

1. Class design: Doubly-linked list

A doubly-linked list is a linked data structure that consists of a set of data records, each having two special link fields that contain references to the previous and to the next record in the sequence. It can be viewed as two singly-linked lists formed from the same data items, in two opposite orders.

A Doubly-Linked List

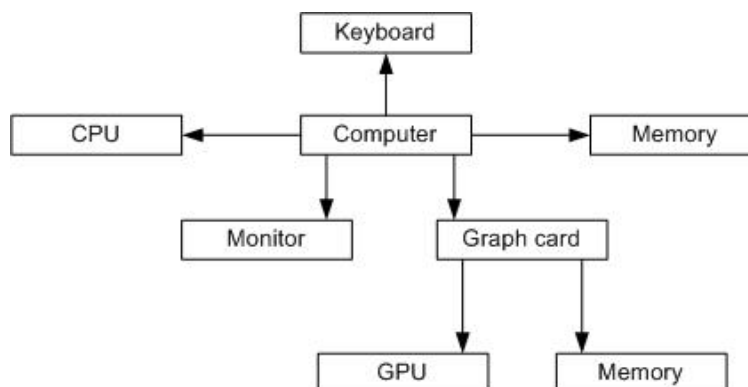


Design and implement your Eiffel class representing this data structure. The design should satisfy the following criteria:

- The design is void-safe.
- The doubly-linked class is generic.
- Basic operations like creating a new list, inserting a node into the list, deleting a node from the list, testing element existence and sorting the nodes in the list (in descending or ascending order of their values) should be available.
- Provide contracts to your classes.
- Provide a test suite for your classes.

2. Class design: composite and visitor pattern

A computer consists of many different devices, such as CPU, memory, keyboard, graphic card. An illustration of such a system is shown below:



Your task is to use the composite pattern and visitor pattern to model such a system:

- There is a deferred class DEVICE which all other devices inherit from.

- Support at least the following devices: CPU, GPU, memory, keyboard, monitor, graph card and computer. CPU, GPU, memory, keyboard and monitor are non-composite devices. Graph card and computer are composite devices, meaning they are composed from other devices.
- Every device has a price feature. For a non-composite device, the price describes the value of that individual piece; for a composite device, the price is the sum of all its component pieces.
- Provide a deferred visitor using the visitor pattern to process all different types of devices.
- Provide a concrete visitor, which calculates the total price for **ONLY** the memory parts in a computer device.