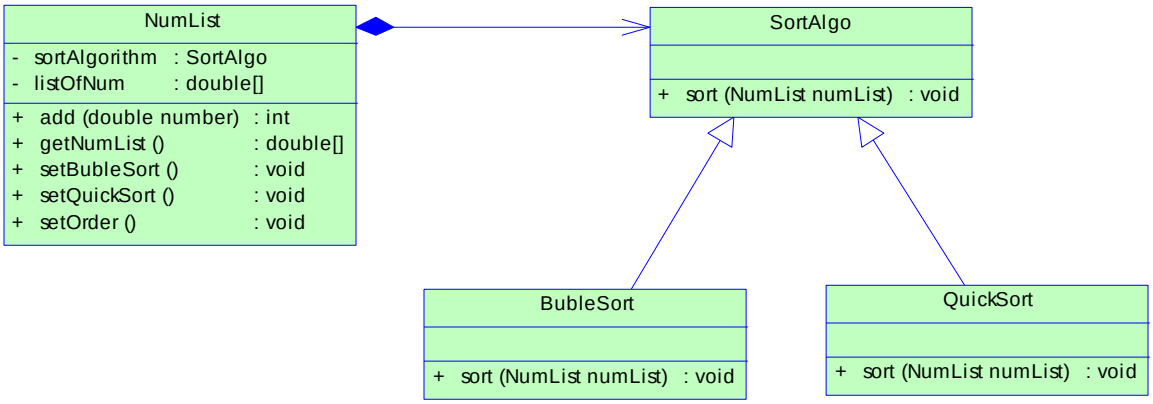
Bài thực hành số 09 – Mẫu thiết kế

Bài 1: Hãy đọc hiểu và cài đặt mẫu thiết kế **Strategy** sau (bổ xung các phương thức và biến thành viên nếu cần thiết)



Mục tiêu:

- Cho phép lựa chọn linh hoạt các thuật toán khác nhau khi thực hiện cùng một công việc. Cụ thể trong mẫu thiết kế trên, cho phép lựa chọn các thuật toán sắp xếp khác nhau ở lớp NumList.

Chú thích:

- Lớp NumList biểu diễn một danh sách các số thực.
- Giải thích một số phương thức:
  - *setBubleSort(), setQuickSort()* : Xác định thuật toán sắp xếp cho lớp NumList
  - *setOrder()*: thực hiện sắp xếp.
- Yêu cầu cài đặt cụ thể 2 thuật toán BubleSort và QuickSort.

<u>Bài 2:</u> Cài đặt mẫu **Proxy** cho bởi hình vẽ dưới (bổ xung các phương thức và biến thành viên nếu cần thiết)

Mục tiêu:

- Tối ưu việc sử dụng bộ nhớ bằng cách xây dựng đối tượng trung gian (proxy) để đại diện cho một đối tượng phức tạp (chiếm dung lượng lớn) . Đối tượng phức tạp đó chỉ được tạo ra (đưa vào bộ nhớ) tại thời điểm mà các phương thức của nó được *thực sự dùng đến*. Ví dụ đối với ảnh trong một trang web, ban đầu trình duyệt chỉ hiển thị một ô vuông biểu thị vị trí của ảnh, khi người dùng thao tác với ảnh thì ảnh mới thực sự được load.
- Client tương tác với đối tượng ảnh (Picture) thông qua lớp giao diện Graphic.

Chú ý:

- getPicture(): Trả về đối tượng Picture thực mà PictureProxy làm đại diện . Chỉ tại thời điểm gọi hàm getPicture, đối tượng Picture mới thực sự được tạo ra và sử dụng (VD trong phương thức draw() mới thực hiện gọi phương thức getPicture để tạo ra đối tượng Picture thực sự).
- getExtent(): Trả về vùng (extent) mà ảnh chiếm.
- Không yêu cầu cài đặt cụ thể các phương thức draw, store, load, handMouse.