

Orientación a Objetos 2

Parcial Miércoles 6 de Junio de 2018

Ejercicio 1: Organización de comisiones

Como es bien sabido, la organización de inscripciones en las comisiones de objetos 2 no es sencilla. Las comisiones tienen un límite de inscriptos que se llama cupo. Si un alumno se inscribe a una comisión con cupo, situación a la que llamamos "**Abierta con cupo**", queda oficialmente inscripto, es decir tiene un lugar en la comisión. Cuando las comisiones se completan, la cátedra considera esta situación como "**Abierta completa**" para lo cual se ideó un sistema de lista de espera donde se anota a los alumnos, pero no se les asegura la inscripción ya que depende de que los alumnos oficialmente inscriptos decidan des-inscribirse. A veces las listas de espera que se generan son demasiados grandes, entonces la cátedra decide cerrar las posibilidades de inscribirse a esa comisión y no permite agregar más personas a la lista de espera, a esta situación la llama "**Cerrada**". Es por eso que se quiere realizar un sistema que permita controlar las inscripciones de los alumnos en las comisiones con el siguiente protocolo:

Clase >> crear:unNombreDeComision cupo:unNumero enEspera: otroNumero

"Crea una comisión con nombre unNombreDeComisión que tiene como cantidad de personas que pueden inscribirse a unCupo y en lista de espera podrán estar otroNumero de personas como máximo. Asuma que unNumero y otroNumero son mayores que 0"

>> inscribir:unaPersona

"retorna true si la persona queda oficialmente inscripta, caso contrario retorna false siguiendo los detalles de la Tabla 1."

>> desinscribir: unaPersona

"elimina de la lista de inscriptos oficialmente a unaPersona (si es que existe en la lista) y siguiendo los detalles de la Tabla 1, si unaPersona no está en la lista de inscriptos no hace nada."

>> imprimir

"Imprime un string siguiendo los detalles de la siguiente Tabla"

	Abierta con cupo	Abierta completa	Cerrada
inscribir: unaPersona	Inscribe oficialmente a la persona en la comisión y retorna true. Si el cupo se completa pasa a estar "Abierta completa"	Anota a la persona en lista de espera y retorna false. Si se llenó pasa a estar "Cerrada".	Retorna false. No hace nada más.
desinscribir: unaPersona	Elimina de la lista de oficialmente inscriptos a la unaPersona, si es que existe.	Elimina a la persona y mueve a la persona que está en primer lugar en la lista de espera a la lista de inscriptos. Si no hay personas en lista de espera pasa a estar "Abierta con cupo"	Elimina de la lista de oficialmente inscriptos a unaPersona (si existe) y mueve a la persona que está en el primer lugar de la lista de espera. Pasa a estar "Abierta con cupo". <i>completar</i>
imprimir	Retorna un String con la forma "Comision abierta, aun quedan X cupos" donde X es la cantidad de cupos disponibles.	Retorna un string con la forma "Comision abierta pero completa (X alumnos). La lista de espera tiene actualmente Y personas." donde X es el cupo de la comisión e Y la cantidad de personas en lista de espera.	Retorna un String con siguiente estructura: "Comision cerrada. X alumnos inscriptos. Y alumnos quedaron en lista de espera." Donde X es el cupo de la comisión e Y la cantidad en lista de espera.

Actividades

Para este ejercicio ud deberá:

1. Realizar el diseño de una solución orientada a objetos con un diagrama UML de clases.
2. Si utilizó patrones de diseño indique cuáles y también indique los participantes de esos patrones en su solución. (Según el libro de Gamma et al.)
3. Programe en Smalltalk la totalidad de su solución.

Ejercicio 2: Refactoring

A continuación, se presenta la implementación de un método que posee una serie de malos olores. Enumere los malos olores detectados, realice los refactorings necesarios y enumérelos.

>>pEnEspr

"retorna la persona que está en lista de espera de acuerdo al orden del modo"

```
| firstEntregas firstPromedio x |
modo = #maximasEntregas
    ifTrue: [ "ordena por cantidad de entregas"
        firstEntregas := listaDeEspera
        asSortedCollection: [ :a :b | a entregasHechasEnObjetos1 size <= b
entregasHechasEnObjetos1 size ] first.
        listaDeEspera remove: firstEntregas.
        x := firstEntregas ].
modo = #mejorPromedio
    ifTrue: [ "ordena por promedio"
        firstPromedio := listaDeEspera
        asSortedCollection:
            [ :a :b |
                (a entregasHechasEnObjetos1 collect: [ :e | e nota ] average)
                <= (b entregasHechasEnObjetos1 collect: [ :e | e nota ] average) ] first.
        listaDeEspera remove: firstPromedio.
        x := firstPromedio ].
```

^ x

Importante: Lea detenidamente el enunciado. Resuelva solo lo pedido. Sea prolj@ y clar@ en su solución.

Preste atención. Entregue todas sus hojas numeradas indicando en cada una el número de hoja sobre el total. En la primera hoja ponga en letra clara su nombre, apellido y legajo.