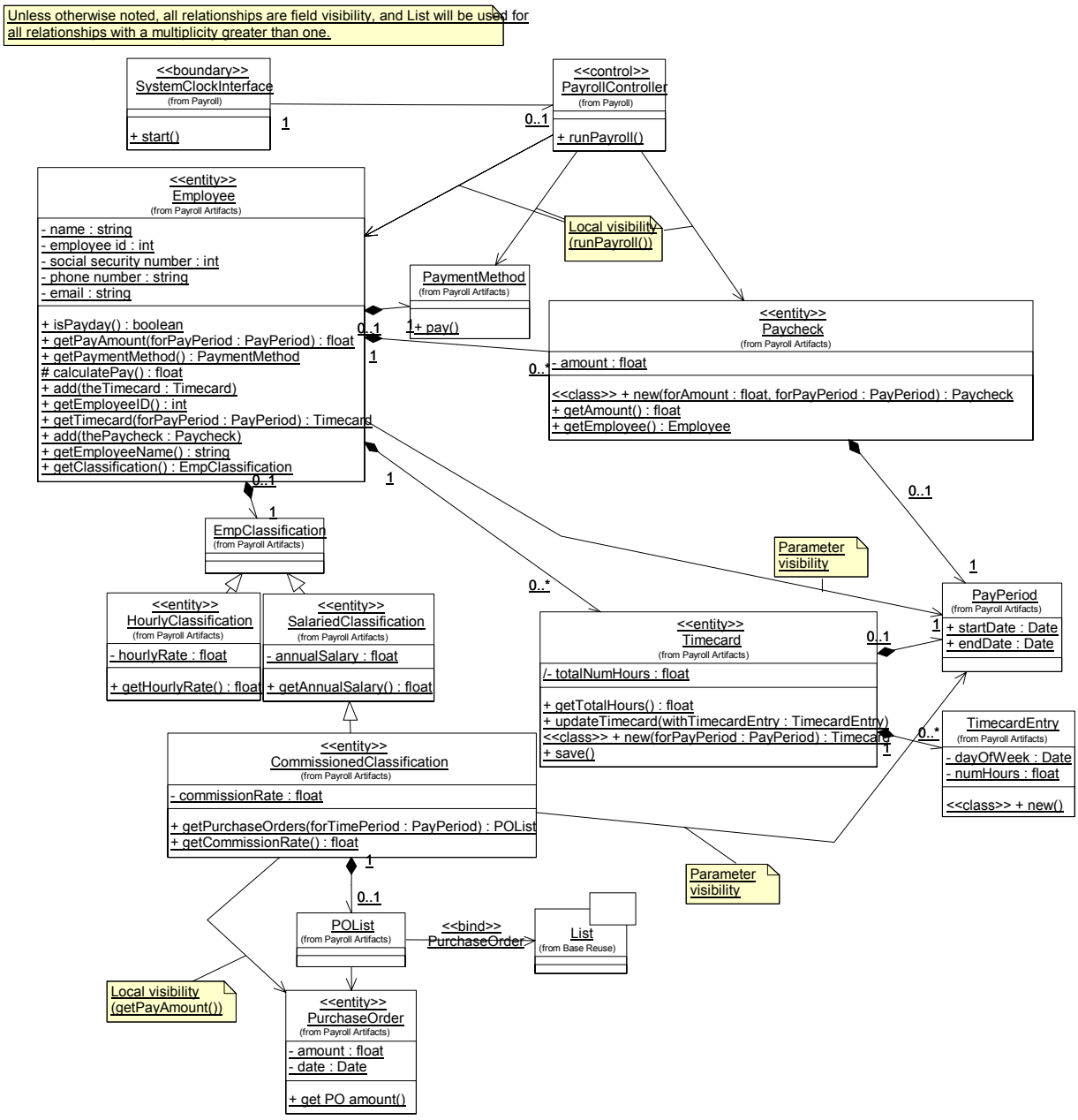
Run Payroll - Basic Flow (with ss interface)



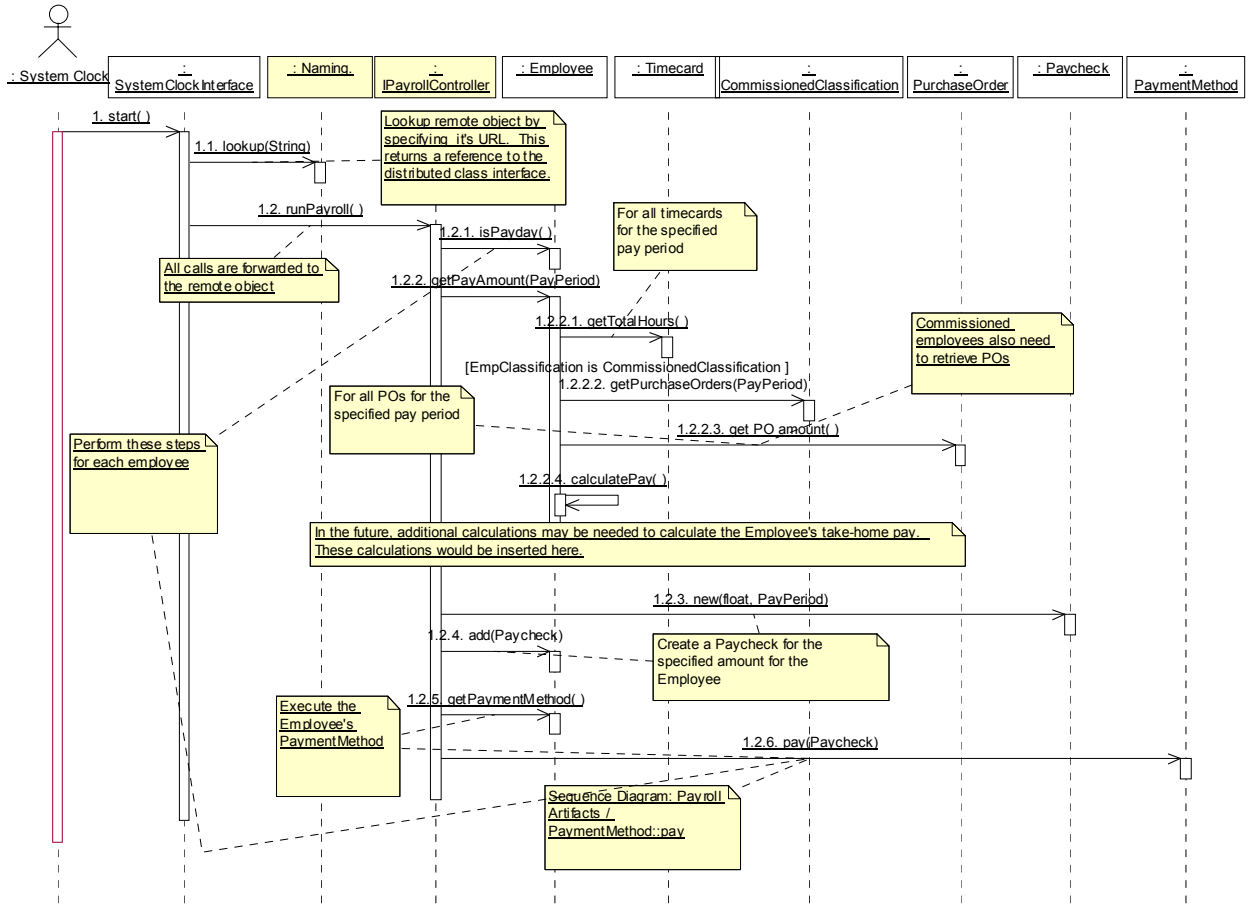
Run Payroll - Basic Flow (with ss interface)



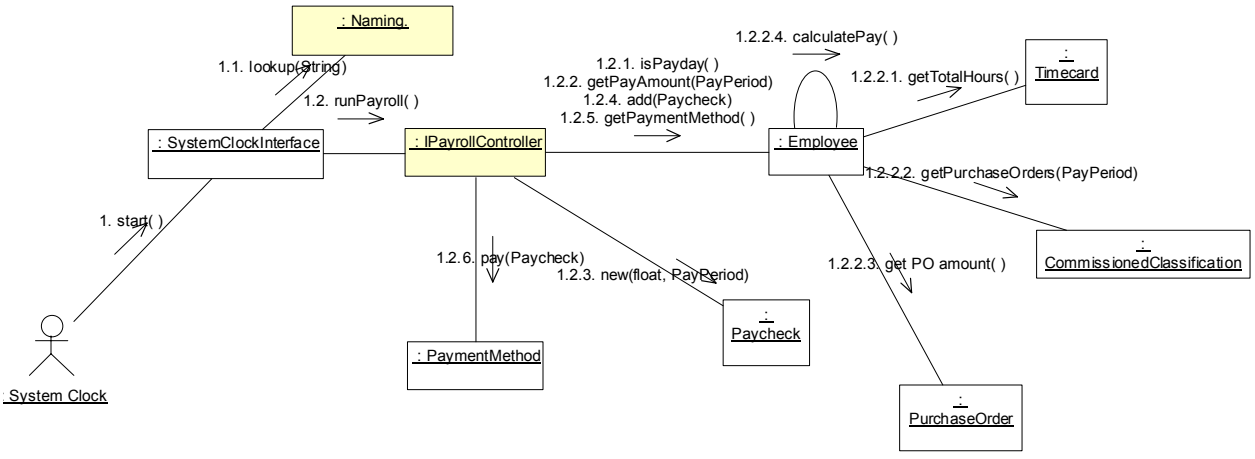
Run Payroll - VOPC (with ss interface)



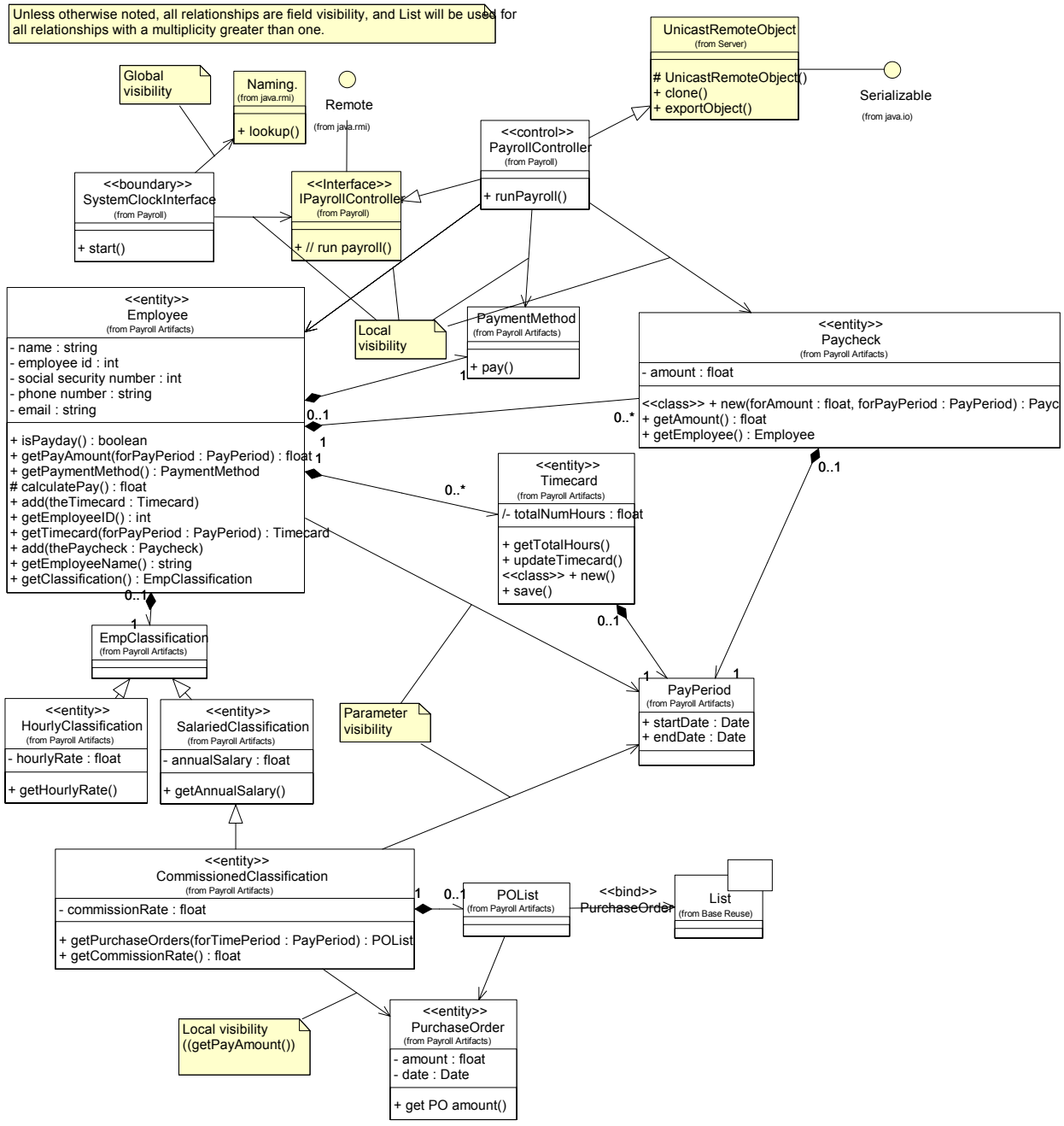
Run Payroll - Basic Flow (with Distribution)



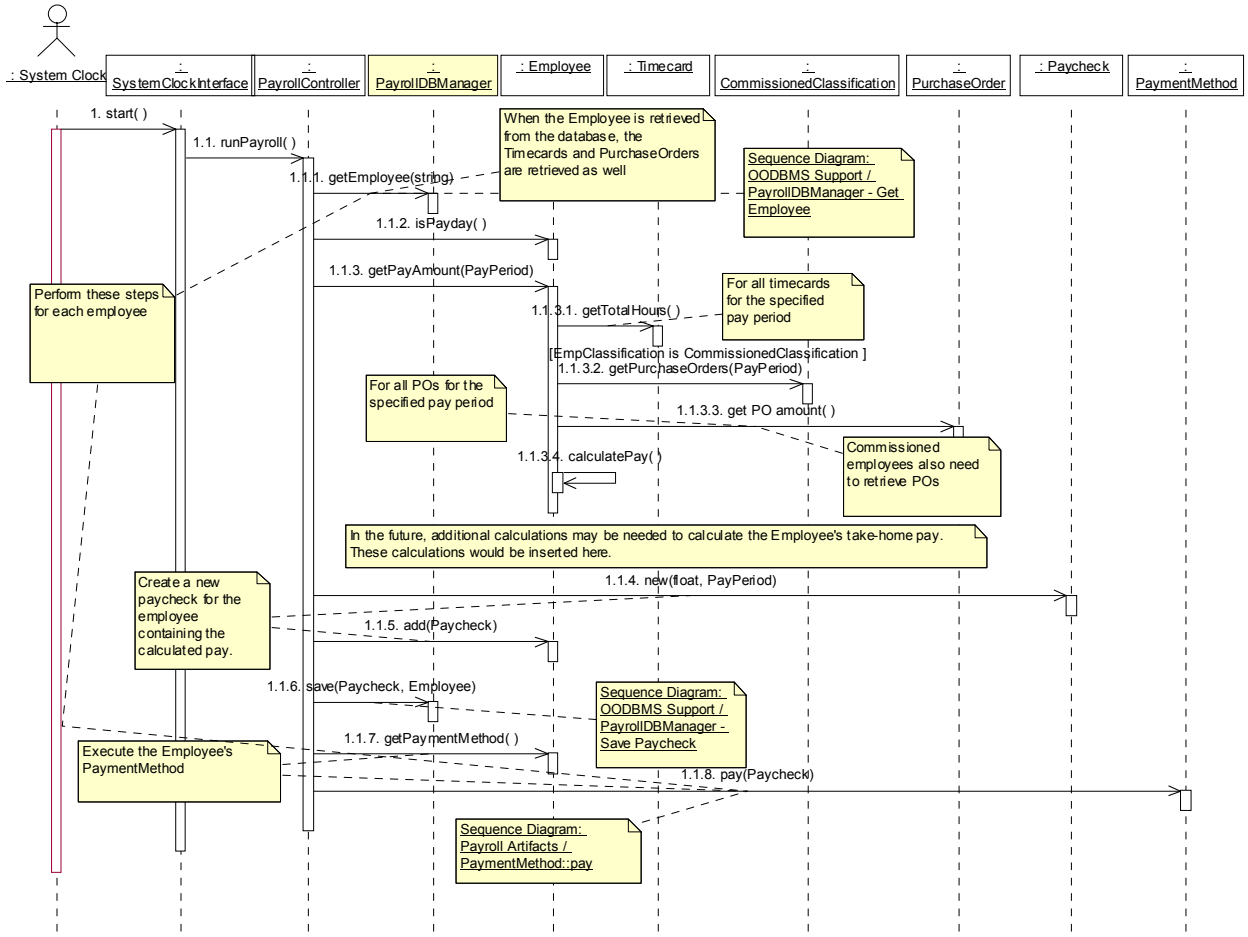
Run Payroll - Basic Flow (with Distribution)



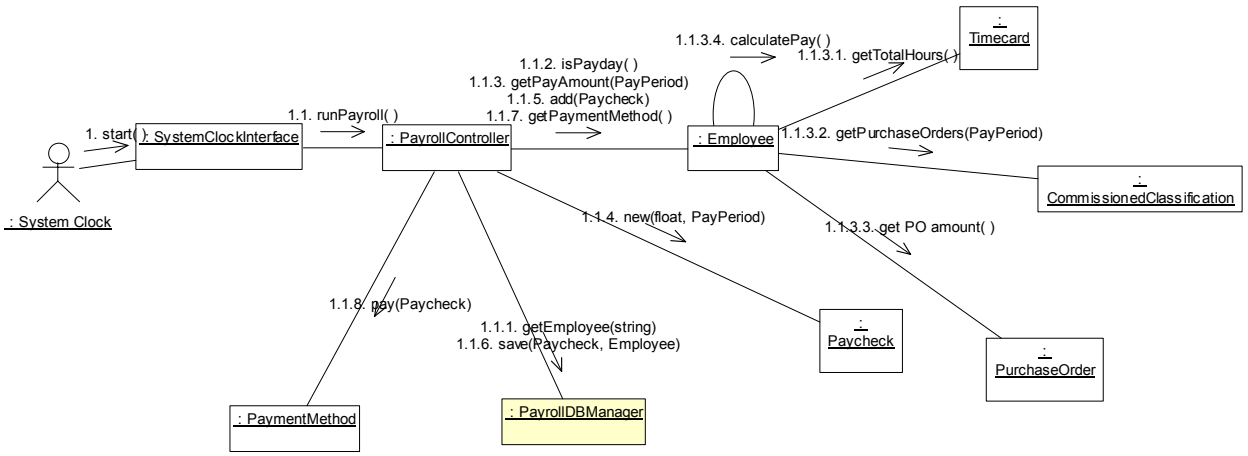
Run Payroll - VOPC (with Distribution)



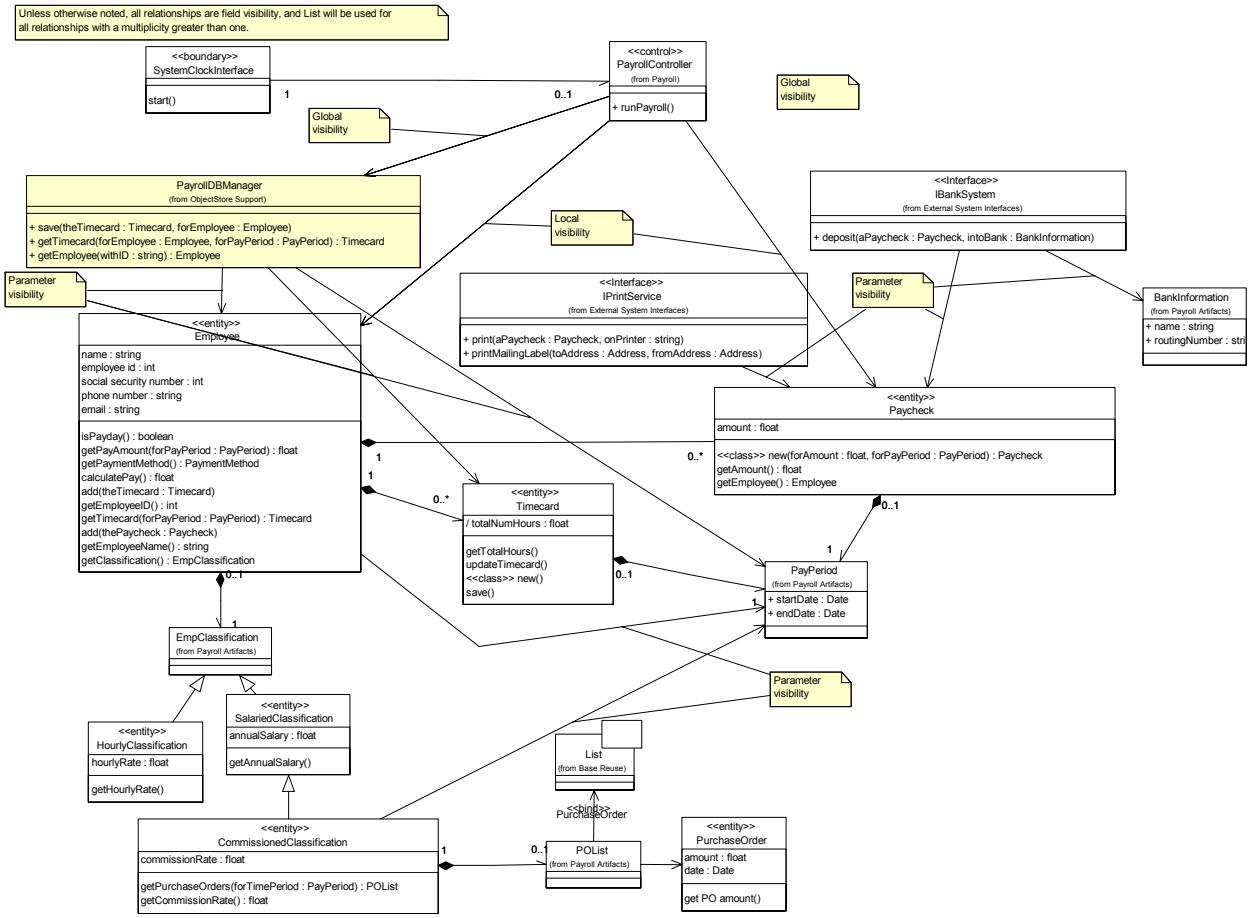
Run Payroll - Basic Flow (with OODBMS Persistency)



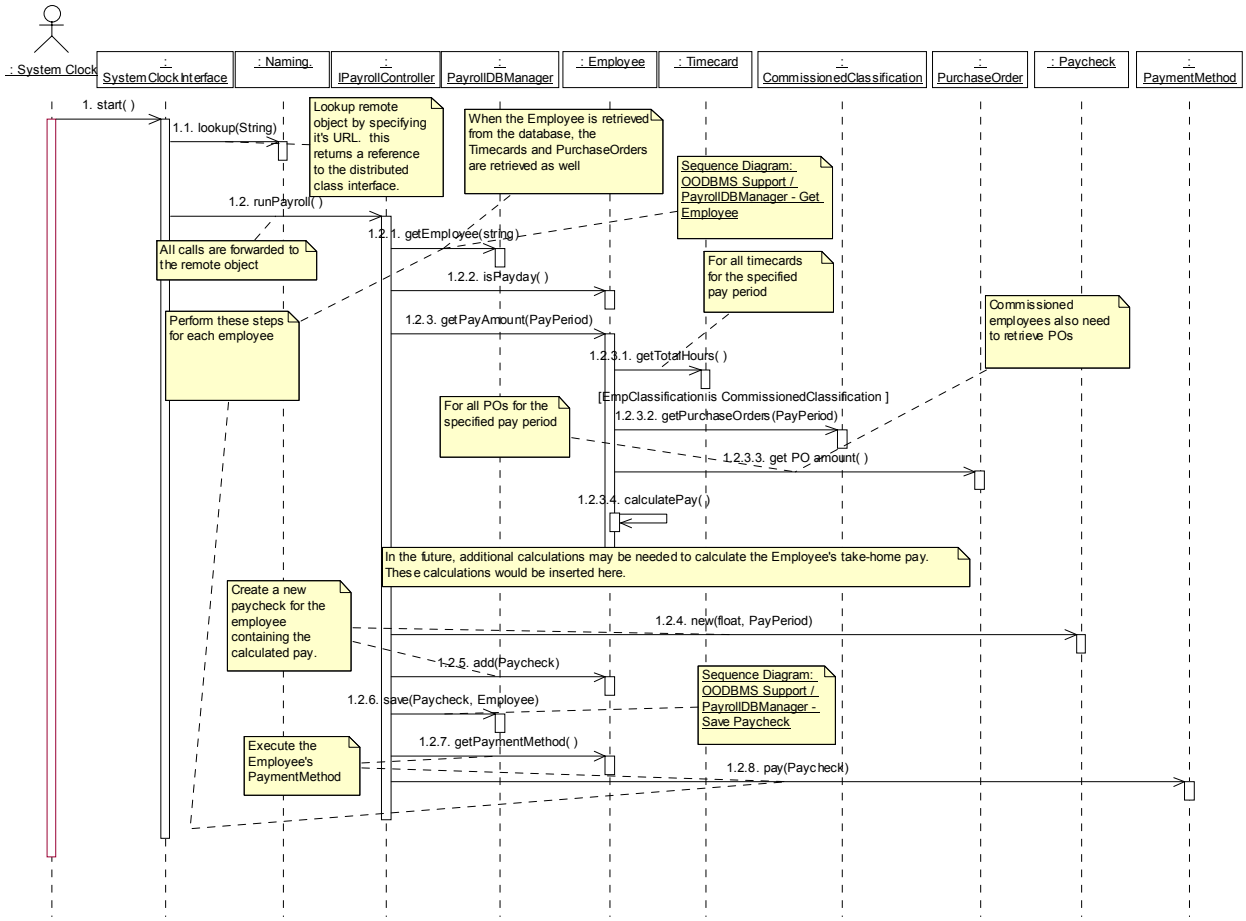
Run Payroll - Basic Flow (with OODBMS Persistency)



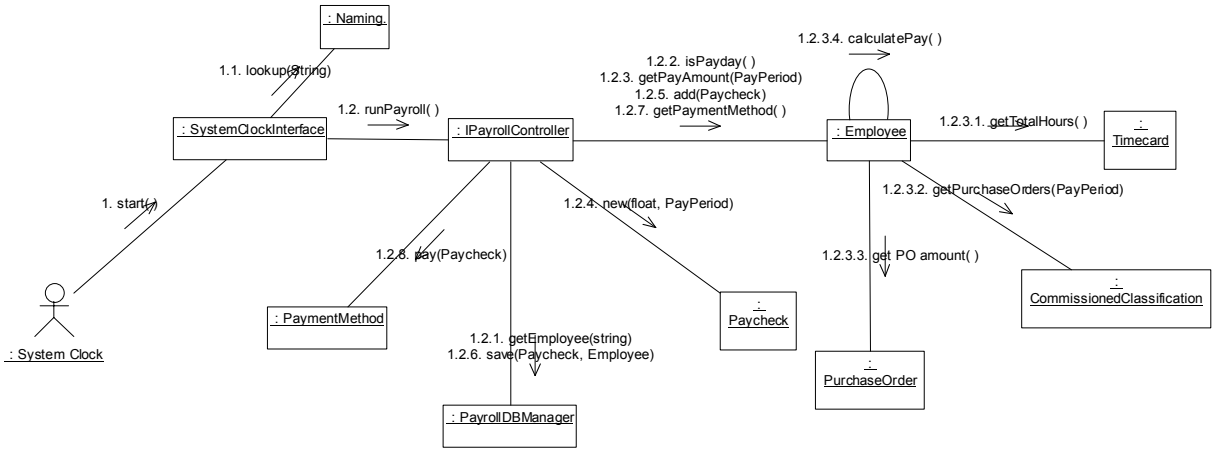
Run Payroll - VOPC (with OODBMS Persistency)



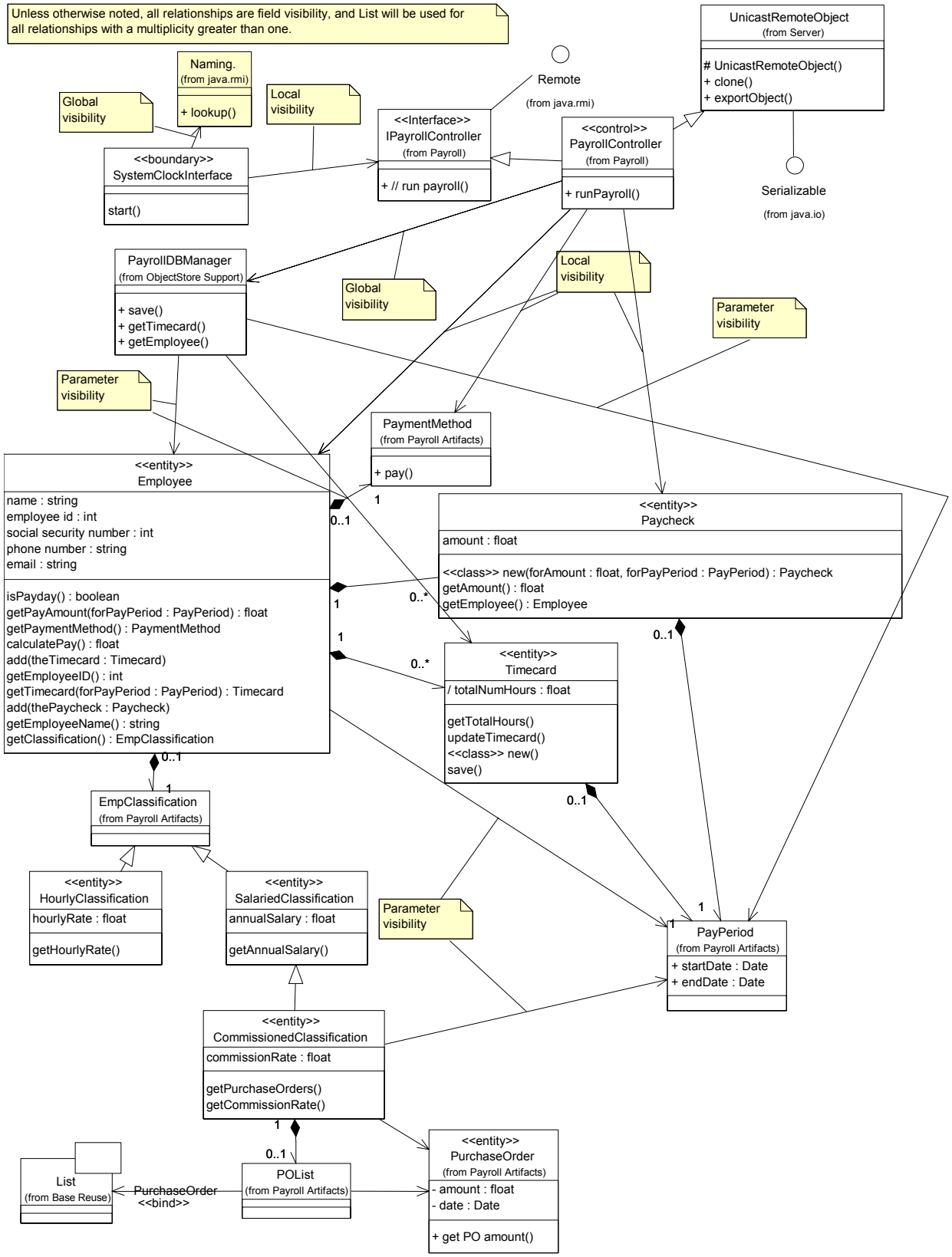
Run Payroll - Basic Flow (with everything)



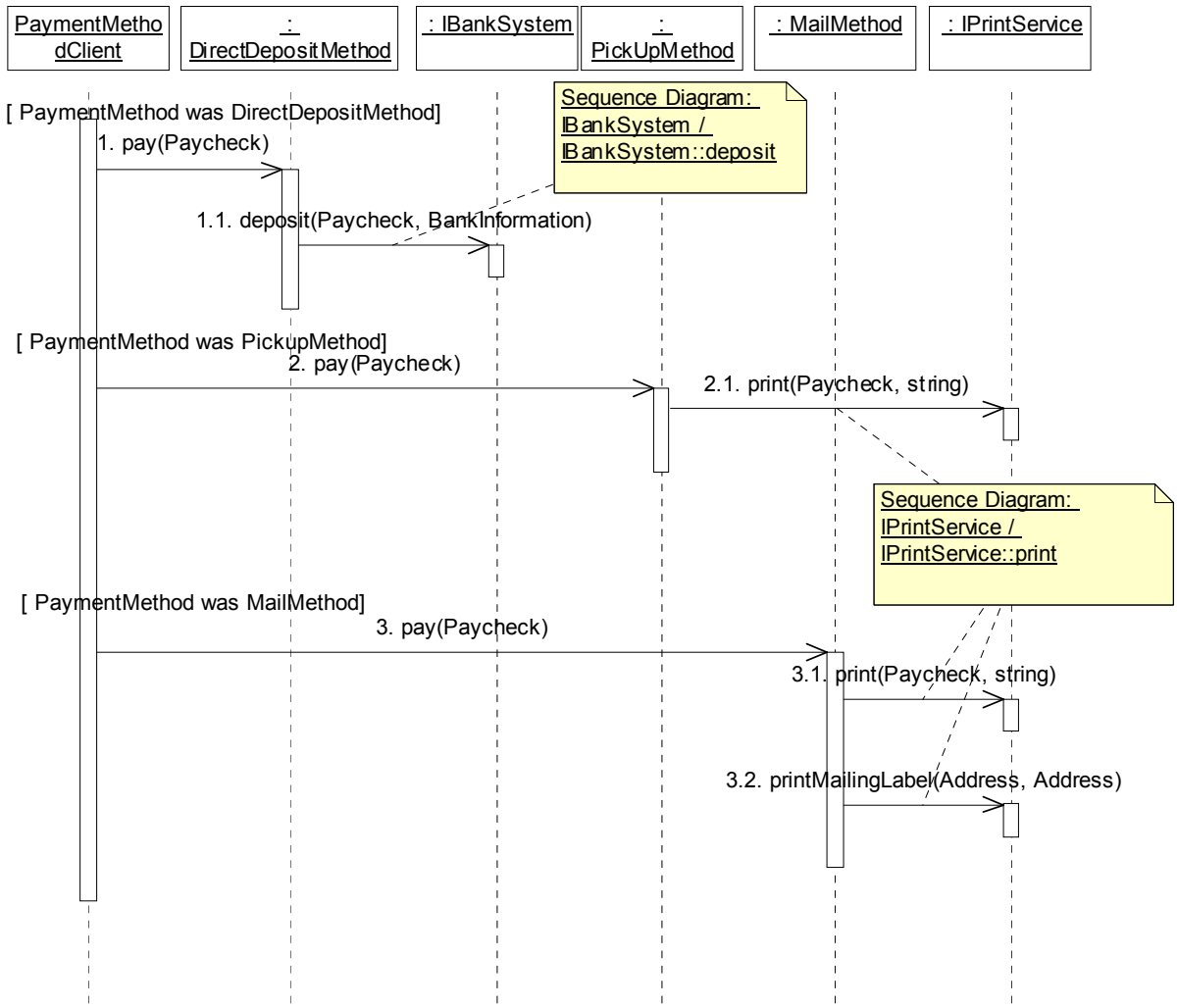
Run Payroll - Basic Flow (with everything)



Run Payroll - VOPC (with everything)



PaymentMethod::pay



PaymentMethod::pay