Rational Software

Payroll System Use-Case Design Solution

Version 2003

Revision History

Date	Issue	Description	Author
09/01/2000	V2000	Generate for beta	Shawn Siemers
10/02/2000	V2000	Final release	Shawn Siemers
01/14/2003	V2003	Final Release	Alex Kutsick

Table of Contents

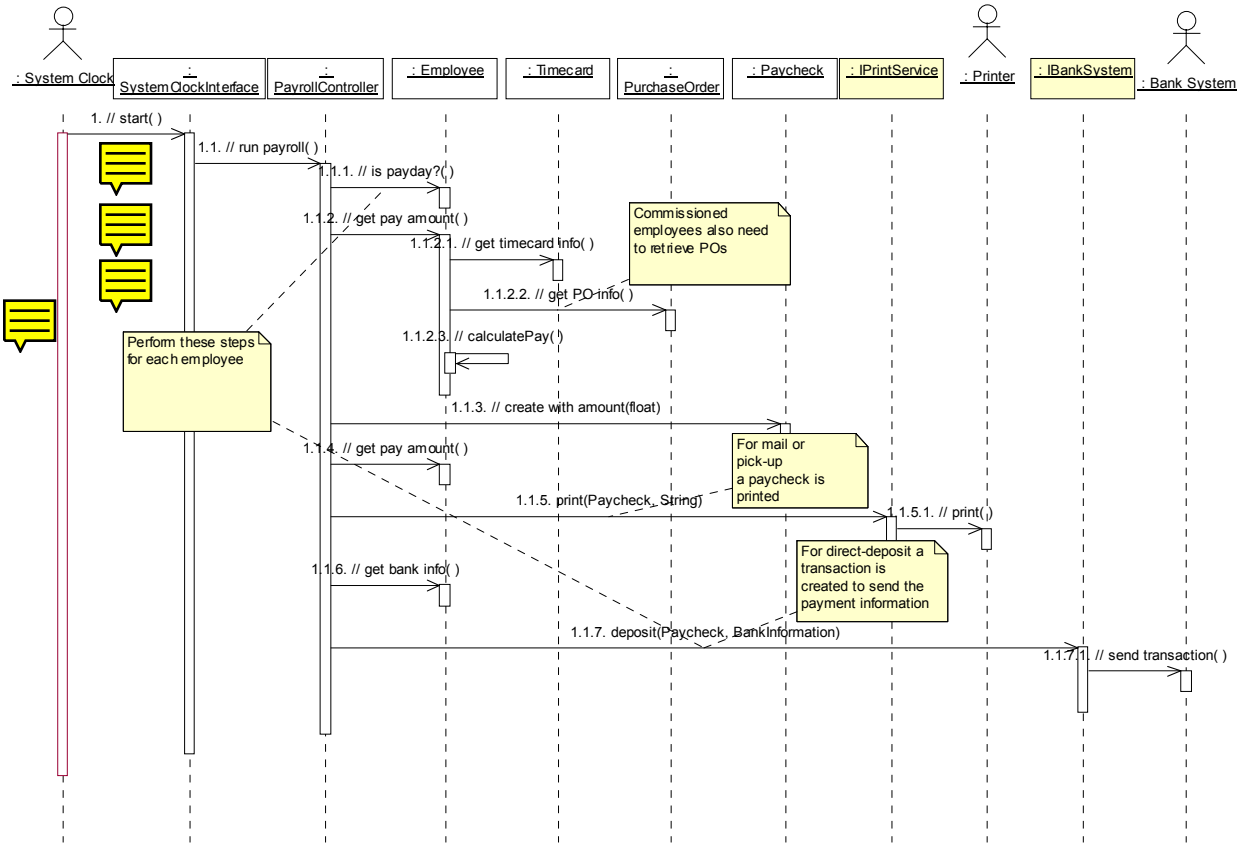
Payroll System Use-Case Design Solution

1. Exercise: Use-Case Design, Part 1

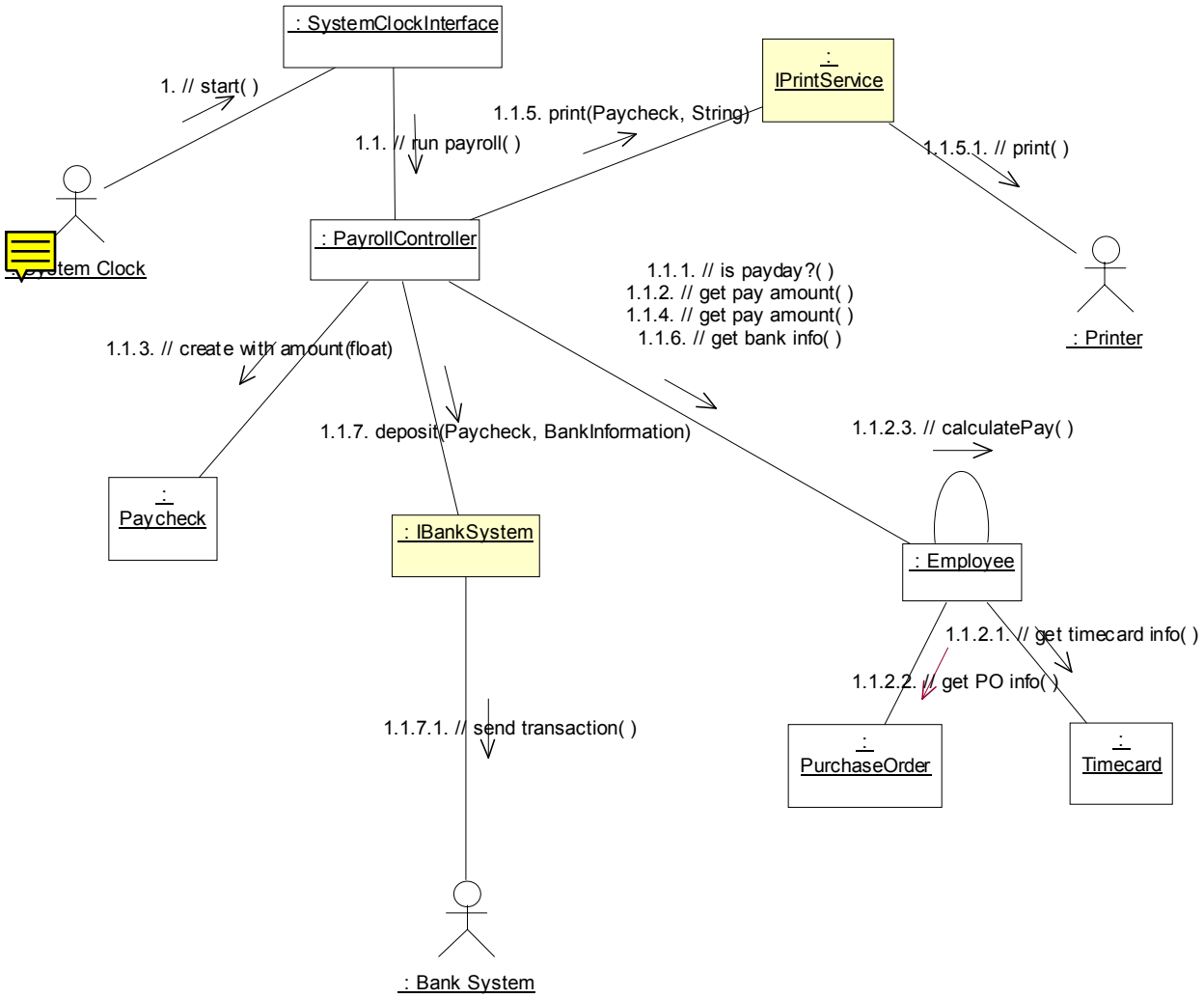
1.1 Use-Case Realization - Run Payroll

1.1.1 Run Payroll (with ss interface)

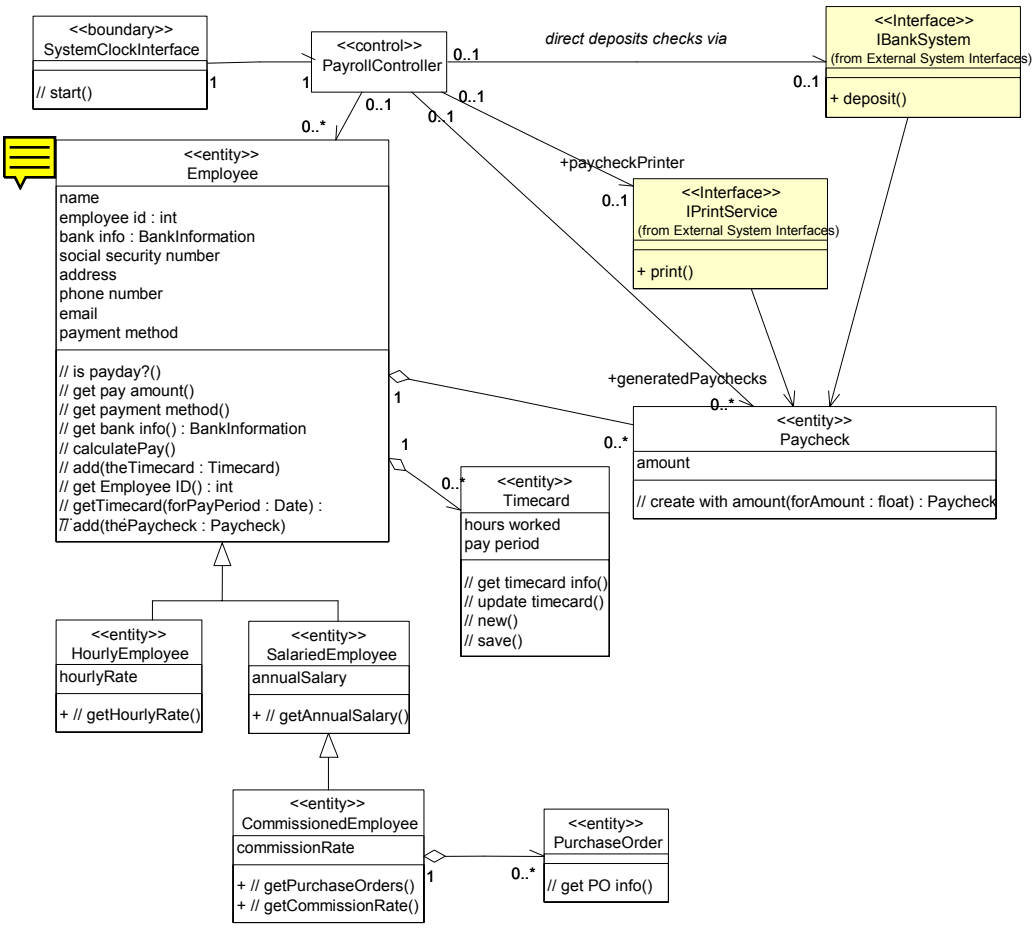
Run Payroll - Basic Flow (with ss interface)



Run Payroll - Basic Flow (with ss interface)



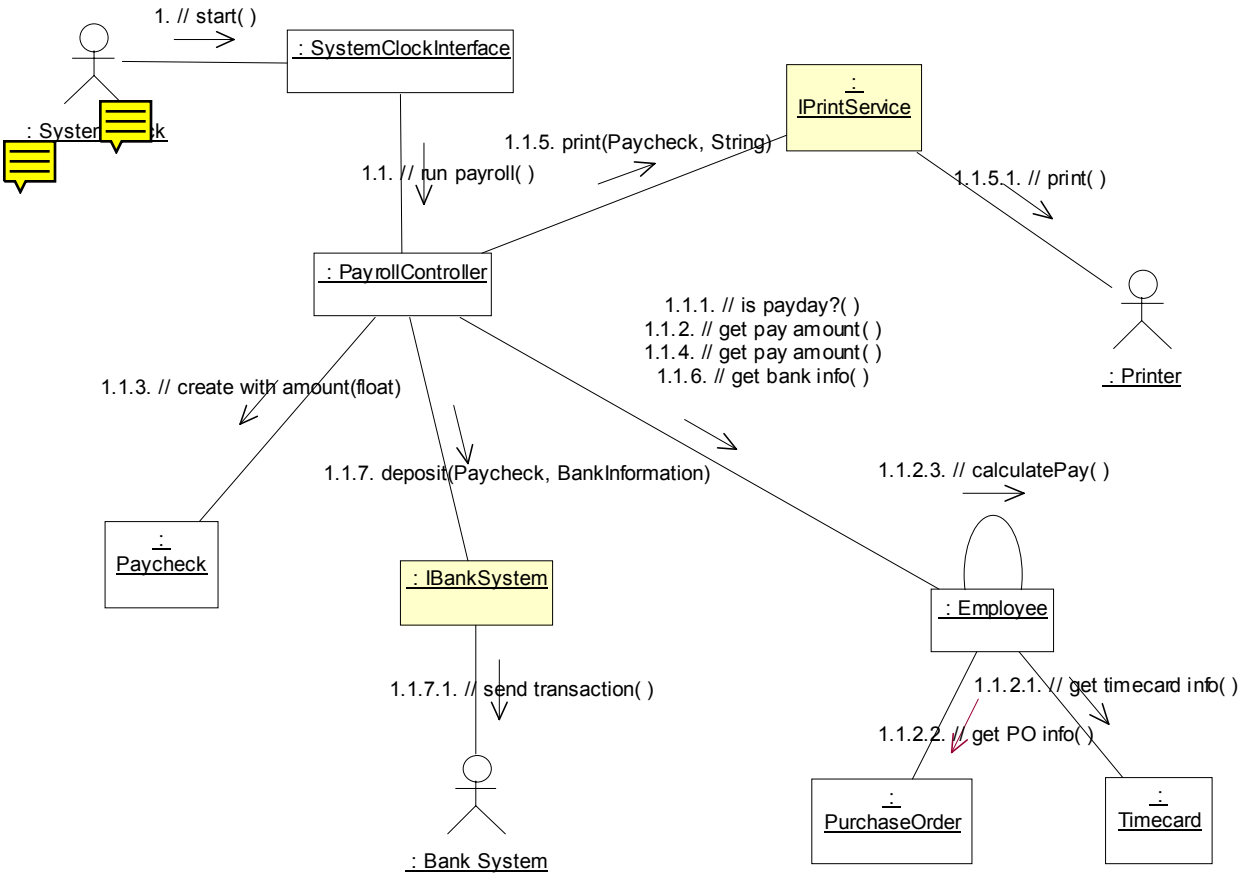
Run Payroll - VOPC (with ss interface)



1.1.2 Run Payroll (with Security)

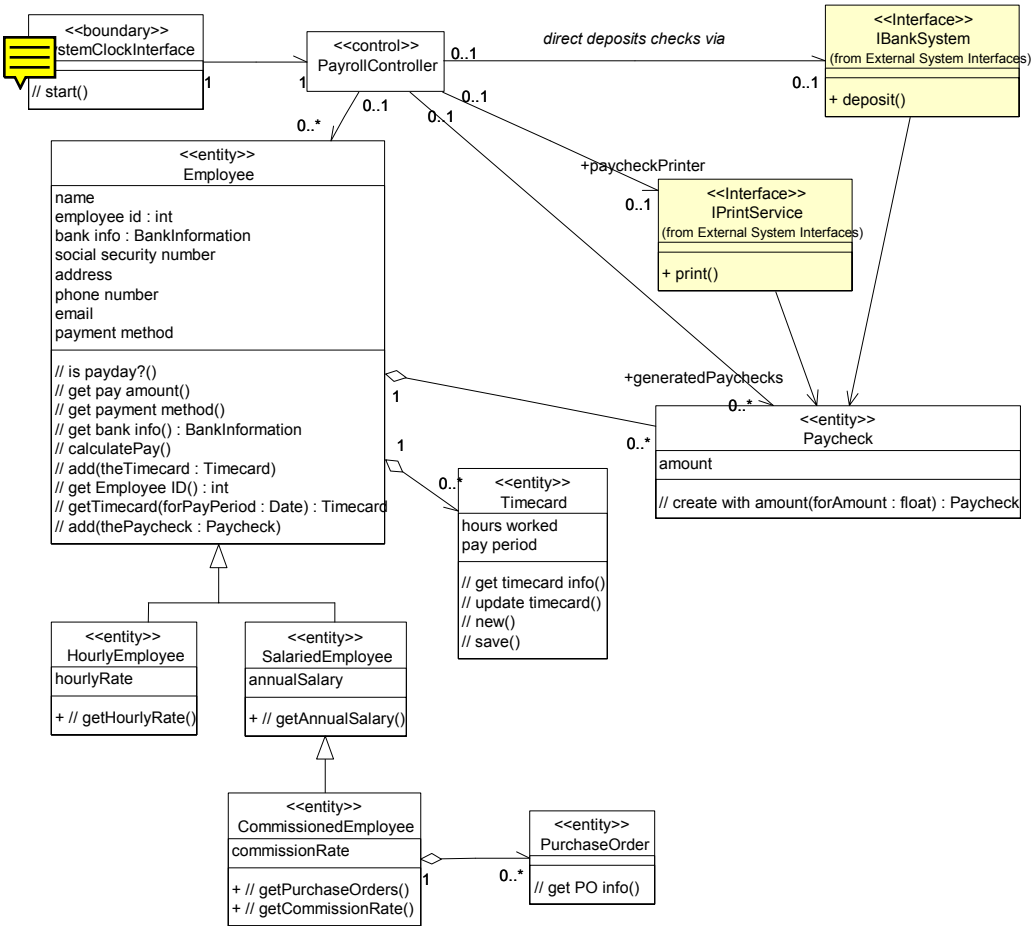
Run Payroll - Basic Flow (with security)

This is the same collaboration diagram as Run Payroll (with ss interface). There are no additional processing steps for Security for Run Payroll, as the PayrollController is meant to be "all-knowing" and "all-seeing" and thus, has open access to all secure data for Employees.



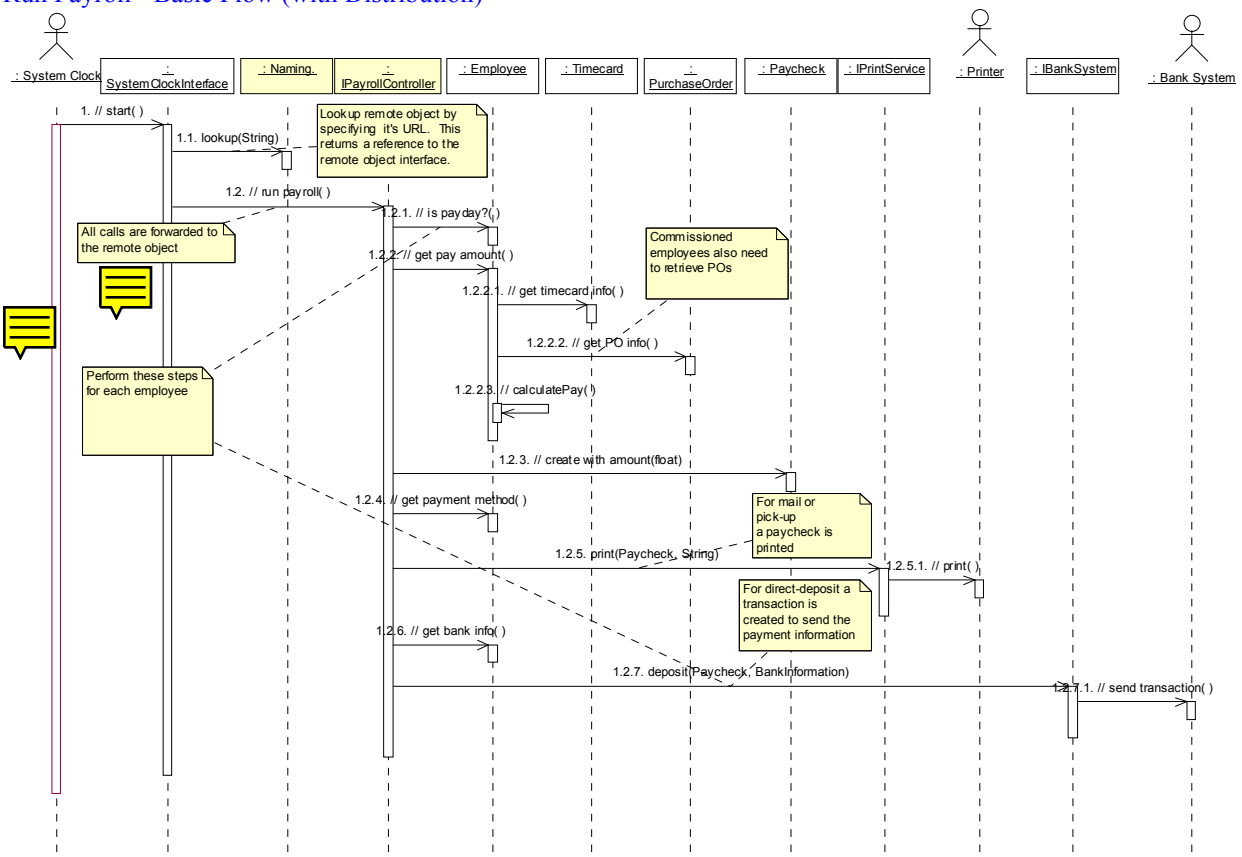
Run Payroll - VOPC (with Security)

This is the same VOPC as Run Payroll (with ss interface).
There are no additional processing steps for Security for Run Payroll, as the PayrollController is meant to be "all-knowing" and "all-seeing" and thus, has open access to all secure data for Employees.

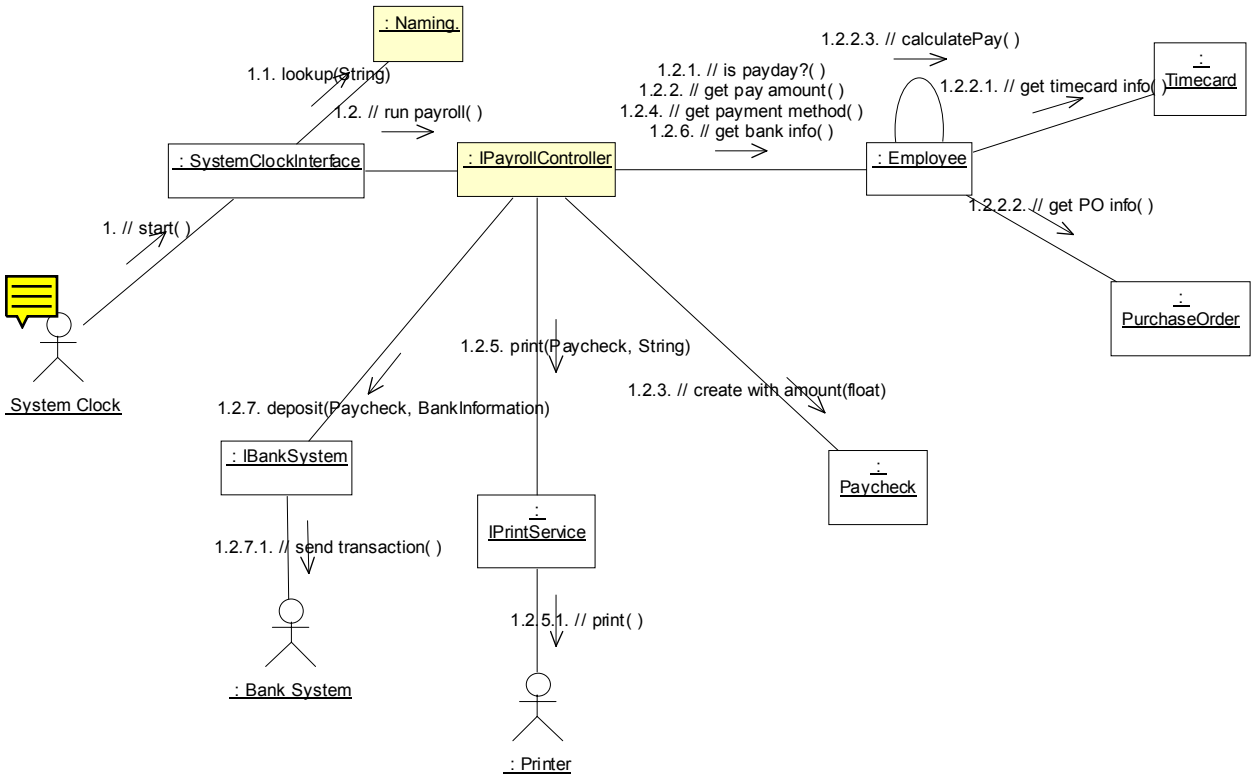


1.1.3 Run Payroll (with Distribution)

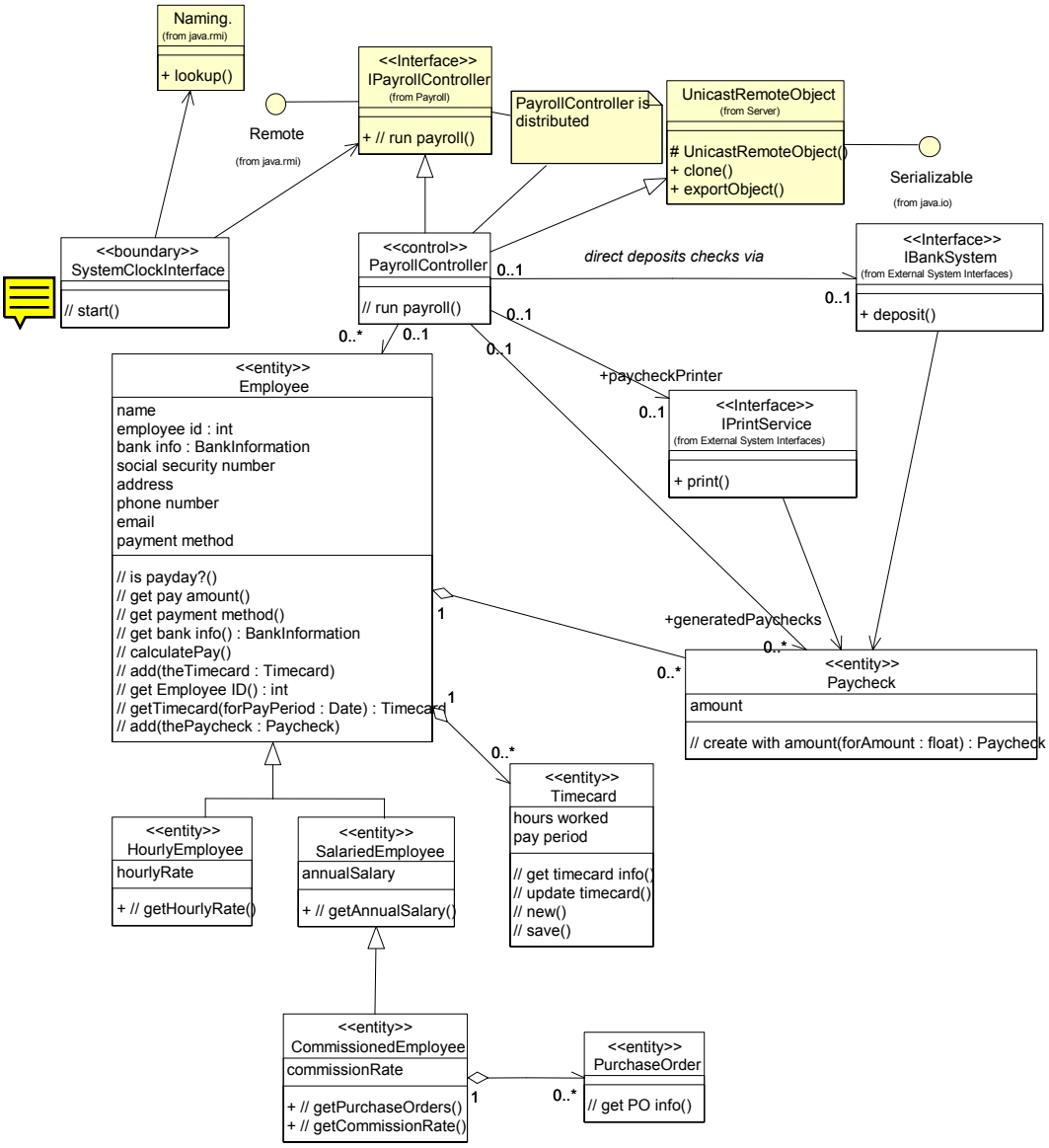
Run Payroll - Basic Flow (with Distribution)



Run Payroll - Basic Flow (with Distribution)

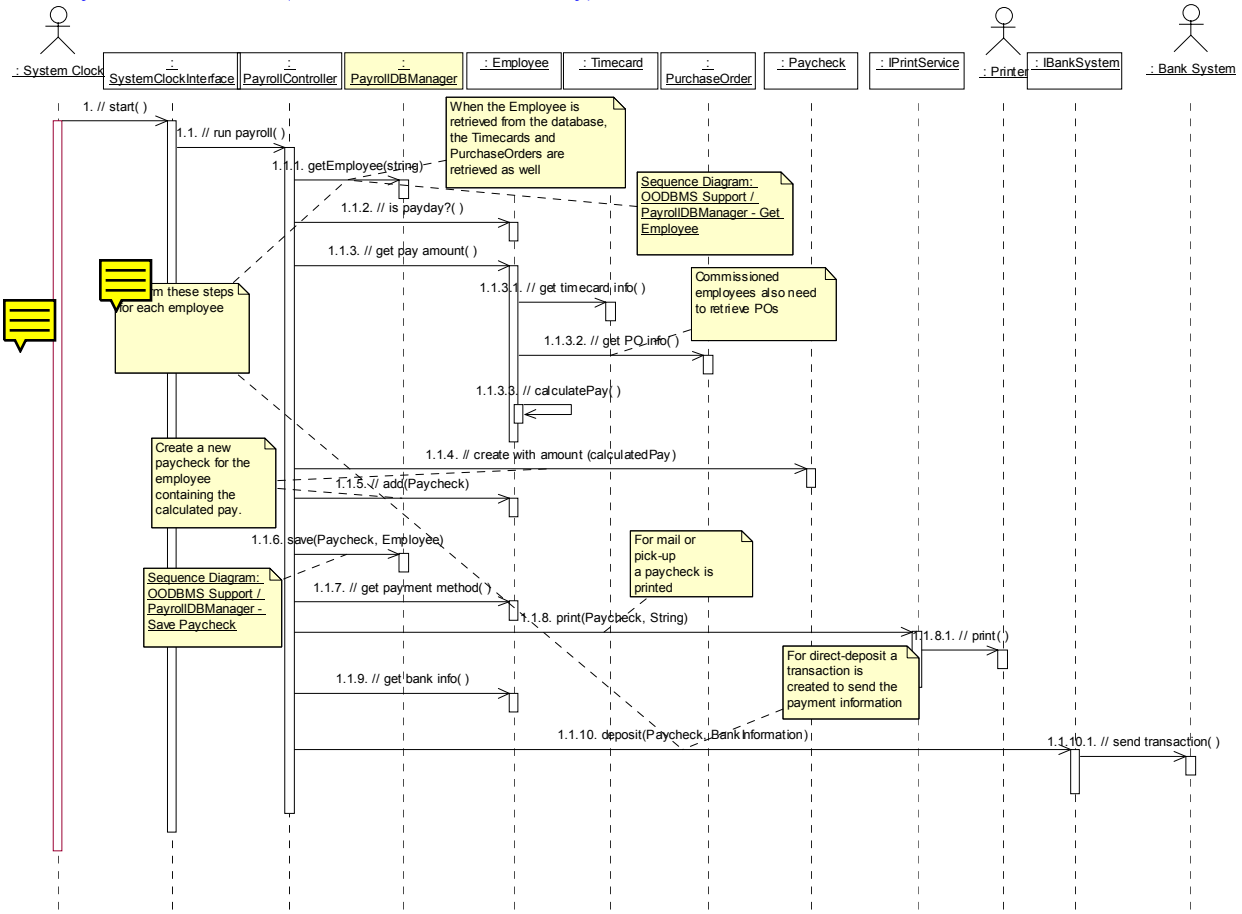


Run Payroll - VOPC (with Distribution)

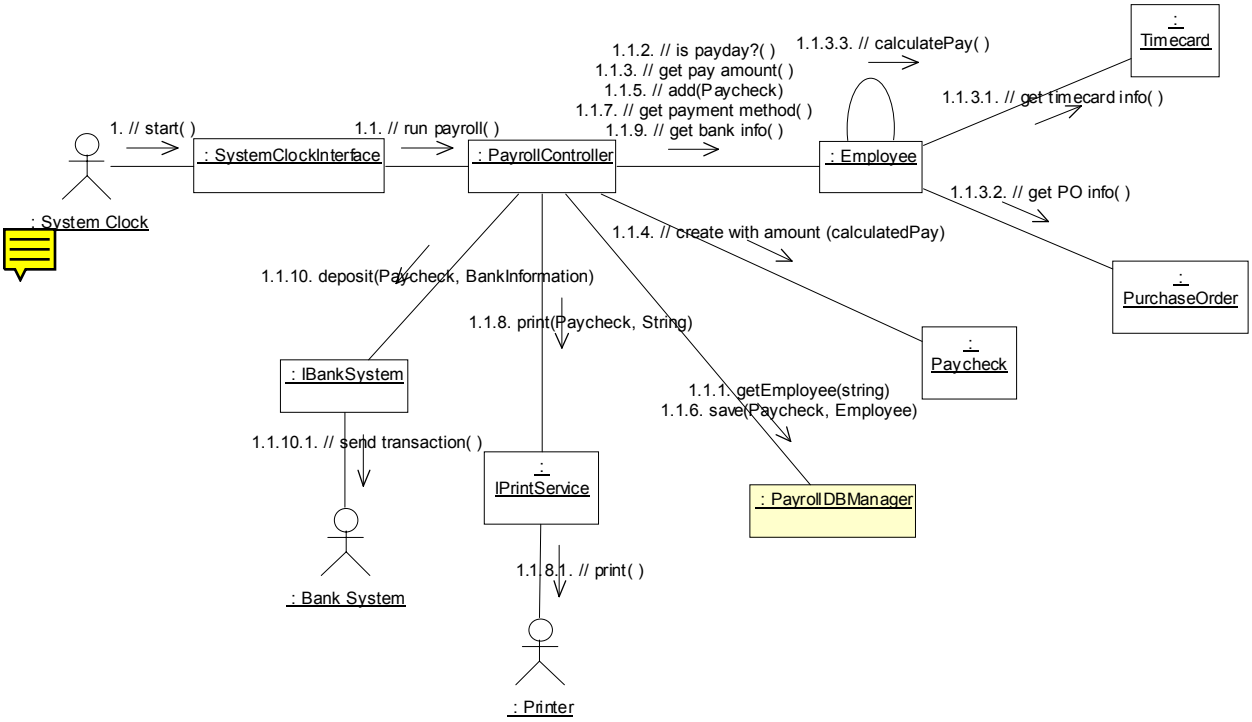


1.1.4 Run Payroll (with OODBMS Persistency)

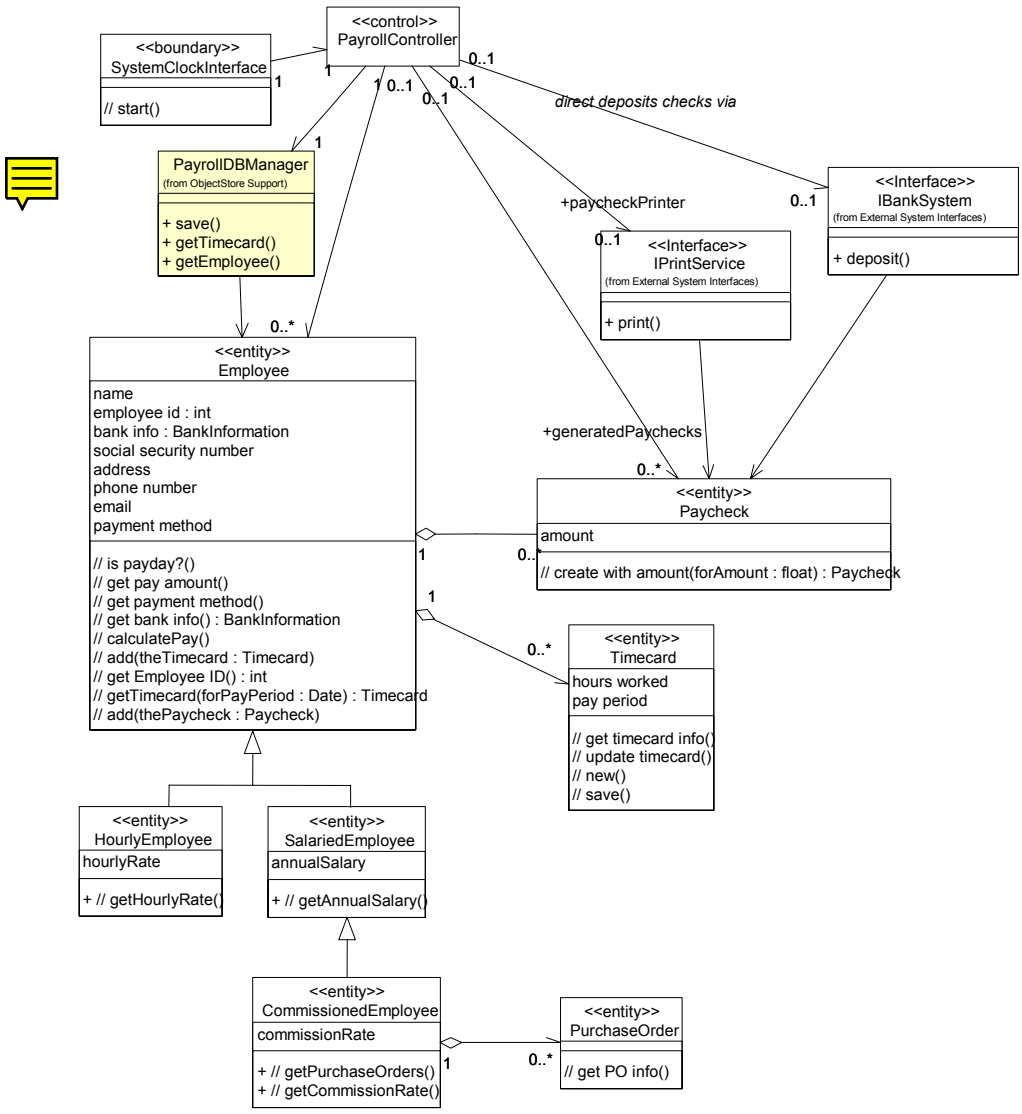
Run Payroll - Basic Flow (with OODBMS Persistency)



Run Payroll - Basic Flow (with OODBMS Persistency)

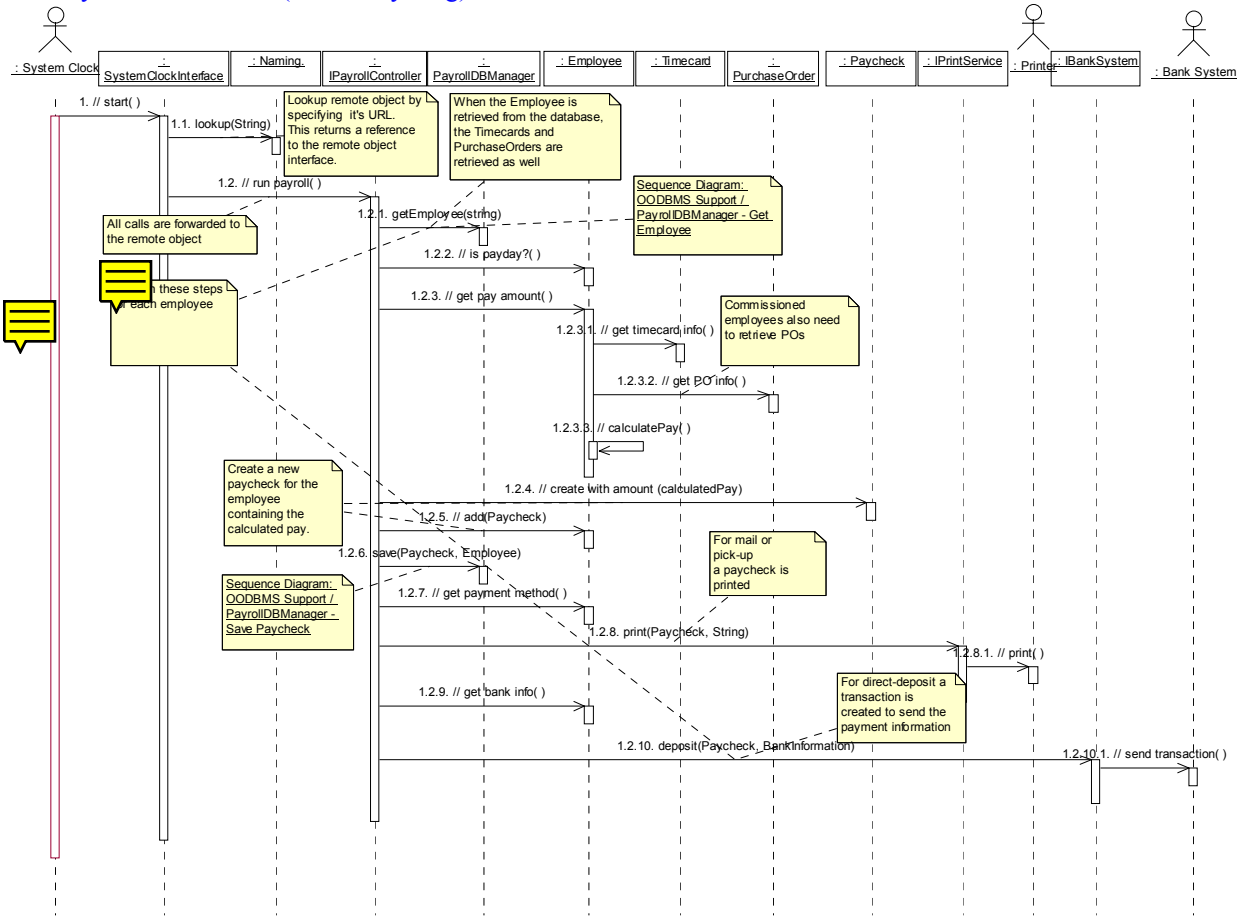


Run Payroll - VOPC (with OODBMS Persistency)

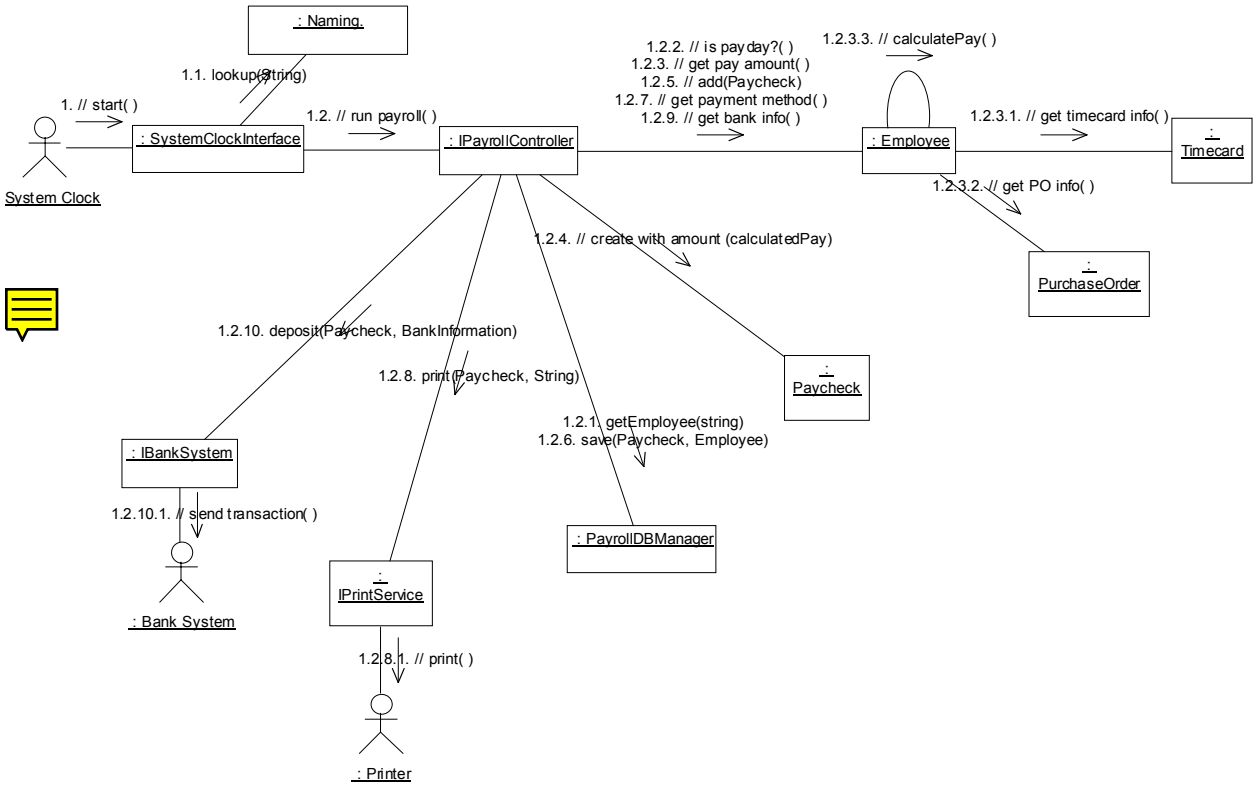


1.1.5 Run Payroll (with everything)

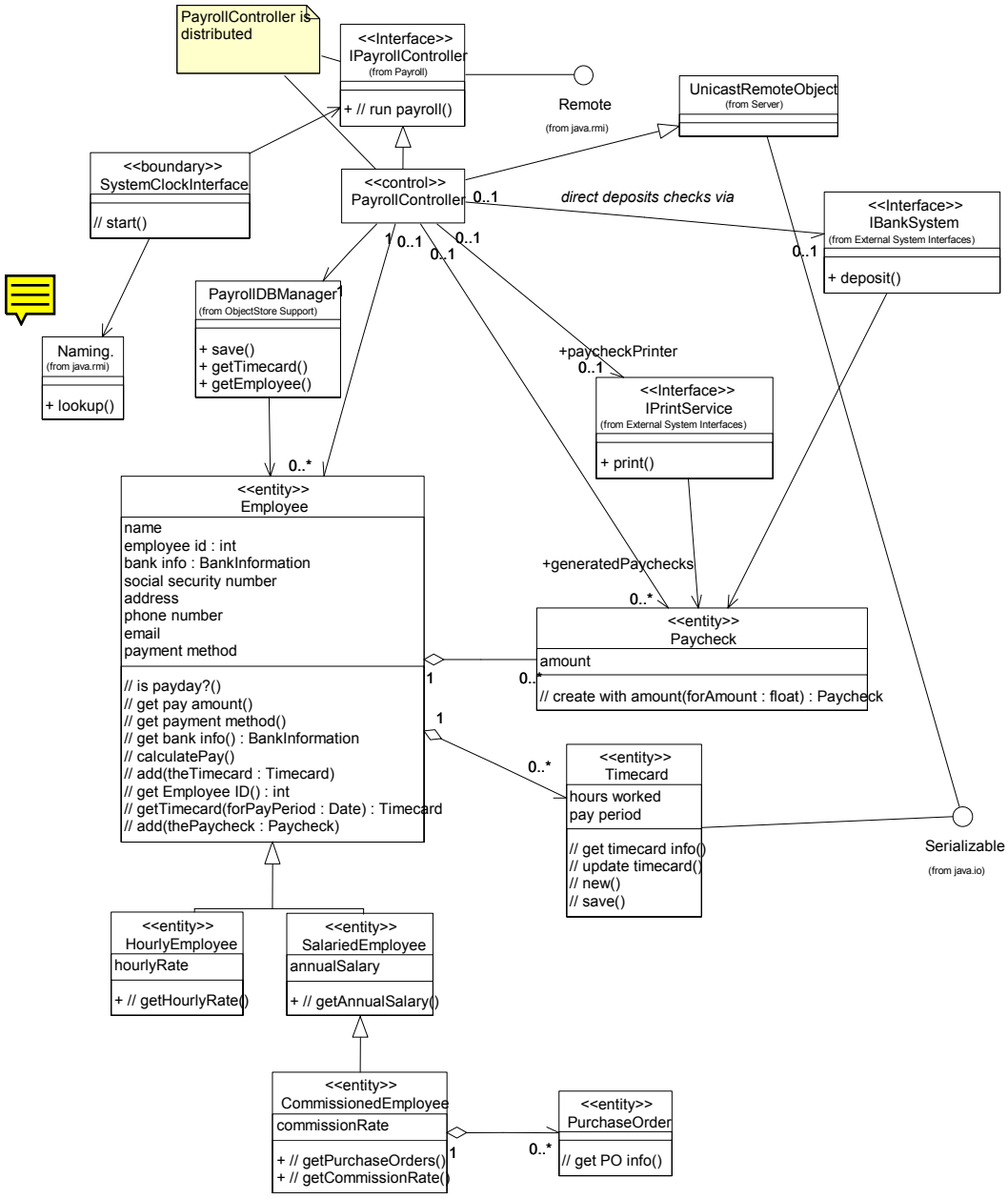
Run Payroll - Basic Flow (with everything)



Run Payroll - Basic Flow (with everything)



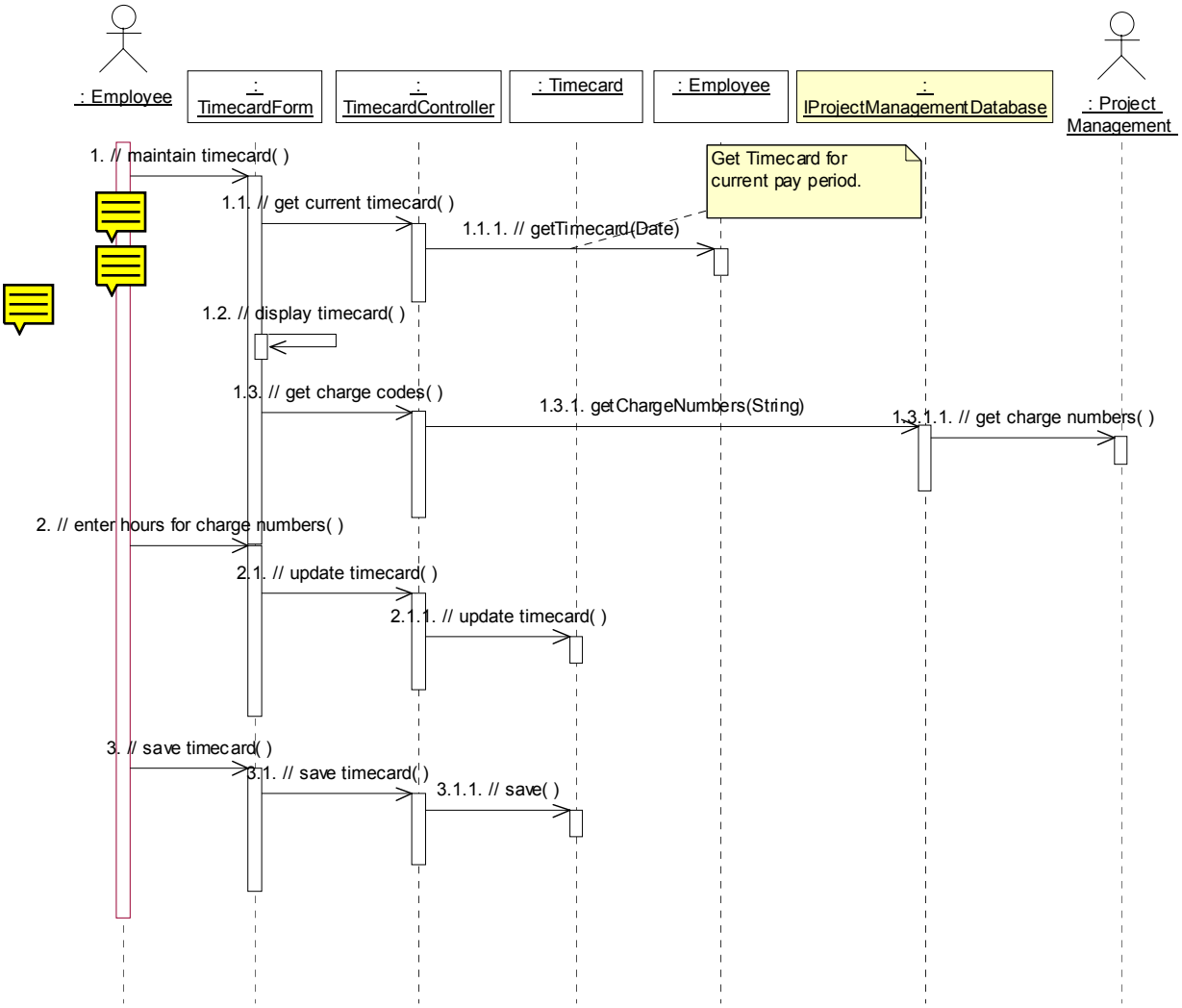
Run Payroll - VOPC (with everything)



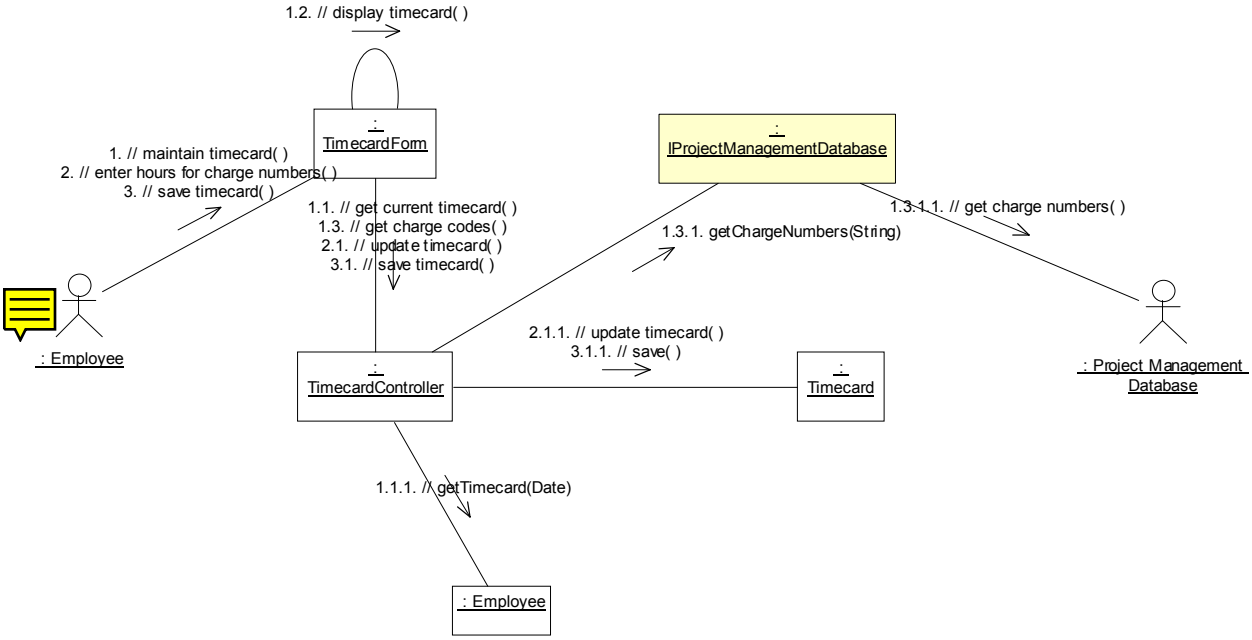
1.2 Use-Case Realization - Maintain Timecard

1.2.1 Maintain Timecard (with ss interface)

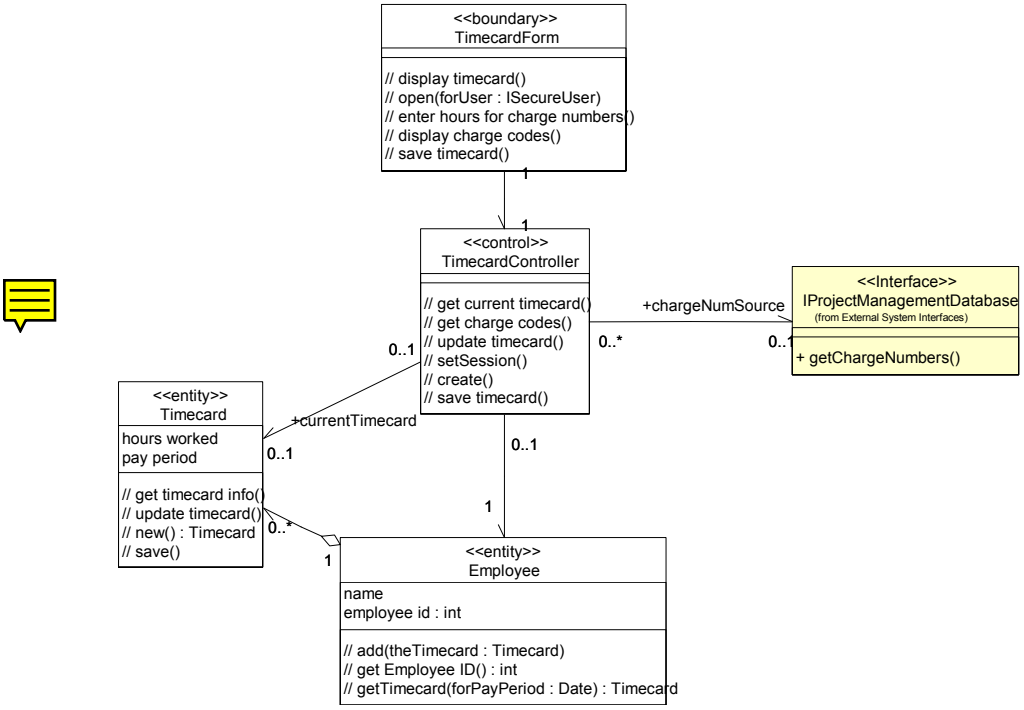
Maintain Timecard - Basic Flow (with ss interface)



Maintain Timecard - Basic Flow (with ss interface)

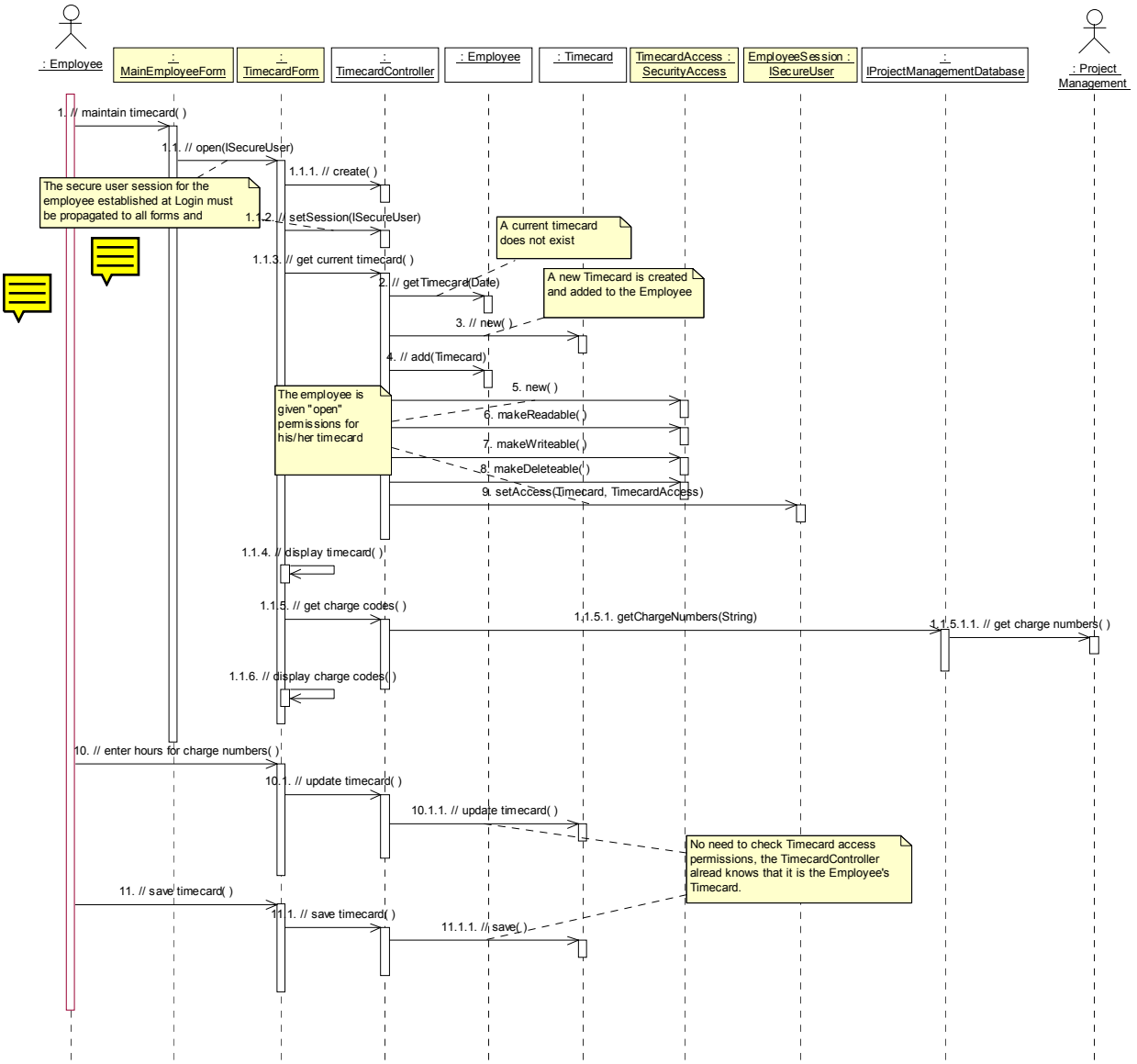


Maintain Timecard - VOPC (with ss interface)

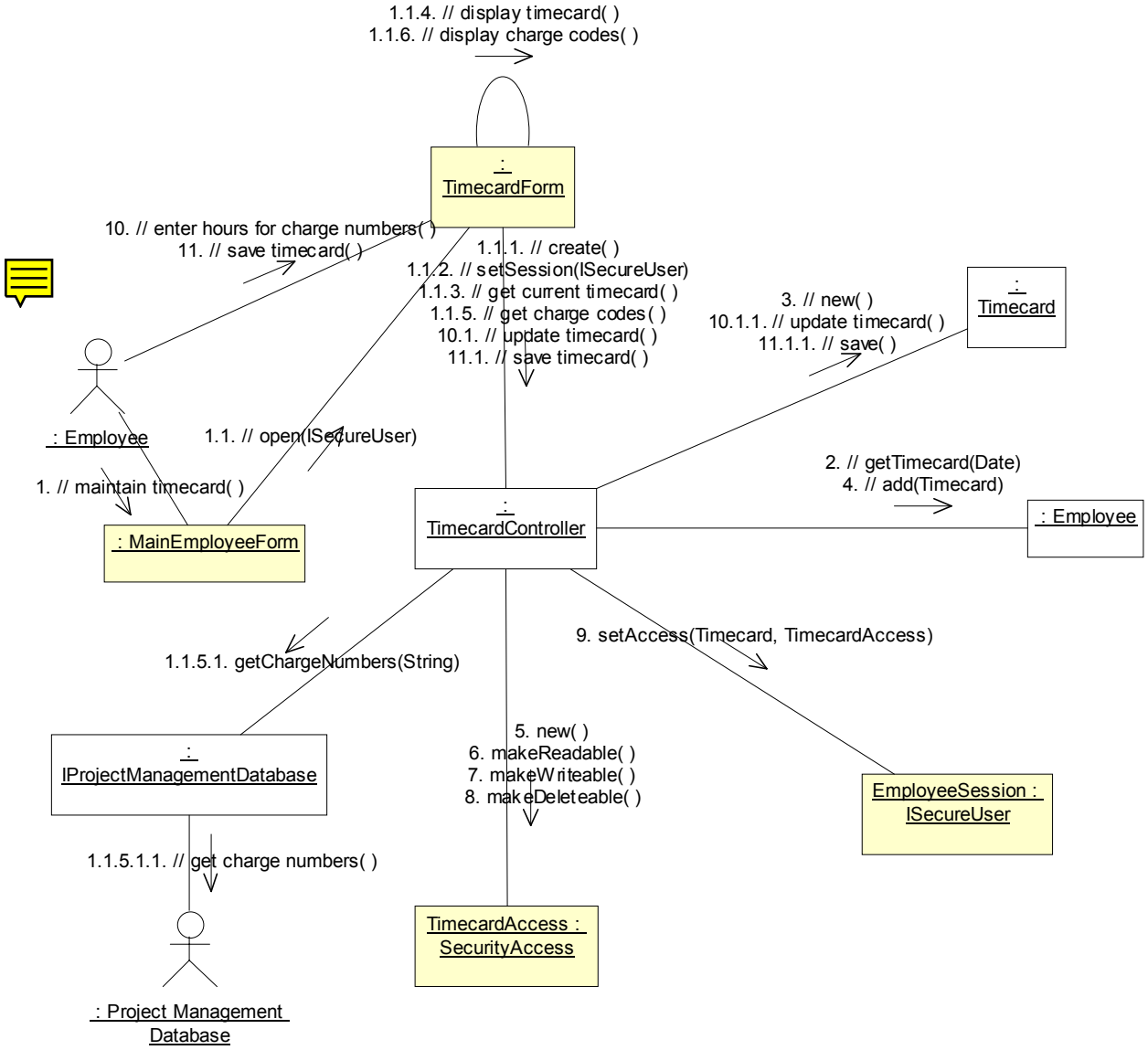


1.2.2 *Maintain Timecard (with Security)*

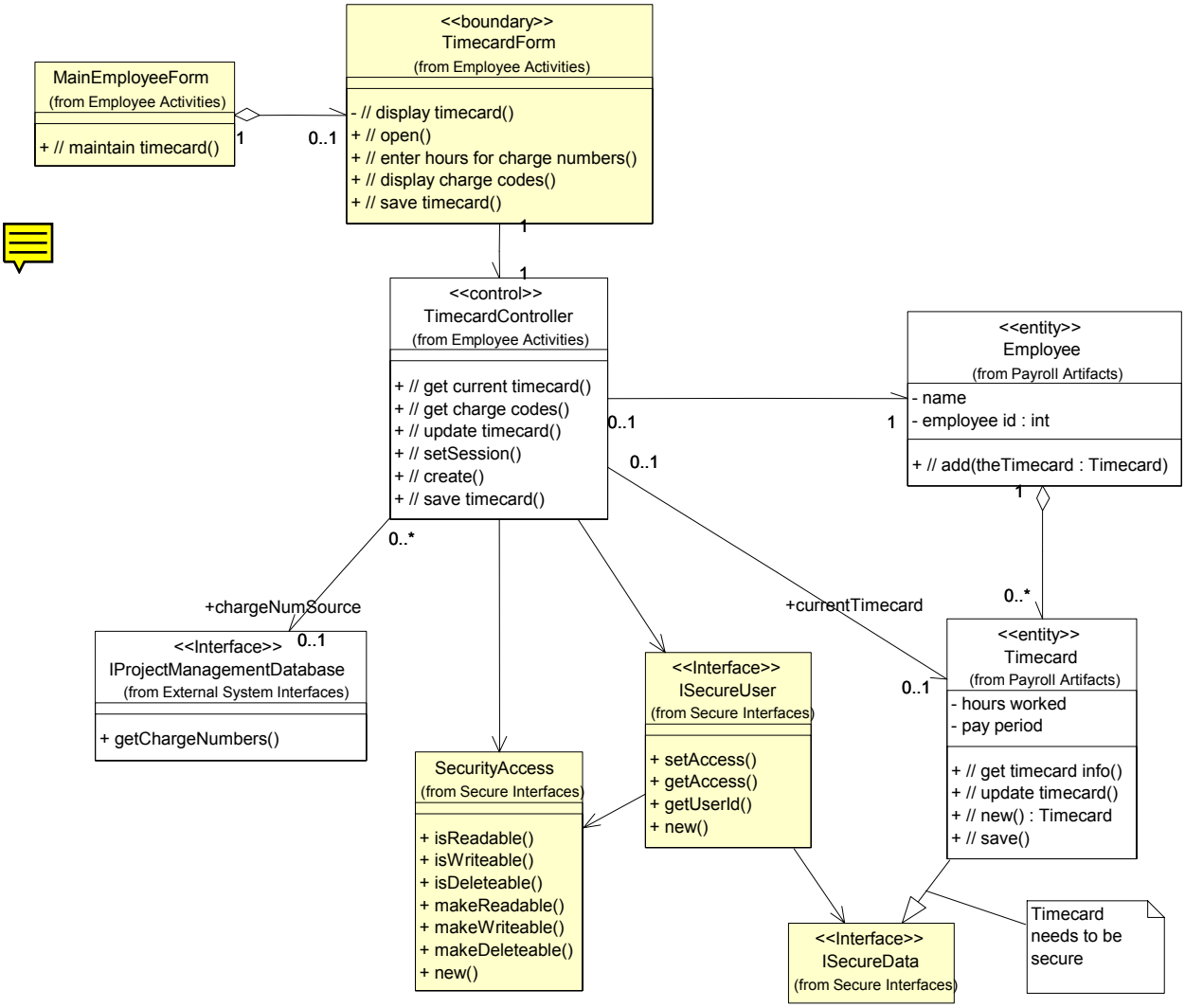
Maintain Timecard - Basic Flow (New Timecard with Security)



Maintain Timecard - Basic Flow (New Timecard with Security)

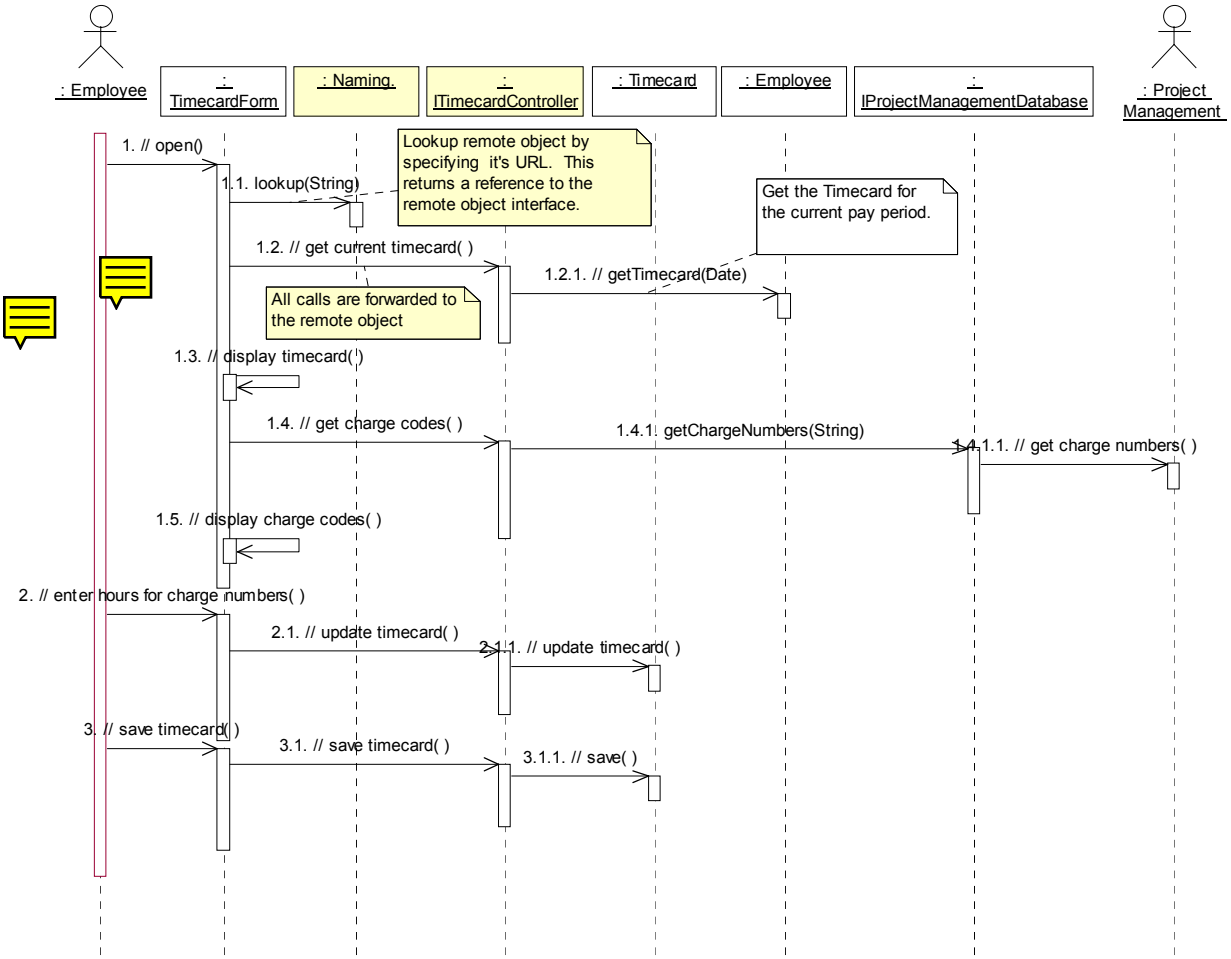


Maintain Timecard - VOPC (with Security)

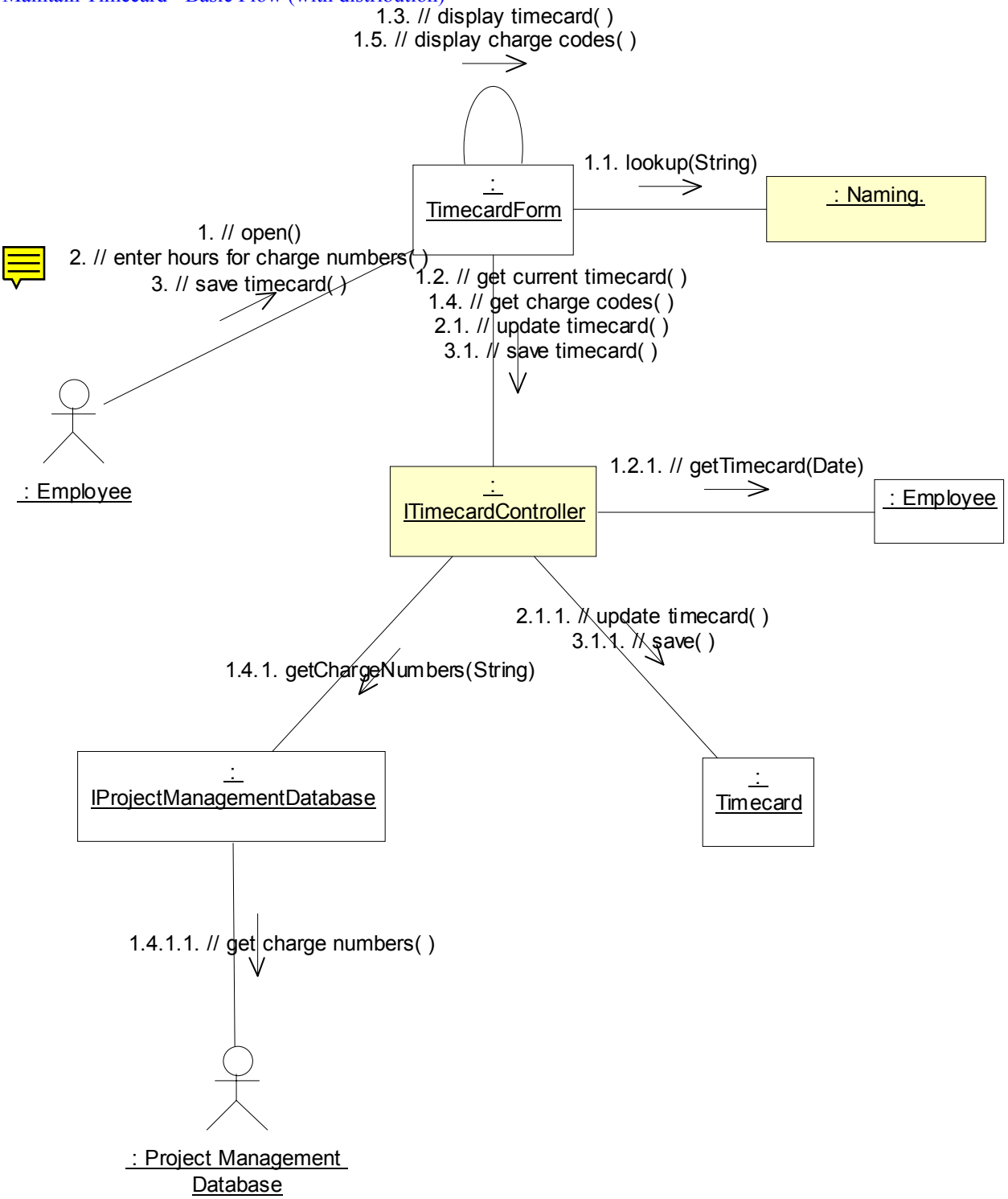


1.2.3 Maintain Timecard (with Distribution)

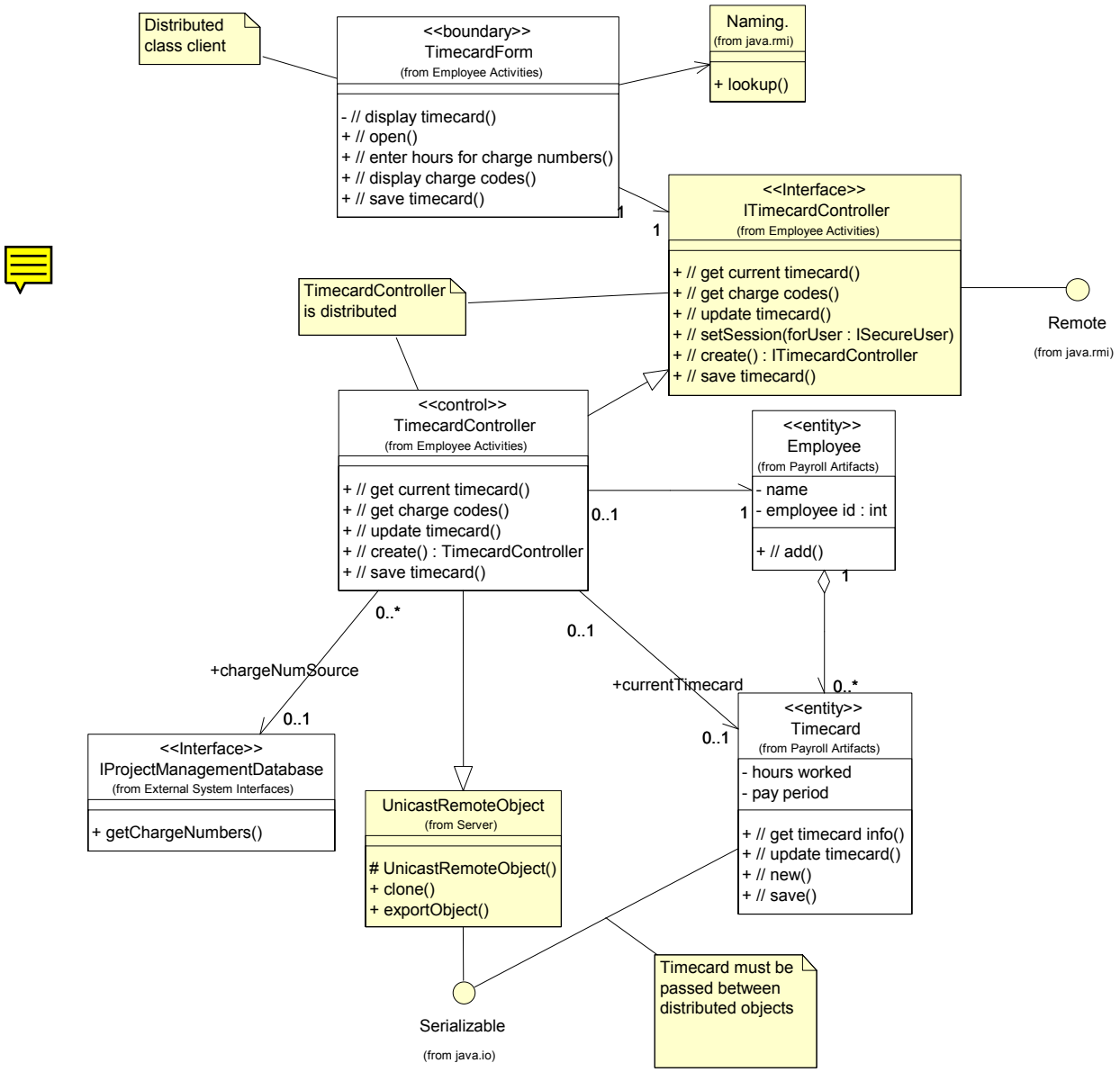
Maintain Timecard - Basic Flow (with distribution)



Maintain Timecard - Basic Flow (with distribution)

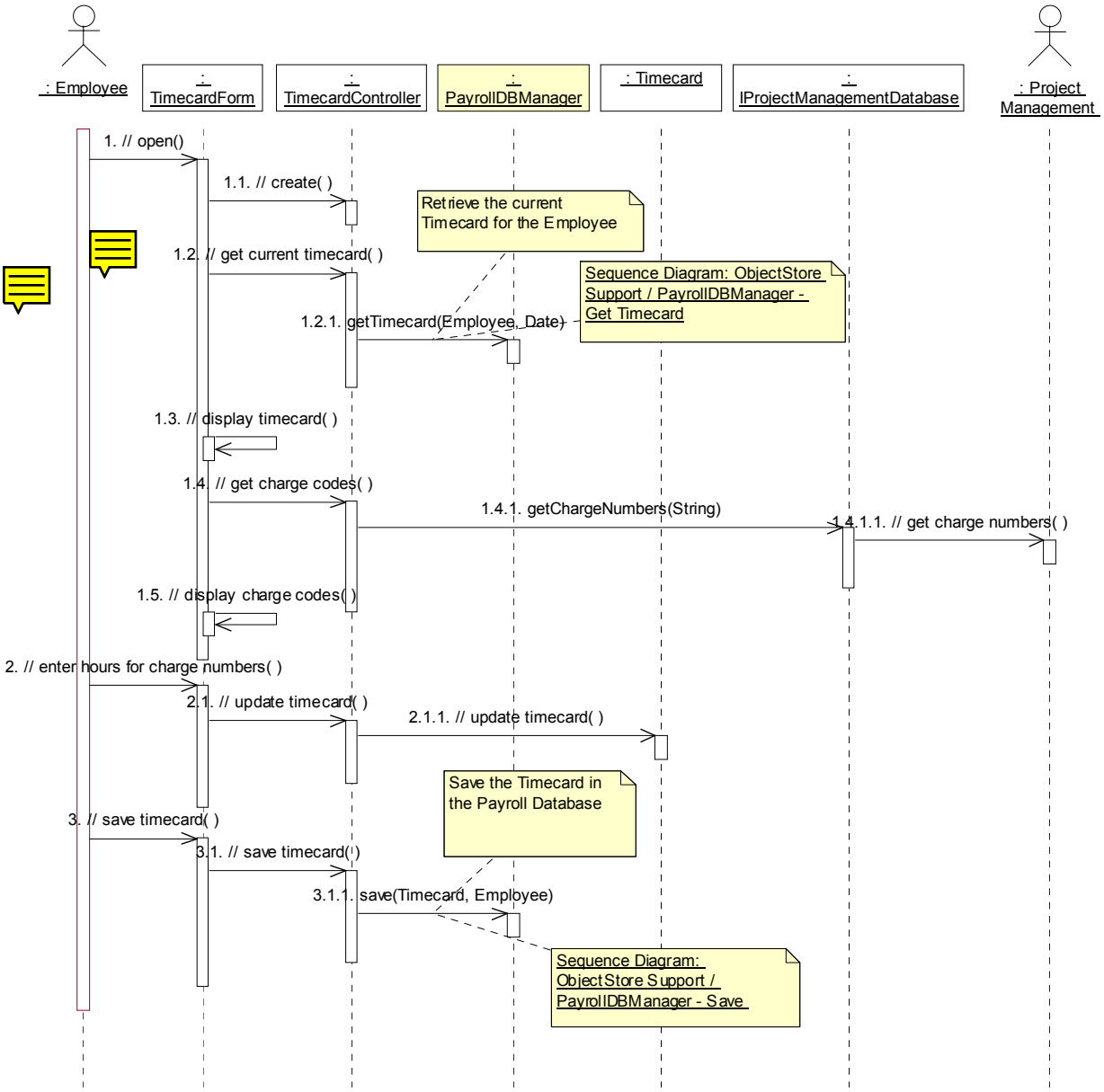


Maintain Timecard - VOPC (with Distribution)

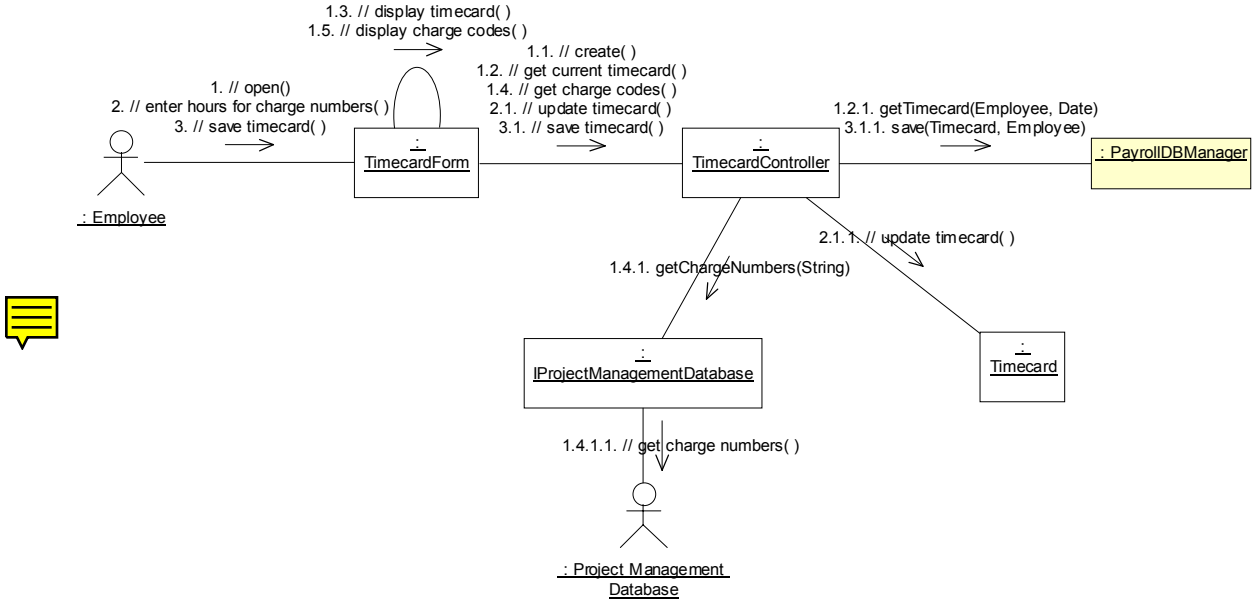


1.2.4 Maintain Timecard (with OODBMS Persistence)

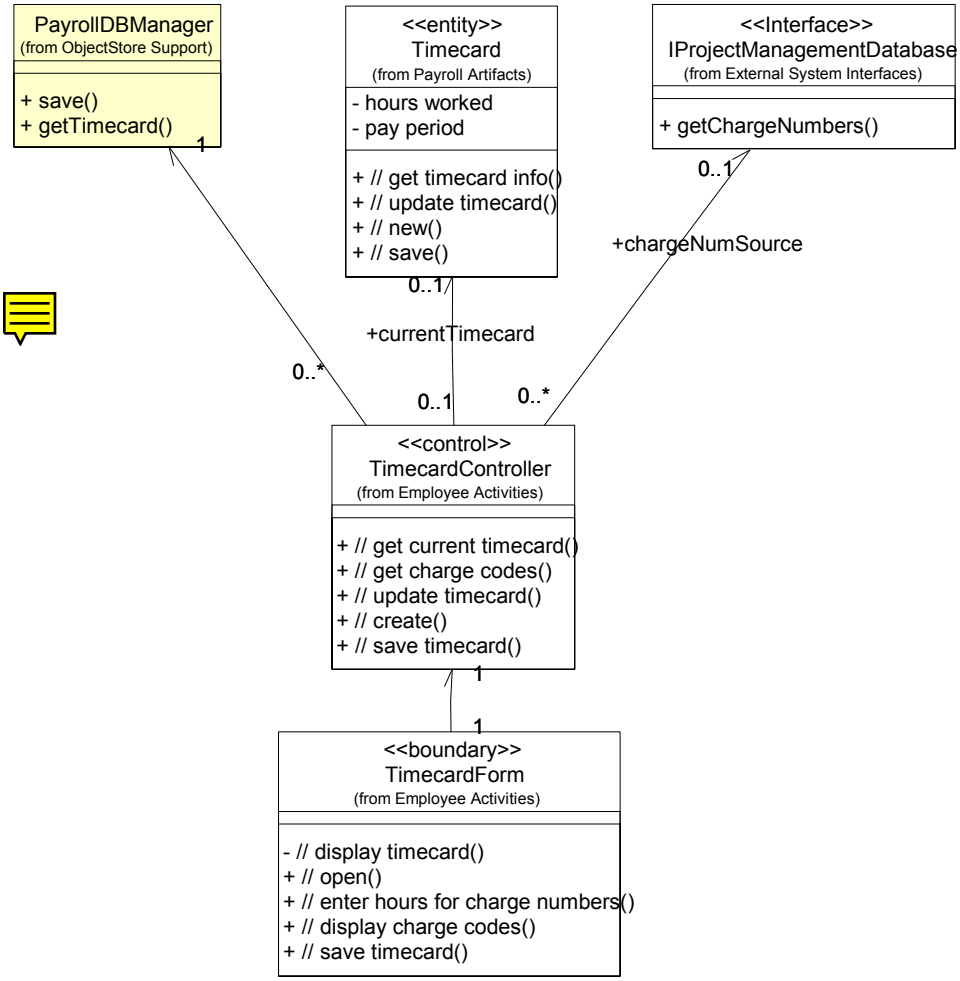
Maintain Timecard - Basic Flow (with OODBMS Persistency)



Maintain Timecard - Basic Flow (with OODBMS Persistency)

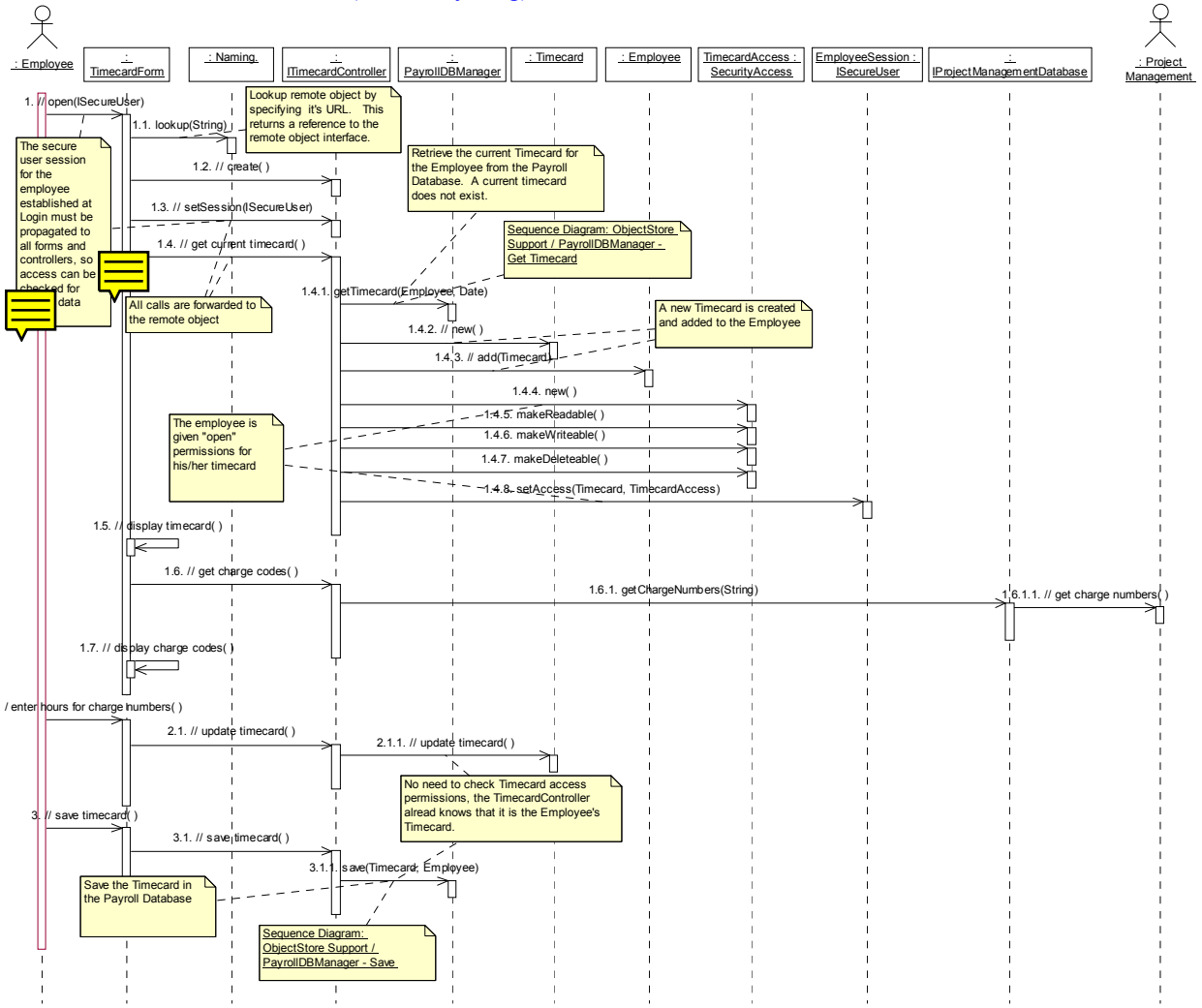


Maintain Timecard - VOPC (with OODBMS Persistency)

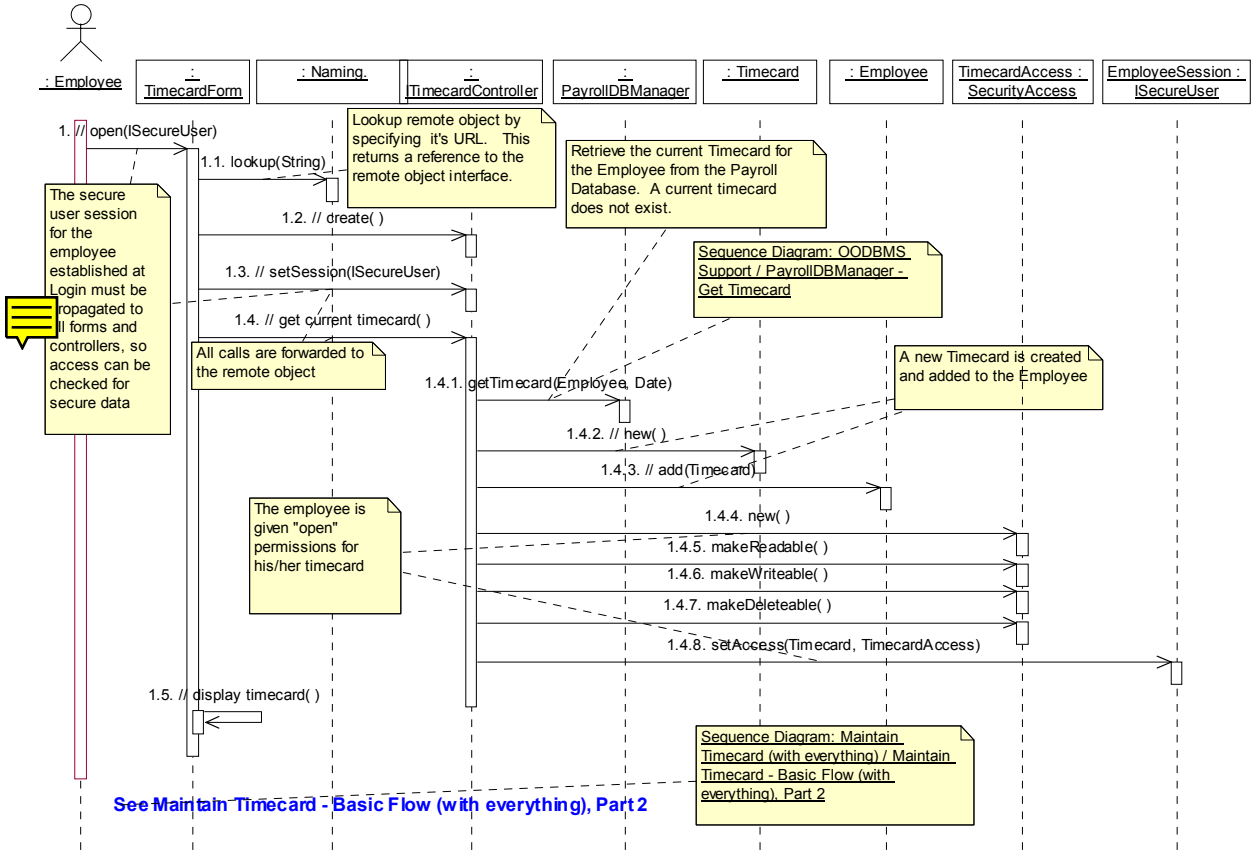


1.2.5 Maintain Timecard (with everything)

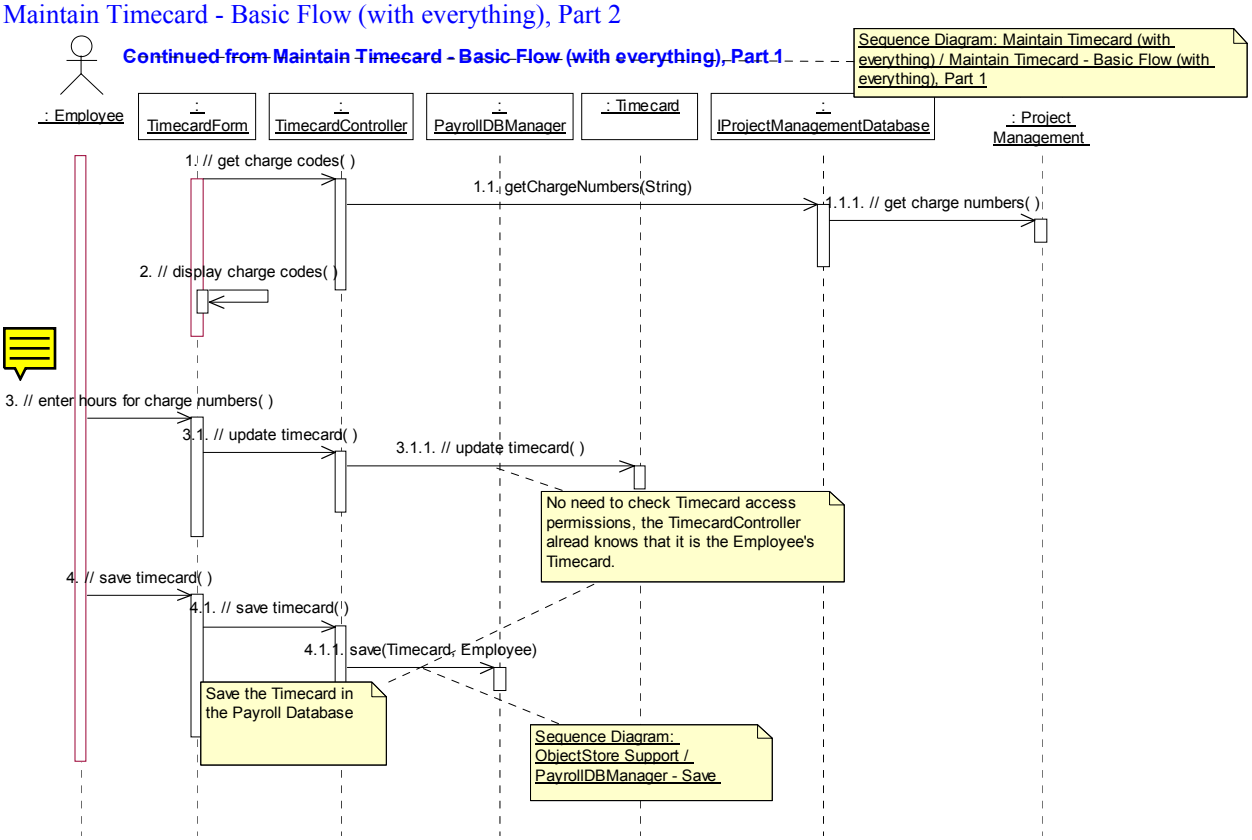
Maintain Timecard - Basic Flow (with everything)



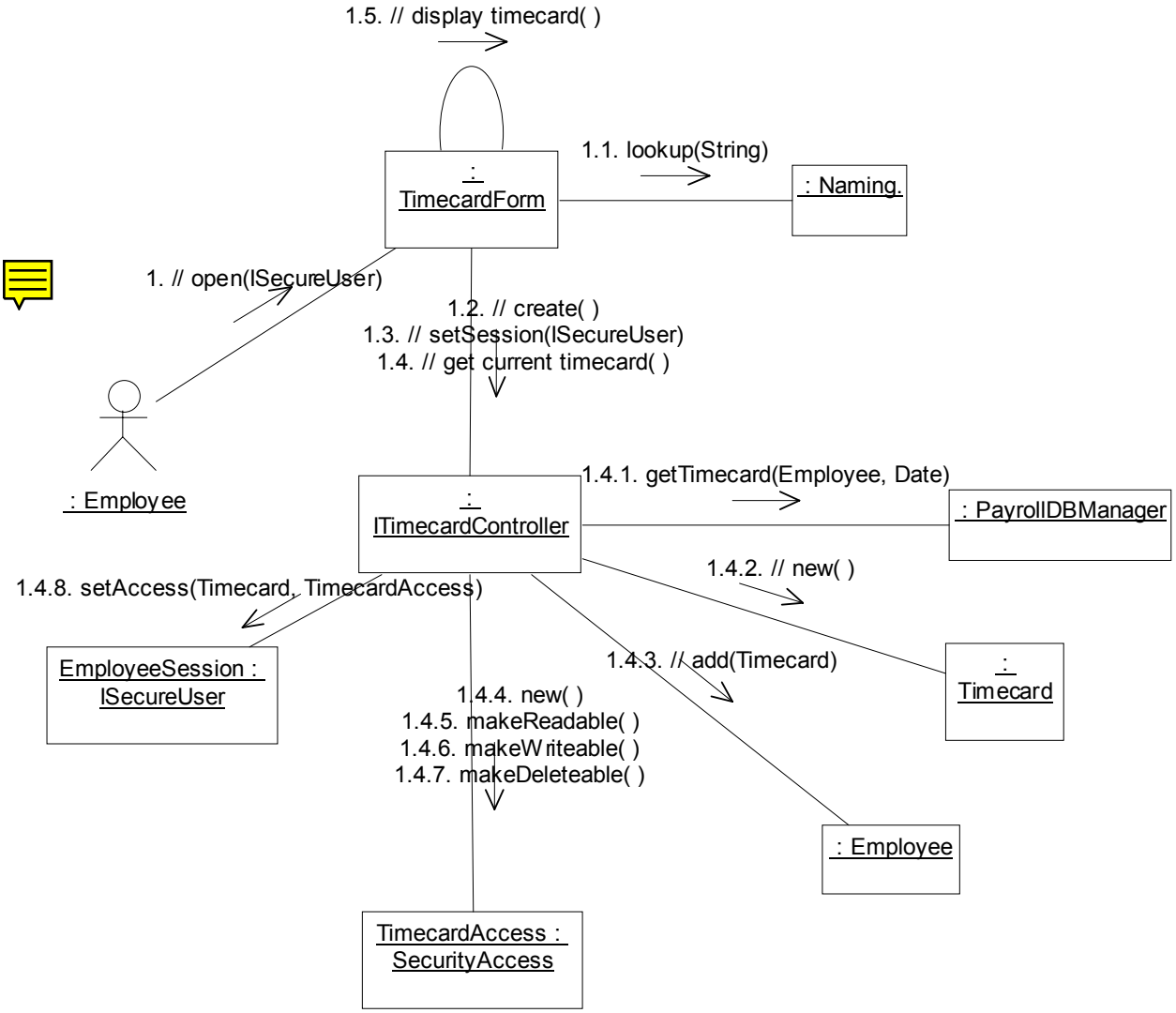
Maintain Timecard - Basic Flow (with everything), Part 1



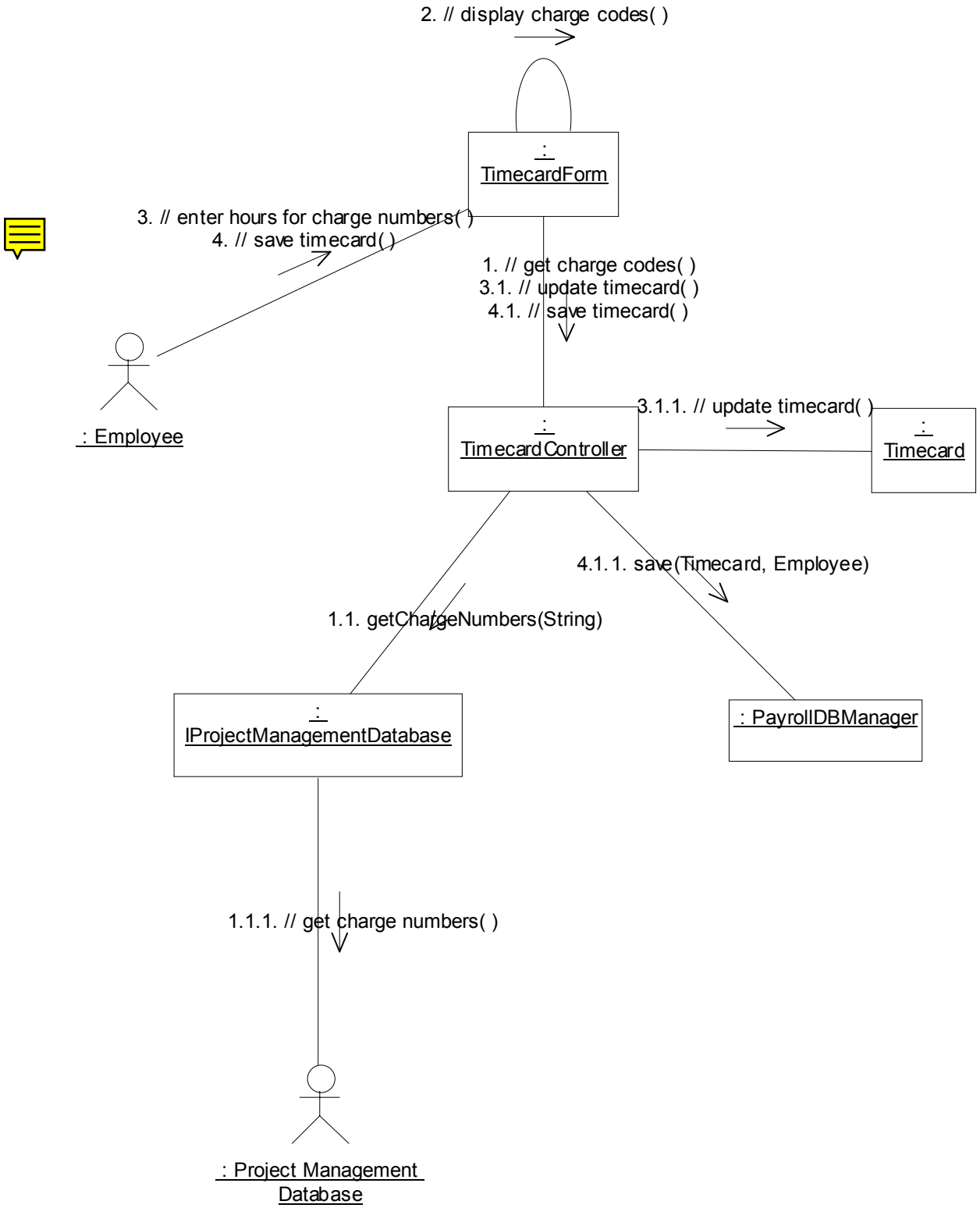
Maintain Timecard - Basic Flow (with everything), Part 2



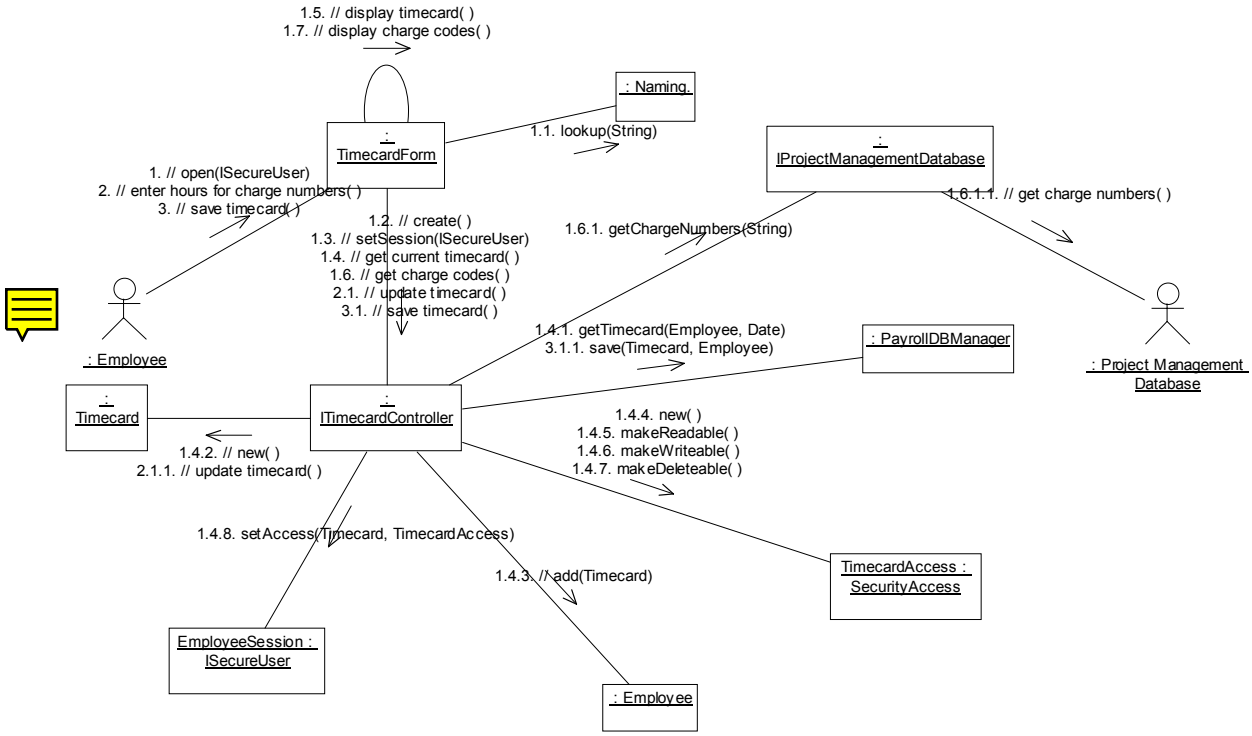
Maintain Timecard - Basic Flow (with everything), Part 1



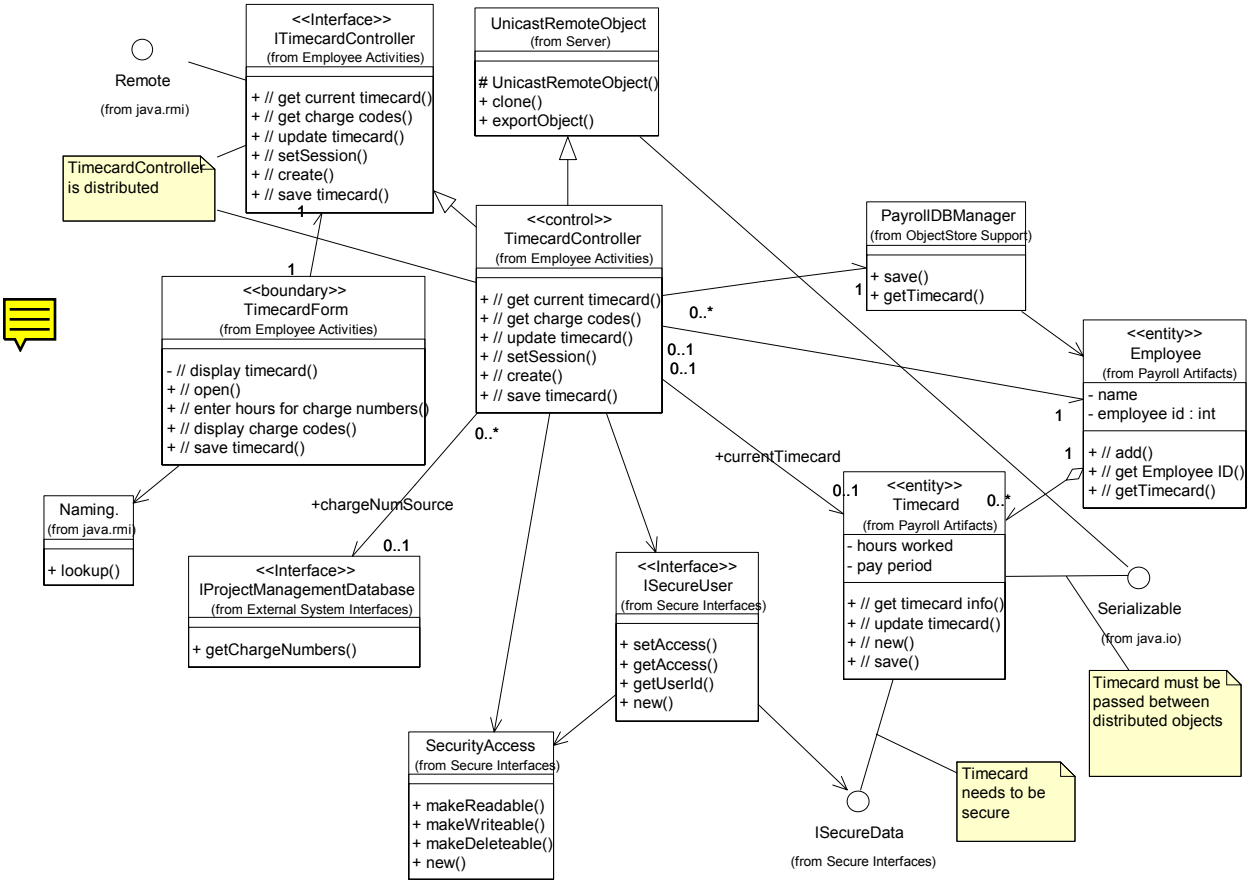
Maintain Timecard - Basic Flow (with everything), Part 2



Maintain Timecard - Basic Flow (with everything)



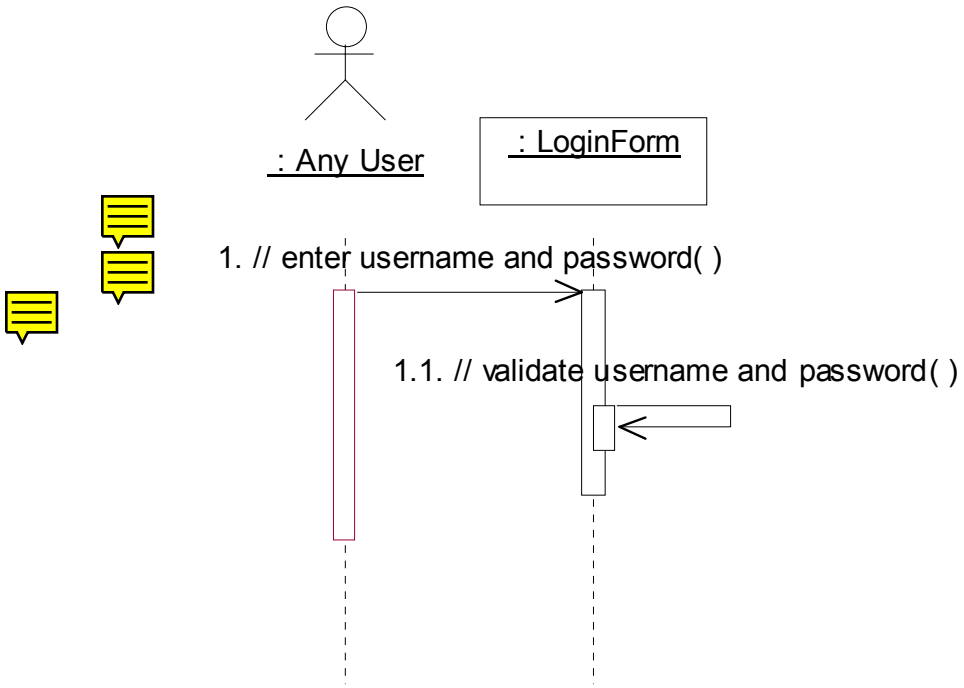
Maintain Timecard - VOPC (with everything)



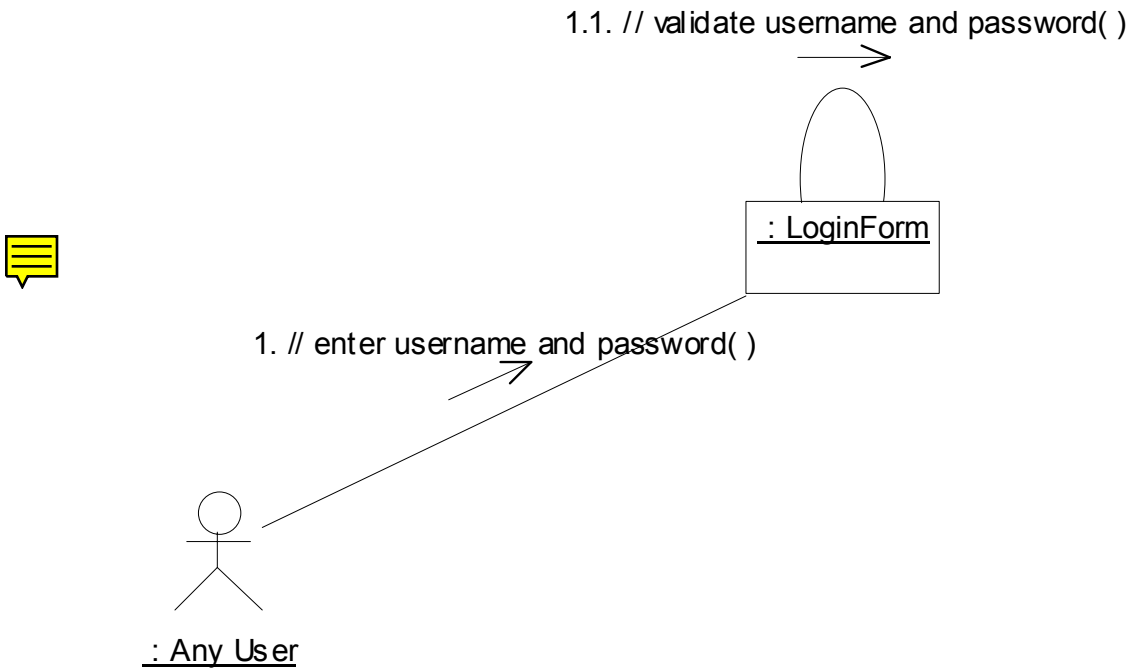
1.3 Use-Case Realization - Login

1.3.1 Login

Login - Basic Flow



Login - Basic Flow

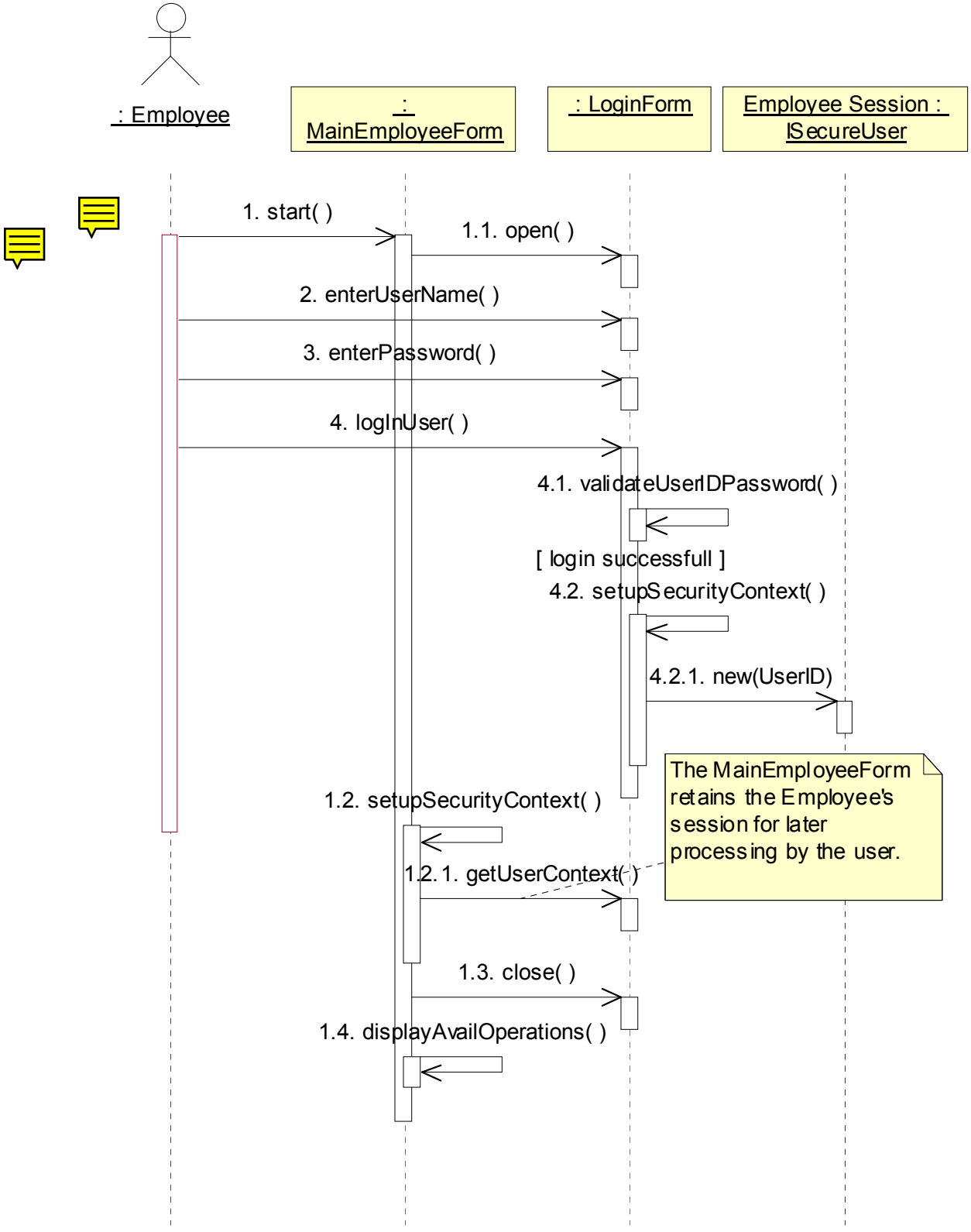


Login - VOPC

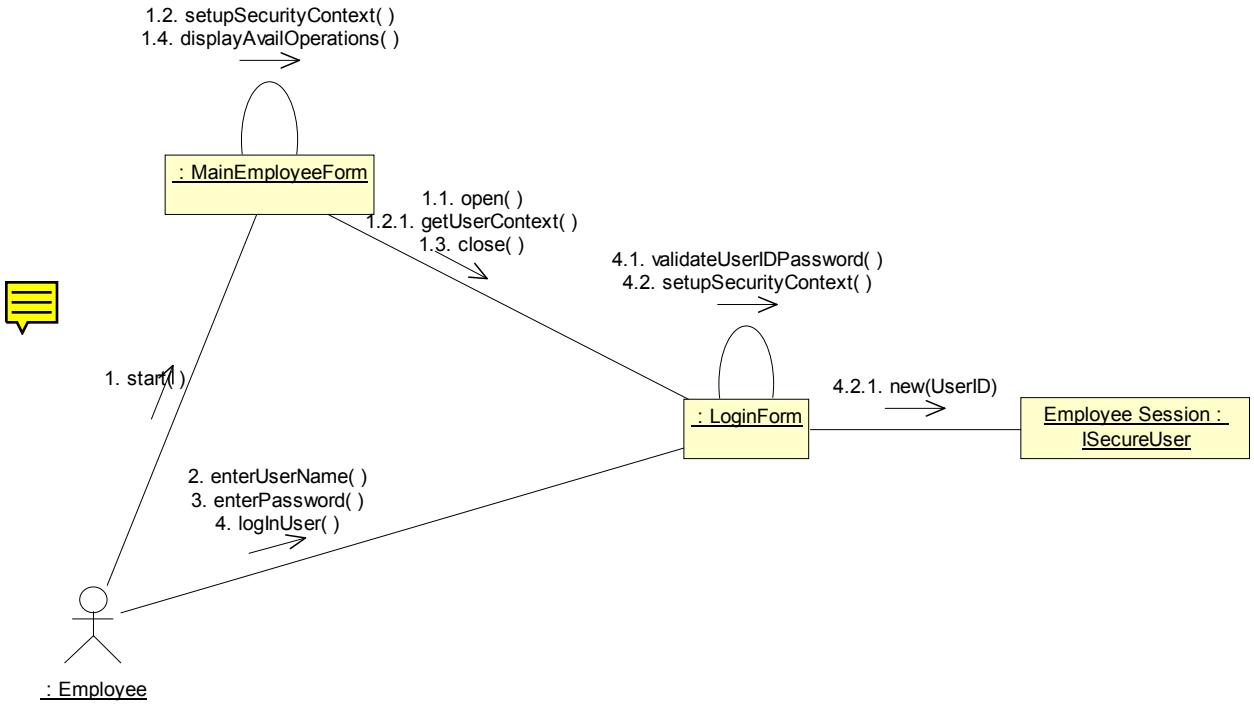

LoginForm
(from GUI Framework)
+ open()
+ enterUserName()
+ validateUserIDPassword(
+ enterPassword()
+ logInUser()
+ setupSecurityContext()
+ getUserContext()

1.3.2 Login (with Security)

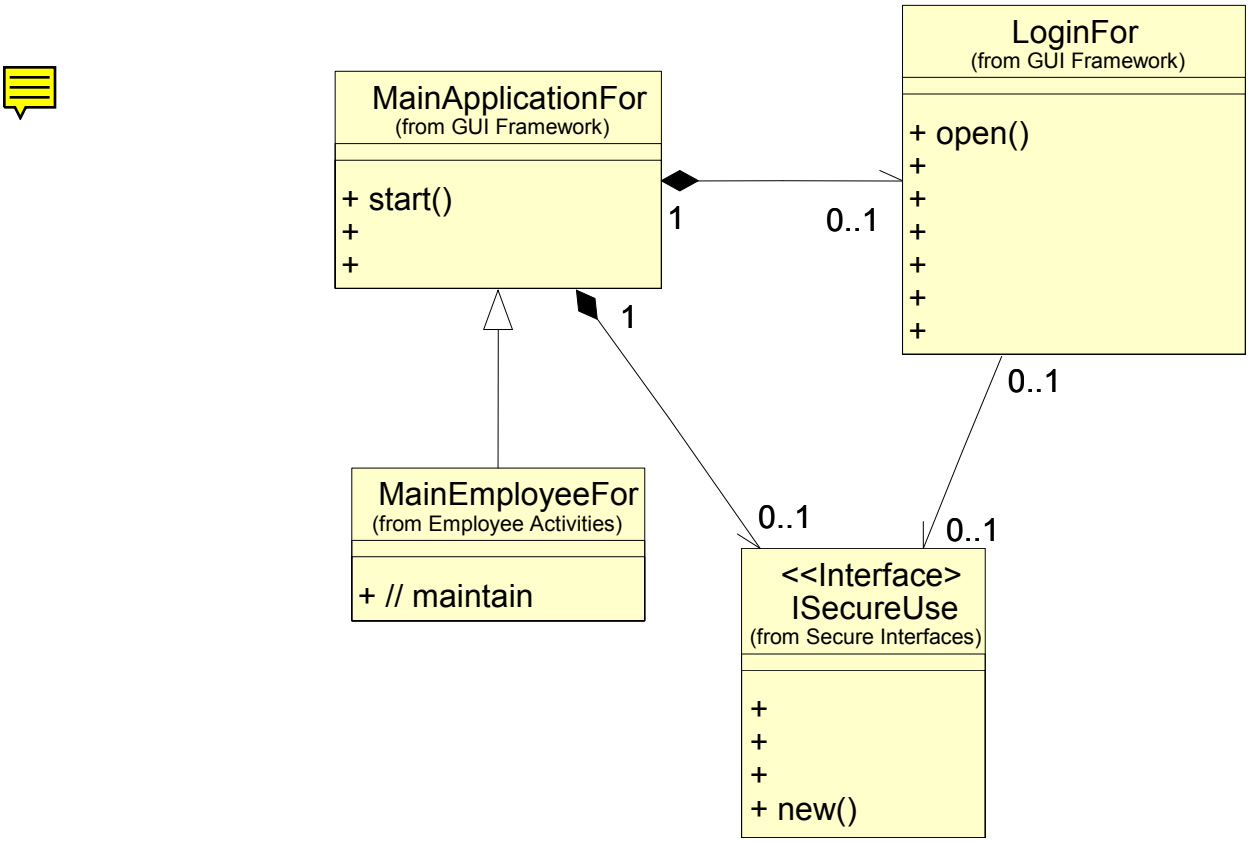
Login - Basic Flow (Employee Login with Security)



Login - Basic Flow (Employee Login with Security)



Login - VOPC (with Security)

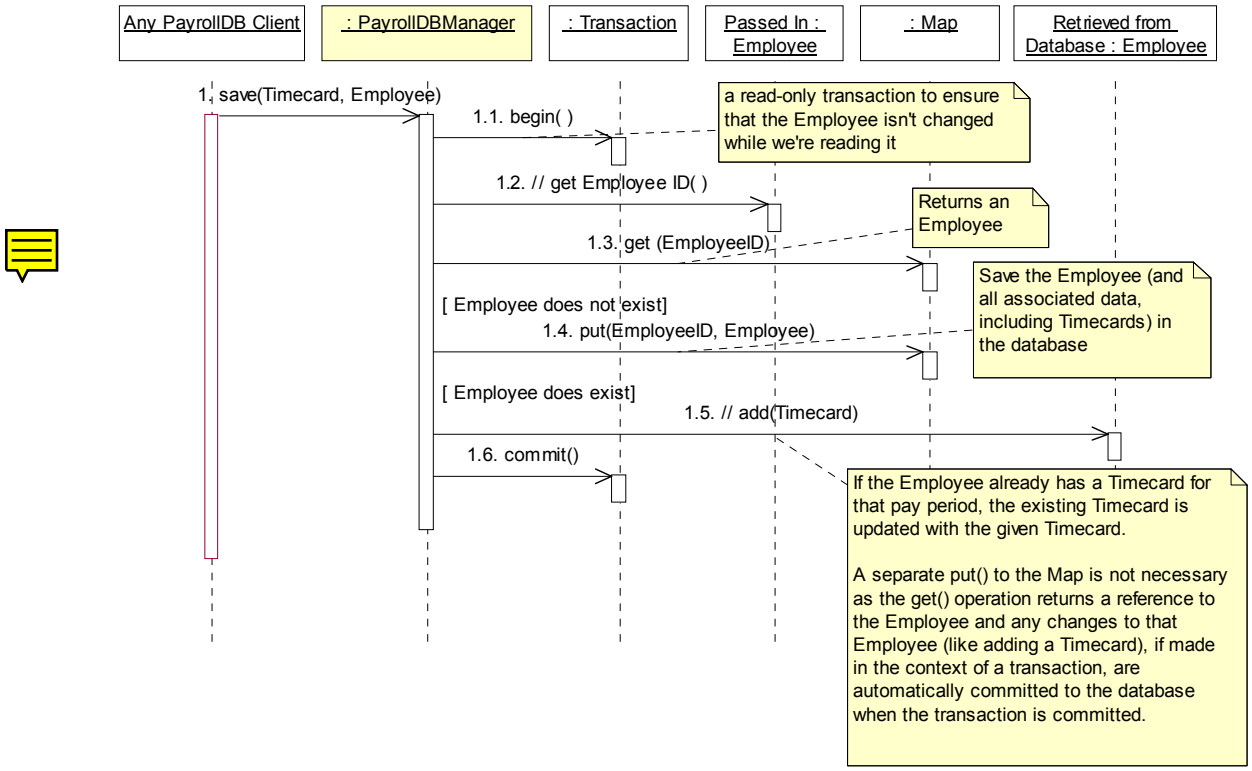


[REPEAT56]

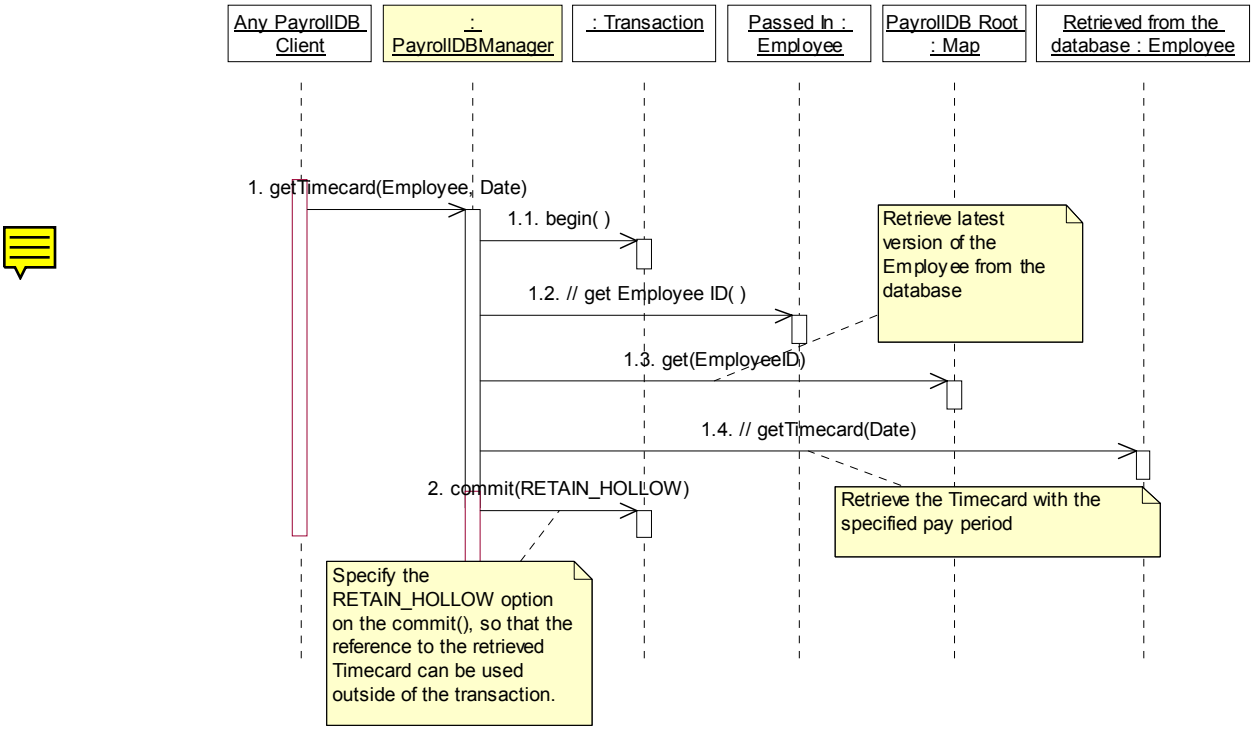
1.4 ObjectStore Support

The following diagrams demonstrate the design of the PayrollDBManager class operations. These are included to supplement the use-case realization diagrams provided above. For the use-case realization diagrams that involve OODBMS persistency, there are references to the diagrams in this section.

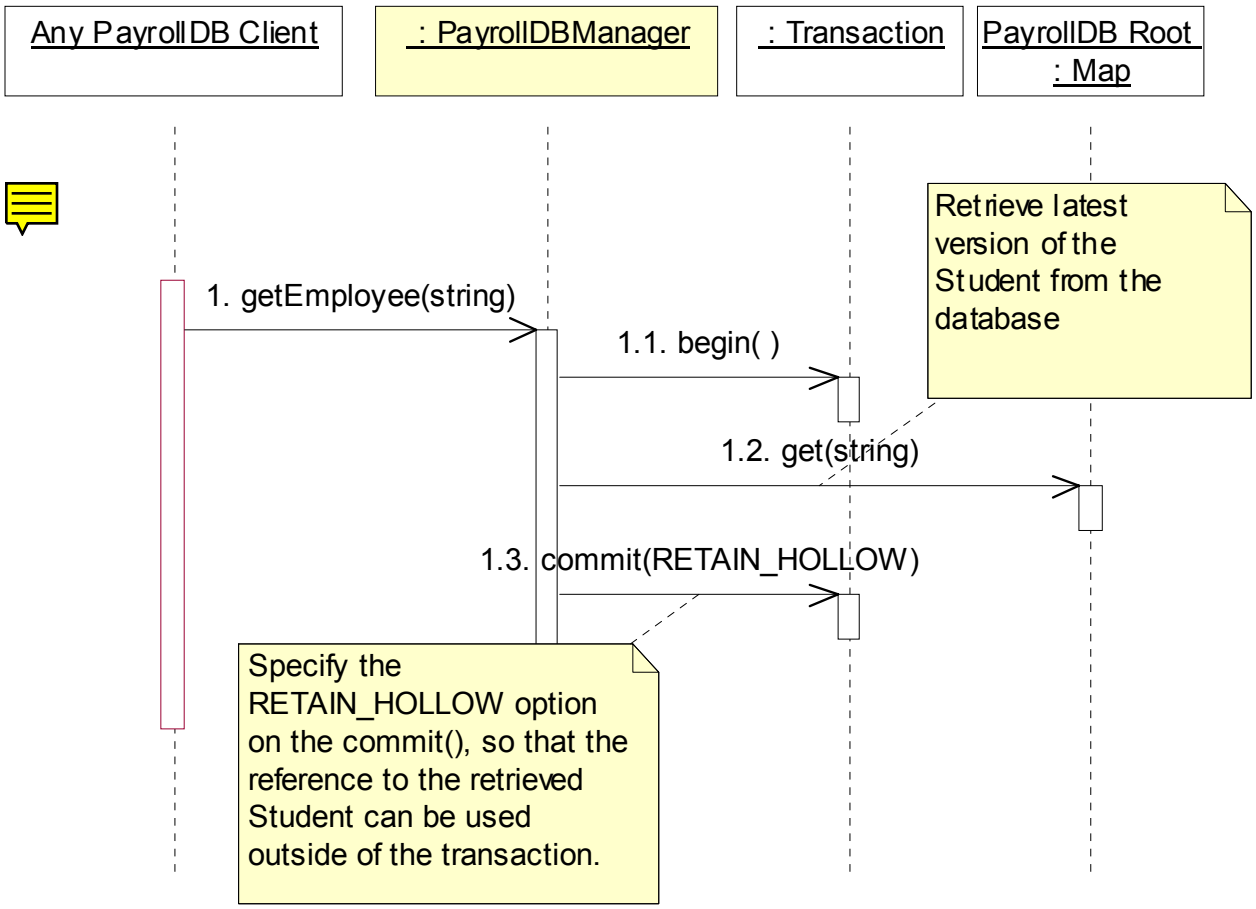
PayrollDBManager - Save Timecard



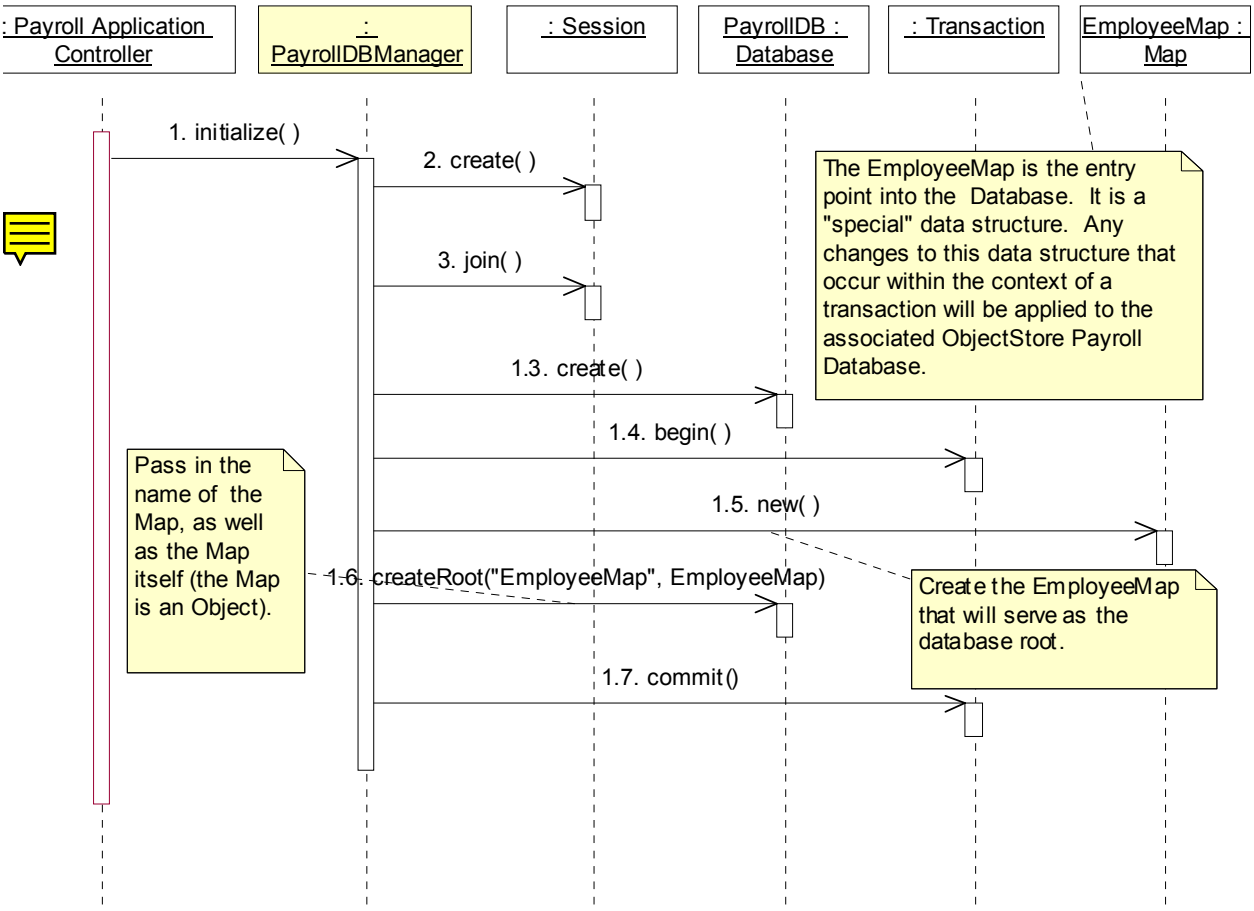
PayrollDBManager - Get Timecard



PayrollDBManager - Get Employee



PayrollDBManager - Initialize



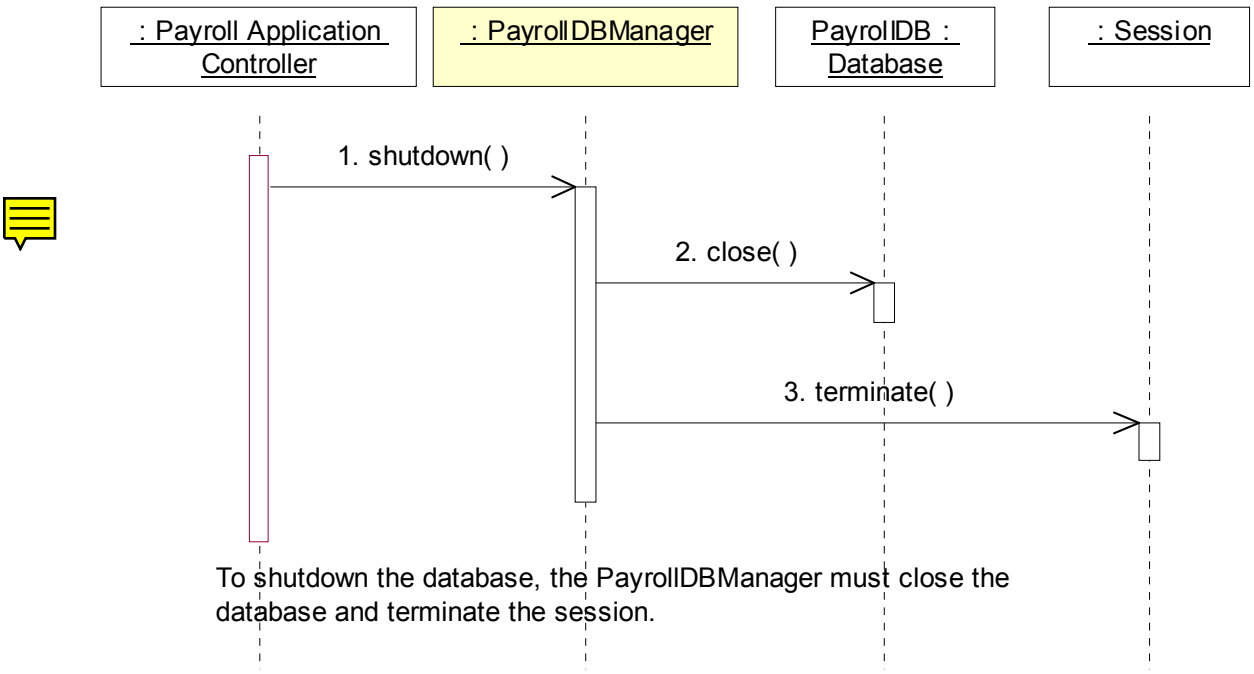
Initialization must occur before any persistent class can be accessed.

Once the session has been created and joined, the PayrollDBManager must open and create the new database.

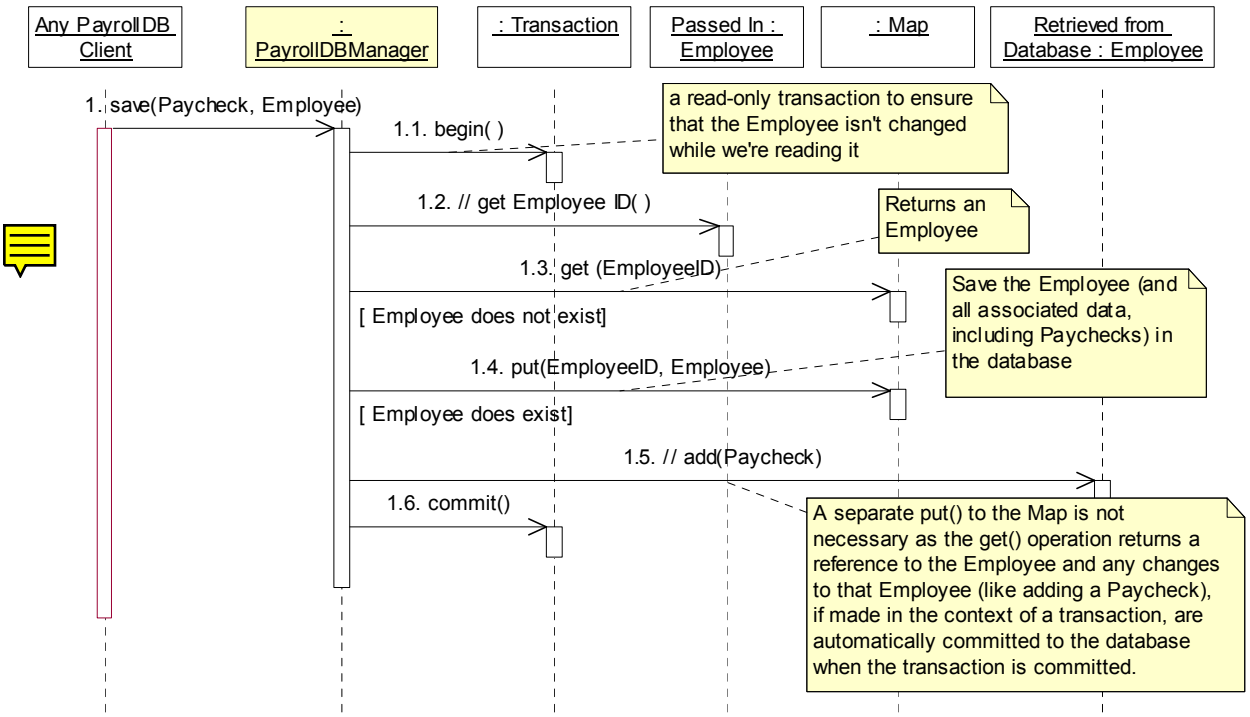
To create the database, the PayrollDBManager creates a new transaction and creates the "root" of the database with the "createRoot()" operation. In our example, the root will be the EmployeeMap data structure. It will contain instances of the Employee class and all “reachable” classes (including Timecards and Purchase Orders). Remember, the root is the entry point into the Payroll Database. It is a "special" data structure. Any changes to this data structure that occur within the context of a transaction will be applied to the associated Payroll ObjectStore Database.

Once the root has been created, the transaction is committed

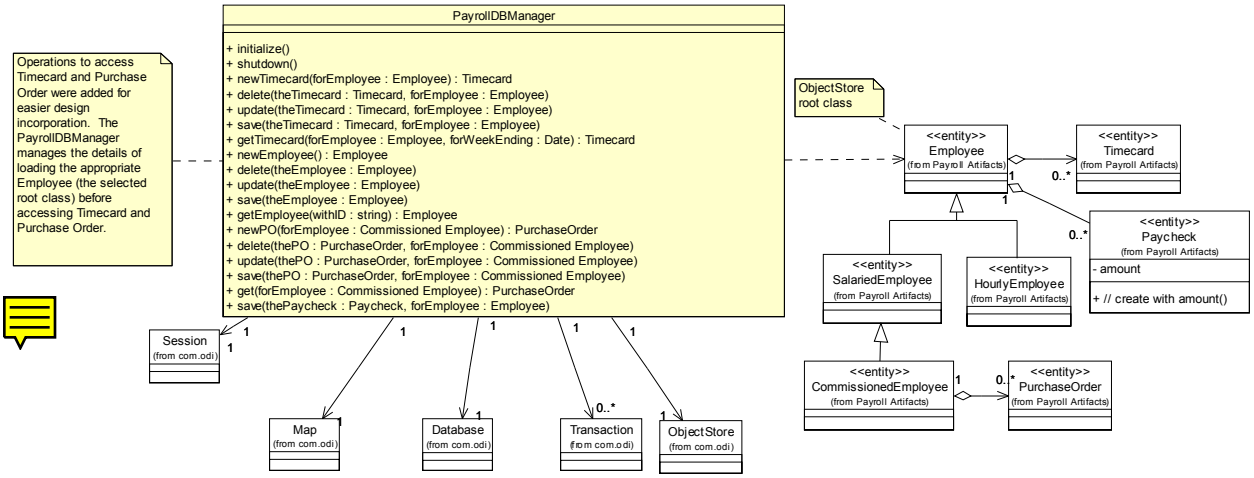
PayrollDBManager - Shutdown



PayrollDBManager - Save Paycheck



Main



Session: The class that represents a database session. A session must be created in order to access the database and any persistent data.

A session is the context in which PSE/PSE Pro databases are created or opened, and transactions can be executed. Only one transaction at a time can exist in a session.

Map: A persistent map container classes that stores key/value pairs.

Database: The Database class represents an ObjectStore database.

Before you begin creating persistent objects, you must create a database to hold the objects. In subsequent processes, you open the database to allow the process to read or modify the objects. To create a database, you call the static create() method on the Database class and specify the database name and an access mode.

Transaction: An ObjectStore transaction. Manages a logical unit of work. All persistent objects must be accessed within a transaction.

ObjectStore: Defines system-level operations that are not specific to any database.

PayrollDBManager: For the Payroll System, there is one ObjectStore database, the Payroll Database, that contains employee, timecard, and purchase order information for the company. There is one PayrollDBManager (i.e., this class is a singleton).

This class is responsible for providing access to the persistent objects in the Payroll Database. It provides a single entry point into the Payroll Database. It contains operations to access entities in the database.

The PayrollDBManager class contains most of the database-specific code, such as starting and ending transactions. There are noPayrollDBManager objects stored in the database, which means that the PayrollDBManager class is not required to be persistence-capable.

The PayrollDBManager class has a static members that keep track of the database that is open. It also has a number of static methods, each of which executes a transaction in the ObjectStore database.

HourlyEmployee: An employee that is paid by the hour.

SalariedEmployee: An employee that receives a salary.

CommissionedEmployee: An employee that receives a commission.

PurchaseOrder: A record of a sale made by an employee.

Timecard: The timecard contains information regarding the hours worked by an employee for a given time period.

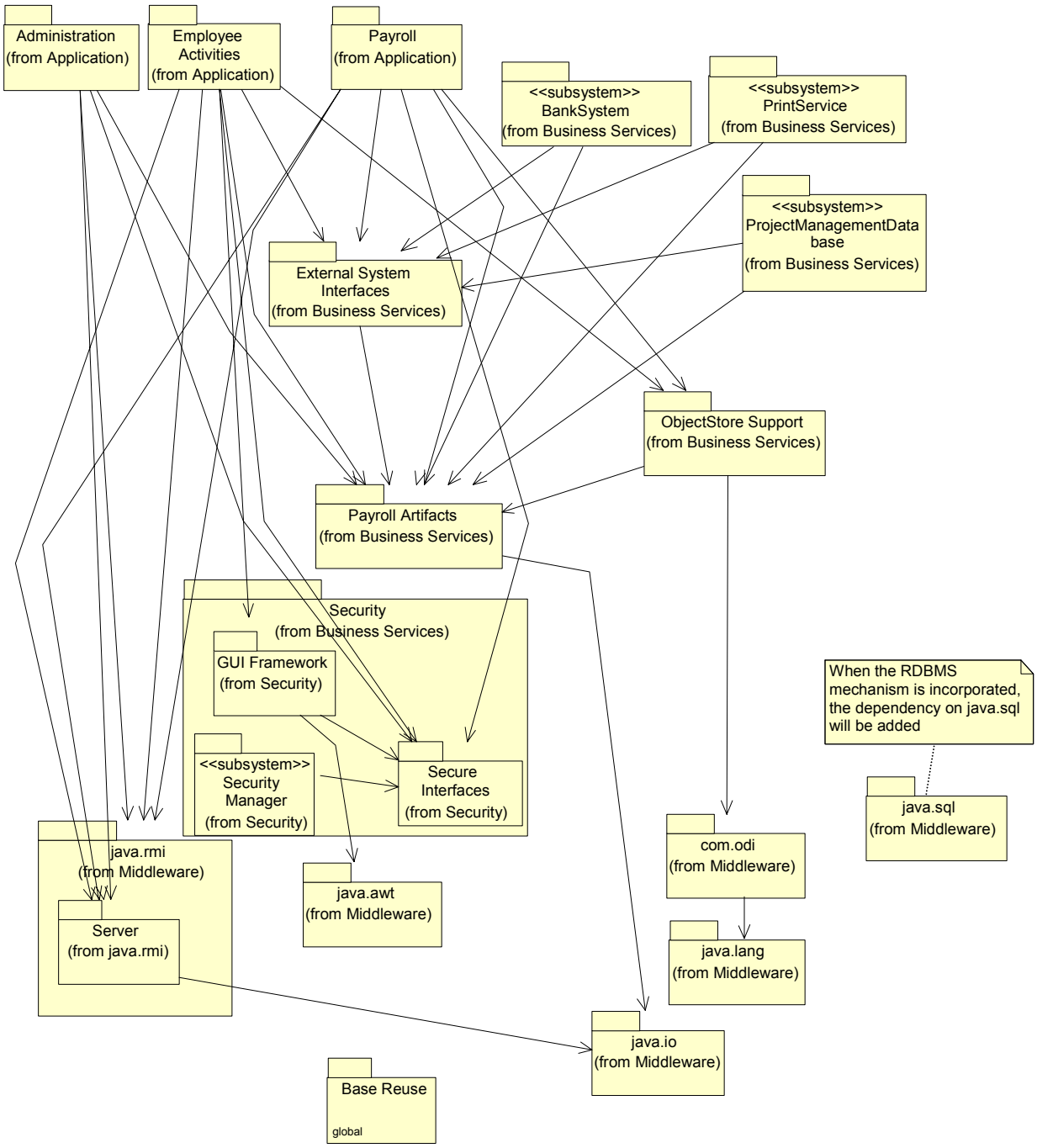
Employee: A person that works for the company.

Paycheck: A record of how much an employee was paid for a given pay period.

2. Exercise: Use-Case Design, Part 2

2.1 Packages and Their Dependencies

Package Dependencies Diagram



2.1.1 Package Descriptions

Administration : Contains the design elements that support the Payroll Administrator's applications.

BankSystem Subsystem: Encapsulates the details involved in communicating with external bank systems.

Base Reuse : Basic reusable design elements.

com.odi : The com.odi package contains the design elements that support the OODBMS persistency mechanism. The name of the package in the model reflects the naming convention for 3rd party Java software. The convention is to use the reverse of the domain name, so if Rational had a Java package called "util" they'd call it" com.rational.util". This com.odi has nothing to do with Microsoft COM/DCOM, they are totally separate. There is nothing COM/DCOM related when using CORBA, RMI, or ObjectStore.

Employee Activities : Contains the design elements that support the Employee's applications.

External System Interfaces : Contains the interfaces that support access to external systems. This is so that the external system interface classes can be version controlled independently from the subsystems that realize them.

GUI Framework : This package comprises a whole framework for user interface management.

It has a ViewHandler that manages the opening and closing of windows, plus window-to-window communication so that windows do not need to depend directly upon each other.

This framework is security-aware, it has a login window that will create a server-resident user context object. The ViewHandler class manages a handle to the user context object.

The ViewHandler also starts up the controller classes for each use case manager.

java.awt : The java.awt package contains the basic GUI design elements for java.

java.io :

java.lang : The package contains some basic java design elements.

java.rmi : The java.rmi package contains the classes that implement the RMI distribution mechanism. This package is commercially available with most standard JAVA IDEs.

java.sql : The package that contains the design elements that support RDBMS persistency.

ObjectStore Support : Contains the business-specific design elements that support the OODBMS persistency mechanism. This includes the DBManager. The DBManager class must contain operations for every OODBMS persistent class.

Payroll : Contains the design elements that support the execution of the payroll processing.

Payroll Artifacts : Contains the core payroll abstractions.

PrintService Subsystem: Provides utilities to produce hard-copy.

ProjectManagementDatabase Subsystem: Encapsulates the interface to the legacy database containing information regarding projects and charge numbers.

Secure Interfaces : Contains the interfaces that provide clients access to security services.

Security : Contains design elements that implement the security mechanism.

Security Manager Subsystem: Provides the implementation for the core security services.

Server :