Squeak
Human Resources Example

Technion - Israel Institute of Technology

Updated: October 2015
**Human Resources Example**

- **Key class:** Employee
  - **Structure:**
    - Number
    - Name
    - Salary
    - Set of skills
  - **Behaviour:**
    - Access attributes
    - Update attributes

**Database Class:**

**EmployeeDataBase**

- **Structure:**
  - Set of employees

- **Behaviour:**
  - Add employees
  - Search for employees
Object subclass: #Employee
  instanceVariableNames:'empName empNum empSalary empSkills'
  classVariableNames: ''
  poolDictionaries: ''
  category: 'Employee-Example'

Full Code is in Employee.st
Basic Methods in Employee Class

! !Employee methodsFor: 'initialize-release'

**initialize**

    empName := 'Steve'.
    empNum := 0.
    empSalary := 0.
    empSkills := Set new.
    ^self

! !

**name**

    ^empName

! !

**name: aName**

    empName := aName.

! !

**empNum**

    ^empNum

! !

...
Object subclass: #Employee

... 

!!!

empNum: anEmpNum

"Employee numbers must be between 1000 and 1999."
"This is a corporate regulation."
((anEmpNum >= 1000) and: [anEmpNum <= 1999])
  ifTrue: [
    empNum := anEmpNum.
    ^self
  ]
  ifFalse: [
    self error: 'Invalid employee number'
  ]

!!!

salary

^empSalary
!!!

...
Object subclass: #Employee

...!

salary: aSalary
    empSalary := aSalary.

!

addSkill: aSkill
    empSkills add: aSkill.

!

hasSkill: aSkill
    ^(empSkills occurrencesOf: aSkill) = 1

!

print
    Transcript show:('name ', empName);cr.
    Transcript show:('number: ', empNum asString);cr.
    Transcript show:('Salary: ', empSalary asString);cr.
    Transcript show: 'Skills:';cr.
    empSkills do:
        [:x | Transcript show: ('   ', x asString);cr ].
Object subclass: #EmployeeDataBase
  instanceVariableNames: 'employees'
classVariableNames: ''
poolDictionaries: ''
category: 'Employee-Example'

initialize
  employees := Dictionary new

! !
add: anEmployee
  (anEmployee isKindOf: Employee)
  ifTrue: [
    employees
      at: (anEmployee empNum)
      put: anEmployee.
      ^self
  ]
  ifFalse: [
    self error: 'You can add employees only!!'
  ]

! !

How does an Employee recognise the isKindOf: message?
Object subclass: #EmployeeDataBase

... ! !

**do:** aBlock

  employees do: aBlock.

! !

**findSkill:** aSkill

| empsWithSkill |
empsWithSkill := Set new.
employees do: [
  :emp |
  (emp hasSkill: aSkill)
  ifTrue: [empsWithSkill add: emp]].
^empsWithSkill.

! !
Testing the Employee Example Using SUnit

TestCase subclass: #EmployeeTest
instanceVariableNames: 'e1 e2 e3 db'

... ! !EmployeeTest methodsFor: 'testing'

setUp

   db := EmployeeDataBase new.
e1 := Employee new.
e1 name: 'George Blogs'; empNum: 1021; salary: 2000;
   addSkill: 'Smalltalk'; addSkill: 'C++'.
e2 := Employee new.
e2 name: 'Jane Lee'; empNum: 1054; salary: 2250;
   addSkill: 'Lisp'; addSkill: 'C++'.
e3 := Employee new.
e3 name: 'Mike Mendez'; empNum: 1088; salary: 1950;
   addSkill: 'Cobol'.
Trying the Employee Example (cont.)

EmployeeTest methodsFor: 'testing'

testAdd
  db add: e1.
  db add: e2.
  db add: e3.
  self assert:
    ((db findSkill: 'C++') size = 2)

• Running the test:
  – Open Test Runner from the Tools flap
  – Select Employee-Example category
  – Click the Run Selected button.
Time for a Change

• Two kinds of employees
  – Monthly:
    • Work the same way as employees did before
  – Hourly:
    • Rate
    • Set number of hours
    • Increase number of hours

• Extending a class:
  – Add new methods
  – Override methods of superclass
  – Extend superclass methods
    • Must use super
Object subclass: #Employee
 instanceVariableNames: 'empName empNum empSalary empSkills'
classVariableNames: ''
 poolDictionaries: ''
category: 'Employee-Example'

initialize

    empName := 'Steve'. empNum := 0.
    empSkills := Set new. ^self

!!

    name
    ^empName

!!

    name: aName
    empName := aName.

!!

    empNum
    ^empNum

...
Object subclass: #Employee

... 

Code identical to previous example was omitted.

salary: aSalary

  self subclassResponsibility.


salary

  self subclassResponsibility.


print

  Transcript show: ('name ', empName);cr .
  Transcript show: ('number: ', empNum asString);cr .
  Transcript show: ('Salary: ', self salary asString);cr.
  Transcript show: 'Skills:';cr.

  empSkills do:

    [:x | Transcript show: (' ', x asString);cr ].
Employee subclass: #MonthlyEmployee
  instanceVariableNames: 'monthlySalary'
  classVariableNames: ''
  poolDictionaries: ''
  category: 'Employee-Example'

initialize
  super initialize.
  monthlySalary := 0.
  ^self.

! !

salary
  ^monthlySalary

! !

salary: aSalary

! !
Employee subclass: #HourlyEmployee

instanceVariableNames: 'hourlyRate numberOfHours'

initialize

    super initialize.
    hourlyRate := 0.  numberOfHours := 0.
    ^self

rate: aRate
    hourlyRate := aRate

salary
    ^hourlyRate * numberOfHours

setHours: numHours
    numberOfHours := numHours

---

Same as Employee except that rate and hours instance variable and related method bodies are added.

Code is in HourlyEmployee.st
Inheritance

• Superclasses and subclasses form an inheritance hierarchy.
• New classes may be derived by generalization or specialization.
  – Generalization: a new superclass is created, containing the common features of its subclasses.
  – Specialization: a new subclass is created, adding features to its superclass, or redefining some methods.
• Generalization normally results in Abstract Classes. They define a common protocol for their subclasses, but can not be used to generate useful instances, since they have only partial implementation.
• When a subclass redefines an inherited method it is **overriding** the implementation of the superclass.
  – The method of the superclass may be called in the subclass through the keyword `super`.

• Ideally, the subclass should be implemented without knowing internal details of the superclass.

• The subclass must respect the **encapsulation** of its superclass.
TestCase subclass: #EmployeeTest2

instanceVariableNames: 'db e1 e2 e3'

setUp

  db := EmployeeDataBase new.
e1 := HourlyEmployee new.
e1 name: 'George Blogswell'; empNum: 1021;
  rate: 10.0; setHours: 160;
  addSkill: 'Smalltalk'; addSkill: 'C++';
e2 := MonthlyEmployee new.
e2 name: 'Jane Lee'; empNum: 1054; salary: 2250;
  addSkill: 'Lisp'; addSkill: 'C++'.
e3 := MonthlyEmployee new.
e3 name: 'Mike Mendez'; empNum: 1088; salary: 1950;
  addSkill: 'Cobol'.

Code is in EmployeeTest2.st
testAdd
  db add: e1.
db add: e2.
db add: e3.
self assert: ((db findSkill: 'C++') size = 2)

!!