7 | Contactos

Meta

Que el alumno aprenda a utilizar correctamente las referencias a objetos.

Objetivos

Al finalizar la práctica el alumno será capaz de:

1. Idenficar los efectos colaterales producidos por el acceso a objetos a través de referencias.
2. Aprovechar el uso de referencias para implementar estructuras lineales recursivas de tamaño variable.

Desarrollo

En esta práctica programarás tu propia lista de contactos. Cualquier lista de contactos que se respete debe permitirle al usuario guardar tantos contactos como el usuario desee (mientras haya memoria disponible, claro). Sin embargo ¿cómo podrías guardar 100 registros sin tener que declarar 100 variables? ¿Qué tal sí sólo necesitabas 15? ¿Qué tal si llegas a Presidente de la República y tienes un millón de contactos? Bueno... tal vez eso sea un poco más complicado. Pero el objetivo es poder almacenar un número variable de elementos, sin saber cuántos serán al momento de hacer el programa.

Para lograrlo utilizarás una técnica muy popular que consiste en hacer que cada objeto sepa dónde se encuentra el que sigue y así sucesivamente. Mientras tú tengas la dirección del primer objeto en tu lista, podrás llegar a cualquier otro siguiendo la fila a partir del primero.

EJERCICIOS

1. Crea una clase `Registro`, cuyos atributos serán `nombre`, `dirección` y `teléfono`. ¿De qué tipo debe ser cada uno de esos atributos? Elígelos tú. Recuerda que estas variables deben ser privadas.

2. Ponle un constructor que reciba estos tres datos como parámetros y los asigne a los atributos correspondientes. Asegúrate de que no sean `null`, si el parámetro valiera `null`, asígnales valores por defecto.

3. Agrega *getters* y *setters* para cada atributo, asegúrate de que los valores asignados no sean `null`. Decide cómo se comportará tu programa en caso de que el parámetro valga `null` y documenta tu decisión en el método correspondiente.

4. Agrega otro atributo que será de tipo `Registro` y cuyo nombre será `siguiente`. Agrega su `getter` y `setter` correspondientes, pero no les pongas acceso. Esto hará que sólo las clases dentro del mismo directorio (`paquete`) puedan mandarlos llamar. Esto permitirá que tu lista de contactos pueda acceder a estos métodos para agregar, consultar, modificar o borrar registros – aunque para esta práctica sólo estarás agregando y consultando. Revisa que tu código compile.

5. Se te da parte del código de la interfaz de usuario en la clase `IUContactos`, completa el contenido del método `solicitaDatosDeContacto()` para que funcione con tu clase `Registro`.

6. Agrega un método `public String toString()` que devuelva una cadena con los datos almacenados en el registro. Ojo: no intentes imprimir en la consola, sólo construye y devuelve la cadena.

7. Crea ahora la clase `Contactos`. Tendrá un atributo de tipo `Registro`, ponle el nombre que quieras pero debe reflejar el hecho de que almacenará la dirección en memoria de tu primer contacto. Al inicio esta variable valdrá `null`, indicando que tu lista de contactos está vacía.

8. Agrega el método `public void insertaContacto(Registro reg)`. Este método debe recibir un objeto de tipo registro con los datos de tu nuevo contacto y lo insertará en orden alfabético. Aquí es donde tu clase `Contactos` necesitará acceder a los métodos `getSiguiente` y `setSiguiente` de la clase `Registro` para encontrar el registro anterior al nuevo e insertar el registro nuevo en su lugar.

9. Agrega un método `public void imprimeContactos()` que recorra tu lista de contactos y los vaya imprimiendo. Deberán aparecen en orden alfabético. Este es un buen momento para probar tanto la inserción como la impresión de tu lista de contactos. Compila, corre tu código y prueba esas opciones del menú. Si encuentras errores, arréglalos de una vez.

10. Agrega el método `public Registro consultar(String nombre)` que recibe el nombre o una parte del nombre del contacto que quieres consultar. Este método deberá devolver el primer registro que contenga la cadena indicada. Revisa el funcionamiento del método `contains` de la clase `String`, lo necesitarás para resolver esta parte. Nota que deberás revisar todos los registros para ver si alguno contiene la cadena indicada, si ninguno la contiene regresa `null`.

11. Compila y corre tu programa. Ya debe estar funcionando, prueba cada opción del menú y verifica que así sea. Si encuentras errores trata de irlos arreglando opción por opción.

12. Documenta todo tu código y corrige cualquier error de indentación en todas las clases.

Preguntas

1. Al buscar por nombre ¿qué necesitarías hacer para devolver todos los registros que contengan la cadena indicada, en lugar de sólo el primero? Descríbelo, no es necesario que lo programes.