Bài tập chương 13

Bài tập bắt buộc:

1. Date
2. Employee
3. Inventory
4. Circle
5. Number Array
6. Trivia Game

1. Date (Bắt buộc)

Thiết kế một lớp được gọi là $Date$. Lớp này lưu thông tin của một thời gian gồm 3 số nguyên: $month$ (tháng), day(ngày) và $year$(năm). Nó có các hàm thành viên để in ra thời gian theo các dạng sau đây:

- 22/05/2021
- Ngày 22 Tháng 5, 2021

Hãy viết 1 chương trình để hoàn thiện lớp này như yêu cầu.

Lưu ý đầu vào: Không cho phép day có giá trị lớn hơn 31 và nhỏ hơn 1. Không cho phép $month$ có giá trị lớn hơn 12 và nhỏ hơn 1.

2. Employee (Bắt buộc)

Hãy viết 1 lớp có tên là $Employee$ (nhân viên) có các biến thành viên sau:

- $name$: 1 *chuỗi ký tự* lưu tên của nhân viên.
- $idNumber$: 1 *số nguyên* lưu số ID của nhân viên (mã nhân viên)
- $department$: 1 *chuỗi ký tự* lưu tên của phòng ban - nơi nhân viên đó làm việc
- $position$: 1 *chuỗi ký tự* lưu vị trí làm việc của nhân viên (job title).

Lớp này phải có các cấu tử (hàm tạo) sau:

- 1 cấu tử cho phép nhập các giá trị sau làm đối số và gán chúng vào các biến thành viên thích hợp: tên nhânh viên, ID nhân viên, phòng ban và vị trí.
- 1 cấu tử cho phép nhập các giá trị sau làm đối số và gán chúng vào các biến thành viên thích hợp: tên nhânh viên, ID nhân viên. Trường $department$ và $position$ sẽ được gán bởi 1 chuỗi rỗng ("").
- 1 cấu tử mặc định sẽ gán các chuỗi rỗng ("") vào các biến thành viên $name$, $department$, $position$ và 0 cho biến thành viên $idNumber$.

Hãy viết các hàm phù hợp để lưu giá trị của các biến thành viên ở trên và các hàm có thể gán hoặc trả về giá trị của chúng (set/get). Sau đó, hãy viết 1 chương trình tạo 3 đối tượng $Employee$ lưu thông tin theo dữ liệu sau:

Name	ID Number	Department	Position

Name	ID Number	Department	Position
Susan Meyers	47899	Accounting	Vice President
Mark Jones	39119	IT	Programmer
Joy Rogers	81774	Manufacturing	Engineer

Chương trình sẽ lưu dữ liệu trên trong 3 đối tượng và biểu diễn dữ liệu của mỗi nhân viên trên màn hình.

3. Car

Hãy viết một lớp có tên là Car (ô tô) có các biến thành viên sau:

- $yearModel$: 1 số nguyên giữ năm model của ô tô.
- $make$: 1 chuỗi ký tự giữ thông tin về cấu tạo của xe.
- $speed$: 1 số nguyên giữ tốc độ hiện tại của xe.

Thêm vào đó, lớp này sẽ có cấu tử và các hàm thành viên như sau:

- **Cấu tử**: Cấu tử sẽ cho phép nhập năm model và cấu tạo của xe như đối số. Các giá trị sẽ bị gán cho đối tượng là biến thành viên $yearModel$ và $make$. Cấu tử sẽ cũng sẽ gán 0 cho biến $speed$.
- **Hàm truy cập**: Các hàm truy cập thích hợp để lấy được các giá trị được lưu trong các biến thành viên của đối tượng $yearModel$, $make$, và $speed$
- **accelerate**: Hàm này sẽ thêm 5 đơn vị vào biến $speed$ sau mỗi lần được gọi.
- **brake**: Hàm này sẽ trừ 5 đơn vị của biến $speed$ sau mỗi lần được gọi.

Viết 1 chương trình chứa lớp như yêu cầu và tạo 1 đối tượng Car. Sau đó, gọi hàm $accelerate$ 5 lần. Sau mỗi lần gọi hàm, hãy lấy tốc độ hiện tại của xe và hiển thị nó. Tiếp theo, gọi hàm $brake$ 5 lần. Sau mỗi lần gọi hàm, hãy lấy tốc độ hiện tại của xe và hiển thị nó.

4. Personal Information

Thiết kế 1 lớp có thể lưu các dữ liệu cá nhân như: tên, địa chỉ, tuổi, và số điện thoại. Viết các hàm gán và trả về (set/get) phù hợp. Hãy viết chương trình sử dụng lớp này để tạo 3 hồ sơ. Một hồ sơ lưu thông tin của bạn, 2 hồ sơ còn lại lưu thông tin của bạn bè hay gia đình của bạn.

5. RetailItem

Viết 1 lớp có tên là $RetailItem$ (Mặt hàng bán lẻ) giữ dữ liệu về 1 mặt hàng trong cửa hàng bán lẻ. Lớp này sẽ có những biến thành viên như sau:

- $description$: 1 chuỗi ký tự giữ thông tin mô tả ngắn gọn về mặt hàng.
- $unitsOnHand$: 1 số nguyên cho biết số lượng sản phẩm còn lại hiện tại trong kho.
- $price$: 1 số thực cho biết giá của mặt hàng.

Viết 1 cấu tử cho phép các đối số ứng với mỗi biến thành viên, các hàm truy cập gán/lấy giá trị (set/get) phù hợp với mỗi biến thành viên. Sau khi có lớp này, hãy viết 1 chương trình cho phép tạo 3 đối tượng $RetailItem$ và lưu trữ chúng theo dữ liệu như bảng sau:

	Description	Unit On Hand	Price
Item #1	Jacket	12	59.95
Item #2	Designer Jeans	40	34.95
Item #3	Shirt	20	24.95

6. Inventory (Bắt buộc)

Thiết kế lớp $Inventory$ - giữ thông tin và tính toán cho các mặt hàng trong một kho hàng cửa hàng bán lẻ. Lớp này có các biến thành viên $private$ như sau:

- $itemNumber$: 1 số nguyên cho biết số định danh của mặt hàng.
- $quantity$: 1 số nguyên cho biết số lượng của mặt hàng trong kho.
- $cost$: 1 số thực cho biết giá của 1 đơn vị mặt hàng này.
- $totalCost$: 1 số thực cho biết tổng giá của kho hàng của mặt hàng này (số lượng nhân với giá).

Lớp này sẽ có các hàm thành viên $public$ sau:

- **Cấu tử mặc định**: Gán toàn bộ biến thành viên giá trị 0.
- **Cấu tử #2**: Cho phép nhập số định danh, giá, số lượng như đối số của hàm. Hàm này sẽ sao chép giá trị tử các đối số này vào các biến tương ứng và sau đó gọi hàm **setTotalCost**.
- **setItemNumber**: Cho phép nhập 1 số nguyên như 1 đối số, số này sẽ được sao chép vào biến thành viên $itemNumber$.
- **setQuantity**: Cho phép nhập 1 số nguyên như 1 đối số, số này sẽ được sao chép vào biến thành viên $quantity$.
- **setCost**: Cho phép nhập 1 số nguyên như 1 đối số, số này sẽ được sao chép vào biến thành viên $cost$.
- **setTotalCost**: Tính toán tổng giá trị của kho hàng cho mặt hàng này sau đó lưu giá trị vào biến thành viên $totalCost$.
- **getItemNumber**: Trả về giá trị của biến $itemNumber$.
- **getQuantity**: Trả về giá trị của biến $quantity$.
- **getCost**: Trả về giá trị của biến $cost$.
- **getTotalCost**: Trả về giá trị của biến $totalCost$.

Viết chương trình thực thi lớp này.

Lưu ý: Không cho phép nhập vào giá trị âm cho số định danh, số lượng hay giá của mặt hàng.

7. TestScores

Thiết kế 1 lớp có tên là $TestScores$. Lớp này có các biến thành viên để lưu trữ 3 điểm kiểm tra. Nó sẽ có 1 cấu tử, hàm truy cập trả về (get) và hàm gán (set) các trường điểm kiểm tra và 1 hàm thành viên trả về trung bình của các điểm kiểm tra. Thực thi lớp này bằng 1 chương trình. Trong chương trình này, hãy tạo 1 đối tượng của lớp $TestScores$, sau đó yêu cầu người dùng nhập vào 3 điểm kiểm tra. 3 điểm này sẽ được lưu vào 3 biến thành viên của đối tượng vừa tạo. Chương trình sẽ thể hiện giá trị trung bình của điểm kiểm tra, điểm này được báo cáo bởi đối tượng $TestScores$ vừa tạo.

8. Circle (Bắt buộc)

Viết 1 lớp $Circle$ (hình tròn) có các biến thành viên như sau:

- $radius$: 1 số thực, thể hiện bán kính của hình tròn.
- pi: 1 số thực được khởi tạo với giá trị 3.14159.

Lớp này sẽ có các hàm thành viên như sau:

- **Cấu tử mặc định**: Cấu tử mặc định sẽ gán bán kính là 0.0.
- **Cấu tử #2**: Cho phép chuyền vào bán kính của đường tròn như 1 đối số.
- **setRadius**: Hàm gán giá trị cho biến $radius$.
- **getRadius**: Hàm lấy giá trị của biến $radius$.
- **getArea**: Hàm trả về diện tích của hình tròn, được tính bởi công thức: $area = pi * radius * radius$.
- **getDiameter**: Hàm trả về đường kính của hình tròn, được tính bởi công thức: $diameter = radius * 2$
- **getCircumference**: Hàm trả về chu vi của hình tròn, được tính bởi công thức: $circumference = 2 * pi * radius$.

Viết 1 chương trình sử dụng lớp $Circle$ để yêu cầu người dùng nhập vào bán kính của đường tròn rồi tạo 1 đối tượng $Circle$, và sau đó hiển thị diện tích, đường kính, và chu vi của hình tròn được tính ra bởi đối tượng vùa tạo.

9. Population

Trong dân số, tỉ lệ sinh và tử được tính theo công thức:

- Tỉ lệ sinh= Số lượng sinh / dân số
- Tỉ lệ tử = Số lượng tử / dân số

Ví dụ, dân số có 100000 thì có 8000 người được sinh ra và 6000 người mất đi mỗi năm thì:

- Tỉ lệ sinh là= 8000/100000= 0.08
- Tỉ lệ tử= 6000/100000= 0.06

Hãy thiết kế một lớp $Population$ (dân số) lưu thông tin về dân số, số người được sinh ra, số người mất đi mỗi năm. Các hàm thành viên sẽ trả về tỉ lệ sinh và tỉ lệ tử. Thực thi lớp này trong chương trình.

Lưu ý: Không cho phép dân số nhỏ hơn 1 và số lượng sinh và số lượng mất nhỏ hơn 0.

10. Number Array (Bắt buộc)

Thiết kế 1 lớp có tên là $NumberArray$ cho phép thể hiện một mảng số được khai báo động. Hàm tạo phải cho phép chấp nhận 1 đối số là 1 số nguyên thể hiện kích thước của mảng, hàm này sẽ tự động cấp phát động mảng để có thể lưu trữ nhiều số. Hàm hủy sẽ giải phóng bộ nhớ đợc giữ bởi mảng. Ngoài ra, lớp này có phải có những hàm thành viên để thực hiện những công việc sau:

- Lưu trữ 1 số bất kỳ và bất kỳ phần tử nào của mảng.

- Lấy 1 số từ bất kỳ phần tử nào của mảng.
- Trả về giá trị của số lớn nhất được lưu trữ trong mảng.
- Trả về giá trị của số thấp nhất được lưu trữ trong mảng.
- Trả về giá trị trung bình của tất cả các số được lưu trữ trong mảng

Viết chương trình để thực thi lớp này.

11. Payroll

Thiết kế một lớp $PayRoll$ có các biến thành viên cho mức lương hàng giờ của nhân viên, số giờ làm việc và tổng số tiền lương trong tuần. Viết chương trình với một mảng gồm 7 đối tượng $PayRoll$. Chương trình phải hỏi người dùng về số giờ mỗi nhân viên đã làm việc và sau đó sẽ hiển thị số tiền lương mà mỗi người đã kiếm được.

Lưu ý: Không chấp nhận các giá trị lớn hơn 60 cho số giờ đã làm việc.

12. Coin Toss Simulator

Viết một lớp có tên là $Coin$. Lớp $Coin$ phải có biến thành viên $sideUp$. $sideUp$ là một chuỗi ký tự, sẽ giữ giá trị "heads" hoặc "tails" cho biết mặt của đồng xu ngửa hoặc sấp.

Lớp $Coin$ phải có các hàm thành viên sau:

- Một cấu tử mặc định xác định ngẫu nhiên mặt của đồng xu (sấp hoặc ngửa) và khởi tạo biến thành viên $sideUp$ cho phù hợp.
- Một hàm thành viên có tên là **toss** có chức năng mô phỏng việc tung đồng xu. Khi nào hàm thành viên này được gọi, nó xác định ngẫu nhiên mặt của đồng xu (sấp hoặc ngửa) và đặt biến thành viên $sideUp$ cho phù hợp.
- Một hàm thành viên có tên **getSideUp** trả về giá trị của thành viên $sideUp$

Viết chương trình thể hiện lớp $Coin$. Chương trình sẽ tạo ra một đối tượng của lớp và hiển thị mặt ban đầu của đồng xu. Sau đó, sử dụng một vòng lặp tung đồng xu 20 lần. Mỗi lần tung đồng xu, hiển thị mặt của đồng xu (sấp hay ngửa). Chương trình nên đếm số lần đồng xu ngửa và hiển thị các giá trị đó sau khi vòng lặp kết thúc.

13. Tossing Coins for a Dollar

Đối với nhiệm vụ này, bạn sẽ tạo một chương trình trò chơi bằng cách sử dụng lớp $Coin$ từ bài tập số 12. Chương trình phải có ba đối tượng của lớp $Coin$: một đại diện cho 0.25 xu, một đại diện cho 0.1 xu và một đại diện cho 0.05 xu.

Khi trò chơi bắt đầu, số dư ban đầu của bạn là 0 đô la. Trong mỗi vòng của trò chơi, chương trình sẽ tung các đồng tiền mô phỏng. Khi tung đồng xu, giá trị của đồng xu (số xu mà nó đại điện) sẽ được thêm vào số dư của bạn nếu nó xuất hiện mặt ngửa. Ví dụ: nếu đối tượng đại diện cho 0.25 xu ngửa thì 25 xu được thêm vào số dư của bạn. Không có gì được thêm vào số dư của bạn nếu mọi đối tượng để cho ra mặt sấp. Trò chơi kết thúc khi số dư của bạn đạt 1 đô la trở lên. Nếu số dư của bạn chính xác là 1 đô la, bạn thắng trò chơi. Bạn sẽ thua nếu số dư của bạn vượt quá 1 đô la.

14. Fishing Game Simulation

(Bỏ nếu không có lớp Die)

Đối với bài tập này, bạn sẽ viết một chương trình mô phỏng trò chơi câu cá. Trong này trò chơi, một con xúc xắc sáu mặt được lăn để xác định những gì người dùng đã bắt được. Mỗi vật phẩm có thể có giá trị là một số điểm câu cá nhất định. Điểm sẽ không được hiển thị cho đến khi người dùng câu cá xong, và sau đó một thông báo sẽ hiển thị chúc mừng người dùng tùy thuộc vào số điểm câu cá đạt được.

Dưới đây là một số gợi ý cho thiết kế của trò chơi:

- Mỗi vòng của trò chơi được thực hiện như một lần lặp đi lặp lại càng lâu khi người chơi muốn câu cá để có nhiều vật phẩm hơn.
- Vào đầu mỗi vòng, chương trình sẽ hỏi người dùng liệu họ có muốn tiếp tục câu cá.
- Chương trình mô phỏng sự lăn của một con xúc xắc sáu mặt (sử dụng lớp Die đã được được trình bày trong chương này).

- Mỗi vật phẩm có thể bắt được được đại diện bởi một số được tạo ra từ con xúc xắc. Ví dụ: 1 cho “một con cá lớn”, 2 cho “một chiếc giày cũ”, 3 cho “một con cá nhỏ”, v.v.

- Mỗi vật phẩm mà người dùng bắt được có giá trị số điểm khác nhau.
- Vòng lặp giữ tổng số điểm câu cá của người dùng đang hoạt động.
- Sau khi kết thúc vòng lặp, tổng số điểm câu được hiển thị, cùng với một tin nhắn khác nhau tùy thuộc vào số điểm kiếm được.

15. Mortgage Paymen

Thiết kế một lớp xác định khoản tiền phải thanh toán hàng tháng cho khoản tiền thế chấp nhà. Các khoản thanh toán hàng tháng với lãi suất cộng gộp hàng tháng có thể được tính như sau:

$$Payment = \frac{Loan * \frac{Rate}{12} * Term}{Term - 1}$$

Với $Term = (1 + \frac{Rate}{12})^{12*Year}$

- Peyment: Khoản tiền thanh toán hàng tháng
- Loan: Khoản tiền vay
- Rate: Lãi suất hàng năm
- Years: Số lượng năm vay

Lớp sẽ có các hàm thành viên để thiết lập số tiền cho vay, lãi suất và số năm của khoản vay. Nó cũng phải có các hàm thành viên để trả về số tiền thanh toán hàng tháng và tổng số tiền trả cho ngân hàng khi kết thúc gia đoạn vay. Thực hiện lớp này với một chương trình hoàn chỉnh.

Lưu ý: Không chấp nhận số âm cho bất kỳ giá trị khoản vay nào.

16. Freezing and Boiling Points

Bảng sau liệt kê các điểm đông đặc và sôi của một số chất:

Substance	Freezing Point	Boiling Point
Ethyl Alcohol	−173	172
Oxygen	−362	−306
Water	32	212

Thiết kế một lớp lưu trữ nhiệt độ trong biến thành viên $temperature$ và có chức năng truy cập và biến đổi thích hợp. Ngoài các cấu tử thích hợp, lớp phải có các hàm thành viên sau:

- **isEthylFreezing**: Hàm này sẽ trả về giá trị $bool$ là $true$ nếu nhiệt độ được lưu trữ trong biến $temperature$ bằng hoặc thấp hơn điểm đóng băng của ethylalcohol. Nếu không, hàm sẽ trả về $false$.
- **isEthylBoiling**: Hàm này sẽ trả về giá trị $bool$ là $true$ nếu nhiệt độ được lưu trữ trong biến $temperature$ bằng hoặc cao hơn điểm sôi của rượu etylic. Ngược lại, hàm sẽ trả về giá trị $false$.
- **isOxygenFreezing**: Hàm này sẽ trả về giá trị $bool$ là $true$ nếu nhiệt độ được lưu trữ trong biến $temperature$ bằng hoặc thấp hơn điểm đóng băng của oxy, ngược lại, hàm sẽ trả về giá trị $false$.
- **isOxygenBoiling**: Hàm này sẽ trả về giá trị $bool$ là $true$ nếu nhiệt độ được lưu trữ trong biến $temperature$ bằng hoặc cao hơn điểm sôi của oxy. Ngược lại, hàm sẽ trả về giá trị $false$.
- **isWaterFreezing**: Hàm này sẽ trả về giá trị $bool$ là $true$ nếu nhiệt độ được lưu trữ trong biến $temperature$ bằng hoặc thấp hơn điểm đóng băng của nước, ngược lại, hàm sẽ trả về giá trị $false$.
- **isWaterBoiling**: Hàm này sẽ trả về giá trị $bool$ là $true$ nếu nhiệt độ được lưu trữ trong biến $temperature$ bằng hoặc cao hơn điểm sôi của nước, ngược lại, hàm sẽ trả về giá trị $false$.

Viết chương trình thể hiện lớp. Chương trình sẽ yêu cầu người dùng nhập nhiệt độ và sau đó hiển thị danh sách các chất sẽ đóng băng ở nhiệt độ đó và những chất sẽ sôi ở nhiệt độ đó. Ví dụ, nếu nhiệt độ là -20 thì lớp sẽ báo rằng nước sẽ đóng băng và oxy sẽ sôi ở nhiệt độ đó.

17. Cash Register

(Bỏ nếu $InventoryItem$ chưa được làm)

Thiết kế một lớp $CashRegister$ có thể được sử dụng với lớp $InventoryItem$ được thảo luận trong chương này. Lớp $CashRegister$ sẽ thực hiện những điều sau:

1. Hỏi người dùng về mặt hàng và số lượng được mua.
2. Lấy chi phí của mặt hàng từ đối tượng $InventoryItem$.
3. Thêm 30% lợi nhuận vào chi phí để nhận được đơn giá của mặt hàng.
4. Nhân đơn giá với số lượng được mua để có tổng giá trị mua.
5. Tính thuế bán hàng 6% trên tổng phụ để có được tổng số tiền mua hàng.
6. Hiển thị tổng phụ, thuế và tổng số tiền mua hàng trên màn hình.
7. Trừ số lượng được mua từ biến $onHand$ của đối tượng lớp $InventoryItem$.

Thực hiện cả hai lớp trong một chương trình hoàn chỉnh. Vui lòng sửa đổi lớp $InventoryItem$ theo bất kỳ cách nào cần thiết.

Lưu ý: Không chấp nhận giá trị âm cho số lượng mặt hàng đang được mua.

18. A Game of 21

(Bỏ nếu Die chưa cài đặt)

Đối với nhiệm vụ này, bạn sẽ viết một chương trình cho phép người dùng đấu với người đánh bài trong một biến thể của trò chơi bài xì dách phổ biến. Trong biến thể này của trò chơi, hai viên xúc xắc sáu mặt được sử dụng thay cho thẻ bài. Xúc xắc được tung và người chơi cố gắng đánh bại tổng số ẩn của máy tính mà không vượt quá 21.

Dưới đây là một số gợi ý về thiết kế của trò chơi:

- Mỗi vòng của trò chơi được thực hiện như một lần lặp lại một vòng lặp đi lặp lại miễn là người chơi đồng ý tung xúc xắc, và tổng của người chơi không vượt quá 21.
- Vào đầu mỗi vòng, chương trình sẽ hỏi người dùng xem họ có muốn tung xúc xắc để tích lũy điểm hay không.
- Trong mỗi vòng, chương trình mô phỏng việc tung hai xúc xắc sáu mặt. Đầu tiên nó tung xúc xắc cho máy tính, sau đó hỏi người dùng xem họ có muốn tung xúc xắc hay không. (Sử dụng lớp Die đã được trình bày trong chương này để mô phỏng xúc xắc).
- Vòng lặp giữ tổng số hoạt động của cả máy tính và điểm của người dùng.
- Tổng điểm của máy tính sẽ được ẩn cho đến khi kết thúc vòng lặp - Sau khi kết thúc vòng lặp, tổng điểm của máy tính sẽ được tiết lộ và người chơi có số điểm cao nhất mà không vượt quá 21 chiến thắng.

19. Trivia Game (Bắt buộc)

Trong thử thách lập trình này, bạn sẽ tạo một trò chơi đố đơn giản cho hai người chơi. Chương trình sẽ hoạt động như thế này:

- Bắt đầu với người chơi 1, mỗi người chơi sẽ có lượt trả lời năm câu hỏi đố. (Có tổng cộng 10 câu hỏi.) Khi một câu hỏi được hiển thị, bốn câu trả lời mà người chơi có thể chọn cũng được hiển thị. Chỉ một trong số các câu trả lời là đúng và nếu người chơi chọn câu trả lời đó thì người đó sẽ kiếm được điểm.
- Sau khi các câu trả lời đã được chọn cho tất cả các câu hỏi, chương trình sẽ hiển thị số điểm mà mỗi người chơi kiếm được và tuyên bố người chơi có số điểm cao nhất là người thắng cuộc.

Trong chương trình này, bạn sẽ thiết kế một lớp $Question$ (Câu hỏi) để chứa dữ liệu cho một câu hỏi đố. Lớp $Question$ phải có các biến thành viên cho dữ liệu sau:

- Một câu hỏi đố
- Câu trả lời có thể #1
- Câu trả lời có thể #2
- Câu trả lời có thể #3
- Câu trả lời có thể #4
- Câu trả lời đúng (1, 2, 3 hoặc 4)

Lớp $Question$ phải có (các) hàm tạo, trình truy cập, lấy và gán giá trị cho biến thích hợp.

Chương trình nên tạo một mảng gồm 10 đối tượng $Question$ tương ứng với 10 câu đố. Tự đặt ra các câu hỏi đố của riêng bạn về chủ đề hoặc các chủ đề bạn chọn cho các đối tượng.

20. Patient Fees (Bài tập nhóm)

1. Chương trình này nên được thiết kế và viết bởi một nhóm sinh viên. Dưới đây là các đề xuất:

- Một hoặc nhiều học sinh có thể làm việc trên một lớp duy nhất .
- Các yêu cầu của chương trình phải được phân tích để mỗi học sinh được cung cấp cùng một khối lượng công việc .
- Các tham số và kiểu trả về của từng hàm và các hàm thành viên của lớp nên được quyết định trong trước.
- Chương trình sẽ được thực hiện tốt nhất dưới dạng chương trình nhiều tệp.

1. Bạn phải viết một chương trình tính toán hóa đơn của bệnh nhân cho thời gian nằm viện. Các thành phần khác nhau của chương trình là:

- Lớp $PatientAccount$
- Lớp $Surgery$
- Lớp $Pharmacy$
- Chương trình $main$

Ở đây:

- Lớp $PatientAccount$ sẽ giữ tổng số phí của bệnh nhân. Nó cũng sẽ theo dõi số ngày nằm trong bệnh viện. Nhóm phải quyết định mức phí hàng ngày của bệnh viện.
- Lớp $Surgery$ sẽ lưu trong đó chi phí cho ít nhất năm loại phẫu thuật. Nó có thể cập nhật biến phí của lớp $PatientAccount$.
- Lớp $Pharmacy$ sẽ lưu trong nó giá của ít nhất năm loại thuốc. Nó có thể cập nhật biến phí của lớp $PatientAccount$.
- Sinh viên thiết kế chương trình $main$ sẽ thiết kế một menu cho phép người dùng nhập một loại phẫu thuật và một loại thuốc, đồng thời kiểm tra bệnh nhân của bệnh viện. Khi bệnh nhân kiểm tra, tổng chi phí sẽ được hiển thị.

In [ ]: