Trabajo Práctico 1 – Repaso y entrada en calor

Objetivo: Revisaremos errores comunes antes de comenzar con los temas propios de Objetos 2. Comenzaremos a utilizar UML y diseño con Diagrama de Clases UML.

Importante: durante la clase práctica del **Viernes 18 de Marzo**, se hará la corrección de un ejercicio de este TP a realizar en clase.

Ejercicio 1 – Evaluación de protocolos de una clase

Sea la clase Rectángulo con 4 variables de instancia (esquinaSuperiorIzquierda, esquinaSuperiorDerecha, esquinaInferiorIzquierda y esquinaInferiorDerecha). Elija uno de los protocolos presentados a continuación y justifique su elección. Recuerde que el protocolo es el conjunto de mensajes que entiende una clase o tipo.

Opción 1)

```
Class Rectangulo>>new
Rectangulo>>esquinaSuperiorIzquierda: unPunto
Rectangulo>>esquinaSuperiorDerecha: unPunto
Rectangulo>>esquinaInferiorIzquierda: unPunto
Rectangulo>>esquinaInferiorDerecha: unPunto
```

Opción 2)

```
Class Rectangulo>>newEnOrigenEsquinaSuperiorIzquierda: unPunto
                                    alto: unNumero ancho:unNumero
Rectangulo>>reubicarEsquinaSuperiorIzquierdaEn: unPunto
Rectangulo>>ancho: unNumero
Rectangulo>>alto: unNumero
```

Ejercicio 2 – Distribución de Responsabilidades

En una oficina hay un jefe que tiene un secretario el cual administra un fichero. Dadas las siguientes implementaciones seleccione la mejor alternativa y justifique.

Opción 1)

```
Jefe>>trabajarConFicha: unaFicha
      self secretario fichero buscar: unaFicha.
```

Opción 2)

```
Jefe>>trabajarConFicha: unaFicha
      self secretario buscharEnFichero: unaFicha
Secretario>>buscarEnFichero: unaFicha
      self fichero buscar: unaFicha
```

Ejercicio 3 - Encapsulamiento

a) Defina la clase Persona y modélela en Smalltalk. Una persona tiene un nombre y una fecha de nacimiento, por lo que debe ser posible pedirle su nombre, fecha de nacimiento y edad.

b) Un objeto cualquiera que le pide la edad a una Persona, ¿conoce cómo ésta calcula u obtiene tal valor? ¿Cómo se llama el mecanismo de abstracción que permite esto?
c) Agregue a la clase Persona definida anteriormente el método #< que recibe como parámetro otra persona y retorna true en caso de que el receptor sea menor en edad que el parámetro, false en caso contrario.
d) Agregue a la clase Persona un método (de clase) de creación llamado:
conNombre: unString nacidaEl: unaFecha
que recibe como parámetros el nombre y la fecha de nacimiento de la persona a crear, crea una nueva instancia de persona y la retorne inicializada con los valores de los parámetros.
e) Utilice un objeto de la clase SortedCollection para guardar 5 nuevas Personas ordenadas de acuerdo a su nombre.
f) Utilice otro objeto de la clase SortedCollection para guardar las mismas 5 Personas ya creadas, pero ahora ordenadas por fecha de nacimiento.

Ejercicio 4 - Polimorfismo

a) Defina la clase Mascota y modélela en Smalltalk. Una mascota tiene un nombre y una raza (String), por lo que debe ser posible pedirle tales datos.
b) Instancie dos Personas, dos Mascotas e inserte los cuatro objetos en una colección. Itere sobre la colección e imprima un listado con los nombres de todos los objetos de la misma.
c) Durante la iteración, ¿fue necesario distinguir si el objeto era una Persona o una Mascota para poder imprimir su nombre? ¿Cómo se llama el mecanismo de abstracción que permite esto?

Ejercicio 5 – Equipos

a) Defina la clase Equipo y modélela en Smalltalk. Un equipo tiene un nombre y un conjunto de integrantes (que son instancias de Persona).
b) Un Equipo debe saber responder su nombre y el promedio de edad de sus integrantes.
c) Instancie un Equipo, instancie 5 Personas y agregue éstas a aquél. Pida al equipo el promedio de edad de sus integrantes e imprima el resultado devuelto.

Ejercicio 6 – Más polimorfismo

Existen *cuentas bancarias* que pueden ser de tipo *caja de ahorro* o *cuenta corriente*. A continuación se presentan diferentes alternativas en Smalltalk que intentan resolver la extracción. Indique los defectos de cada una de las opciones.

Opción 1)

```
CuentaBancaria>>extraer:unMonto
|rojo|
   self tipo = 'cuentacorriente'
      ifTrue:[rojo:= self rojoPermitido.]
      ifFalse:[rojo := 0.]
  (self saldo + rojo >= unMonto)
      ifTrue: [self saldo: self saldo - unMonto].
```

Opción 2)

```
CuentaBancaria>>extraer:unMonto
    |rojo|
    self class = CuentaCorriente ifTrue:[rojo:= self rojoPermitido.]
                                 ifFalse:[rojo := 0.]
```

```
    (self saldo + rojo >= unMonto)
            ifTrue: [self saldo: self saldo - unMonto].
```

Donde CuentaCorriente es subclase de CuentaBancaria.

Opción 3)

```
CuentaBancaria>>extraer:unMonto
    ^self subclassResponsibility
CajaDeAhorro>>extraer:unMonto
    (self saldo >= unMonto)
         ifTrue: [self saldo: self saldo - unMonto].
CuentaCorriente>>extraer:unMonto
    (self saldo + self rojoPermitido >= unMonto )
         ifTrue: [self saldo: self saldo - unMonto].
```

Opcion 4)

```
CuentaBancaria>>extraer:unMonto
    (self chequearSaldoParaExtraccion:unMonto)
         ifTrue: [self saldo: self saldo - unMonto].
CajaDeAhorro>>chequearSaldoParaExtraccion:unMonto
    ^self saldo >= unMonto
CuentaCorreinte>>chequearSaldoParaExtraccion:unMonto
    ^self saldo  + self rojoPermitido >= unMonto
```

Ejercicio 7 – Accessors

En el libro de Kent Beck Smalltalk Best Practices Patterns (que se encuentra disponible en el sitio de la materia) se discute la utilización de accessors.
Lea las secciones Direct Variable Access hasta Collection Accessor Method.

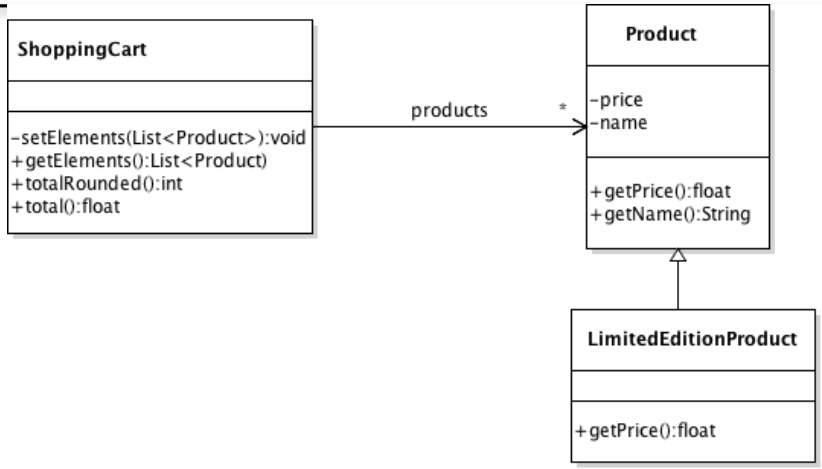
Responda las siguientes preguntas:

1. ¿Qué significa acceder directamente a las variables? Ejemplificar.
2. ¿Qué significa acceder indirectamente a las variables? Ejemplificar.
3. Según Kent Beck, ¿cuando utilizaría cada una de las alternativas?[1]
4. ¿Cómo proporciona acceso a variables que referencian a una colección?

Ejercicio 8 - UML

Escriba en lenguaje Smalltalk todo aquello que puede establecer a partir del siguiente diagrama de clases UML.

[1] Es particularmente importante en esta respuesta que se esfuercen en realizar una argumentación solida. Noten que deben trasladar las consideraciones, en este caso, de Kent Beck.


ShoppingCart
-setElements(List<Product>):void
+getElements():List<Product)
+totalRounded():int
+total():float
products
*
Product
-price
-name
+getPrice():float
+getName():String
LimitedEditionProduct
+getPrice():float