

Übung 10

Listen-Suchfunktionen, Polymorphie und Interfaces

1. Ergänzen Sie in der Klasse **WeihnachtsbaumNode** aus der vorigen Aufgabe die Funktion **WeihnachtsbaumNode find(Weihnachtsbaum w)**. Diese soll in der doppelt verketteten Liste nach dem als Parameter übergebenen Weihnachtsbaum-Fenster **w** suchen, und das Listenelement, in dem sich das Fenster befindet, zurückgeben. Testen Sie die Funktion, indem Sie an geeigneter Stelle (z.B. bei einer Fenster-Aktion) die Adresse des Listenelementes, in dem das Fenster untergebracht ist, ausgeben.
2. Erzeugen Sie ein **interface**, das einen Nachrichtenaustausch zwischen zwei Objekten mit Hilfe einer Funktion
void senden(String nachricht, Object empfaenger)
und
void empfangen(String nachricht, Object sender)
realisieren soll.
3. Implementieren Sie zwei Testobjekte (evtl. verschiedener selbstdefinierter Klassen), die das in der vorigen Aufgabe erstellte **interface** implementieren, und sich gegenseitig Nachrichten verschicken, indem aus dem jeweils anderen Objekt die komplementäre Methode aufgerufen wird. D.h. beim **senden()** einer Nachricht wird **empfangen()** aus dem anderen Objekt aufgerufen.¹
4. Implementieren Sie ein Testprogramm, um das Senden und Empfangen von Nachrichten anhand der in der vorigen Aufgabe erstellten Testobjekte beobachten zu können.
5. Schreiben Sie eine abstrakte Klasse **Pflanze**, sowie eine nicht-abstrakte Klasse **Baum** und **Blume**, die jeweils von **Pflanze** als Basisklasse erben. Wenn Sie nun in einem Testprogramm von jeder der 3 Klassen eine Variable erzeugen, welche der Variablen können Sie der anderen zuweisen, und welche nicht?
6. Schreiben Sie für jede der 3 Klassen aus der vorigen Aufgabe einen Konstruktor, der mit **System.out.println()** den Klassennamen des gerade erzeugten Objektes ausgibt.

¹Sie werden ggf. in **Software Engineering** ein **Design Pattern** bzw. ein Java-Interface namens **Observer** kennenlernen, das solche Nachrichtensysteme stark vereinfacht.