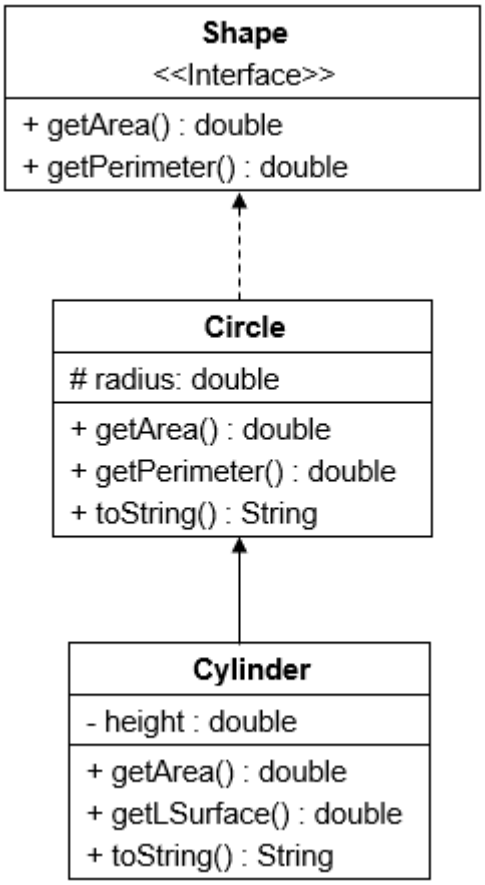
Đề ôn tập cuối kỳ

Môn: **Cấu trúc dữ liệu và giải thuật 1**

Câu 1:

Dựa vào sơ đồ lớp, sinh viên hiện thực các lớp sau bằng ngôn ngữ Java:



Trong đó:

- Tương ứng lớp Circle và Cylinder trong sơ đồ, sinh viên hiện thực thêm 3 phương thức khởi tạo đầy đủ tham số, không tham số và sao chép.

- Phương thức **getArea()** thực hiện chức năng trả về diện tích của đối tượng tương ứng mỗi lớp:

  + S_{Circle} = 3.14 x radius x radius
  + $S_{Cylinder}$ = S_{Circle} x height

- Phương thức **getPerimeter()** thực hiện chức năng trả về chu vi của đối tượng lớp Circle:

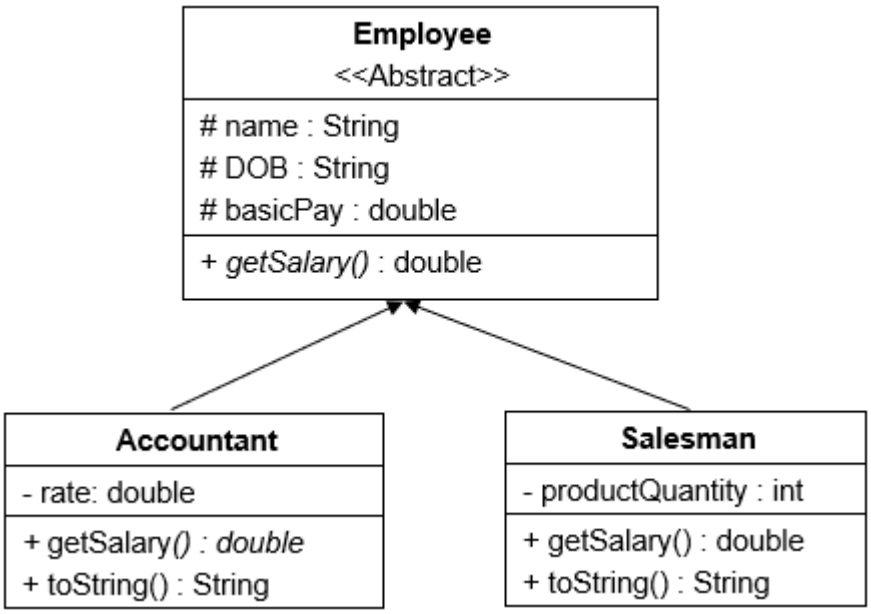
  + P_{Circle} = 3.14 x 2 x radius

- Phương thức **getLSurface()** thực hiện chức năng trả về diện tích xung quanh của đối tượng lớp Cylinder:
  - $A_{Cylinder} =$ 3.14 x 2 x radius x height

- Phương thức **toString()** trả về chuỗi chứa giá trị của các thuộc tính tương ứng mỗi đối tượng. Ví dụ với đối tượng của lớp Cylinder thì chuỗi trả về bảo đảm chứa thuộc tính radius và height, định dạng tùy sinh viên.

Câu 2:

Dựa vào sơ đồ lớp, sinh viên hiện thực các lớp sau bằng ngôn ngữ Java:

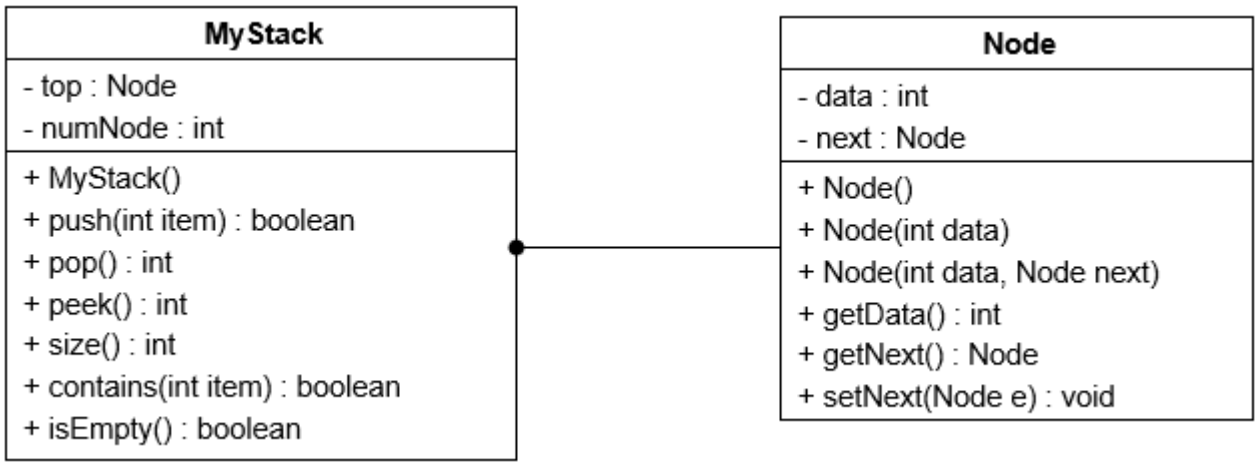


- Lớp Abstract Employee định nghĩa thêm 2 phương thức khởi tạo không tham số và có tham số. Tương ứng lớp Accountant và Salesman trong sơ đồ, sinh viên hiện thực thêm 3 phương thức khởi tạo đầy đủ tham số, không tham số và sao chép.

- Phương thức **getSalary()** thực hiện chức năng trả về tiền lương của đối tượng tương ứng mỗi lớp:
  - $S_{Accountant} =$ rate $*$ basicPay + basicPay
  - $S_{Salesman} = productQuantity * 0.05 +$ basicPay

- Phương thức **toString()** trả về chuỗi chứa giá trị của các thuộc tính tương ứng mỗi đối tượng. Ví dụ với đối tượng của lớp Accountant thì chuỗi trả về bảo đảm chứa 3 thuộc tính của Employee và 1 thuộc tính của Accountant, định dạng tùy sinh viên.

Câu 3: Một ngăn xếp (Stack) có đặc tả bằng UML như sau:



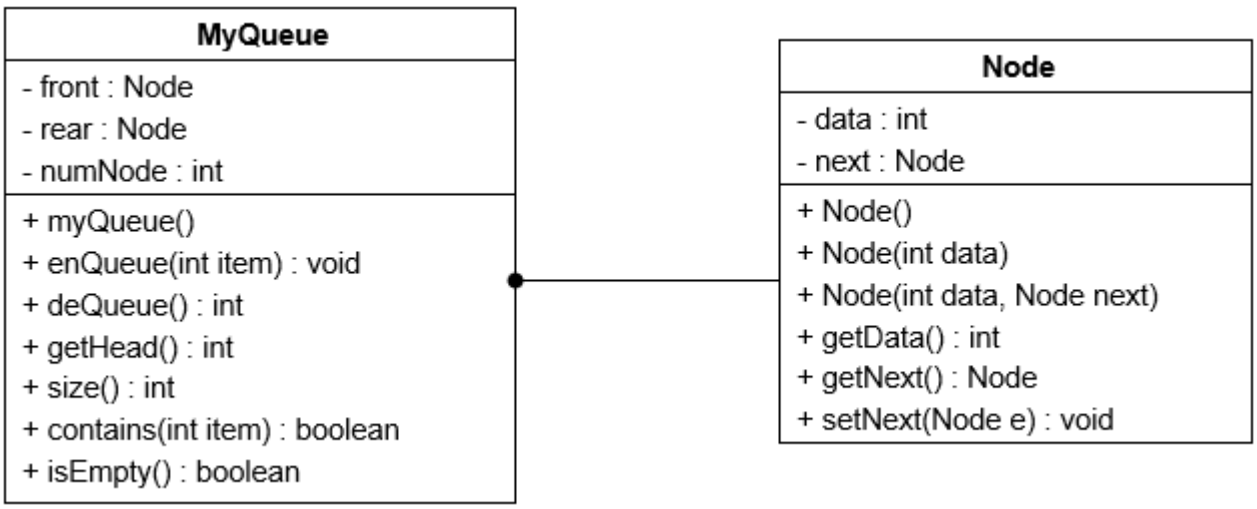
Sinh viên hiện thực lại các phương thức sau:

a. Hiện thực phương thức **push(int item):boolean** để thêm một phần tử mới vào ngăn xếp số nguyên. Phương thức trả về true nếu thêm thành công phần tử mới vào danh sách; trả về false nếu phần tử mới có giá trị lớn hơn đỉnh của ngăn xếp, trường hợp ngăn xếp null thì thêm bình thường.

b. Hiện thực phương thức **pop():int** để lấy phần tử từ ngăn xếp số nguyên. Phương thức trả về giá trị của phần tử vừa lấy. Nếu trường hợp ngăn xếp rỗng thì sử dụng NoSuchElementException.

c. Hiện thực phương thức **peek():int** để biết được giá trị hiện tại của đỉnh ngăn xếp. Phương thức trả về giá trị của phần tử đỉnh trong ngăn xếp.

Câu 4: Một hàng đợi (Queue) có đặc tả bằng UML như sau:



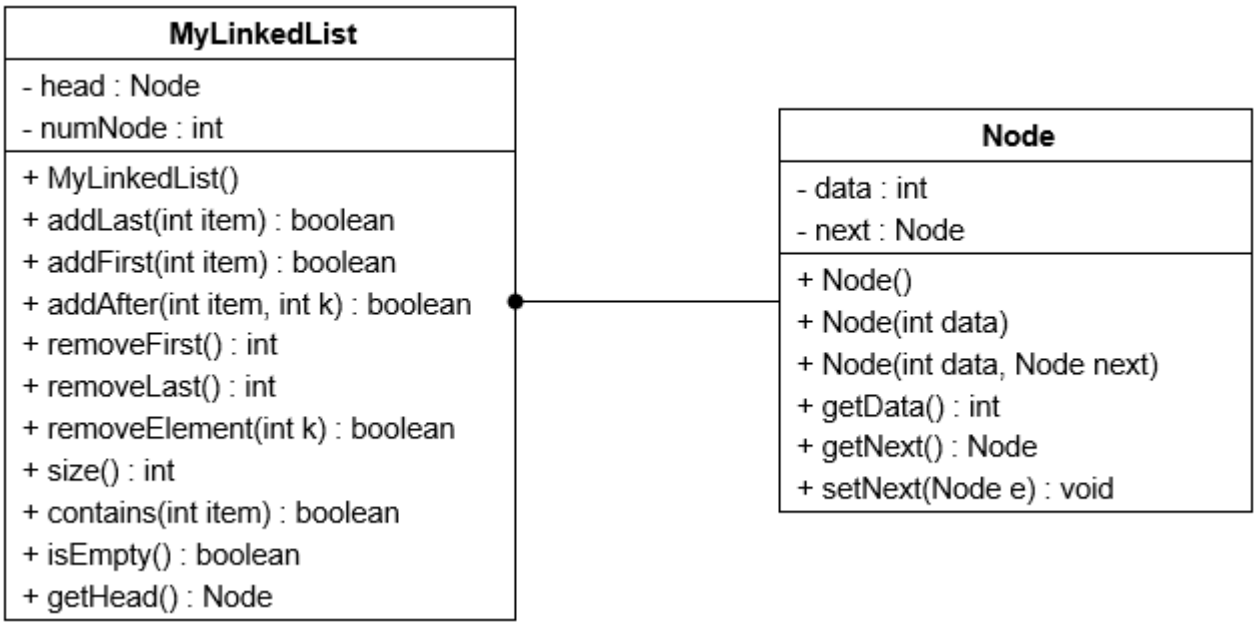
Sinh viên hiện thực lại các phương thức sau:

a. Hiện thực phương thức **enQueue(int item):void** để thêm một phần tử mới vào hàng đợi các số nguyên.

b. Hiện thực phương thức **deQueue():int** để lấy phần tử từ hàng đợi danh sách các số nguyên ra ngoài. Phương thức trả về giá trị của phần tử được lấy ra. Nếu trường hợp hàng đợi rỗng thì sử dụng NoSuchElementException.

c. Hiện thực phương thức **getHead():int** để lấy giá trị của phần tử đầu tiên trong hàng đợi. Phương thức trả về giá trị của phần tử đầu hàng đợi.

Câu 5: Một danh sách liên kết có đặc tả bằng UML như sau:



Sinh viên hiện thực lại các phương thức sau:

a. Hiện thực phương thức **addFirst(int item):boolean** để thêm một phần tử mới vào đầu danh sách các số nguyên. Phương thức trả về true nếu thêm thành công phần tử mới vào danh sách; trả về false nếu phần tử mới có giá trị bé hơn phần tử đầu tiên có sẵn trong danh sách, trường hợp danh sách null thì thêm bình thường.

b. Hiện thực phương thức **addLast(int item):boolean** để thêm một phần tử mới vào cuối danh sách các số nguyên. Phương thức trả về true nếu thêm thành công phần tử mới vào danh sách; trả về false nếu phần tử mới đã có sẵn trong danh sách.

c. Hiện thực phương thức **addAfter(int item, int k):boolean** để thêm phần tử item liền sau giá trị ***k*** đầu tiên tìm được trong danh sách. Phương thức trả về true nếu thêm thành công phần tử mới vào danh sách; trả về false nếu phần tử mới đã có sẵn trong danh sách hoặc không tồn tại giá trị ***k***.

d. Hiện thực phương thức **removeFirst():int** để xóa phần tử đầu tiên trong danh sách số nguyên và trả về giá trị của phần tử vừa xóa. Nếu trường hợp danh sách rỗng thì sử dụng NoSuchElementException.

e. Hiện thực phương thức **removeLast():int** để xóa phần tử cuối cùng trong danh sách số nguyên và trả về giá trị của phần tử vừa xóa. Nếu trường hợp danh sách rỗng thì sử dụng NoSuchElementException.

f. Hiện thực phương thức **removeElement(int k):boolean** để xóa phần tử đầu tiên chứa giá trị ***k*** trong danh sách, nếu xóa thành công thì trả về true, nếu phần tử không tồn tại trong danh sách thì trả về false. Nếu trường hợp danh sách rỗng thì sử dụng NoSuchElementException.

Câu 6: Sử dụng kỹ thuật đệ quy để giải các bài toán sau:

a. Tính tích các chữ số của số nguyên dương **n** với prototype là **int multiply(int n)**

b. Tìm số fibonacci thứ n biết công thức của dãy fibonacci như sau:

$$fibonacci(0) = 0$$
$$fibonacci(1) = 1$$
$$fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)$$

c. Tìm chữ số lớn nhất trong số nguyên dương **n** với prototype là **int maxDigit(int n, int max)**

Câu 7: Sử dụng kỹ thuật đệ quy để giải các bài toán sau:

a. $A(n) = \begin{cases} 4, & n = 0 \\ 3, & n = 1 \\ A(n-1) - A(n-2) - 1, & n > 1 \end{cases}$

b. $A(n) = \begin{cases} 1, & n = 1 \\ 3, & n = 2 \\ \frac{A(n-1) + A(n-2)}{2}, & n > 2 \end{cases}$

c. $A(n,k) = \begin{cases} 0, & k > n \\ 1, & k = 0 \ or \ k = n \\ A(n-1,k) + A(n-1,k-1), & otherwise \end{cases}$ (giả sử n được cho luôn lớn hơn k)

Câu 8: Giải thuật sắp xếp chọn (Selection Sort)

a. Trình bày ý tưởng của giải thuật sắp xếp chọn.

b. Áp dụng giải thuật sắp xếp chọn, trình bày từng bước quá trình sắp xếp dãy số sau theo thứ tự **tăng dần từ trái sang phải**.

31 1 20 0 99 12 27 2 36

c. Hiện thực hàm **public void selectionSort(int arr[])** để sắp xếp mảng arr theo thứ tự tăng dần.

Câu 9: Giải thuật sắp xếp nổi bọt (Bubble Sort)

a. Trình bày ý tưởng của giải thuật sắp xếp nổi bọt.

b. Áp dụng giải thuật sắp xếp nổi bọt, trình bày từng bước quá trình sắp xếp dãy số sau theo thứ tự **giảm dần từ trái sang phải**.

12 55 22 60 57 15 27 2 36

c. Hiện thực hàm **public void bubbleSort(int arr[])** để sắp xếp mảng arr theo thứ tự tăng dần.

Câu 10: Giải thuật sắp xếp chèn (Insertion Sort)

a. Trình bày ý tưởng của giải thuật sắp xếp chèn.

b. Áp dụng giải thuật sắp xếp chèn, trình bày từng bước quá trình sắp xếp dãy số sau theo thứ tự **tăng dần từ trái sang phải**.

50 28 12 27 88 45 30 65 36

c. Hiện thực hàm **public void insertionSort(int arr[])** để sắp xếp mảng arr theo thứ tự tăng dần.

Câu 11: Tìm độ phức tạp tính toán của các đoạn mã sau:

a.
```
int i, j;
int s = 0;
for (i = 0; i < n; ++i)
{
	for (j = 0; j < n; ++j)
	{
		s += j;
	}
}
```

b.
```
for(i = 0; i < n; i++) {
	for(k = n*n - 1; k >= 0; k--) {
		print("Hi");
	}
}
```

c.
```
int s = 1;
for(i = 0; i < n; i++) {
	index = 1;
	while(index < n) {
		s = s * 3;
		index *= 5;
	}
}
```

d.
```
int i, int j;
for(i = 0; i < n-1; i++) {
	for(j = i+1; j < n; j++) {
		tmp = A[i][j];
		A[i][j] = A[j][i];
		A[j][i] = tmp
	}
}
```

Câu 12: Sinh viên chèn dãy số sau vào bảng băm 7 phần tử, với ***Hash(k) = k mod 7***. Trong trường hợp xảy ra đụng độ (conflict), áp dụng phương pháp **dò tuyến tính** (linear probing) bằng cách tăng lên 1 vị trí, đến khi tìm được vị trí thích hợp.

44 35 22 43 54 41

Câu 13: Sinh viên chèn dãy số sau vào bảng băm 13 phần tử, với ***Hash(k) = k mod 13***. Trong trường hợp xảy ra đụng độ (conflict), áp dụng phương pháp **dò tuyến tính** (linear probing) bằng cách tăng lên 1 vị trí, đến khi tìm được vị trí thích hợp.

13 20 36 23 20 71 31 22 60 28 38 39

-- HẾT --