Lista de Exercícios 1

1. Implemente o seguinte programa:

   a) Construir a classe **Funcionário**:
      i) variáveis de instância (visibilidade privada):
         - Nome (String *nome*)
         - Função (String *funcao*)
         - Salário (double *salario*)
      ii) variável estática (visibilidade privada):
         - Quantidade de Funcionários (int *numFuncionarios*)
      iii) Métodos públicos
         - Construtor default;
         - Construtor com três parâmetros, cada um deverá receber valores que irão iniciar cada variável de instância;
         - Métodos **set** e **get** para o acesso às variáveis;
         - **aumentarSalario**, que recebe um percentual de aumento e faz o cálculo;
         - **exibir**, para a visualização dos valores das variáveis de instância;

   b) Construir a classe **Professor**:
      i) herdar da classe Funcionário;
      ii) Métodos Públicos:
         - **Construtor default** (padrão), onde o número mínimo de disciplinas é 2;
         - Um **construtor com parâmetros** (com valores para as variáveis de instância da classe Funcionário e um valor para o número de disciplinas) que inicie todas as variáveis de instância e chame, para ajudá-lo nisso, o construtor da classe base (superclasse);
         - Sobrescrever o método **aumentarSalario**, acrescentando ao percentual de aumento do Professor, um bônus de R$ 100,00 para cada disciplina que o mesmo leciona;
         - Sobrescrever o método **exibir**, para a visualização dos valores das variáveis de instância.

   c) Crie a classe **Escola** para testar as classes acima, onde, essa classe, terá tanto uma instância de **Funcionário**, quanto uma instância de **Professor**. Execute os métodos sobrescritos **aumentarSalario** e **exibir**.

2. Implemente o seguinte programa:

   a) Construir a classe **Transporte**:
      i) variáveis de instância (visibilidade protegida):
         - Marca do Transporte (String *marca*)
         - Modelo do Transporte (String *modelo*)
         - Quilometragem (String *quilometragem*)
         - Capacidade do Tanque (double *capacidadeTanque*)
      ii) Métodos Públicos:
         - Construtor default (padrão);
         - Construtor com quatro parâmetros, cada um deverá receber valores que irão iniciar cada variável de instância;
         - **mover**, que recebe os quilômetros que deve se mover, aumentando sua quilometragem;
         - **abastecer**, que recebe os litros que são colocados no tanque, adicionando à capacidade já existente;
         - **exibir**, para a visualização dos valores das variáveis de instância.

   b) Construir a classe **Automóvel**:

i) Herdar da classe **Transporte**;
ii) Variável de instância (visibilidade privada):
   - Placa (String *placa*)

iii) Métodos Públicos:
   - Essa classe deverá conter os seguintes métodos:
   - Um construtor default (padrão);
   - Um construtor com parâmetros (contendo valores para as variáveis de instância da classe **Transporte** e um valor da placa do automóvel) que inicie todas as variáveis de instância e chame, para ajudá-lo nisso, o construtor da classe base;
   - Sobrescrever o método **moverSe**, que aumenta a quilometragem percorrida, reduzindo 1 litro do tanque a cada 8 quilômetros deslocados;
   - **exibir**, para a visualização dos valores das variáveis de instância, invocando (chamando) o método **exibir** da superclasse e depois exibindo sua placa;

c) Construir a classe **Avião**:
i) Herdar da classe **Transporte**;
ii) Variável de instância (visibilidade privada):
   - Código (String *codigo*)

iii) Métodos Públicos:
   - Um construtor default (padrão);
   - Um construtor com parâmetros que inicie todas as variáveis de instância e chame, para ajudá-lo nisso, o construtor da classe base (esse construtor recebe como parâmetros, valores para as variáveis de instância da classe **Transporte** e um valor do código do Avião);
   - Sobrescrever o método **moverSe**, que aumenta a quilometragem percorrida, reduzindo 10 litros do tanque a cada quilômetro deslocado;
   - **exibir**, para a visualização dos valores das variáveis de instância, invocando (chamando) o método exibir da superclasse e depois exibindo seu código.

d) Construir uma classe para testar a questão acima, onde, essa classe instanciará objetos das três classes (Transporte, Automóvel e Avião) e executará os métodos sobrescritos **moverSe** e **exibir**.

3. Implemente um aplicativo em Java referente a um sistema bancário. Em todas as classes crie dois construtores: um construtor default e outro com argumentos a serem inicializados para todos os seus atributos.
   a) Cada livro possui um código identificador, nome e editora.
   b) Somente usuários cadastrados podem alugar livros da biblioteca. Um usuário normal possui código, nome e CPF. Ele pode alugar um livro gratuitamente por até 15 dias. A partir daí ele pagará multa diária de R$ 1,00 por cada dia excedido. Um monitor é um tipo especial de usuário que possui ainda um determinado valor de bolsa de estudos. Além disso, o mesmo pode alugar um livro gratuitamente por até 30 dias com o mesmo valor de multa caso exceda esse período. Implemente o cálculo da multa a ser paga para todo usuário no caso de aluguel dos livros. Para tanto, implemente um método que receba o número de dias que o usuário passou com o livro e retorne o valor da multa do mesmo.
   c) Construa uma classe principal para instanciar 2 livros, 1 usuário normal e um 1 usuário monitor. Efetue o cálculo da multa para cada um dos usuário instanciados supondo que ambos devolveram o livro após 40 dias de alugado. Mostre na tela quantos livros a biblioteca possui.