

DET111 [Programação de Computadores I]
Lista de Exercícios - Campos e Métodos Estáticos [22/09/2011]

1. A distância média da Terra à Lua é de aproximadamente 382.000 quilômetros. Usando a classe `ConversaoDeUnidadesDeComprimento`, escreva um programa em Java que mostre qual é a distância média da Terra à Lua em milhas e pés. Escreva métodos adicionais para a classe `ConversaoDeUnidadesDeComprimento`, se necessário.

```
public class ConversaoDeUnidadesDeComprimento {
    public static double polegadasParaCentimetros(double polegadas) {
        double centimetros = polegadas * 2.54;
        return centimetros;
    }

    public static double pésParaCentimetros(double pés) {
        double centimetros = pés * 30.48;
        return centimetros;
    }

    public static double milhasParaQuilometros(double milhas) {
        double quilometros = milhas * 1.609;
        return quilometros;
    }
}
```

2. Escreva a classe `ConversaoDeUnidadesDeArea` com métodos estáticos para conversão das unidades de área segundo a lista abaixo.
 - 1 metro quadrado = 10.76 pés quadrados
 - 1 pé quadrado = 929 centímetros quadrados
 - 1 milha quadrada = 640 acres
 - 1 acre = 43.560 pés quadrados
3. A área de um campo de futebol é de 8.250 metros quadrados. Usando a classe `ConversaoDeUnidadesDeArea`, exercício 2, escreva um programa em Java que mostre qual é a área de um campo de futebol em pés quadrados, acres e centímetros quadrados. Escreva métodos adicionais para a classe `ConversaoDeUnidadesDeArea`, se necessário.
4. Escreva a classe `ConversaoDeUnidadesDeVolume` com métodos estáticos para conversão das unidades de volume segundo a lista abaixo.
 - 1 litro = 1000 centímetros cúbicos
 - 1 metro cúbico = 1000 litros
 - 1 metro cúbico = 35.32 pés cúbicos
 - 1 galão americano = 231 polegadas cúbicas
 - 1 galão americano = 3.785 litros
5. O volume de uma piscina olímpica é de 1.890 metros cúbicos. Usando a classe `ConversaoDeUnidadesDeVolume`, exercício 5, escreva um programa em Java que mostre qual é o volume de uma piscina olímpica em litros, pés cúbicos e centímetros cúbicos. Escreva métodos adicionais para a classe `ConversaoDeUnidadesDeVolume`, se necessário.
6. Escreva a classe `ConversaoDeUnidadesDeTempo` com métodos estáticos para conversão aproximada das unidades de velocidade segundo a lista abaixo.

- 1 minuto = 60 segundos
- 1 hora = 60 minutos
- 1 dia = 24 horas
- 1 semana = 7 dias
- 1 mês = 30 dias
- 1 ano = 365.25 dias

7. O tempo de gestação de um elefante indiano é de aproximadamente 624 dias. Usando a classe `ConversaoDeUnidades-DeTempo`, exercício 7, escreva um programa em Java que mostre qual é o tempo de gestação de um elefante indiano em dias, horas, minutos e segundos. Escreva métodos adicionais para a classe `ConversaoDeUnidadesDeTempo`, se necessário.
8. Escreva uma classe `ConversaoDeTemperatura` que contenha métodos estáticos para calcular a conversão entre diferentes escalas de temperatura. Considere as fórmulas de conversão abaixo:
- De graus Celsius (C) para graus Fahrenheit (F): $F = (9 \times C / 5) + 32$
 - De graus Fahrenheit (F) para graus Celsius (C): $C = (F - 32) \times 5 / 9$
 - De graus Celsius (C) para graus Kelvin (K): $K = C + 273.15$
 - De graus Kelvin (K) para graus Celsius (C): $C = K - 273.15$
 - De graus Celsius (C) para graus Réaumur (Re): $Re = C \times 4 / 5$
 - De graus Réaumur (Re) para graus Celsius (C): $C = Re \times 5 / 4$
 - De graus Kelvin (K) para graus Rankine (R): $R = K \times 1.8$
 - De graus Rankine (R) para graus Kelvin (K): $K = R / 1.8$

Veja que já que existem cinco sistemas de medidas de temperatura, deve haver 20 diferentes métodos de conversão de temperatura. Alguns podem ser escritos indiretamente, por exemplo, para converter de Celsius para Rankine, podemos converter de Celsius para Kelvin e converter esse resultado para Rankine.

9. Escreva um programa em Java que, usando a classe `ConversaoDeTemperatura`, exercício 9, que mostre quantos graus Kelvin e Fahrenheit correspondem a zero e cem graus Celsius.
10. Escreva uma classe que contenha métodos estáticos para retornar o maior e o menor de dois, três, quatro e cinco valores (com um total de oito métodos), considerando que os argumentos e retorno dos métodos podem ser dos tipos `int` e `double`. *Dica:* Os métodos podem ser chamados em cascata: para calcular o maior de três valores a , b e c , pode-se calcular o maior valor de a e b , e comparar esse resultado com c .
11. Escreva uma classe que contenha métodos estáticos para calcular as médias e somas de dois, três, quatro e cinco valores, considerando que os argumentos e retorno dos métodos podem ser dos tipos `int` e `double`. Um total de 16 métodos deverão ser criados.
12. Escreva uma versão da classe `RegistroAcademico` que tenha o campo `numeroDeMatricula` declarado como `static`, e que incremente o valor desse campo cada vez que uma instância da classe for criada. Escreva também uma aplicação que crie algumas instâncias da classe para demonstrar seu funcionamento. Use a classe abaixo como base.

/**

```

* Esta classe gera dados acadêmicos referentes aos alunos que nessa instituição são
matriculados.
*
* @author ()
* @version ( versão 1, 04/06/09)
*/
public class RegistroAcademico {
    private String nome;
    private int numeroMatricula;
    private byte codigoCurso;
    private float percentualCobranca;

    /**
     * Construtor para objetos da classe RegistroAcademico, inicializando todos os campos com
     zero.
     */
    public RegistroAcademico(){
        nome = "";
        numeroMatricula = 0;
        codigoCurso = 0;
        percentualCobranca = 0;
    }

    /**
     * Construtor para objetos da classe RegistroAcademico, inicializando todos os campos
     com os valores
     * dos parâmetros entre os parentes somente o percentualCobranca não recebe nada, será
     * inicializado com zero.
     */
    public RegistroAcademico(String nome, int matricula, byte curso){
        this.nome = nome;
        this.numeroMatricula = matricula;
        this.codigoCurso = curso;
        this.percentualCobranca = 0;
    }

    /**
     * Permite alterar o nome do estudante
     * @param é utilizado o parametro n, para receber o valor nome
     * @return não retorna nada
     */
    public void setNome(String n){
        nome = n;
    }

    /**
     * Permite alterar o código do curso
     * @param é utilizado o parametro c, para receber o valor nome
     * @return não retorna nada
     */
    public void setCurso(byte c){
        codigoCurso = c;
    }

    /**
     * Permite alterar a matrícula do estudante
     * @param é utilizado o parametro d, para receber o valor nome
     * @return não retorna nada
     */
}

```

```

public void setMatricula(int d){
    numeroMatricula = d;
}

/**
 * Permite alterar o percentual de cobrança
 * @param é utilizado o parametro n, para receber o valor nome
 * @return não retorna nada
 */
public void setPercentualCobranca(int n){
    percentualCobranca = n;
}

/**
 * Retorna o nome do estudante
 * @param não utiliza parametro
 * @return retorna o valor do nome
 */
public String getNome(){
    return nome;
}

/**
 * Retorna a matrícula do estudante
 * @param não utiliza parametro
 * @return retorna o valor do do Numero da Matricula
 */
public int getMatricula(){
    return numeroMatricula;
}

/**
 * Retorna uma string contendo o nome, por extenso, do curso
 * @param não utiliza parametro
 * @return retorna o nome do curso (1-Medicin, 2-Nutrição, 3-Administração, 4-
Engenharia de Alimentos
 */
public String getCurso(){
    switch (codigoCurso) {
        case 1: return "Medicina";
        case 2: return "Nutrição";
        case 3: return "Administração";
        case 4: return "Engenharia de Alimentos";
        default: return "Sem curso definido ";
    }
}

/**
 * Retorna a mensalidade do estudante, esta varia de acordo com o curso
 * @param não utiliza parametro
 * @return retorna o valor da mensalidade (1: 2300,00, 2:1150,00, 3: 1200,00, 4:
1750,00
 */
public float getMensalidade(){
    switch (codigoCurso) {
        case 1: return (float)(2300.00 * percentualCobranca)/ 100;
        case 2: return (float)(1500.00 * percentualCobranca)/ 100;
        case 3: return (float)(1200.00 * percentualCobranca)/ 100;
        case 4: return (float)(1750.00 * percentualCobranca)/ 100;
    }
}

```

```

        default: return 0;
    }
}

/**
 * Retorna uma String contendo a ficha completa do estudante
 * @param não utiliza parametro
 * @return retorna o ficha completa do aluno
 */
public String toString(){
    String ficha = "Nome: "+nome+"\n";
    ficha += "Matricula: "+numeroMatricula+ "\n";
    ficha += "Curso: "+codigoCurso+ "\n";
    ficha += "Nome do Curso: "+getCurso()+ "\n";
    ficha += "Percentual de Cobrança: "+percentualCobranca+ "\n";
    ficha += "Mensalidade: "+getMensalidade()+ "\n";
    return ficha;
}
}
/**
 * Esta classe gera dados acadêmicos referentes aos alunos que nessa instituição são matriculados.
 *
 * @author ()
 * @version ( versão 1, 04/06/09)
 */
public class RegistroAcademico {
    private String nome;
    private int numeroMatricula;
    private byte codigoCurso;
    private float percentualCobranca;

    /**
     * Construtor para objetos da classe RegistroAcademico, inicializando todos os campos com zero.
     */
    public RegistroAcademico(){
        nome = "";
        numeroMatricula = 0;
        codigoCurso = 0;
        percentualCobranca = 0;
    }

    /**
     * Constructor for objects of class RegistroAcademico, inicializando todos os campos com os valores
     * dos parametros entre os parentes somente o percentualCobranca não recebe nada, ou seja recebe zero..
     */
    public RegistroAcademico(String nome, int matricula, byte curso){
        this.nome = nome;
        this.numeroMatricula = matricula;
        this.codigoCurso = curso;
        this.percentualCobranca = 0;
    }

    /**
     * Permite alterar o nome do estudante
     * @param é utilizado o parametro n, para receber o valor nome

```

```

* @return não retorna nada
*/
public void setNome(String n){
    nome = n;
}

/**
* Permite alterar o código do curso
* @param é utilizado o parametro c, para receber o valor nome
* @return não retorna nada
*/
public void setCurso(byte c){
    codigoCurso = c;
}

/**
* Permite alterar a matrícula do estudante
* @param é utilizado o parametro d, para receber o valor nome
* @return não retorna nada
*/
public void setMatricula(int d){
    numeroMatricula = d;
}

/**
* Permite alterar o percentual de cobrança
* @param é utilizado o parametro n, para receber o valor nome
* @return não retorna nada
*/
public void setPercentualCobranca(int n){
    percentualCobranca = n;
}

/**
* Retorna o nome do estudante
* @param não utiliza parametro
* @return retorna o valor do nome
*/
public String getNome(){
    return nome;
}

/**
* Retorna a matrícula do estudante
* @param não utiliza parametro
* @return retorna o valor do do Numero da Matricula
*/
public int getMatricula(){
    return numeroMatricula;
}

/**
* Retorna uma string contendo o nome, por extenso, do curso
* @param não utiliza parametro
* @return retorna o nome do curso (1-Medicin, 2-Nutrição, 3-Administração, 4-
Engenharia de Alimentos
*/
public String getCurso(){
    switch (codigoCurso) {

```

```

        case 1: return "Medicina";
        case 2: return "Nutrição";
        case 3: return "Administração";
        case 4: return "Engenharia de Alimentos";
        default: return "Sem curso definido ";
    }
}

/**
 * Retorna a mensalidade do estudante, esta varia de acordo com o curso
 * @param não utiliza parametro
 * @return retorna o valor da mensalidade (1: 2300,00, 2:1150,00, 3: 1200,00, 4:
1750,00
 */
public float getMensalidade(){
    switch (codigoCurso) {
        case 1: return (float)(2300.00 * percentualCobranca)/ 100;
        case 2: return (float)(1500.00 * percentualCobranca)/ 100;
        case 3: return (float)(1200.00 * percentualCobranca)/ 100;
        case 4: return (float)(1750.00 * percentualCobranca)/ 100;
        default: return 0;
    }
}

/**
 * Retorna uma String contendo a ficha completa do estudante
 * @param não utiliza parametro
 * @return retorna o ficha completa do aluno
 */
public String toString(){
    String ficha = "Nome: "+nome+"\n";
    ficha += "Matricula: "+numeroMatricula+ "\n";
    ficha += "Curso: "+codigoCurso+ "\n";
    ficha += "Nome do Curso: "+getCurso()+ "\n";
    ficha += "Percentual de Cobrança: "+percentualCobranca+ "\n";
    ficha += "Mensalidade: "+getMensalidade()+ "\n";
    return ficha;
}
}

```

13. Escreva uma versão da classe `ContaBancariaSimplificada` que tenha um campo `numeroDaConta` declarado como `static`, e que incremente o valor desse campo cada vez que uma instância da classe for criada. Escreva também uma aplicação que crie algumas instâncias da classe para demonstrar seu funcionamento. Use a classe abaixo como base.

```

public class ContaBancariaSimplificada {
    // campos ou variáveis de instâncias
    private String nomeDoCorrentista;
    private float saldo;
    private boolean contaÉEspecial;
    private char sexo;

    /**
     * Constrói objetos da classe ContaBancariaSimplificada
     */
    public ContaBancariaSimplificada() {
        // inicializa campos da classe
    }
}

```

```

    nomeDoCorrentista = "";
    saldo = 0;
    contaÉEspecial = false;
    sexo = ' ';
}

/**
 * Atualiza o nome do correntista
 * @param pNome Recebe o valor passado para o campo nome
 * */

public void setNomeDoCorrentista(String pNome) {
    nomeDoCorrentista = pNome;
}

/**
 * Retorna o nome do correntista
 */
public String getNome() {
    return nomeDoCorrentista;
}

/**
 * Altera o status da conta
 */
public void setContaÉEspecial(boolean pContaEspecial) {
    contaÉEspecial = pContaEspecial;
}

/**
 * Retorna true se a conta é do tipo especial e false caso contrário
 */
public boolean getContaEspecial() {
    return contaÉEspecial;
}

public void setSexo(char s) {
    sexo = s;
}

/**
 * Retorna o sexo do correntista
 */
public char getSexo() {
    return sexo;
}

/**
 * Acrescenta ao campo saldo o valor do depósito passado como parâmetro
 */
public void deposita(float valor) {
    saldo = saldo + valor;
}

/**
 * Retira do campo saldo o valor passado como parâmetro
 */
public void retira(float valor) {
    if (saldo > valor )

```

```

        saldo = saldo - valor;
    else
        System.out.println("Saldo insuficiente");
    }

/**
 * Retorna o saldo da conta
 */
public float getSaldo() {
    return saldo;
}

/**
 * Retorna uma String composta por várias linhas contendo informações armazenadas no
objeto
 */
public String toString () {
    String retorno = "Nome do correntista: "+getNome()+"\n";
    retorno += "Saldo: "+getSaldo()+"\n";
    if (contaÉEspecial)
        retorno += "Tipo: ESPECIAL\n";
    else
        retorno += "Tipo: CONTA COMUM\n";
    if (sexo == 'F') retorno += "Correntista do sexo: feminino\n";
    else retorno += "Correntista do sexo: masculino\n";
    return retorno;
}
}

```

14. Existe um problema em potencial com a classe `SimuladorDeCaixaDeBanco`, mostrada no código abaixo: se uma nova instância da classe for criada em uma aplicação onde já existam algumas instâncias sendo usadas, o número do cliente será "resetado" (voltará a ser zero). Modifique a classe para prevenir esse problema.

```

public class SimuladorCaixaBanco {
    static private int numeroDoCliente;
    private int numeroDoCaixa;

    SimuladorCaixaBanco(int n) {
        numeroDoCaixa = n;
        numeroDoCliente = 0;
        System.out.println("Caixa "+numeroDoCaixa+" iniciou operação.");
    }

    public void proximoAtendimento(){
        numeroDoCliente = numeroDoCliente + 1;
        System.out.print("Cliente com a senha nº "+numeroDoCliente+" , favor ");
        System.out.println("dirigir-s ao caixa nº "+numeroDoCaixa+".");
    }
}

```