Paradigmas de Programación
Delegación y Responsabilidad

Ejercicio 1 – Trenes y Depósitos

Una administradora ferroviaria necesita una aplicación que le ayude a manejar las formaciones que tiene disponibles en distintos depósitos.
Una formación es lo que habitualmente llamamos “un tren”, tiene una o varias locomotoras, y uno o varios vagones. Hay vagones de pasajeros y vagones de carga.
En cada depósito hay: formaciones ya armadas, y locomotoras sueltas que pueden ser agregadas a una formación.

De cada vagón de pasajeros se conoce el largo en metros, y el ancho útil también en metros. La cantidad de pasajeros que puede transportar un vagón de pasajeros es:

- Si el ancho útil es de hasta 2.5 metros: metros de largo * 8.
- Si el ancho útil es de más de 2.5 metros: metros de largo * 10.

P.ej., si tenemos dos vagones de pasajeros, los dos de 10 metros de largo, uno de 2 metros de ancho útil, y otro de 3 metros de ancho útil, entonces el primero puede llevar 80 pasajeros, y el segundo puede llevar 100.
Un vagón de pasajeros no puede llevar carga.

De cada vagón de carga se conoce la carga máxima que puede llevar, en kilos. Un vagón de carga no puede llevar ningún pasajero.
No hay vagones mixtos.

El peso máximo de un vagón, medido en kilos, se calcula así:

- Para un vagón de pasajeros: cantidad de pasajeros que puede llevar * 80.
- Para un vagón de carga: la carga máxima que puede llevar + 160 (en cada vagón de carga van dos guardas).

De cada locomotora se sabe: su peso, el peso máximo que puede arrastrar, y su velocidad máxima. P.ej. puedo tener una locomotora que pesa 1000 kg, puede arrastrar hasta 12000 kg, y su velocidad máxima es de 80 km/h. Obviamente se tiene que arrastrar a ella misma, entonces no le puedo cargar 12000 kg de vagones, solamente 11000; diremos que este es su “arrastre útil”.

Modelar la situación descripta de acuerdo al paradigma de objetos, escribiendo el código en lenguaje Wollok, de manera de poder saber:

1. El total de pasajeros que puede transportar una formación.
2. Cuántos vagones livianos tiene una formación; un vagón es liviano si su peso máximo es menor a 2500 kg.
3. La velocidad máxima de una formación, que es el mínimo entre las velocidades máximas de las locomotoras.
4. Si una formación es eficiente; es eficiente si cada una de sus locomotoras arrastra, al menos, 5 veces su peso (el de la locomotora misma).

5. Si una formación puede moverse. Una formación puede moverse si el arrastre útil total de las locomotoras es mayor o igual al peso máximo total de los vagones.
6. Cuántos kilos de empuje le faltan a una formación para poder moverse, que es: 0 si ya se puede mover, y (peso máximo total de los vagones – arrastre útil total de las locomotoras) en caso contrario.
7. Dado un depósito, el conjunto formado por el vagón más pesado de cada formación; se espera un conjunto de vagones.
8. Si un depósito necesita un conductor experimentado.
   Un depósito necesita un conductor experimentado si alguna de sus formaciones es compleja. Una formación es compleja si: tiene más de 20 unidades (sumando locomotoras y vagones), o el peso total (sumando locomotoras y vagones) es de más de 10000 kg.
9. Agregar, dentro de un depósito, una locomotora a una formación determinada, de forma tal que la formación pueda moverse.
   Si la formación ya puede moverse, entonces no se hace nada.
   Si no, se le agrega una locomotora suelta del depósito cuyo arrastre útil sea mayor o igual a los kilos de empuje que le faltan a la formación. Si no hay ninguna locomotora suelta que cumpla esta condición, no se hace nada.

O sea: indicar qué clases se necesitan, qué variables de instancia se necesitan en cada clase, qué mensajes van a entender las instancias de cada clase, y escribir los métodos correspondientes.
Para cada punto, indicar a qué objeto se le pide lo que se indica, con qué mensaje, qué parámetros, y qué devuelve.
Para el punto 8, indicar en qué otros objetos delega el responsable de hacer lo que se pide, y qué delega (indicar lo que se delega en castellano). Si hay una cadena de delegaciones (al objeto 1 le piden algo, entonces delega algo en el objeto 2, y el objeto 2 para hacer lo que le pidió el 1 tiene que delegar otra cosa en otro objeto 3) indicarla.

Ejercicio 2 – Mascota Virtual

Modelar una mascota virtual, onda Tamagotchi, incluyendo los mensajes correspondientes a las acciones de comer y jugar, y la pregunta acerca de si puede jugar o no.

También hay que poder conocer el nivel de contenta de una mascota, que es un número entero mayor o igual que 0, donde a mayor nivel, más contenta está la mascota.

Una mascota puede estar aburrida, hambrienta o contenta; y su comportamiento depende de en qué estado esté. Veamos:

Cuando una mascota come, pasa lo siguiente:

- Si está hambrienta, se pone contenta.
- Si está contenta, su nivel se incrementa en una unidad.
- Si está aburrida, y hace más de 80 minutos que está aburrida, entonces se pone contenta.
- Si está aburrida desde hace 80 minutos o menos, entonces no le pasa nada, no cambia nada.

Cuando una mascota juega, pasa lo siguiente:

- Si está contenta, su nivel se incrementa en dos unidades.
- Si está aburrida, se pone contenta.
- No produce ningún efecto jugar con la mascota si está hambrienta.

Una mascota puede jugar si está contenta o aburrida, si está hambrienta no.

NO SE PUEDE CONSULTAR DE NINGUNA MANERA EL ESTADO ACTUAL DE LA MASCOTA.
Esto quiere decir que está **prohibido** hacer comparaciones del tipo ***`estado == "contento"`*** o cualquiera similar utilizando mensajes especiales.

Responder las siguientes preguntas:

1. Indique en palabras los pasos necesarios para incorporar un nuevo estado “Triste” en la mascota, de manera que quede listo para funcionar.
2. Indique cuál sería la prueba en un test similar para darles de comer a todas las mascotas que están dentro de una colección “mascotas”.

Ejercicio 3 – Manejo de Proyectos

Implementar, usando el paradigma de objetos, una solución que ayude a una organización a... organizarse.
En la organización se llevan a cabo distintos proyectos, y trabajan varias personas. Para cada proyecto hay que llevar a cabo tareas.

La solución debe incluir la posibilidad de:

- Asignar una tarea a una persona.
- Registrar que una persona trabajó en una tarea una cantidad de tiempo, medida en horas.
- Registrar que se terminó una tarea.
- Conocer las tareas pendientes de una persona.
- Conocer qué personas están trabajando actualmente en un proyecto. Una persona está trabajando en un proyecto si tiene asignada alguna tarea pendiente de ese proyecto.
- Conocer qué personas colaboraron con un proyecto. Una persona colaboró con un proyecto si tiene asignada alguna tarea finalizada de ese proyecto.
- Qué tareas no asignadas tiene un proyecto.
- El total de horas trabajadas en un proyecto.

Ejercicio 4 – impresiones

Etapa 1

Una red atiende pedidos de impresión de documentos de distintos usuarios, los resuelve mediante varios servidores de impresión que conoce, a su vez cada servidor conoce a una o varias impresoras.

Cada impresora tiene una carga máxima de peso de los documentos que soporta en su cola de impresión.

Hay dos tipos de documentos: libros (postscript) y archivos de imágenes (jpg). El peso de los libros es su cantidad de páginas / 10 (parte entera - floor), el de los archivos de imágenes es: 1 hasta 100K, 2 de 100K a 300K, 3 si tiene más de 300K.

Una impresora no puede imprimir un documento si su peso + el peso total de los documentos en su cola de impresión supera su carga máxima.

Además, una impresora puede o no imprimir fotos, y puede o no imprimir postscript.

Un servidor puede aceptar un documento si y sólo si alguna de las impresoras que maneja puede imprimirlo.

El objetivo de esta etapa es que el modelo soporte lo que se necesita para implementar los pedidos de impresión.
Ante algunos pedidos (p.ej. de administradores) hay que mostrarle la lista de servidores que pueden encargarse del trabajo, y cuando el usuario elige un servidor se envía el documento a imprimirse al servidor elegido.

Ante otros pedidos (p.ej. usuarios comunes) se muestra una pantalla de confirmación, y cuando el usuario confirma se elige un servidor al azar, y se le envía el documento para que lo imprima.

Por ahora no tener en cuenta el evento de "fin de impresión de un documento", eso es para el paso siguiente.

Etapa 2

Manejar la información sobre los documentos ya impresos y en cola de impresión, según estas reglas: Cuando se envía un documento a una impresora, se agrega a una cola de impresión, o sea va "al final de todo".
Cuando una impresora física termina con un documento, envía una señal que puede ser traducida como el envío de un mensaje sin parámetros a un objeto dentro de nuestro modelo.

Se pide:

- Saber qué documentos se están imprimiendo en la red.
- Poder subir un documento arriba de todo en la impresora que lo tenga en su cola, mediante un mensaje a la red. Esto hace que se convierta en el siguiente documento a imprimir en esa impresora, después de terminar el actual.
- Conocer, para un servidor, las impresoras que están ociosas.
- Saber qué documentos están pendientes de impresión.
- Conocer el peso total de los documentos que se están imprimiendo
    - En toda la red
    - En un servidor.
- Saber qué servidor tiene registrado un documento, nil si no está registrado.

Etapa 3

Manejar la posibilidad de que se conecten a la red empresas especializadas en impresión (p.ej. Taller 4), al mismo nivel que los servidores:

- Pueden recibir pedidos de impresión que encolan (se les envía electrónicamente, aunque esto no hace falta modelarlo, sólo el control)
- Envían una señal cuando terminaron de imprimir un documento, esta señal puede traducirse a un mensaje sin parámetros a un objeto.
- Se cuentan para los ítems de la etapa 2, excepto el de impresoras ociosas (la red no conoce la organización interna de la empresa), incluyendo el de poner un documento arriba de todo, en este caso es respecto de todos los documentos enviados a la empresa.

Etapa 4

Mejorar la asignación de documentos para los cuales el usuario no eligió servidor:

- Estos no se envían a empresas, sólo a servidores.
- Ee envían al servidor con menos carga en documentos pendientes de impresión.

Etapa 5

Manejar la posibilidad de agregar una impresora. Cada servidor puede aceptar una cantidad máxima de impresoras.
Cuando el administrador de la red consulta, le deben aparecer los servidores que pueden aceptar una impresora. Luego hay que avisarle al sistema cuando se conecta la impresora en un servidor.

Etapa 6 (heeeaaavyyyy)

que se pueda enviar a un servidor un trabajo = .zip con varios documentos, y que en realidad la impresora le avisa al servidor y es el servidor que conoce los trabajos terminados.