Trabajo práctico 3: **Lollapatuza**

Normativa

Límite de entrega: Sábado 1 de julio, 23:59hs. Segunda fecha de entrega: domingo 16 de julio.
Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.
(`http://campus.exactas.uba.ar`)
Versión: 1.0 del 16 de junio de 2023

El objetivo de este TP es implementar en C++ todos los módulos correspondientes al diseño presentado por su grupo en el TP2. El código que entreguen debería respetar el diseño propuesto en el TP2 de la manera más fiel posible. Obviamente se permite y se espera que corrijan todos los potenciales *bugs* que puedan llegar a encontrar en el diseño. Las implementaciones deben cumplir con las complejidades definidas en su solución del TP 2 (salvo mejoras que hayan encontrado), incluyendo las restricciones de complejidad establecidas en el enunciado del mismo.

Código producido

La resolución debe tener un archivo `.h` y `.cpp` por cada módulo del TP 2 (o eventualmente un archivo `.hpp` si se trata de un módulo paramétrico que se implemente con `templates`). Estos archivos deberán ubicarse en el directorio `src`, respetando el esqueleto disponible en la página de la materia.

Código de la cátedra y tests

Como parte del enunciado, la cátedra provee un **esqueleto de TP3**. El esqueleto tiene la misma estructura de directorio que los talleres, con un directorio `src` para los archivos fuente y un directorio `tests` para el código de los tests. Proveemos un conjunto de casos de test que deberán ser pasados con éxito. Los tests provistos por la cátedra no son necesariamente exhaustivos, por lo tanto, deben *escribir sus propios tests de unidad* para evaluar aquellos casos que no estén contemplados en los tests de la cátedra. Se exige para la aprobación del TP haber incluido *al menos 1 test propio*.

Los archivos que comienzan con el prefijo `fachada_` que se encuentran en el directorio `src` son provistos como parte del esqueleto del TP para poder utilizarlos en los tests. Deben completar estos archivos agregando instancias de las clases diseñadas por ustedes en la parte privada de cada clase e implementado los métodos de forma tal que utilicen la interfaz provista por sus propios módulos.

El archivo `src/tipos.h` define algunos tipos auxiliares y renombres de tipos. En particular, se define el struct `aed2_Puesto`. Este struct agrupa los parámetros que se necesitan para crear un puesto y sú único próposito es para facilitar los tests de la cátedra.

La adaptación de la interfaz de sus módulos a los requeridos en en las fachadas puede conllevar operaciones con un costo no inmediato (ej. copiar un conjunto a una lista, recorrer un diccionario, etc.). Los requisitos de complejidad a cumplir aplican solamente a las funciones de la interfaz de los módulos. Los costos asociados a la traducción de su interfaz a la nuestra no tienen restricciones.

Importante: sugerimos implementar la menor lógica posible en los archivos `Fachada`, y crear clases independientes que respeten el diseño hecho por ustedes en el TP1. Estas clases se llaman "Fachada" porque deberían delegar todos los mensajes que reciben a las clases diseñadas por ustedes. Así, la implementación de todos los métodos de las clases `Fachada_` debería ser breve (ej. una o dos líneas).

Módulos básicos

Pueden utilizar las siguientes clases de la STL de C++ para los respectivos módulos provistos por la cátedra:

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

Entrega

Para la entrega deben hacer `commit` y `push` de todo el esqueleto, incluyendo el código fuente (`*.cpp`, `*.h`, `*.hpp`), los tests, y el archivo `CMakeLists.txt` en el repositorio **grupal** en el directorio `tpg2/`. No incluir los archivos binarios generados por el compilador (`*.o`, `*.a`, `*.exe`) en el repositorio.

Fechas de entrega

La primera fecha de entrega es el sábado 1 de julio, 23:59hs. Intenten llegar a esta fecha con el mayor avance posible. Aprovechen las instancias de consultas para despejar dudas que puedan ir surgiendo.

La fecha de re-entrega es el domingo 16 de julio, 23:59hs. No hay más re-entregas luego de esa fecha.

Rúbricas

Agregamos a continuación rúbricas que exponen qué se espera de la entrega. Las mismas presentan una serie criterios con los que se evaluarán las producciones entregadas. En términos generales, se considera que entregas con soluciones que solo logren los criterios parcialmente deberán ser reentregados con correcciones en estos aspectos en particular.

Por ser criterios generales, pueden no cubrir todos los detalles relacionados con este enunciado. No obstante buscamos que sean lo más completas posibles. Esperamos que las mismas les sirvan de orientación para la resolución del TP.

	Logra Totalmente	Logra	Logra Parcialmente	No Logra
Correctitud	La implementación respeta la especificación y satisface los tests de la cátedra.	La implementación respeta la especificación y satisface los tests de la cátedra.	La implementación no respeta la especificación y o no satisface todos los tests de la cátedra.	La implementación no respeta la especificación y o no satisface todos los tests de la cátedra.
Complejidad	Se cumplen todas las restricciones de complejidad del TP de diseño, teniendo en cuenta los costos de copia de variables y las complejidades de las estructuras auxiliares.	Se cumplen todas las restricciones de complejidad del TP de diseño, teniendo en cuenta los costos de copia de variables y las complejidades de las estructuras auxiliares.	No se cumplen todas las restricciones de complejidad del TP de diseño.	No se cumplen varias de las restricciones de complejidad del TP de diseño.
Uso de memoria	La implementación no presenta malos usos de memoria (pérdidas, doble deletes, escritura/accesos inválidos).	La implementación presenta malos usos de memoria (pérdidas, doble deletes, escritura/accesos inválidos).	La implementación presenta malos usos de memoria (pérdidas, doble deletes, escritura/accesos inválidos).	La implementación presenta malos usos de memoria (pérdidas, doble deletes, escritura/accesos inválidos).
Prolijidad	El código es legible, está bien formateado, los nombres de variables y funciones son declarativos.	El código es mayormente legible, está bien formateado, los nombres de variables y funciones comunican la idea aunque quieren contexto.	Hay secciones del código donde es necesario leerlo al detalle para entender que función cumple.	Hay secciones del código que no pueden comprenderse.
Modularización (clases)	Los módulos sólo se comunican mediante su interfaz pública. No exhiben su representación mediante punteros o referencias (la referencia explicita detalles de implementación).	Los módulos sólo se comunican mediante su interfaz pública. No exhiben su representación mediante punteros o referencias (la referencia explicita detalles de implementación).	En alguna circunstancia se necesitan conocer la representación interna de otros módulos. Se exportan punteros o referencias que explicitan detalles de implementación.	Varios módulos necesitan conocer la representación interna de otros módulos. Se exportan punteros o referencias que explicitan detalles de implementación.
Modularización (funciones)	Las funciones resuelven problemas acotados y entendibles. Lógicas muy sofisticadas o extensas se dividen en funciones auxiliares.	Alguna función es particularmente larga pero mediante auxiliares es fácil de leer.	Alguna función es particularmente larga y difícil de entender.	Es común tener funciones súmamente extensas que resuelven diversos problemas simultáneamente.
Buenas prácticas	El código no presenta situaciones que constituyen malas prácticas en cuanto a declaratividad, correctitud o eficiencia que sean importante aprender como corregir.	El código no presenta situaciones que constituyen malas prácticas en cuanto a declaratividad, correctitud o eficiencia que sean importante aprender como corregir.	El código presenta alguna situación que constituye malas prácticas.	El código presenta numerosas situaciones que constituye malas prácticas.