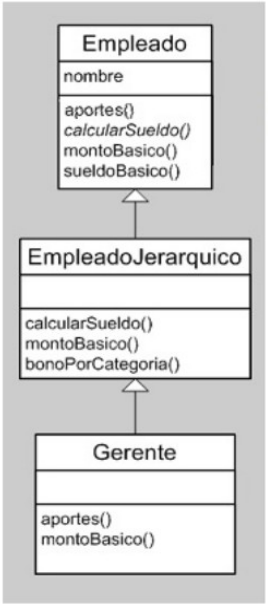
Orientación a Objetos 1 - 2012
Práctica 8

Ejercicio 1

Dada la jerarquía mostrada a la izquierda de la Figura 1, cuya implementación se muestra en la tabla de la derecha, indique qué retornan las siguientes expresiones:



Empleado	EmpleadoJerarquico	Gerente
sueldoBasico ^self montoBasico - self aportes	**calcularSueldo** ^super sueldoBasico + self bonoPorCategoria	**aportes** ^self montoBasico * 0.05
montoBasico ^ 700	**montoBasico** ^ 850	**montoBasico** ^1000
aportes ^10	**bonoPorCategoria** ^ 200	

Figura 1

```
| gerente |
unObjeto := Gerente new.
a) gerente aportes
b) gerente calcularSueldo
```

Ejercicio 2

Considere una computadora con algunos de sus componentes y sus correspondientes características:

- Memoria (cantidad de GB)
- Procesador (tipo de juego de instrucciones del procesador (x86, ARM, Itanium), Ghz de velocidad)
- Disco Rígido (GB de capacidad)

Cuando se crea una instancia de computadora debe tener todos sus componentes correctamente inicializados.

Orientación a Objetos 1 - 2012
Práctica 8

Cuando la computadora recibe el mensaje on debe reportar su configuración, es decir cuanta memoria tiene, de que procesador se trata y su velocidad, y la capacidad del disco rígido. Nuestra computadora usará el Transcript para reportar esta información.
Considere además que la Memoria y el Disco Rígido deben proveer un método (con implementación vacía ya que en la siguiente práctica implementará el cuerpo de estos métodos) para almacenar Strings que se le envían.

Tareas:

1. Realice el diagrama de clases.
2. Realice el diagrama de secuencia donde se muestra lo que ocurre cuando el mensaje `on` es enviado a la computadora.
3. Implemente en Smalltalk.

Ejercicio 3

Se desea extender el ejercicio del `ToDoItem` comenzado en la prácticas anteriores, utilizando una ventana diferente a la empleada en esa práctica.
Tareas:

1. Cargue el parcel provisto por la Cátedra.
2. Mediante el siguiente script compruebe que su interface funciona.

```
|anItem changer|
anItem := ToDoItem new.
anItem text: 'Comprar verduras'.
changer := ToDoChangerView new.
changer item: anItem.
changer open.
anItem inspect.
```

El resultado debe ser semejante al que muestra la Figura 2.

Figura 2

Orientación a Objetos 1 - 2012
Práctica 8

El boton Accept cambia el texto del ToDoItem por el contenido del campo de texto superior. Edite el texto, oprima el botón Accept, y luego el botón Close. Verifique mediante el inspector que el texto haya cambiado en la instancia de ToDoItem. Para comprender cómo se construyó la ventana, siga el tutorial correspondiente a esta práctica.

Ejercicio 4

Indique qué efecto tienen en una `OrderedCollection` los mensajes: `#add:`, `#remove:`, `#size`, `#last`, `#first`, `#includes:` y `#at:put:`. Clasifíquelos según si el objetivo principal del mensaje es modificar al objeto receptor u obtener información. ¿Los mensajes que modifican el objeto receptor retornan algo?

Ejercicio 5

Para cada uno de los mensajes enunciados abajo: lea los comentarios, analice y compare (codificando ejemplos si lo cree conveniente) su comportamiento para cada una de estas clases: `OrderedCollection`, `SortedCollection`, `Array`, `Dictionary`, `Bag` y `Set`.

```
#add:              #at:              #at:put:
#size              #do:              #detect:
#select:           #collect:         #reject:
#inject:into:      #includes:        #isEmpty
```

Responda a las siguientes preguntas:

1. ¿Es posible que algunos mensajes no sean aplicables para algunas colecciones? Por ejemplo, ¿Se le puede enviar el mensaje `#add:` a un `Array`? ¿Y a un `Set`? ¿Se le puede enviar el mensaje `#at:` y `#at:put:` a un `Set`?
2. En respuesta al mensaje `#select:`, ¿qué retorna un `Array`? ¿Y un `Dictionary`? ¿Y una `SortedCollection`?
3. En respuesta al mensaje `#size:`, ¿qué retorna un `Array` creado con `Array new:10`? ¿Qué retorna una `OrderedCollection` creada con `OrderedCollection new:10`?
4. ¿Cómo se elimina un elemento de un `Array`? ¿Es posible?
5. ¿Como se averigua la posición de un elemento en un `Array`? ¿Y la del primer elemento que cumple una condición? ¿Es posible hacerlo en un `Set`? .
6. Indique la diferencia entre `#detect:` y `#detect:ifNone:`. ¿Para qué sirve el bloque que se envía como parámetro en `#ifNone:`?
7. ¿Cómo crea una `SortedCollection` para contener instancias de `String` ordenadas por tamaño? ¿Cómo crea una `SortedCollection` para contener instancias de `String` ordenadas alfabéticamente?

Orientación a Objetos 1 - 2012
Práctica 8

8. ¿Cómo consigue los elementos en un `Array` eliminado las repeticiones?
9. ¿Cuál es el problema con la siguiente expresión si `col` es un `Set` con elementos? ¿Y si fuera una `OrderedCollection`?

```
col do: [ :each | col remove: each]
```

10. ¿Cuál es el efecto de enviar el mensaje `#add:` con `nil` como parámetro a una `OrderedCollection`? ¿Por qué?
11. ¿Qué diferencia hay entre los mensajes `#includes:` y `#contains:`?

Ejercicio 6: ToDoItemManager

Extienda la solución del ejercicio de la práctica 3 agregándole a las instancias de `ToDoItem`, un `deadline`, que representa la fecha limite para la cual la tarea debería ser completada. Además, implemente la clase `ToDoItemManager`. Una instancia de `ToDoItemManager` debe responder al menos los siguientes mensajes:

```
#toDoItems
   "Retorna la colección de tareas"

#addToDoItem: aToDotItem
   "Agrega una nueva tarea a la colección"

#removeToDoItem: aToDotItem
   "Elimina una tarea de la colección"

#pendingTasks
   "Retorna todas las tareas que no están completas"

#orderByPriority
   "Devuelve una colección con las tareas pendientes y ordenadas por
   prioridad"

#orderByDeadline
   "Devuelve una colección con las tareas pendientes y ordenada por el
   deadline"
```