Herança e polimorfismo

EXERCÍCIO 46
Leia o capítulo 7 da apostila e resolva os exercícios.

EXERCÍCIO 47
Considere, como subclasse da classe `Pessoa` a classe `Fornecedor`. Cada instância da classe Fornecedor tem, além dos atributos que caracterizam a classe Pessoa, os atributos `creditoMax` (correspondente ao crédito máximo atribuído pelo fornecedor a determinado cliente) e `valorEmDivida` (montante da dívida para com o fornecedor). Implemente na classe Fornecedor, além dos usuais métodos setters/getters e modificadores, um método `obterSaldo` que devolve a diferença entre os valores dos atributos `creditoMax` e `valorEmDivida`.

EXERCÍCIO 48
Depois de implementada a classe Fornecedor, crie um programa de teste adequado que lhe permita verificar o funcionamento dos métodos implementados na classe Fornecedor e os herdados da classe Pessoa.

EXERCÍCIO 49
Considere, como subclasse da classe `Pessoa`, a classe `Empregado`. Considere que cada instância da classe Empregado tem, além dos atributos que caracterizam a classe Pessoa, os atributos `numeroSecao`, `salarioBase` (vencimento base) e IR (porcentagem retida para o Imposto de Renda). Implemente a classe Empregado com métodos setters/getters e modificadores e um método `calcularSalario` (salarioBase - IR). Escreva um programa de teste adequado para esta classe.

EXERCÍCIO 50
Considere, como subclasse da classe Pessoa a classe `Cliente`. Considere que cada instância da classe Cliente tem, além dos atributos que caracterizam a classe Pessoa, os atributos `creditoMax` (correspondente ao crédito máximo concedido ao cliente) e `valorEmDivida`. Implemente na classe Cliente, um método `obterSaldo` que devolve a diferença entre os valores dos atributos `creditoMax` e `valorEmDivida`. Escreva um programa de teste adequado para esta classe.

EXERCÍCIO 51
A classe Funcionário tem, além dos atributos da classe Pessoa, os atributos `numeroSecao`, `salarioBase` (vencimento base) e IR (porcentagem retida para o Imposto de Renda). Implemente a classe Funcionário com métodos setters/getters e modificadores e um método `calcularSalario` (salarioBase - IR).

Implemente a classe `Funcionário` de duas maneiras:
a) Funcionário tem como atributo um objeto pessoa (composição de classes);
b) Funcionário é subclasse de `Pessoa`.

OBS: classe `Pessoa` da lista 5

EXERCÍCIO 52
Implemente a classe `Administrador` como subclasse da classe Funcionário. Um administrador tem, além dos atributos da classe `Funcionário`, o atributo `ajudasDeCusto` (ajudas referentes a viagens, estadias, ...). Note que deverá redefinir na classe Administrador o método herdado `calcularSalario`. O salário de um administrador é equivalente ao salário de um funcionário usual acrescido das ajudas de custo. Escreva um programa de teste adequado para esta classe.

EXERCÍCIO 53
Crie um programa de teste que lhe permita verificar o funcionamento dos métodos implementados na classe `Administrador` e os herdados da classe `Funcionário`. No programa teste use as duas implementações da classe `Funcionário`.

EXERCÍCIO 54
A utilização de OO cria possibilidades interessantes. Por exemplo, partindo da classe `Bacteria` do exercício 19 (lista 2), suponha que o peso da bactéria aumenta um pouco a cada dia (devido ao lixo ingerido, algo como 10% é adicionado ao seu peso corporal). Analise onde deveria ser alterado o código. Faça as seguintes modificações:

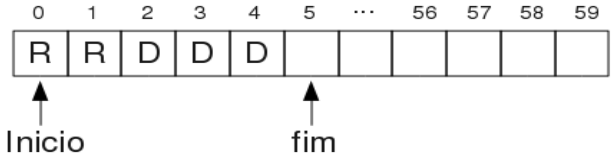
- Adicione 10% ao peso da bactéria ao final de cada 12 horas;
- Nem todas as bactérias durarem exatamente 25 horas, mas uma variação aleatória entre 25 e 45 horas;
- Que 80% das bactérias podem se dividir a cada 3 horas.

Implemente essas modificações na classe `BacteriaMutante`. Faça também o programa simulador (classe `Colonia`) usando `List`. Ver exercício 27 da lista 3.

EXERCÍCIO 55
A empresa trabalha com dois tipos de contêineres: drybox e refrigerado. O contêiner tem as seguintes informações: identificador (único), origem, destino e peso. O contêiner drybox tem um indicador informando se o contêiner pode transportar animais vivos. O contêiner refrigerado guarda dados da temperatura e o tipo de alimentação (elétrica, gasolina ou diesel). Implemente as classes `Container`, `ContainerDryBox` e `ContainerRefrigerado`.

EXERCÍCIO 56
O problema foi modelado em torno de uma classe `Sistema`, que usa o sistema de fila de prioridades para organizar o envio dos contêineres. O sistema obedece as seguintes regras de despacho:

- O próximo *contêiner* a ser despachado é o que está no início da fila;
- Um *contêiner* recebido é inserido no final da fila;
- Um *contêiner* refrigerado tem prioridade no despacho em relação a um drybox;
- Não existe prioridade entre *contêineres* do mesmo tipo.



Implemente a classe Sistema que deve ter métodos para:

a) Adicionar um *contêiner*;
b) Despachar um *contêiner*;
c) Verificar o número de *contêineres drybox* na fila de despacho;
d) Retornar as informações de um contêiner através do seu identificador.

Devido às limitações físicas no pátio da empresa, esta consegue armazenar no máximo 60 *contêineres*.

OBS: Implemente *getters*/*setters* e métodos adicionais apenas quando for necessário.

BOM ESTUDO!