UML Design Format Part 2B – Craig Scott
Assign 3) Class Diagram

Classes

- employeeRecord
- Employee
  - a.) hourlyEmployee
  - b.) salEmployee
  - c.) pieceEmployee
- Employee Handler
- payCalculator(main)

Class Associations

employeeHandler --- 1:1 (contains) Employee[]
employeeHandler --- 1:2 (contains) Employee
hourlyEmployee --- (inherits) --- (1)Employee
salEmployee --- (inherits) --- (1)Employee
pieceEmployee --- (inherits) --- (1)Employee
payCalculator --- 1:1 (Uses) --- employeeHandler
payCalculator --- 1:m (Uses) --- Employees

Class Attributes

employeeRecord

(+)String firstName,lastName
(+)float hoursWorked, payRate, grossPay, taxAmount, netPay, unitsSold,ppu,sal

Employee

(#) employeeRecord rec = new employee()
(-)double tax
(+) String firstName
(+)void setFirstName(String n)
(+) String lastName()
(+) void setLastName(String n)
(+) float hoursWorked()
(+) void setHoursWorked(float n)
(+) float payRate()

(+) void setPayRate(float n)
(+) float grossPay()
(+) void setGrossPay(float n)
(+) float taxAmount()
(+)void setTaxAmount(float n)
(+)float netPay()
(+)void setNetPay(float n)
(+)double getTax()
(+) void addHours(int n)
(+) void addHours(float n)
(+) String convertString()
(+) String convertStringStruct()

hourlyEmployee

(+)hourlyEmployee()
(+)hourlyEmployee(float hrs, float pay)
(+)hourlyEmployee(hourlyEmployee e)
(+)convertString()
(+)getHrs(),getPay()
(+)setHrs(),setPay()

salEmployee

(+)salEmployee()
(+)salEmployee(float sal)
(+)salEmployee(salEmployee s)
(+)convertString
(+)getSal()
(+)setSal()

pieceEmployee

(+)pieceEmployee()
(+)pieceEmployee(float unit, float ppu)
(+)pieceEmployee(pieceEmployee m)
(+)convertString
(+)getUnit(),getPPU()
(+)setUnit(),setPPU()

employeeHandler

(-) Employee employeeDB[], averages, totals
(-) Boolean isUsed[] //parallel array to tell if spot is open
(+)employeeHandler()
(+)employeeHandler(Employee[] n)
(+)employeeHandler(Employee n)
(+) void setEmps(Employee[] n)
(+)Employee getAverage()
(+)void setAverage(Employee n)
(+)Employee getTotal()
(+)Employee setTotal()
(+)void calculateTotals()
(+)void calculateAverages()
(+)float total (Employee[] emp, String type)
(+)float averages (Employee[] emp, String type)
(+) float grossPay(float payrate, float hoursWorked)
(+)float taxAmount(float grosPay, float tax)
(+)float netPay (float grossPay, float taxAmount)
(+)void alphabetize
(+)int getLength() //gets length of array that holds employees 5 employes but 10 spots array, returns 5
(+)void add(Employee e)// Add next open spot at end of array
(+)void add (int index, Employee e)// Replaces index with employee
(+)void delete(int index) // deletes object at location index
(+) void sortNull()// shuffles employees to take open spots in array ie. Delete employee shuffles
(+)Boolean[] getBool()

Use Case Model (employeeHandler)

Normal Case Scenario 1:

1. Manager inputs employees using add() function entering basic information
2. Manager deletes employees that have been fired using delete() function
3. System automatically sorts employees alphabetically
4. Manager runs reports with calculateTotal & calculateAverage
5. System gives total report in output file

Abnormal Scenario 1:

1. Manager inputs non-valid values for employeeHandler
   1a) employeeHandler recognizes this and asks for re entry
2. Manager recognizes error and puts proper employee integers for last name, first name, pay rate and hours worked.
3. The report is printed to the output file
4. Manager leaves application

State Diagram employeeRecord
None – Container

State Diagram Employee
---Employee() -→ s0
---Employee(String n, Strings, float a, float d) → s0
---Employee(Employee n) →
s0 – calcGross() → s1
s0 – calcTaxes() → s_{err}
s0 – calcNet() → s_{err}

s1 – calcTaxes() →
s1 – calcNet() →

s2 – calcNet() → s3

s3 --> employeeContainer.add(Employee) $s_{terminal}$

State Diagram employeeContainer
--- employeeContainer() → s0
--- employeeContainer(Employee[] n) → s0
--- employeeContainer(employeeContainer s) →

s0 – add(Employee) – s1 (n times with integer n>0);

s0 – add(Employee) →s_{error} (add incorrectly)

s1 – remove(Employee) →s2(n times where n = container.length)
s1 – remove(Employee) →s_{error} (removing employees in empty array)

s2 – alphabetize() → $s_{terminal}$

Main Pseudo-Code

```
payCalculator{

    main{
        Add employee1;
        Add employee2;
        Add employee3;
        Add employee4;
        Add employee5;
        Calculate averages
        Calculate totals;
        Alphabetize;
        Display data→Print report to file;
    }
```