BÀI TẬP VIẾT CLASS

====

BÀI 1. Viết chương trình tạo lập lớp Sinh viên với các thuộc tính mã sinh viên, tên sinh viên, lớp, khóa. Khởi tạo một sinh viên với các giá trị nhập từ bàn phím. In thông tin sinh viên ra màn hình.

BÀI 2. Sử dụng lớp sinh viên trong bài 1, nhập vào danh sách n sinh viên và thực hiện:

a. In toàn bộ danh sách ra màn hình

b. Sắp xếp danh sách theo tên sinh viên

c. In danh sách sinh viên theo lớp

BÀI 3. Viết chương trình thực hiện các yêu cầu sau:

- Tạo lớp đối tượng hình chữ nhật có tên là Rectangle theo lược đồ UML dưới đây:

Rectange	
- width: double	Chiều rộng hình chữ nhật
- height: double	Chiều dài hình chữ nhật
- color: String	Màu hình chữ nhật
+ Rectange()	Tạo HCN có c.dài = 1, c.rộng = 1
+ Rectange(width: double, height: double, color: String)	Tạo HCN có c.dài, c.rộng xác định qua tham số
+ getWidth() :double	Trả về chiều rộng
+ setWidth(width:double): void	Thiết lập chiều rộng mới
+ getHeight(): double	Trả về chiều dài
+ setHeight(height:double): void	Thiết lập chiều dài mới
+ getColor(): String	Trả về màu của HCN
+ setColor(color): void	Thiết lập màu mới cho HCN
+ findArea(): double	Tính và trả về diện tích HCN
+ findPerimeter(): double	Tính và trả về chu vi HCN

- Từ chương trình chính, tạo một đối tượng hình chữ nhật với kích thước và màu sắc nhập vào từ bàn phím. Đưa ra màn hình các thuộc tính, diện tích và chu vi của đối tượng hình chữ nhật đã tạo.

BÀI 4. Viết chương trình thực hiện các yêu cầu sau:

- Tạo lớp đối tượng ngăn xếp chứa các số nguyên có tên là StackOfIntergers theo lược đồ UML dưới đây:

StackOfIntegers	
- elements: int[]	Mảng chứa các số nguyên trong stack
- size: int	Số lượng số nguyên đang chứa trong stack
+ StackOfIntegers()	Tạo một stack rỗng có dung lượng = 16
+StackOfIntegers (capacity: int)	Tạo một stack rỗng có dung lượng xác định trong tham số
+ isEmpty(): boolean	Trả về true nếu stack rỗng
+ isFull(): boolean	Trả về true nếu stack đầy
+ peak(): int	Trả về số nguyên ở đỉnh stack
+ push(value:int): void	Đẩy một số nguyên vào đỉnh stack
+ pop(): int	Lấy ra và trả về số nguyên tại đỉnh stack
+ getSize(): int	Trả về số lượng số nguyên đang chứa trong stack

Viết chương trình sử dụng lớp StackOfIntergers để đưa ra màn hình tất cả các số nguyên tố nhỏ hơn một số nguyên dương n (nhập vào từ bàn phím) theo thứ tự giảm dần.

BÀI 5. Tạo lớp StackOfIntergers như bài 4. Viết chương trình nhập vào một số nguyên dương rồi hiển thị các thừa số nguyên tố nhỏ nhất của nó theo thứ tự ngược. Ví dụ, nếu số nguyên là 120, các thừa số nguyên tố nhỏ nhất được hiển thị là 5, 3, 2, 2, 2. Sử dụng lớp StackOfIntergers để chứa các thừa số rồi lấy và hiển thị chúng theo thứ tự ngược.

BÀI 6. Tạo lớp StackOfIntergers như bài 4, viết chương trình đổi một số nguyên bất kỳ sang cơ số 2.

BÀI 7. Viết chương trình thực hiện các yêu cầu sau:

- Tạo lớp đối tượng ngăn xếp chứa các ký tự có tên là StackOfChars theo lược đồ UML dưới đây:

StackOfChars	
- elements: char[]	Mảng chứa các ký tự trong stack
- size: int	Số lượng ký tự đang chứa trong stack
+ StackOfChars()	Tạo một stack rỗng có dung lượng = 16
+ StackOfChars (capacity: int)	Tạo một stack rỗng có dung lượng xác định trong tham số
+ isEmpty(): boolean	Trả về true nếu stack rỗng
+ isFull(): boolean	Trả về true nếu stack đầy
+ peak(): char	Trả về ký tự ở đỉnh stack
+ push(ch:char): void	Đẩy một ký tự vào đỉnh stack
+ pop(): char	Lấy ra và trả về ký tự tại đỉnh stack
+ getSize(): int	Trả về số lượng ký tự đang chứa trong stack

Viết chương trình sử dụng lớp StackOfChars để tính giá trị của biểu thức số học dạng trung tố có dấu ngoặc đầy đủ. Giả sử trong biểu thức số học chỉ chứa các phép toán cộng, trừ, nhân, chia và các số hạng là các số chỉ có một chữ số.

Ví dụ: Nhập vào biểu thức số học ((3+7)×(9-(6-2))), đưa ra kết quả là 50.

BÀI 8. Tạo lớp StackOfChars như bài 7. Viết chương trình sử dụng lớp StackOfChars để chuyển một biểu thức dạng trung tố về dạng hậu tố.

BÀI 9. Tạo lớp StackOfChars như bài 7. Viết chương trình sử dụng lớp StackOfChars để đổi số nguyên n bất kỳ sang cơ số b bất kỳ (1<b<36).

BÀI 10. Viết chương trình thực hiện các yêu cầu sau:

- Tạo lớp đối tượng điểm trong mặt phẳng tọa độ OXY có tên là MyPoint theo lược đồ UML dưới đây:

MyPoint	
- x: double	Tọa độ x
- y: double	Tọa độ y
+ MyPoint()	Tạo đối tượng mặc định
+ MyPoint(x: double, y: double)	Tạo đối tượng có tọa độ xđ trong tham số
+ MyPoint(p: MyPoint)	Tạo đối tượng là bản sao của đối tượng trong t.số
+ getX(): double	Trả về tọa độ X
+ getY(): double	Trả về tọa độ Y
+ distance(secondPoint: MyPoint): double	Trả về khoảng cách từ điểm này tới điểm thứ hai
+distance(p1: MyPoint, p2: MyPoint): double	Trả về khoảng cách giữa hai điểm

Viết chương trình tạo n đối tượng điểm MyPoint có tọa độ nhập vào từ bàn phím. Tìm hai điểm có khoảng cách lớn nhất. Đưa ra màn hình tọa độ của hai điểm tìm được và giá trị khoảng cách giữa chúng.

BÀI 11. Sử dụng lớp MyPoint ở câu trên và nhập vào 3 điểm trong không gian 2 chiều. Kiểm tra xem có tạo thành một tam giác hay không, có phải là tam giác dạng đặc biệt (vuông, cân, vuông cân, hay đều) hay không.

BÀI 12. Viết chương trình thực hiện các yêu cầu sau:

- Tạo lớp đối tượng ma trận có tên là Matrix theo lược đồ UML dưới đây:

Matrix	
- a: float[][]	Mảng chứa các phần tử của ma trận
+ Matrix()	Tạo ma trận có số hàng và số cột bằng 3
+ Matrix(rows: int, cols : int)	Tạo ma trận có số hàng và số cột xđ trong tham số
+ add(m: Matrix): Matrix	Tính tổng ma trận này với ma trận trong tham số, trả về ma trận tổng. Trong phương thức có kiểm tra sự hợp lệ về số hàng, số cột của hai ma trận.
+ sub(m: Matrix): Matrix	Tính hiệu ma trận này với ma trận trong tham số, trả về ma trận hiệu. Trong phương thức có kiểm tra sự hợp lệ về số hàng, số cột của hai ma trận.
+ neg(): Matrix	Đổi dấu các phần tử của ma trận, trả về ma trận đã đổi dấu các phần tử
+ transpose(): Matrix	Trả về ma trận chuyển vị
+ mul(m: Matrix): Matrix	Tính tích ma trận này với ma trận trong tham số, trả về ma trận tích. Trong phương thức có kiểm tra sự hợp lệ về số hàng, số cột của hai ma trận.
+ print(): void	Đưa ma trận ra màn hình
+ input(): void	Nhập vào các phần tử của ma trận

- Viết chương trình sử dụng lớp Maxtrix để thực hiện một số phép toán về ma trận. Chương trình có các mục menu:

1. Tính tổng và hiệu hai ma trận;
2. Tính tích hai ma trận;
3. Tìm chuyển vị của một ma trận;

4. Kết thúc chương trình.

Khi người sử dụng chọn các mục từ 1 đến 3 thì cho nhập vào ma trận, thực hiện tính toán và đưa ra kết quả; khi người sử dụng chọn 4 thì kết thúc chương trình.

BÀI 13. Viết chương trình thực hiện các yêu cầu sau:

- Tạo lớp đối tượng phân số có tên là PhanSo theo lược đồ UML dưới đây:

PhanSo	
- ts: int	Tử số
- ms: int	Mẫu số
+ PhanSo()	Tạo phân số có tử số bằng 0, mẫu số bằng 1
+ PhanSo(ts: int, ms: int)	Tạo phân số có tử số và mẫu số xđ trong tham số
+ cong(sp2: PhanSo): PhanSo	Cộng hai phân số
+ tru(sp2: PhanSo): PhanSo	Trừ hai phân số
+ nhan(sp2: PhanSo): PhanSo	Nhân hai phân số
+ chia(sp2: PhanSo): PhanSo	Chia hai phân số
+ nghichDao(): PhanSo	Nghịch đảo phân số và trả về phân số nghịch đảo
+ doiDau(): PhanSo	Đổi dấu phân số và trả về phân số đã đổi dấu
+ toiGian(): PhanSo	Tối giản phân số và trả về phân số đã tối giản
+ bangNhau(ps2: PhanSo): boolean	So sánh bằng
+ lonHon(ps2: PhanSo): boolean	So sánh lớn hơn
+ nhoHon(ps2: PhanSo): boolean	So sánh nhỏ hơn
+ hien(): void	Đưa phân số ra màn hình ở dạng ts/ms
+ nhap(): void	Nhập vào phân số dạng ts/ms

- Viết chương trình sử dụng lớp PhanSo. Nhập vào hai phân số; tính tổng, hiệu, tích, thương hai phân số; tối giản và so sánh hai phân số.

BÀI 14. Viết chương trình thực hiện các yêu cầu sau:

- Tạo lớp đối tượng số phức có tên là SoPhuc theo lược đồ UML dưới đây:

SoPhuc	Số phức a + jb
- a: float	Phần thực a
- b: float	Phần ảo b
+ SoPhuc()	Tạo số phức có phần thực và ảo bằng 0
+ SoPhuc(thuc: float, ao: float)	Tạo số phức có phần thực và ảo xđ trong tham số
+ cong(sp2: SoPhuc): SoPhuc	Cộng hai số phức
+ tru(sp2: SoPhuc): SoPhuc	Trừ hai số phức
+ nhan(sp2: SoPhuc): SoPhuc	Nhân hai số phức
+ chia(sp2: SoPhuc): SoPhuc	Chia hai số phức
+ nghichDao(): SoPhuc	Trả về số phức nghịch đảo
+ bangNhau(sp2: SoPhuc): boolean	So sánh bằng
+ lonHon(sp2: SoPhuc): boolean	So sánh lớn hơn (theo modul)
+ nhoHon(sp2: SoPhuc): boolean	So sánh nhỏ hơn (theo modul)
+ hien(): void	Đưa số phức ra màn hình ở dạng a + jb
+ nhap(): void	Nhập vào số phức dạng a + jb

Viết chương trình sử dụng lớp SoPhuc. Nhập vào hai số phức; tính tổng, hiệu, tích, thương hai số phức; tính nghịch đảo và so sánh hai số phức.

BÀI 15. Cho lớp Sinh viên được mô tả như sau:

SinhVien	
- maSV: int	Mã SV: có 4 chữ số
- ten: String	Tên sinh viên, gồm cả họ và tên. Không được để trống
- lop: String	Lớp: Theo quy tắc đặt tên của PTIT.
- dtb: double	Điểm trung bình: từ 0 đến 10.
+ SinhVien()	Khởi tạo đối tượng mặc định
+ SinhVien(m: int, t: String, l : String, d: double)	Khởi tạo đối tượng có tham số
…….	*(… cần bổ sung các phương thức cho phù hợp với yêu cầu …)*

Hãy thực hiện các yêu cầu sau:

1. Cho file dữ liệu SV.INP trong đó mỗi sinh viên ghi trên 4 dòng lần lượt là mã, tên, lớp, điểm. Nhập dữ liệu và lưu vào mảng. Chú ý kiểm tra các ngoại lệ theo ràng buộc của các thuộc tính, nếu sai bỏ qua sinh viên đó (tức là chỉ nhận các sinh viên có thông tin đúng theo ràng buộc).
2. In danh sách sinh viên đã nhập ra màn hình. Mỗi sinh viên ghi trên một dòng, mỗi thông tin cách nhau đúng một khoảng trống.
3. Sắp xếp danh sách sinh viên theo lớp và hiển thị danh sách sinh viên trong từng lớp theo điểm trung bình giảm dần. Ghi ra file SX.OUT. Mỗi sinh viên ghi trên 1 dòng, các thông tin cách nhau đúng một khoảng trống.
4. Phân loại sinh viên theo 4 mức: Giỏi (đtb>8), Khá (7<đtb<8), Trung Bình (5<đtb<7), Yếu (đtb<5). Sắp xếp danh sách đã phân loại theo tên sinh viên (tách tên và sắp xếp theo ABC, nếu trùng tên thì sắp xếp theo họ và tên đệm). Ghi ra file XEPLOAI.OUT.

Chú ý: Sinh viên cần bổ sung các lớp cho phù hợp.