Mini-TP: Diseño
Partes

Normativa

Fecha de entrega: 18 de noviembre de 2018 a las 23:59 hs.
Normas de entrega: Ver **Inicio** en el sitio Web de la materia.
(`http://campus.exactas.uba.ar`)

1. Introducción

Este trabajo tiene los siguientes objetivos:

- Descomponer un problema de tamaño mediano en módulos con interfaces claramente definidas.
- Justificar la plausibilidad de implementación de cada módulo a través de la elección de una estructura de representación, acompañada de un invariante de representación y una función de abstracción[1].

2. Descripción del problema

Se quiere implementar un sistema cuyo objetivo es registrar la compra, venta y reparación de **partes** electrónicas e industriales. Suponemos que cada parte puede tener subpartes, las subpartes pueden tener subpartes, y así recursivamente. Las partes que no tienen subpartes se llaman partes **atómicas**. Una parte atómica puede encontrarse en buen estado o en estado defectuoso.

Cuando se registra la compra de una parte, se le asigna un identificador único a la parte adquirida, pero no se registra ningún detalle sobre su estado (bueno o defectuoso) y no se registra si la parte tiene o no tiene subpartes. Es decir, el sistema no debe reflejar la "realidad" sino el conocimiento que se tiene de la realidad.

Una parte puede ser subparte de una parte más grande, o puede estar **suelta**, si no es subparte de otra parte. Suponemos que solamente se compran y venden partes sueltas. (Por ejemplo, no es posible vender un condensador de flujo que compone una máquina del tiempo sin primero *desarmar* la máquina del tiempo).

Lxs empleadxs del emprendimiento se identifican por un número de legajo. En todo momento unx empleadx puede iniciar una **transacción** sobre una parte suelta P, en cuyo caso decimos que la transacción *afecta* a la parte P y transitivamente a todas sus subpartes. Si unx empleadx tiene una transacción abierta, puede realizar alguna de las siguientes acciones:

- Registrar que una parte P se encuentra compuesta por n subpartes $\{P_1, \ldots, P_n\}$. En ese momento el sistema debe asignarle un nuevo identificador a cada una de las n subpartes. La parte P debe estar afectada por la transacción de lx empleadx. Las nuevas partes $P_1, \ldots, P_n$ pasan también a estar afectadas por dicha transacción.
- Registrar que una parte atómica P se encuentra en buen estado o en estado defectuoso. Por ejemplo, si la parte se encontraba en estado defectuoso y pasa a estar en buen estado, la acción se considera una *reparación*. La parte P debe estar afectada por la transacción de lx empleadx.
- Si una parte P está suelta y no es atómica, se la puede desarmar en sus subpartes $\{P_1, \ldots, P_n\}$. Al desarmar una parte, esa parte deja de tener entidad en el sistema, y sus subpartes pasan a estar sueltas. La parte P debe estar afectada por la transacción de lx empleadx.

Las modificaciones que efectúa unx empleadx a lo largo de una transacción son invisibles para el resto del sistema. Cuando unx empleadx finaliza su trabajo sobre una parte, puede **confirmar** la transacción, en cuyo caso la transacción se cierra y los cambios se publican en el sistema. Unx empleadx también puede **cancelar** una transacción, en cuyo caso la transacción también se cierra pero los cambios no se reflejan. Además, en todo momento debe ser posible registrar en el sistema la compra y la venta de partes sueltas.

Observaciones:

- Unx empleadx puede tener a lo sumo una transacción abierta en cada momento.
- Dos transacciones distintas no pueden afectar a las mismas partes.
- No es posible vender partes que se encuentren afectadas por una transacción.

[1] en realidad esto va a ser aproximado, ver explicación más adelante

En todo momento debe ser posible conocer:

- La situación **global** del sistema: las partes del sistema, subpartes de cada parte, estado (bueno o defectuoso) de las partes atómicas.
- La situación **local** del sistema desde el punto de vista de cada empleadx. Es decir, debe ser posible responder las mismas preguntas que en el ítem anterior pero teniendo en cuenta también las modificaciones potencialmente realizadas por unx empleadx dentro de su transacción.
- Dadx unx empleadx, debe ser posible determinar si tiene una transacción abierta y cuáles son las partes afectadas por dicha transacción.

3. Enunciado

Se pide diseñar un conjunto de módulos para resolver el problema descripto arriba. Para ello se pide proponer un conjunto de clases de C++:

- Para cada clase propuesta, escribir un *header* (archivo `.h`) detallando la interfaz pública y la representación privada de la clase.
- Cada método en la interfaz pública debe detallar su pre y postcondición, en forma de comentario.
- La representación privada debe detallar el invariante de representación, en forma de comentario.
- En la representación privada se debe detallar qué es lo que representa desde el punto de vista abstracto. Aquí se espera que describa qué representa cada parte privada y se aclaren todos los detalles que sean necesarios para comprender como el módulo resuelve el problema propuesto (notar que no tiene sentido hablar formalmente de la función de abstracción, porque no contamos con una especificación formal).

Observaciones:

- **No** se pide que la precondición, la postcondición y el invariante de representación estén expresados en lenguaje formal, pero sí se espera que sean precisos.
- **No** se pide implementar la funcionalidad en un `.cpp`, pero sí se espera que sea posible implementar los métodos de la interfaz con la representación propuesta.
- Pueden utilizar las siguientes clases de la STL para los respectivos módulos básicos:

Módulo	**Clase**
Lista Enlazada	`std::list`
Pila	`std::stack`
Cola	`std::queue`
Vector	`std::vector`
Diccionario Lineal	`std::map`
Conjunto Lineal	`std::set`

4. Pautas de entrega

El **Mini-TP** se entregará por mail a la dirección `algo2.dc+minitp@gmail.com`. El trabajo práctico puede realizarse en grupos de hasta de 2 integrantes. El código desarrollado se entregará como un archivo comprimido en formato zip. El mail deberá tener como **Asunto** los números de libreta separados por punto y coma (;). Por ejemplo:

To: algo2.dc+minitp@gmail.com
From: alumno-algo2@dc.uba.ar
Subject: 102/09; 98/10
Adjunto: minitp.zip

Dentro del zip se deberá encontrar todo el código fuente del TP (archivos con extensión `.h`).