Introducción a la Programación Orientada a Objetos

DCIC – UNS

APELLIDO Y NOMBRE: …… **LU:** ……...

CANTIDAD DE HOJAS ENTREGADAS (SIN ENUNCIADO): ……

IMPORTANTE: SE EVALUARÁ LA **CORRECTITUD, LEGIBILIDAD Y EFICIENCIA** DE LAS SOLUCIONES.

La interpretación del enunciado es parte del examen. **NO RESUELVA LOS EJERCICIOS EN EL ENUNCIADO.**

1. Defina el modelo computacional de la programación orientada a objetos
2. Dado el siguiente diagrama para la clase **genérica** ColeccionOrdenada:

ColeccionOrdenada
<<Atributos de instancia>> T [] Elemento cant: entero
ColeccionOrdenada(max: entero)
<<Consultas>> estaElem(e:Elemento):boolean

   a. Implemente el método recursivo estaElem que busca el elemento ***e*** usando la estrategia de búsqueda binaria.

   b. Implemente la clase Elemento con los servicios demandados por estaElem de la clase ColeccionOrdenada.

   c. Explique cómo los conceptos de herencia, polimorfismo, vinculación dinámica y clases abstractas permiten soportar la definición de una clase genérica como ColeccionOrdenada.

   d. Implemente la clase Articulo que extiende a Elemento

Articulo
<<Atributos de instancia>> codigo:entero costo: real cantidad:entero
Articulo(c: entero, s:real,n:entero) equals(a:Elemento):boolean mayor(a:Elemento):boolean obtenerPrecio(p:real)

   obtenerPrecio retorna el costo por el porcentaje p

   equals y mayor comparan por código

   e. Implemente la clase Stock que extiende a ColeccionOrdenada

Stock
Stock (max: entero)
<<Consultas>> cantRango(i,f:real): entero

   cantRango computa cuantos artículos del stock tienen el precio mayor que i y menor que f.

3. En un videojuego cada uno de los personajes tiene una misión asignada. Cada misión está formada por una secuencia de acciones. Algunas acciones requieren una calificación especial. Toda acción tiene una duración que se calcula de manera diferente según sea calificada o no.

El modelo parcial del problema es:

Mision
<<Atributos de instancia>> m : [] Accion cant: entero
Mision(max:entero)
<<Consultas>> **subsecuenciaMision (s:String,t:entero): Mision**

Accion
Accion(n : String, d : entero)
<<Consultas>> obtenerNombre():String duracion() esIgual(a:Accion):boolean esMayor(a:Accion):boolean

AccionCalificada
AccionCalificada(n : String, d,r : entero)
<<Consultas>> duracion()

Las clases Accion y AccionCalificada brindan los métodos triviales. AccionCalificada extiende a Accion

El método subsecuenciaMision retorna, si existe, una subsecuencia de acciones dentro de la misión que recibe el mensaje, que comienza con una acción de nombre s y continúa con la **mayor cantidad de acciones posibles**, que tengan una duración total de **t** unidades de tiempo. Si hay más de una subsecuencia con estas características, retorna la primera. Si no hay ninguna, retorna nulo.
Por ejemplo si el nombre de la acción s es Caminar y t es 40 y la misión está formada por las acciones:

nombre	Duración
Saltar	20
Trepar	20
Correr	12
Caminar	10
Saltar	25
Trepar	30
Caminar	20
Trepar	20
Comer	10
Beber	12
Caminar	8
Caminar	**10**
Saltar	**14**
Luchar	**16**
Saltar	11

secuenciaMision debe retornar la secuencia

Caminar	10
Saltar	14
Luchar	16

Porque la secuencia comienza con Caminar, tiene una duración total de 40 unidades de tiempo y es la **subsecuencia más larga** que comienza con Caminar y tiene duración 40.
Implemente la clase Mision de acuerdo a la especificación dada.