BÀI TẬP CLASS
TRONG JAVA – PHẦN 2

<u>Bài 1</u>: Hệ thống quản lý sản phẩm

1. Tạo một lớp có tên Product bao gồm các thuộc tính và phương thức sau:

· String Name

· String Description

· double Price *// 0 < Price <= 100*

· int[ ] rate *// lưu các đánh giá của người dùng cho sản phẩm, giá trị từ 1 - 5*

· void viewInfo() *// hiển thị tên, giá và mô tả về sản phẩm*

2. Tạo lớp Shop gồm các thuộc tính và phương thức sau:

· ArrayList ProductList *// lưu danh sách các sản phẩm của shop*

· void addProduct() *// yêu cầu người dùng nhập thông tin của sản phẩm rồi lưu vào ProductList*

· void removeProduct() *// yêu cầu người dùng nhập vào tên sản phẩm sau đó tìm và xóa sản phẩm có tên tương ứng trong ProductList*

· void iterateProductList() *// hiển thị các sản phẩm trong ProductList, gọi phương thức viewInfo() của lớp Product, tính trung bình cộng đánh giá cho từng sản phẩm và hiển thị thông tin ra màn hình.*

· void searchProduct() *// yêu cầu người dùng nhập vào 2 số, sau đó tìm và hiển thị thông tin của những sản phẩm có giá nằm giữa hai số đó.*

3. Tạo menu:

PRODUCT MANAGEMENT SYSTEM

1. Add new product

2. Remove product
3. Iterate product list
4. Search product
5. Exit

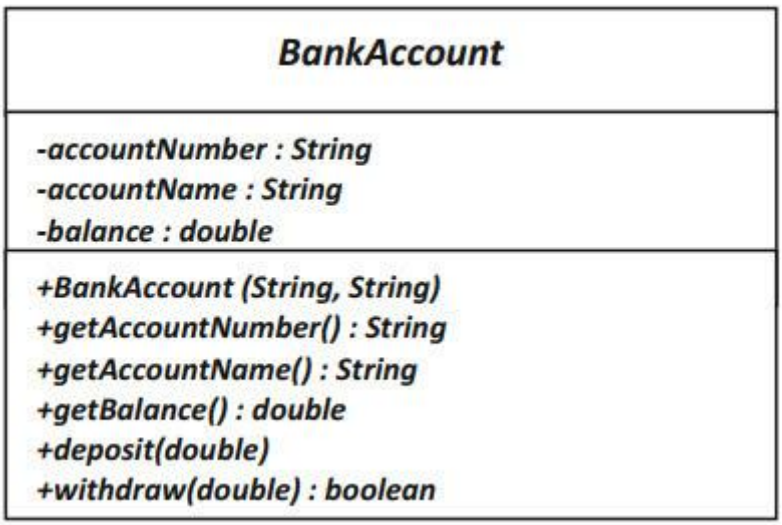
và thực thi các phương thức tương ứng trong lớp Shop với mỗi mục chọn.

Câu hỏi thêm:

Tạo thêm một mục trong Menu ứng với phương thức gọi là sortProduct() đặt trong lớp Shop để sắp xếp các sản phẩm trong ProductList theo giá.

<u>Bài 2</u>: Quản lý tài khoản ngân hàng

Hãy cài đặt lớp BankAcount theo mô tả như hình dưới đây:



Trong đó:

- accountNumber : Số tài khoản ngân hàng

- accountName : Tên chủ tài khoản

- balance : Số tiền có trong tài khoản

+ BankAccount (String, String): Phương thức thiết lập để tạo tài khoản với số tiền =0; tên tài khoản và số tài khoản là tham số truyền vào.

+ getAccountNumber() : Lấy về thông tin số tài khoản

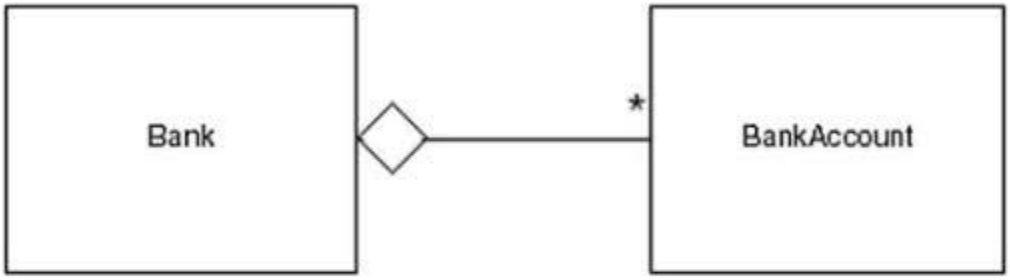
+ getAccountName() : Trả về tên chủ tài khoản

+ getBalance() : Trả về số tiền hiện có trong tài khoản

+ deposit(double): Nạp tiền vào tài khoản

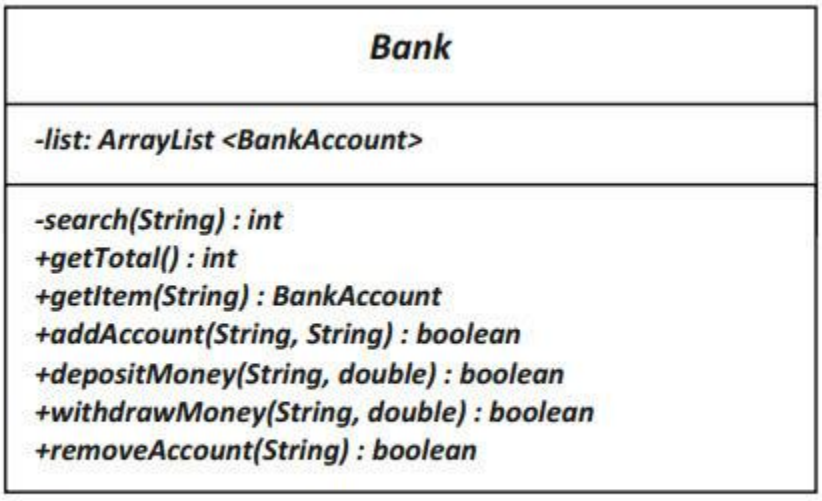
+ withdraw(double) : Chuyển khoản/rút tiền với số tiền được truyền vào là tham số; hàm trả về true nếu số tiền cần rút <= số tiền trong tài khoản; cập nhật lại số tiền trong tài khoản nếu rút thành công; ngược lại hàm trả về false.

Cho lớp Bank có quan hệ với lớp BankAccount như sau



The *Bank* object can contain many *BankAccount* objects

Hãy cài đặt lớp Bank như mô tả dưới đây



.6 The design of the *Bank* class

- list: ArrayList <BankAccount>

Chứa danh sách các tài khoản hiện có tại ngân hàng.

Có 7 phương thức được mô tả như dưới đây:

- search(String) : int

Phương thức này trả về chỉ số của tài khoản ngân hàng có số tài khoản là tham số trong list; nếu không tồn tại tài khoản nào có số tài khoản như tham số thì hàm trả về -1.

+ getTotal(): int

Phương thức này trả về số lượng tài khoản hiện có tại ngân hàng.

+ getItem(String): BankAccount

Phương thức này nhận tham số đầu vào là số tài khoản ngân hàng, và trả về đối tượng BankAccount có số tài khoản là tham số truyền vào; nếu số tài khoản không hợp lệ hàm sẽ trả về giá trị null.

+ addAccount(String, String): boolean

Phương thức này nhận 2 xâu ký tự lần lượt là số tài khoản và tên chủ tài khoản. Nếu số tài khoản đã tồn tại trong hệ thống thì hàm sẽ trả về giá trị false; ngược lại hàm sẽ thêm một tài khoản mới có thông tin là tham số truyền vào và trả về giá trị true.

+ depositMoney(String, double) : boolean

Thực hiện nạp tiền vào tài khoản; nếu tồn tại số tài khoản trong hệ thống thì cập nhật lại số tiền của tài khoản và trả về giá trị true, ngược lại trả về false

+ withdrawMoney(String, double) : boolean

Thực hiện rút tiền từ tài khoản; nếu tồn tại số tài khoản trong hệ thống và số tiền trong tài khoản >= số tiền cần rút thì cập nhật lại số tiền của tài khoản và trả về giá trị true, ngược lại trả về false.

+ removeAccount(String) : boolean

Loại bỏ tài khoản có số tài khoản được truyền vào khỏi danh sách; nếu không tồn tại tài khoản nào có số tài khoản đó thì trả về false; ngược lại trả về true.

Bạn hãy xây dựng chương trình quản lý ngân hàng với các chức năng sau:

- Nhập một danh sách các tài khoản hiện có của ngân hàng
- Sinh ngẫu nhiên số tiền trong tài khoản cho các tài khoản hiện có trong hệ thống
- Hiển thị ra màn hình thông tin các tài khoản cho ngân hàng
- Nạp thêm 500 triệu đồng cho số tài khoản 1050008855
- Thực hiện rút 30 triệu đồng từ tài khoản 1050008854
- Hiển thị in ra màn hình thông báo số dư và tình trạng giao dịch sau khi thực hiện nạp và rút tiền tại các tài khoản ở trên.

Bài 3: Quản lý số

Xây dựng một lớp có tên So gồm các thuộc tính và phương thức sau:

+ Các thuộc tính:

- double number

+ Các phương thức:

- Các phương thức tạo không tham số, đầy đủ tham số
- Các phương thức getter, setter
- inputNumber() để nhập liệu
- showNumber() để hiển thị
- kiemTraNguyen() để kiểm tra number có phải số nguyên không
- kiemTraChanLe()
- kiemTraAmDuong()
- kiemTraChanDuong()
- kiemTraLeAm()
- kiemTraChinhPhuong()
- kiemTraNguyenTo()
- kiemTraHoanHao()
- kiemTraDacBiet()

Bài 4. Viết chương trình OOP quản lý sinh viên đơn giản: Nhập, xuất thông tin, tính điểm TB.
- Viết lớp Sinh viên như sau:
Attributes (private):
- Mã sinh viên là số nguyên.
- Họ tên: chuỗi ký tự.
- Điểm LT, điểm TH : float

Constructor:
- Constructor mặc định (để khởi tạo đối tượng với các thông tin kiểu số là 0, kiểu chuỗi là chuỗi rỗng).
- Constructor thứ hai nhận đầy đủ thông tin để khởi tạo giá trị cho tất cả các biến

instance.
Methods:
- Các getter và setter cho mỗi thuộc tính.
- Tính điểm trung bình.
- Phương thức toString để diễn tả đối tượng ở dạng chuỗi

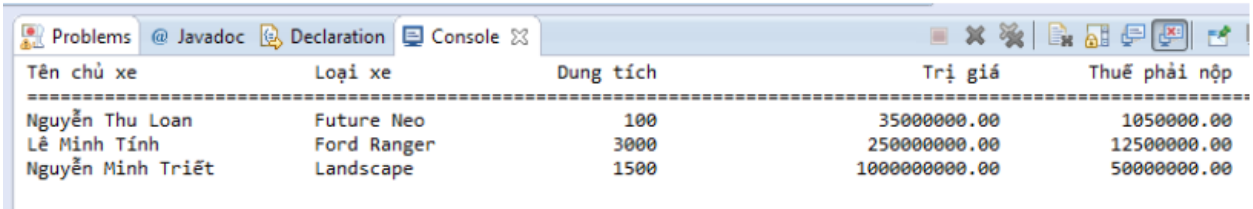
- Xây dựng class chứa hàm main: tạo 3 đối tượng sinh viên sv1, sv2, sv3, trong đó:
- sv1 chứa thông tin của chính mình (tạo bằng constructor đủ thông số, thông tin biết rồi khỏi nhập từ bàn phím).
- sv2 là thông tin người bạn thân nhất của em (tạo bằng constructor đủ thông số, thông tin biết rồi khỏi nhập từ bàn phím).
- sv3 tạo bằng constructor mặc định. Nhập các thông tin cho sv3 từ bàn phím rồi sau đó dùng các setter để gán vào cho các thuộc tính tương ứng.
- In bảng danh sách sinh viên gồm 4 cột là MSSV, họ tên, điểm LT", điểm TH, điểm TB (bảng có 3 dòng cho 3 sinh viên).

HD: *phương thức xuất của đối tượng sinh viên in thông tin trên một dòng có định dạng. Sử dụng*
System.out.printf("chuỗi định dạng", đối số 1, đối số 2, ...); Trong đó chuỗi định dạng giống
c++, ví dụ:
"%-30s": chuỗi, chiếm 30 ký tự, dấu trừ canh lề trái.
"%5.2f" : số thực, chiếm 5 ký tự, bao gồm 2 ký số lẻ.
Ký tự định dạng:
- *s : chuỗi*
- *d: số nguyên (byte, short, int, long)*
- *f: số thực (float, double)*
- *b: boolean*

Bài 5: Sở giao thông cần theo dõi việc đăng ký xe của người dân. Dựa vào thông tin trị giá xe và dung tích xylanh của xe, sở giao thông cũng tính mức thuế phải đóng trước bạ khi mua xe như sau:
- Dưới 100cc, 1% trị giá xe.
- Từ 100 đến 200cc, 3% trị giá xe.
- Trên 200cc, 5% trị giá xe.

Hãy thiết kế và cài đặt class Vehicle với các attributes và methods phù hợp. Class phải có các constructor và phải bảo đảm tính encapsulation.
Xây dựng class chứa hàm main. Hàm main in ra menu lựa chọn các công việc:
1. Nhập thông tin và tạo các đối tượng xe1, xe2, xe3
2. Xuất bảng kê khai tiền thuế trước bạ của các xe.
3. Thoát

Mẫu kết xuất của chương trình:



Bài 6. Ngân hàng ABC muốn lưu trữ thông tin của mỗi tài khoản như sau:
Mỗi tài khoản chứa các thông tin:
- Số tài khoản (Kiểu long),
- Tên tài khoản (kiểu chuỗi),
- Số tiền trong tài khoản (kiểu double)

(a). Thiết kế lớp Account để lưu trữ các thông tin, lớp bao gồm các phương thức sau:
- Constructor: Có 2 constructor (mặc định và đầy đủ tham số)
- Các phương thức get, set cho từng thuộc tính
- Phương thức toString để trả về chuỗi chứa toàn bộ thông tin tài khoản, yêu cầu định dạng tiền tệ.

(b). Thêm các thông tin sau vào lớp Account
- Hằng số lãi suất có giá trị khởi tạo 0.035
- Constructor có 2 đối số: số tài khoản, tên tài khoản. Constructor này sẽ khởi tạo số tiền mặc định là 50.
- Phương thức nạp tiền vào tài khoản: Lấy số tiền hiện tại trong tài khoản + số tiền nạp vào
- Phương thức rút tiền: Lấy số tiền hiện tại trong tài khoản – (số tiền muốn rút+phí rút tiền)
- Phương thức đáo hạn: Mỗi lần đến kỳ đáo hạn thì số tiền trong tài khoản = số tiền trong tài khoản + số tiền trong tài khoản * LAISUAT
- Phương thức chuyển khoản từ tài khoản này sang tài khoản khác

Chú ý: Mỗi thao tác phải kiểm tra số tiền nạp, rút, chuyển có hợp lệ hay không? (VD: tiền nhập vào <0, tiền rút nhiều hơn tiền trong tài khoản thì thông báo không hợp lệ và yêu cầu nhập lại)