Lista de Exercícios Nro. 1

Programação Orientada a Objetos - SCC204

Exercício 1: Descreva algumas diferenças básicas entre programação estruturada e programação orientada a objetos.

Exercício 2: Para que um membro de uma classe-base posam ser acessados por membros da classe derivada, eles devem ser:

A. public B. protected
C. private D. todas as anteriores

Resposta: _ _

Exercício 3: Os membros de uma classe-base podem acessar:

A. membros públicos da classe derivada B. membros protegidos da classe derivada
C. membros privados da classe derivada D. nenhuma das anteriores

Resposta: _ _

Exercício 4: Quais dos seguintes processos são permitidos com classe abstratas?

A. declarar objetos B. retornar um objeto de uma função
C. enviar um objeto como argumento para uma função D. declarar ponteiros

Resposta: _ _

Exercício 5: Para que serve um construtor e como ele pode ser utilizado? Implemente dois exemplos:

- um para ser possível fazer 'Classe("POO")'
- e outro para ser possível fazer 'Classe("POO", 30)'.

Exercício 6: Qual a diferença das linhas de código abaixo?

- private Classe varClasse;
- private Classe varClasse = new Classe();

Exercício 7: Considere os métodos (lebre, os construtores também são métodos) abaixo. Indique se está sendo utilizada sobrecarga de método ou sobrescrita. Explique e diga com que outro método esta sendo realizada a sobrecarga ou sobrescrita (se for o caso).

```
public class A {
    ...
    public A() { ... }
    public A( int x ) { ... }
    public void m1() { ... }
    public void m1( int h ) { ... }
}


public class B extends A {
    ...
    public B() { ... }
    public void m1() { ... }
    public void m1( double x, double y ) { ... }
    public void m2() { ... }
}
```

Exercício 8: Crie uma classe em Java chamada **Data** que inclui três informações como variáveis de instância:

- mês (**int**),
- dia (**int**)
- e ano (**int**).

A classe deve ter métodos get e set para cada variável e um construtor que inicializa as variáveis e assume que os valores fornecidos são corretos. Forneça um método **displayData** que exibe o dia, o mês e o ano separados por barras normais (/). Escreva um aplicativo de teste chamado DataTeste que demonstra as capacidades da classe Data.

Exercício 9: Escreva uma classe chamada **CadernoDeEnderecos** que represente os dados de uma pessoa, como **nome**, **telefone**, **email**, **data de aniversário** e **endereço**. Que campos (variáveis de instância) e métodos essa classe deve ter? Faça uma outra classe para usar/testar a classe CadernoDeEnderecos.

Exercício 10: Crie uma classe **calculadora**. Esta classe deve ser abstrata e implementar as operações básicas (soma, subtração, divisão e multiplicação). Utilizando o conceito de herança crie uma classe chamada **calculadora cientifica** que implementa os seguintes cálculos: raiz quadrada e a potência. Dica utilize a classe Math do pacote java.lang.

Exercício 11: Criar uma estrutura hierárquica que contenha as seguintes classes: **Veiculo** (classe abstrata), **Bicicleta** e **Automóvel**.

Os métodos da classe Veiculo são todos abstratos e possuem a seguinte assinatura:

- public float acelerar(float velocidade);
- public void parar();

Estes métodos são implementados nas subclasses **Automóvel** e **Bicicleta**. Acrescentar na classe Automóvel o método public void mudarOleo(float litros).

Exercício 12: Crie uma classe chamada **Empresa** capaz de armazenar os dados de uma empresa (Nome, Endereço, Cidade, Estado, CEP e Fone). Inclua um construtor sem argumentos e um que receba os dados como argumentos e os inicialize. Escreva duas funções, uma para fazer a interface com o usuário da entrada de dados, Get(), e outra para imprimir os dados, Print().

Use a classe **Empresa** como base para criar a classe Restaurante. Inclua o tipo de comida, o preço médio de um prato, duas funções construtoras, a interface de entrada de dados, Get(), e a função que imprima os dados, Print(). Construa um programa para testar a classe Restaurante.

Exercício 13: Escreva um programa para armazenar dados de veículos. Primeiramente, crie a **classe Motor** que contém NumCilindro (int) e Potenci(int). Inclua um construtor sem argumentos que inicialize os dados com zeros e um que inicialize os dados com os valores recebidos como argumento. Acrescente duas funções, uma para a entrada de dados, Get(), e uma que imprima os dados, Print().

Escreva a **classe Veiculo** contendo Peso em quilos (int), VelocMax em Km/h (int) e Preco em R$ (float). Inclua um construtor sem argumentos que inicialize os dados com os valores recebidos como argumento. Acrescente duas funções, uma para a entrada de dados, Get(), e uma que imprima os dados, Print().

Crie uma **classe CarroPasseio** derivada das classes Motor e Veículo como base. Inclua Cor (string) e Modelo (string). Inclua um construtor sem argumentos que inicialize os dados com zeros e uma que inicialize os dados com os valores recebidos como argumentos. Acrescente duas funções, uma pára a entrada de dados, Get(), e uma que imprima os dados, Print().

Crie uma **classe Caminhao** derivada das classes Motor e Veiculo. Inclua Toneladas (carga máxima), AlturaMax (int) e Comprimento (int). Inclua um construtor sem argumentos que inicialize os dados com zeros e um que inicialize com os valores recebidos como argumento. Acrescente duas funções, uma para a entrada de dados, Get(), e uma que imprima os dados, Print().

Exercício 14: Implemente uma **classe abstracta** de nome **Forma** onde são declarados dois métodos abstractos:

- float calcularArea();
- float cacularPerimetro();

Crie, como subclasse de Forma, uma classe de nome **Rectangulo** cujas instâncias são caracterizadas pelos atributos lado e altura ambos do tipo float. Implemente na classe Rectangulo os métodos herdados de Forma e outros que ache necessários.

Crie, como subclasse de Forma, uma classe de nome **Circulo** cujas instâncias são caracterizadas pelo atributo raio do tipo float. Implemente na classe Circulo os métodos herdados de Forma e outros que ache necessários. Nota: poderá aceder ao valor de Pi fazendo Math.Pi.

Crie, como subclasse de Rectangulo, uma classe de nome **Quadrado** cujas instâncias são caracterizadas por terem os atributos lado e altura com o mesmo valor.

Elabore um programa de teste onde é declarado um array, de dimensão 5, do tipo estático Forma. Nesse array devem ser guardadas instâncias de Rectangulo, Circulo e Quadrado seguindo uma ordem aleatória. Nota: para gerar números aleatórios crie primeiro uma instância da classe Random (presente na biblioteca java.util) e para extrair um inteiro entre 0 e n efectue a evocação nextInt(n). Depois implemente um ciclo que percorra o array evocando, relativamente a cada um dos objectos guardados, os métodos calcularArea e calcularPerimetro.

Exercício 15: Crie uma classe em Java chamada **fatura** para uma loja de suprimentos de informática. A classe deve conter quatro variáveis:

- o número (**String**),
- a descrição (**String**),
- a quantidade comprada de um item (**int**)
- e o preço por item (**double**).

A classe deve ter um construtor e um método get e set para cada variável de instância. Além disso, forneça um método chamado **getTotalFatura** que calcula o valor da fatura e depois retorna o valor como um **double**. Se o valor não for positivo, ele deve ser configurado como 0. Se o preço por item não for positivo, ele deve ser configurado como 0.0. Escreva um aplicativo de teste chamado **FaturaTeste** (em outro arquivo) que demonstra as capacidades da classe Fatura.

Exercício 16: Crie uma classe chamada **Empregado** que inclui três partes de informações como variáveis de instância:

- nome (**String**),
- sobrenome (**String**)
- e um salário mensal (**double**).

A classe deve ter um construtor, métodos get e set para cada variável de instância. Escreva um aplicativo de teste chamado **EmpregadoTeste** que cria dois objetos Empregado e exibe o salário de cada objeto. Então dê a cada Empregado um aumento de 10% e exiba novamente o salário anual de cada Empregado.

Introduza na classe **Empregado** uma variável de classe capaz de contabilizar o numero de empregados que passaram pela empresa até a data.

Exercício 17: Crie uma classe em Java chamada **InteiroSet**. Cada objeto **InteiroSet** pode armazenar inteiros no intervalo de 0 a 100. O conjunto é representado por um array de **booleans**. O elemento do array a[i] é **true** se o inteiro i estiver no conjunto. O elemento do array a[j] é **false** se o inteiro não estiver no conjunto. O construtor sem argumento inicializa o array Java como 'conjunto vazio' (todos os valores **false**).

Forneça os seguintes métodos:

- Método union cria um terceiro conjunto que é a união teórica de dois conjuntos existentes (isto é, aplicação da função lógica OU sobre os conjuntos e retorna o valor lógico **true** ou **false**).
- Método **intersecção** cria um terceiro conjunto que é a intersecção teórica de dois conjuntos existentes (isto é, aplicação da função lógica AND sobre os conjuntos e retorna o valor lógico true ou false).
- Método **insereElemento** insere um novo elemento inteiro k em um conjunto (configurando a[k] como **true**).
- Método **deleteElemento** exclui o inteiro m (configurando a[m] como **false**).
- Método **toSetString** retorna uma string contendo um conjunto como uma lista de números separados por espaço. Inclua somente os elementos que estão presentes no conjunto. Utilize -"para representar um conjunto vazio.
- Método **ehIgualTo** determina se dois conjuntos são iguais.

Exercício 18: Considere a necessidade de desenvolver uma aplicação para armazenar as notas de um Estudante. Apresenta-se a seguir a classe Estudante com algumas partes importantes omitidas.

```
import java.util.*; // classe Vector
public class Estudante {
    // As notas de cada disciplina de cada ano são armazenadas num Vector.
    // Como há 5 anos de estudo o estudante tem um array de 5 Vectors.
    private static final int NUM_ANOS = 5;
    private String nome;
    private int numEstudante;
    private Vector[] notas = new Vector[NUM_ANOS];

    public Estudante(String nome, int numEstudante) {
        // Omitido
    }
    // adiciona nota de uma disciplina de um dado ano ao registro do estudante
    public void adicionaNota (int nota, String disciplina, int ano) {
        NotaDisciplina nd = new NotaDisciplina(nota, disciplina);
        notas[ano].addElement(nd);
    }
    // Terá outro estudante a mesma nota a uma determinada disciplina?
    public boolean igualNota (Object outroEstudante, String disciplina) {
        // Omitido
    }
} // class Estudante
```

- Complete o construtor da classe **Estudante**.
- Crie a classe **NotaDisciplina** com informação da nota de uma disciplina.
- Complete o método **igualNota** da classe **Estudante**. Este método devolve true se **outroEstudante** tem a mesma nota a uma determinada disciplina. Note que o parâmetro **outroEstudante** pode ser **null** ou pertencer a uma classe errada.
- Inclua na classe **Estudante** um método público, que permita determinar a maior nota obtida pelo estudante. Note que o registro do estudante pode conter só algumas notas ou eventualmente nenhuma.