Payroll System Class Design Solution

Version 2003
<table>
<thead>
<tr>
<th>Date</th>
<th>Issue</th>
<th>Description</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/01/2000</td>
<td>V2000</td>
<td>Generate for beta</td>
<td>Shawn Siemers</td>
</tr>
<tr>
<td>10/03/2000</td>
<td>V2000</td>
<td>Final release</td>
<td>Shawn Siemers</td>
</tr>
<tr>
<td>01/14/2003</td>
<td>V2003</td>
<td>Final Release</td>
<td>Alex Kutsick</td>
</tr>
</tbody>
</table>

**Revision History**
# Table of Contents

1. Exercise: Class Design  
   1.1 Exercise: Define Operations  
   1.2 Exercise: Define States  
   1.3 Exercise: Define Attributes  
   1.4 Exercise: Define Dependencies and Associations  
      1.4.1 Use-Case Realization - Run Payroll  
         1.4.1.1 Run Payroll (with ss interface)  
         1.4.1.2 Run Payroll (with Security)  
         1.4.1.3 Run Payroll (with Distribution)  
         1.4.1.4 Run Payroll (with OODBMS Persistency)  
         1.4.1.5 Run Payroll (with everything)  
      1.4.2 Use-Case Realization - Maintain Timecard  
         1.4.2.1 Maintain Timecard (with ss interface)  
         1.4.2.2 Maintain Timecard (with Security)  
         1.4.2.3 Maintain Timecard (with Distribution)  
         1.4.2.4 Maintain Timecard (with OODBMS Persistence)  
         1.4.2.5 Maintain Timecard (with everything)  
      1.4.3 Use-Case Realization - Login  
         1.4.3.1 Login  
         1.4.3.2 Login (with Security)  
      1.4.4 BankSystem  
      1.4.5 PrintService  
      1.4.6 ProjectManagementDatabase  
   1.5 Exercise: Define Generalizations
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
Payroll - VOPC (with ss interface; ops only)

In this diagram, the attributes have been suppressed.
Use-Case Realization - Maintain Timecard
Maintain Timecard - VOPC (with ss interface; ops only)

```
IProjectManagementDatabase
  + getChargeNumbers(criteria : String) : chargeNumList
  + initialize()

<<Interface>>
```

```
IProjectManagementDatabase
+ getChargeNumbers(criteria : String) : chargeNumList
+ initialize()
```

```
Employee
+ add(theTimecard : Timecard)
+ getEmployeeID() : int
```

```
ChargeNumList
+ create()
+ add()
+ delete()
```

```
Timecard
+ getTotalHours() : float
+ updateTimecard(withTimecardEntry : TimecardEntry)
+ save()
```

```
TimecardEntry
- dayOfWeek : Date
- numHours : float
```

```
PayPeriod
+ startDate : Date
+ endDate : Date
```

```
Any User
```

```
Any User
```

```
LoginForm
+ open()
+ enterUserName()
0..# validateUserIDPassword() : boolean
+ enterPassword()
+ logInUser()
```

In this diagram, the attributes have been suppressed.

1.2 Exercise: Define States

Employee
Apply

Hired Hourly

Hired[ hourly ]

changed to hourly

changed to salaried

Hired[ salary ]

changed to hourly

changed to hourly

Hired[ commissioned ]

changed to hourly

changed to salaried

changed to commissioned

Salaried

Salaried

changed to commissioned

changed to commissioned

changed to salaried

Commissioned

Commissioned

[ age = 65 ]

[ age = 65 ]

Terminated

Terminated

fired

quit

return

request leave

Retired

Retired

Confidential ©Rational Software, 2003 Page 7 of 36
1.3 Exercise: Define Attributes

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

is diagram, the operations have been defined.

At this time, PaymentMethod is intended to be an enumerated type.

Confidential ©Rational Software, 2003 Page 8 of 36
Use-Case Realization - Maintain Timecard
Maintain Timecard - VOPC (with ss interface; attr only)

In this diagram, the operations have been suppressed.

At this time, PaymentMethod is intended to be an enumerated type not being explicitly modeled.

At this time, Date is intended to be an abstract data type not being explicitly modeled.

Use-Case Realization - Login
Login - VOPC (attr only)

In this diagram, the operations have been suppressed.
1.4 Exercise: Define Dependencies and Associations

1.4.2 Case Realization - Run Payroll

1.4.2.1 Payroll (with ss interface)

PayrollController

Employee

HourlyEmployee

- hourlyRate : float
- getHourlyRate() : float

SalariedEmployee

- annualSalary : float
- getAnnualSalary() : float

CommissionedEmployee

- commissionRate : float
- getCommissionRate() : float

POList

PurchaseOrder

- amount : float
- date : Date
+ get PO amount()

Paycheck

- amount : float
+ new(forAmount : float, forPayPeriod : PayPeriod) : Paycheck
+ getAmount() : float
+ getEmployee() : Employee

Employee

- name : string
- employee id : int
- social security number : int
- address : string
- phone number : string
- email : string
- payment method : PaymentMethod

BankInformation

- name : string
- routingNumber : string

PaymentMethod

- calculatePay() : float

Employee

- isPayday() : boolean
- getPayAmount(forPayPeriod : PayPeriod) : float
- getPaymentMethod() : PaymentMethod
- getBankInfo() : BankInformation

BankInformation

- name : string
- routingNumber : string

PaymentMethod

- calculatePay() : float

PayPeriod

- startDate : Date
- endDate : Date

Timecard

- totalNumHours : float
+ getTotalHours() : float
+ updateTimecard(withTimecardEntry : TimecardEntry)

TimecardEntry

- dayOfWeek : Date
- numHours : float

PayrollController

- runPayroll()

SystemClockInterface

- start()

IBankSystem

- deposit(aPaycheck : Paycheck, intoBank : BankInformation)

IPrintService

- print(aPaycheck : Paycheck, onPrinter : string)

Local visibility

(runPayroll())

Global visibility

Parameter visibility

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

1.4.1.3 Payroll (with Distribution)

Otherwise noted, all relationships are field visibility, and List will be used for relationships with a multiplicity greater than one.
### Payroll System Design Solution

#### 1.4.1.4 Run Payroll (with OODBMS Persistency)

- **HourlyEmployee**
  - hourRate : float
  - getHourlyRate()

- **SalariedEmployee**
  - annualSalary : float
  - getAnnualSalary()

- **PurchaseOrder**
  - amount : float
  - date : Date
  - getPOamount()

- **Global**
  - visibility

- **Local**
  - visibility

- **Parameter**
  - visibility

- **PayrollDBManager**
  - + save(theTimecard : Timecard, forEmployee : Employee)
  - + getTimecard(forEmployee : Employee, forPayPeriod : PayPeriod) : Timecard
  - + getEmployee(withID : string) : Employee
  
  *(from ObjectStore Support)*

- **List**
  - *(from Base Reuse)*

- **CommissionedEmployee**
  - commissionRate : float
  - getPurchaseOrders(forTimePeriod : PayPeriod) : POList
  - getCommissionRate() : float

- **POList**
  - *(from Payroll Artifacts)*

- **Global**
  - visibility

- **Local**
  - visibility

- **Parameter**
  - visibility

- **BankInformation**
  - + name : string
  - + routingNumber : string
  
  *(from Payroll Artifacts)*

- **Paycheck**
  - amount : float
  - new(forAmount : float, forPayPeriod : PayPeriod) : Paycheck
  - getAmount()
  - getEmployee()

  *(from Payroll Artifacts)*

- **Employee**
  - name : string
  - employee id : int
  - social security number : int
  - address : string
  - phoneNumber : string
  - email : string
  - payment method : PaymentMethod
  - isPayday() : boolean
  - getPayAmount(forPayPeriod : PayPeriod) : float
  - getPaymentMethod()
  - getBankInfo()
  - calculatePay() : float
  - add(theTimecard : Timecard)
  - getEmployeeID()
  - getTimecard(forPayPeriod : PayPeriod) : Timecard
  - add(thePaycheck : Paycheck)
  - getEmployeeName()

- **Global**
  - visibility

- **Local**
  - visibility

- **Parameter**
  - visibility

- **PayPeriod**
  - + startDate : Date
  - + endDate : Date
  
  *(from Payroll Artifacts)*

- **Timecard**
  - / totalNumHours : float
  - getTotalHours()
  - updateTimecard()
  - new()
  - save()

- **PayrollController**
  - + runPayroll()
  
  *(from Payroll)*

- **SystemClockInterface**
  - start()

- **IPrintService**
  - + print(aPaycheck : Paycheck, onPrinter : string)

- **IBankSystem**
  - + deposit(aPaycheck : Paycheck, intoBank : BankInformation)

*Note: Unless otherwise noted, all relationships are field visibility, and List will be used for relationships with a multiplicity greater than one.*
1.4.1.5 Payroll (with everything)

Payroll System Class Design Solution

- Employee
  - name : string
  - employeeId : int
  - socialSecurityNumber : int
  - address : string
  - phoneNumber : string
  - email : string
  - paymentMethod : PaymentMethod
  - isPayday() : boolean
  - getPayAmount(forPayPeriod : PayPeriod) : float
  - getPayAmount() : float
  - getEmployeeID() : int
  - getEmployeeName() : string

- HourlyEmployee
  - hourlyRate : float
  - getHourlyRate() : float

- SalariedEmployee
  - annualSalary : float
  - getAnnualSalary() : float

- CommissionedEmployee
  - commissionRate : float
  - getCommissionRate() : float

- Paycheck
  - amount : float
  - new(amount : float, forPayPeriod : PayPeriod) : Paycheck
  - getAmount() : float
  - getEmployee() : Employee

- Timecard
  - totalNumHours : float
  - getTotalHours() : float
  - updateTimecard() : void
  - new() : Timecard

- PayPeriod
  - startDate : Date
  - endDate : Date

- BankInformation
  - name : string
  - routingNumber : string

- BankSystem
  - deposit(aPaycheck : Paycheck, intoBank : BankInformation)

- PayrollController
  - runPayroll()

- IPayrollController
  - runPayroll()

- IRemoteObject
  - clone()
  - exportObject()

- Serializable

- UnicastRemoteObject
  - # UnicastRemoteObject()
  - clone()
  - exportObject()

- Remote
  - UnicastRemoteObject()

- IBankSystem
  - deposit(aPaycheck : Paycheck, intoBank : BankInformation)

- IPrintService
  - print(aPaycheck : Paycheck, onPrinter : string)

- SystemClockInterface
  - start()

- UnicastRemoteObject
  - # UnicastRemoteObject()
  - clone()
  - exportObject()
1.4.2 Case Realization - Maintain Timecard

1.4.2.1 Maintain Timecard (with ss interface)

Maintain Timecard - VOPC (with ss interface)

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

Global visibility

ChargeNumList

Parameter visibility

TimecardForm

TimecardEntry

Employee

Employee id : int

addTimecard : Timecard

getEmployeeID() : int

<<class>> + new()

PayPeriod

startDate : Date

dateEnd : Date

<<class>> + new()

Timecard

totalNumHours : float

getTotalHours() : float

updateTimecard(withEntry : TimecardEntry)

<<class>> + new()

IProjectManagementDatabase

getChargeNumbers(criteria : String) : chargeNumList

initialize()

<<Interface>>

ChargeNumList

+ create()

+ add()

+ delete()

TimecardForm

displayTimecard()

create()

add()

delete()

TimecardEntry

dayOfWeek : Date

numHours : float

<<class>>

Employee

employee id : int

addTimecard : Timedcard

getEmployeeID() : int

<<class>>

PayPeriod

startDate : Date

dateEnd : Date

<<class>>

Timecard

totalNumHours : float

getTotalHours() : float

updateTimecard(withTimecardEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>

TimecardController

getCurrentTimecard(forEmployee : Employee) : Timecard

getChargeNums() : ChargeNumList

updateTimecard(withEntry : TimecardEntry)

<<class>>
1.4.2.2 Maintain Timecard (with Security)

Main Timecard - VOPC (with Security)

- isReadable()
- isWriteable()
- isDeleteable()
- makeReadable()
- makeWriteable()
- makeDeleteable()
1.4.2.3 Maintain Timecard (with Distribution)

Maintain Timecard - VOPC (with Distribution)

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

Global visibility

Distributed class client

Global visibility

Remote

Naming

UnicastRemoteObject

# UnicastRemoteObject()
+ clone()
+ exportObject()

(from Server)

TimecardEntry

- dayOfWeek : Date
- numHours : float
+ new()

(from Payroll Artifacts)

Employee

- employeeId : int
+ add(theTimecard : Timecard)

(from Payroll Artifacts)

<<entity>>

Timecard

/ - totalNumHours : float
+ getTotalHours() : float
+ updateTimecard(withTimecardEntry : TimecardEntry)
+ new(forPayPeriod : PayPeriod) : Timecard
+ save()

(from Payroll Artifacts)

<<entity>>

TimecardController

+ getCurrentTimecard(forEmployee : Employee) : Timecard
+ getChargeNums() : ChargeNumList
+ updateTimecard(withEntry : TimecardEntry)
+ new() : TimecardController
+ saveTimecard()

(from Employee Activities)

<<control>>

TimecardForm

# displayTimecard()
+ new(forUser : ISecureUser) : TimecardForm
+ displayChargeCodes()
+ maintainTimecard()

(from Employee Activities)

<<boundary>>

ITimecardController
+ get current timecard()
+ get charge codes()
+ update timecard()
+ setSession()
+ create()

(from Employee Activities)

<<Interface>>

IProjectManagementDatabase
+ getChargeNumbers(criteria : String) : chargeNumList
+ initialize()

(from External System Interfaces)

Serializable

(from java.io)

Serialization.

# lookup()

(from java.rmi)

Naming.

Remote

(from java.rmi)

UnicastRemoteObject

+ lookup()

(from java.rmi)
1.4.2.4 Maintain Timecard (with OODBMS Persistence)

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

Global visibility

ChargeNumList
+ create()
+ add()
+ delete()
(from Payroll Artifacts)

Parameter visibility

TimecardEntry
- dayOfWeek : Date
- numHours : float
<<class>> + new()
(from Payroll Artifacts)

Employee
(from Payroll Artifacts)
<<entity>>

Timecard
/- totalNumHours : float
+ getTotalHours() : float
+ updateTimecard(withTimecardEntry : TimecardEntry)
<<class>> + new(forPayPeriod : PayPeriod) : Timecard
+ save()
(from Payroll Artifacts)
<<entity>>

PayPeriod
+ startDate : Date
+ endDate : Date
(from Payroll Artifacts)

TimecardController
+ getCurrentTimecard(forEmployee : Employee) : Timecard
+ getChargeNums() : ChargeNumList
+ updateTimecard(withEntry : TimecardEntry)
<<class>> + new() : TimecardController
+ saveTimecard()
(from Employee Activities)
<<control>>

TimecardForm
# displayTimecard()
<<class>> + new(forUser : ISecureUser) : TimecardForm
+ // enter hours for charge numbers()
+ displayChargeCodes()
+ saveTimecard()
(from Employee Activities)
<<boundary>>

IProjectManagementDatabase
+ getChargeNumbers(criteria : String) : chargeNumList
+ initialize()
(from External System Interfaces)
<<Interface>>

PayloadDBManager
+ saveTheTimecard : Timecard, forEmployee : Employees
+ getTimecard(forEmployee : Employee, forPayPeriod : PayPeriod, : Timecard
(from Objective Support)

Global visibility

PayrollDBManager
+ save(theTimecard : Timecard, forEmployee : Employee)
+ getTimecard(forEmployee : Employee, forPayPeriod : PayPeriod) : Timecard
(from ObjectStore Support)
1.4.2.5 Maintain Timecard (with everything)

Maintain Timecard - VOPC (with everything)

Note: Most all relationships are field visibility, and List will be used for lists with a multiplicity greater than one.

Employee, Timecard, TimecardEntry, and ChargeNumList must be passed between distributed objects.

1.4.3 Use-Case Realization - Login

1.4.3.1 LoginForm

LoginForm
(from GUI Framework)

+ open()
+ enterUserName()
# validateUserNamePassword() : boolean
+ enterPassword()
+ loginUser()
# setupSecurityContext()
+ getUserContext() : ISecureUser
1.4.3.2 Login (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 System

Main

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

Main

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

Main

A "plural" DBClass was defined because the charge numbers always are retrieved as a set (a list of the available charge numbers).

```
ResultSet + getString() : string (from java.sql)
```

```
Connection + createStatement() : Statement (from java.sql)
```

```
DriverManager + getConnection(url, user, pass) : Connection (from java.sql)
```

```
Statement + executeQuery(sql : string) : ResultSet
+ executeUpdate(sql : string) : int (from java.sql)
```

```
Parameter + visibility
+ create() + add(aChargeNum : ChargeNum) + delete(aChargeNum : ChargeNum) (from Payroll Artifacts)
```

```
Local + visibility
+ projectName : string - value : string
+ getProjectName() : string + getValue() : string + create(projectName, value)
```

```
Global + visibility
+ getChargeNumbers(criteria : String) : chargeNumList + initialize()
```

```
DBChargeNumbers + getChargeNums(criteria : string) : ChargeNumList + initialize()
```

```
ChargeNumList + create()
+ add(aChargeNum : ChargeNum)
+ delete(aChargeNum : ChargeNum) (from Payroll Artifacts)
```

```
ChargeNum + projectName : string - value : string + getProjectName() + getValue() + create(projectName, value)
```

```
IProjectManagementDatabase + getChargeNumbers(criteria : String) + initialize() (from External System)
```

Confidential ©Rational Software, 2003 Page 22 of 36
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)

- PickUpMethod
  - thePrinter : string
  + deposit(aPaycheck : Paycheck, intoBank : BankInformation)
  (from External System Interfaces)
  <<Interface>>

- BankInformation
  + name : string
  + routingNumber : string

- DirectDepositMethod

- IPrintService
  + print(aPaycheck : Paycheck, onPrinter : string)
  + printMailingLabel(toAddress : Address, fromAddress : Address)
  (from External System Interfaces)
  <<Interface>>

- PaymentMethod
  + pay()

Employee (EmpClassification metamorphosis)

- Timecard
  - totalNumHours : float
  + getTotalHours()
  + updateTimecard()
  <<class>>
  + new()
  + save()

- Address
  + street : string
  + city : string
  + state : string
  + country : string
  + zipCode : string

- MailMethod

- ReturnAddress
  + mailingAddress

- HomeAddress
  + homeAddress

- Paycheck

- Employee
  - name : string
  - employeeId : int
  - socialSecurityNumber : int
  - phoneNumber : string
  - email : string
  + isPayday()
  + getPayAmount()
  + getPaymentMethod()
  + calculatePay()
  + add()
  + getEmployeeID()
  + getClassification()

- BankSystem (denotes External System Interface)
  + deposit(aPaycheck : Paycheck intoBank : BankInformation)

- Employee
  + print(aPaycheck : Paycheck, onPrinter : string)
  + printMailingLabel(toAddress : Address, fromAddress : Address)
  <<Interface>>

- Paycheck
class_design_solution_rpt.doc

<<entity>>
Employee
- name : string
- employee id : int
- social security number : int
- phone number : string
- email : string

+ isPayday()
+ getPayAmount()
+ getPaymentMethod()
# calculatePay()
+ add()
+ getEmployeeID()
+ getTimecard()
+ add()
+ getEmployeeName()
+ getClassification()

<<entity>>
HourlyClassification
- hourlyRate : float

+ getHourlyRate()

<<entity>>
SalariedClassification
- annualSalary : float

+ getAnnualSalary()

<<entity>>
CommissionedClassification
- commissionRate : float

+ getPurchaseOrders()
+ getCommissionRate()

<<class>> + new()
+ save()

<<entity>>
Timecard
/- totalNumHours : float

+ getTotalHours()
+ updateTimecard()
<<class>> + new()
+ save()

<<entity>>
EmpClassification

<<entity>>
POList

<<bind>>
List
(from Base Reuse)

<<entity>>
PurchaseOrder

+ get PO amount()
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)

1. start()
1.1. runPayroll()
1.1.1. isPayday()
1.1.2. getPayAmount(PayPeriod)
1.1.2. getHours() (CommissionedClassification)
1.1.2.2. getPurchaseOrders(PayPeriod)
1.1.2.3. getPOamount()
1.1.2.4. calculatePay()
1.1.3. new(float, PayPeriod)
1.1.4. add(Paycheck)
1.1.5. getPaymentMethod()
1.1.6. pay(Paycheck)

For all timecards for the specified pay period
For all POs for the specified pay period
Commissioned employees also need to retrieve POs
In the future, additional calculations may be needed to calculate the Employee's take-home pay.
These calculations would be inserted here.

Perform these steps for each employee
Create a Paycheck for the specified amount for the Employee
Execute the Employee's PaymentMethod

Sequence Diagram: Payroll
Artifacts / PaymentMethod::pay
Payroll - Basic Flow (with ss interface)

1. start()
   1.1. runPayroll()
       1.1.1. isPayday()
       1.1.2. getPayAmount(PayPeriod)
       1.1.3. new(float, PayPeriod)
       1.1.4. add(Paycheck)
       1.1.5. getPaymentMethod()
   1.1.6. pay(Paycheck)

   : SystemClock
   1.1. start()

   : PayrollController
   1.1. runPayroll()
       1.1.1. isPayday()
       1.1.2. getPayAmount(PayPeriod)
       1.1.3. new(float, PayPeriod)
       1.1.4. add(Paycheck)
       1.1.5. getPaymentMethod()
       1.1.6. pay(Paycheck)

   : PaymentMethod

   : SystemClockInterface

   : Employee
   1.1.2. calculatePay()
       1.1.2.1. getTotalHours()
       1.1.2.2. getPurchaseOrders(PayPeriod)
       1.1.2.3. getPO amount()
       1.1.2.4. getPO amount()
       1.1.2.5. getPO amount()

   : CommissionedClassification

   : Paycheck
   1.1. new(float, PayPeriod)

   : Timecard

   : PurchaseOrder
Payroll - VOPC (with ss interface)

Unless otherwise noted, all relationships are field visibility, and List will be used for all relationships with a multiplicity greater than one.
Payroll - Basic Flow (with Distribution)

Run Payroll - Basic Flow (with Distribution):

1. System Clock

1.1. lookup(String)

1.2. runPayroll()

2. IPayrollController

2.1. isPayday()

2.2. getPayAmount(PayPeriod)

2.2.1. getTotalHours()

2.2.2. getPurchaseOrders(PayPeriod)

2.2.3. get PO amount()

2.2.4. calculatePay()

2.2.5. getPaymentMethod()

2.2.6. pay(Paycheck)

3. Employee

3.1. lookup(String)

4. CommissionedClassification

5. PaymentMethod

6. Timecard

6.1. isPayday()

6.2. getPayAmount(PayPeriod)

6.2.1. getTotalHours()

6.2.2. getPurchaseOrders(PayPeriod)

6.2.3. get PO amount()

6.2.4. calculatePay()

6.2.5. getPaymentMethod()

6.2.6. pay(Paycheck)

7. Paycheck

7.1. new(float, PayPeriod)

7.2. add(Paycheck)

7.3. pay(Paycheck)

8. PurchaseOrder

In the future, additional calculations may be needed to calculate the Employee's take-home pay. These calculations would be inserted here.

Perform these steps for each employee.

Commissioned employees also need to retrieve POs for the specified pay period.

Sequence Diagram: Payroll Artifacts / PaymentMethod::pay

Create a Paycheck for the specified amount for the Employee.

Execute the Employee's PaymentMethod.
Run Payroll - VOPC (with Distribution)

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

Local visibility
((getPayAmount()))

Paycheck
- amount : float
+ getAmount() : float
+ getEmployee() : Employee

Employee
- name : string
- employee id : int
- social security number : int
- phone number : string
- email : string
+ isPayday() : boolean
+ getPayAmount(forPayPeriod : PayPeriod) : float
# calculatePay() : float
+ addTheTimecard : Timecard
+ getEmployeeName() : string
+ getClassification() : EmpClassification

Timecard
- totalNumHours : float
+ getTotalHours() : float
+ updateTimecard() : float
+ new() : Timecard
+ save() : bool

PayPeriod
+ startDate : Date
+ endDate : Date

POList
- amount : float
date : Date
+ getPOAmount() : float

EmpClassification

HourlyClassification
- hourlyRate : float
+ getHourlyRate() : float

SalariedClassification
- annualSalary : float
+ getAnnualSalary() : float

CommissionedClassification
- commissionRate : float
+ getPurchaseOrders(forTimePeriod : PayPeriod) : POList
+ getCommissionRate() : float

POList
- amount : float
date : Date
+ getPOAmount() : float

Remote
+ start()

SystemClockInterface

PayrollSystem

PayrollController
+ runPayroll()

IPayrollController
+ // run payroll()

Employee

Timecard

POList

POOrder

PurchaseMethod

PaymentMethod

Employee

EmpClassification

EmpClassification

HourlyClassification

SalariedClassification

CommissionedClassification

Local visibility

Global visibility

Naming
+ lookup()
Run Payroll - Basic Flow (with OODBMS Persistency)

Perform these steps for each employee

1. start()
   1.1. runPayroll()
      1.1.1. getEmployee(string)
      1.1.2. isPayday()
      1.1.3. getPayAmount(PayPeriod)
      1.1.4. new(float, PayPeriod)
      1.1.5. add(Paycheck)
      1.1.6. save(Paycheck, Employee)
      1.1.7. getPaymentMethod()
      1.1.8. pay(Paycheck)

Sequence Diagram: OODBMS Support / PayrollDBManager - Get Employee

When the Employee is retrieved from the database, the Timecards and PurchaseOrders are retrieved as well.

Sequence Diagram: OODBMS Support / PayrollDBManager - Save Paycheck

Create a new paycheck for the employee containing the calculated pay.

Sequence Diagram: Payroll Artifacts / PaymentMethod::pay

For all POs for the specified pay period

In the future, additional calculations may be needed to calculate the Employee's take-home pay. These calculations would be inserted here.

In the future, additional calculations may be needed to calculate the Employee's take-home pay. These calculations would be inserted here.

For all POs for the specified pay period

Sequence Diagram: OODBMS Support / PayrollDBManager - Get Employee

Commissioned employees also need to retrieve POs

For all POs for the specified pay period

When the Employee is retrieved from the database, the Timecards and PurchaseOrders are retrieved as well.

Commissioned employees also need to retrieve POs

Perform these steps for each employee

1. start()
   1.1. runPayroll()
      1.1.1. getEmployee(string)
      1.1.2. isPayday()
      1.1.3. getPayAmount(PayPeriod)
      1.1.4. new(float, PayPeriod)
      1.1.5. add(Paycheck)
      1.1.6. save(Paycheck, Employee)
      1.1.7. getPaymentMethod()
      1.1.8. pay(Paycheck)

Sequence Diagram: OODBMS Support / PayrollDBManager - Get Employee

When the Employee is retrieved from the database, the Timecards and PurchaseOrders are retrieved as well.

Sequence Diagram: OODBMS Support / PayrollDBManager - Save Paycheck

Create a new paycheck for the employee containing the calculated pay.

Sequence Diagram: Payroll Artifacts / PaymentMethod::pay

For all POs for the specified pay period

In the future, additional calculations may be needed to calculate the Employee's take-home pay. These calculations would be inserted here.

In the future, additional calculations may be needed to calculate the Employee's take-home pay. These calculations would be inserted here.

For all POs for the specified pay period

Sequence Diagram: OODBMS Support / PayrollDBManager - Get Employee

Commissioned employees also need to retrieve POs

For all POs for the specified pay period

When the Employee is retrieved from the database, the Timecards and PurchaseOrders are retrieved as well.

Commissioned employees also need to retrieve POs

Perform these steps for each employee

1. start()
   1.1. runPayroll()
      1.1.1. getEmployee(string)
      1.1.2. isPayday()
      1.1.3. getPayAmount(PayPeriod)
      1.1.4. new(float, PayPeriod)
      1.1.5. add(Paycheck)
      1.1.6. save(Paycheck, Employee)
      1.1.7. getPaymentMethod()
      1.1.8. pay(Paycheck)
Payroll - VOPC (with OODBMS Persistency)

more than one relationships are field visibility, and can be used to
shapes with a multiplicity of more than one.

PayrollDBManager

Global

+ save (theTimecard : Timecard, forEmployee : Employee)
+ getTimecard (forEmployee : Employee, forPayPeriod : PayPeriod) : Timecard
+ getEmployee (withID : string) : Employee
(from ObjectStore Support)

Local

Parameter

Visibility

IBankSystem

+ deposit (aPaycheck : Paycheck, intoBank : BankInformation)
(from External System Interfaces)

Parameter

Visibility

IPrintService
+ print (aPaycheck : Paycheck, onPrinter : string)
+ printMailingLabel (toAddress : Address, fromAddress : Address)
(from External System Interfaces)

Parameter

Visibility

PayrollController
+ runPayroll()
(from Payroll)

SystemClockInterface

start()
(from Boundary)

Parameter

Visibility

PayPeriod

+ startDate : Date
+ endDate : Date
(from Payroll Artifacts)

Paycheck

amount : float
<<class>> new (forAmount : float, forPayPeriod : PayPeriod) : Paycheck
getAmount () : float
getEmployee () : Employee
(from Entity)

Timecard

/ totalNumHours : float
getTotalHours ()
updateTimecard()
<<class>> new ()
save()
(from Entity)

Employee

name : string
employeeID : int
socialSecurityNumber : int
phoneNumber : string
email : string
isPayday () : boolean
getPayAmount (forPayPeriod : PayPeriod) : float
getPaymentMethod () : PaymentMethod
calculatePay () : float
add (theTimecard : Timecard)
getEmployeeID () : int
getTimecard (forPayPeriod : PayPeriod) : Timecard
add (thePaycheck : Paycheck)
getEmployee Name () : string
getClassification () : EmpClassification
(from Entity)

EmpClassification

(from Payroll Artifacts)

0..1

0..1

0..1

0..1

CommissionedClassification

0..1

0..1

0..1

0..1

SalariedClassification

0..1

0..1

0..1

0..1

HourlyClassification

0..1

0..1

0..1

0..1

PurchaseOrder

amount : float
<<entity>> new ()
POLine

getPOAmount ()
<<entity>> new ()

getPO amount ()

CommissionedClassification

commissionRate : float
getPurchaseOrders (forTimePeriod : PayPeriod) : POList
getCommissionRate () : float
(from Entity)

POList

1

0..1

1

0..1

0..1

0..1

0..1

0..1

0..1

0..1

PurchaseOrder

get PO amount ()
<<entity>> new ()
Run Payroll - Basic Flow (with everything)

1. start()
   1.2. runPayroll()

   1.2.1. getEmployee(string)
   1.2.2. isPayday()
   1.2.3. getPayAmount(PayPeriod)
      1.2.3.1. getTotalHours()
      1.2.3.2. getPurchaseOrders(PayPeriod)
      1.2.3.3. get PO amount()
      1.2.3.4. calculatePay()

   1.2.4. new(float, PayPeriod)
   1.2.5. add(Paycheck)
   1.2.6. save(Paycheck, Employee)
   1.2.7. getPaymentMethod()
   1.2.8. pay(Paycheck)

1.2. runPayroll()
   1.1. lookup(String)
   1.2. runPayroll()

   1.2.1. getEmployee(string)
   1.2.2. isPayday()
   1.2.3. getPayAmount(PayPeriod)
      1.2.3.1. getTotalHours()
      1.2.3.2. getPurchaseOrders(PayPeriod)
      1.2.3.3. get PO amount()
      1.2.3.4. calculatePay()

   1.2.4. new(float, PayPeriod)
   1.2.5. add(Paycheck)
   1.2.6. save(Paycheck, Employee)
   1.2.7. getPaymentMethod()
   1.2.8. pay(Paycheck)

Payroll - Basic Flow (with everything)

1. start()
   1.2. runPayroll()

   1.2.1. getEmployee(string)
   1.2.2. isPayday()
   1.2.3. getPayAmount(PayPeriod)
      1.2.3.1. getTotalHours()
      1.2.3.2. getPurchaseOrders(PayPeriod)
      1.2.3.3. get PO amount()
      1.2.3.4. calculatePay()

   1.2.4. new(float, PayPeriod)
   1.2.5. add(Paycheck)
   1.2.6. save(Paycheck, Employee)
   1.2.7. getPaymentMethod()
   1.2.8. pay(Paycheck)
**Payroll - VOPC (with everything)**

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

<table>
<thead>
<tr>
<th>Class</th>
<th>Visibility</th>
<th>Method(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>Local</td>
<td>name, employeeId, phone number, email, isPayday, getPaymentMethod, getPayAmount, getEmployeeID, getEmployeeName, getClassification</td>
</tr>
<tr>
<td>HourlyClassification</td>
<td>Local</td>
<td>hourlyRate, getHourlyRate</td>
</tr>
<tr>
<td>SalariedClassification</td>
<td>Local</td>
<td>annualSalary, getAnnualSalary</td>
</tr>
<tr>
<td>CommissionedClassification</td>
<td>Local</td>
<td>getPurchaseOrders, getCommissionRate</td>
</tr>
<tr>
<td>Paycheck</td>
<td>Local</td>
<td>amount, getAmount, getEmployee, getPayAmount, getPaymentMethod, getEmployeeID, getDate, getEmployeeName, getClassification</td>
</tr>
<tr>
<td>Timecard</td>
<td>Local</td>
<td>totalNumHours, getTotalHours, add, getTimecard, add, getEmployeeID, getEmployeeName, getClassification</td>
</tr>
<tr>
<td>List</td>
<td>Local</td>
<td>PurchaseOrder</td>
</tr>
<tr>
<td>List</td>
<td>Local</td>
<td>POList</td>
</tr>
<tr>
<td>UnicastRemoteObject</td>
<td>Remote</td>
<td>clone, exportObject</td>
</tr>
<tr>
<td>Naming</td>
<td>Remote</td>
<td>lookup</td>
</tr>
<tr>
<td>SystemClockInterface</td>
<td>Local</td>
<td>start</td>
</tr>
<tr>
<td>PayrollDBManager</td>
<td>Local</td>
<td>save, getTimecard, getEmployee</td>
</tr>
<tr>
<td>PayrollController</td>
<td>Local</td>
<td>runPayroll</td>
</tr>
<tr>
<td>PayPeriod</td>
<td>Local</td>
<td>startDate, endDate</td>
</tr>
<tr>
<td>Paycheck</td>
<td>Local</td>
<td>amount, getAmount, getEmployee, getPayPeriod, calculatePay, add, getEmployeeID, getTimecard, add, getEmployeeName, getClassification</td>
</tr>
<tr>
<td>PaymentMethod</td>
<td>Local</td>
<td>pay</td>
</tr>
<tr>
<td>Employee</td>
<td>Local</td>
<td>name, employeeId, social security number, phone number, email, isPayday, getPayAmount, getPaymentMethod, getPayPeriod, calculatePay, add, getEmployeeID, getEmployeeName, getClassification</td>
</tr>
<tr>
<td>EmpClassification</td>
<td>Local</td>
<td>0..1</td>
</tr>
<tr>
<td>PurchaseOrder</td>
<td>Local</td>
<td>amount, date, getPOamount</td>
</tr>
<tr>
<td>POList</td>
<td>Local</td>
<td>1, 0..1</td>
</tr>
</tbody>
</table>

Confidential ©Rational Software, 2003 Page 34 of 36
PaymentMethod::pay

1. pay(Paycheck)
   1.1. deposit(Paycheck, BankInformation)

2. pay(Paycheck)
   2.1. print(Paycheck, string)

3. pay(Paycheck)
   3.1. print(Paycheck, string)
   3.2. printMailingLabel(Address, Address)

[PaymentMethod was DirectDepositMethod]
[PaymentMethod was PickupMethod]
[PaymentMethod was MailMethod]

Sequence Diagram: IBankSystem / IBankSystem::deposit
Sequence Diagram: IPrintService / IPrintService::print
```plaintext
PaymentMethod::pay

1. pay(Paycheck)
   1.1. deposit(Paycheck, BankInformation)

2. pay(Paycheck)
   2.1. print(Paycheck, string)

3. pay(Paycheck)
   3.1. print(Paycheck, string)
   3.2. printMailingLabel(Address, Address)

PickUpMethod

MailMethod

DirectDepositMethod

IBankSystem

IPrintService
```