Segunda lista de exercícios
Abstração, Encapsulamento e Herança

Exercício 1 (Unidade 1 - Semana 2):

a) O que é uma classe ?

b) O que é um objeto ?

c) O que é um atributo?

d) O que é um método?

e) O que é a assinatura de um método?

f) O que é a palavra **void** na definição de um método e por que alguns métodos não tem esta na sua definição?

g) O que é um construtor e quando ele é invocado?

h) Para que serve o método main?

i) O que significa a sobrecarga de métodos?

j) O que são atributos e métodos estáticos (static) e para que eles servem?

k) Qual a diferença entre um atributo de instância e um atributo de classe?

l) O que é o escopo de uma variável?

m) Para que servem os modificadores de acesso?

Exercício 2 (Unidade 1 - Semana 2):

2.1 – Precisa-se criar um cadastro dos moradores de um edifício para que possam ter acesso através de um código ou digital. Para isso, é preciso criar uma classe para representar os moradores do condomínio com informações importantes para o cadastro. Essas informações são: nome, cpf, celular, data de nascimento, sexo, bloco, apartamento e código de acesso.

2.2 – Criar também uma classe que será responsável pelo cadastro dos moradores. Essa classe deve permitir que o usuário entre com os dados de cada morador e para cada morador, criar o objeto (instância) e mostra os dados do objeto criado, ou seja, do morador cadastrado.

2.3 – Criar uma outra classe que também será uma classe de cadastro porém utilizando um meio de armazenamento em vetores. O usuário deve entrar com os dados do morador, o sistema cria a instância e armazena em um vetor de moradores até o término a entrada de dados. Após o término, o sistema varre o vetor e mostra os dados de todos os moradores cadastrados.

Obs: a entrada de dados do usuário deve ser através de um laço a cada cadastro deve ter uma pergunta se deseja ou não cadastrar mais um morador.

Exercício 3 (Unidade 1 – Semana 3):

3.1 - Em um novo projeto ou pacote, crie uma nova classe de moradores, igual à do item 1 do Exercício 2, mas aplicando as regras de **encapsulamento**, ou seja, **escondendo os atributos e publicando métodos de acesso**. A nova classe deve ter um atributo chamado **código_sequencial** que será gerado automaticamente, de forma sequencial, a cada nova instância criada (usar atributo estático).

3.2 – No mesmo projeto ou pacote, crie uma nova classe de cadastro para instanciar alguns moradores usando os métodos de acesso criados no item 1. As instâncias criadas devem ser armazenadas em listas (usar **ArrayList** ou outro tipo de lista dinâmica). Ao final varrer a lista e mostrar todos os moradores cadastrados.

3.3 – Criar uma classe para representar Robôs aspiradores que se deslocam nos eixos X,Y (usando as regras do encapsulamento).

Um Robô deve ter sua identificação (número), status (ligado, andando, parado e desligado), posicaoX, posicaoY, quantidade de pó, limite de pó.
Um robô executa as seguintes operações: ligar, desligar, aspirar, andar, parar.

Crie um construtor que inicializa o robô como desligado nas posições X=0, Y=0, quantidade de pó = 0 e limite de pó (a definir).

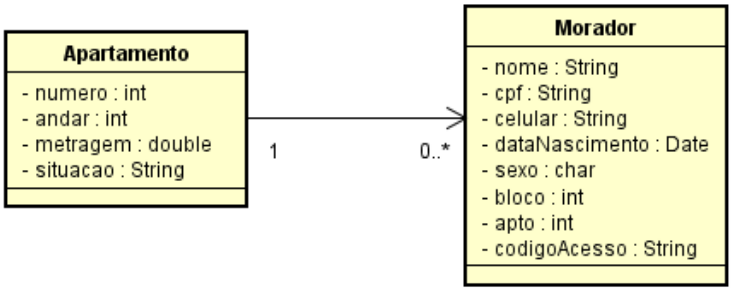
Requisitos das operações:
- as operações de andar, parar e aspirar não são possíveis se o robô estiver desligado, ele precisa primeiramente ser ligado.
- ligar : altera o status do robô para ligado.
- desligar : alterar o status do robô para desligado.
- andar : recebe como parâmetro dois valores) que mostram quantos pontos nos eixos x e y ele vai andar. Soma esses valores recebidos aos atributos posiçãoX e posiçãoY, também altera o status para andando.
- aspirar : recebe um valor referente à quantidade de pó a ser aspirado e soma esse valor ao valor do atributo quantidade de pó. Deve-se observar o limite de pó do aspirador. Caso atinja o limite, o aspirador deve ser automaticamente desligado e informado sobre o limite.
- parar : altera o status para parado

Criar um método toString para mostrar todos os dados do robô.

Criar também uma classe principal que instancie 2 robôs e execute as operações de ligar, desligar, andar, parar, aspirar, mostrando os dados do robô depois de cada operação.

Exercício 4 (Unidade 1 – Semana 4):

4.1 – Crie uma classe para representar os apartamentos de um edifício onde cada apartamento tem as seguintes informações: número (ex:101,201,404,etc), andar, metragem, situação (alugado, financiado, quitado) e uma lista de moradores, conforme o diagrama.
A lista de moradores deve ser um ArrayList da classe “Morador” criada no exercício anterior (Exercício 3 – item 1).



Exercício 5 (Unidade 1 – Semana 5):

Considere a seguinte situação:

Um banco deseja controlar os dados de todos os tipos de pessoas que se relacionam com os seus sistemas, tanto **funcionários** quanto **clientes**. Para isso, foram levantadas algumas informações sobre características de cada tipo de **pessoa**.

- Um caixa de banco é um funcionário e tem como características: nome, telefone, matricula, salário, horário;
- Um gerente é um funcionário e tem como características: nome, telefone, matrícula, salário, bonificação (fixa mensal) e um tipo (PF para gerente de pessoa física ou PJ para jurídica);
- Um cliente tem como características: nome, telefone, idade, cpf, status (“A” – ativo, “I” - inativo)

5.1 - Escreva as classes, com seus atributos, necessárias para representar o enunciado usando o conceito de herança e as boas práticas (atributos protegidos e métodos públicos).

5.2 – Escreva os métodos das classes de acordo com os seguintes requisitos:

- Todos os funcionários do banco podem ter aumento de salário sendo que este é aplicado através da chamada de um método que recebe como parâmetro um percentual e atualiza o salário somando o salário atual com esse percentual;
- Todos os funcionários devem ter um método que retorna o total de recebimento anual do funcionário sendo que os gerentes recebem, além do salário, a bonificação mensal;
- Todo funcionário, ao ser criado, deve ter obrigatoriamente a matrícula e o nome;
- Para a criação de um cliente devem ser informados todos os dados e o status inicial de criação é sempre ativo (criação de construtor);
- Também é possível desativar um cliente invocando um método chamado *desativar* que verifica se o cliente está ativo e troca o seu estado para inativo;
- O cliente deve ter um método para mostrar seus dados concatenados com a informação se ativo ou inativo;
  - ex: “Maria – Tel:2345-6697 – idade: 20 anos – cpf: 8889989898 - ativo”
- Todas as classes devem ter os métodos **gets** e **sets** necessários.

5.3 - Escrever uma classe de Cadastro para instanciar tipos de pessoas diferentes:

- criar um funcionário caixa;

- configurar seu horário, telefone e salário;
- aumentar o salário com um percentual de 10%;
- mostrar os dados do funcionário e o recebimento anual;
- criar um funcionário gerente;
- configurar telefone, salário, bonificação e tipo;
- aumentar o salário em 20%;
- mostrar os dados do funcionário e o recebimento anual;
- criar 3 instâncias de cliente : cliente1, cliente2, cliente3;
- desativar um dos clientes usando o método criado anteriormente;
- mostrar seus dados através do método criado para mostrar os dados do cliente;

FIM DA LISTA 1