Universidade Federal de Lavras
Departamento de Ciência da Computação
GCC – 110 – Programação Orientada a Objetos

Lista de Exercícios 2:

Assunto: Conceitos Básicos de Orientação a Objetos

I – Observações:

1) O trabalho é individual. É permitido discutir os problemas e estratégias de solução com outros colegas, mas quando se tratar de escrever ou implementar computacionalmente as soluções, isto deve ser feito separadamente. O trabalho pode ser feito em qualquer Linguagem OO.

2) Forma de entrega: O trabalho deve ser entregue em formato digital por meio do ***Moodle***. Utilizar a opção ***"Lista de Exercícios 2".*** Anexe **um único arquivo .zip** contendo todos os arquivos do trabalho**.** O nome do arquivo .zip deve conter o primeiro e o último nome do aluno. Por exemplo: **Cristiano_Castro.zip. Obs:** O arquivo deve ser .zip mesmo! Não vale .tar, .tgz, .bz2, e outros formatos de compactação.

3) Trabalhos copiados receberão nota zero para todas as cópias. Trabalhos com erros de compilação não serão avaliados e receberão nota zero**.** Em qualquer erro deve-se emitir uma mensagem adequada ao erro. O programa deve ser desenvolvido seguindo as boas normas de programação.

II – Tarefas:

1. Leia o capítulo denorminado "*Everthing is an Object*" do livro *Thinking in Java*, *Fourth Edition*, páginas 41 a 62. Esse livro pode ser obtido no seguinte endereço: http://mindview.net/Books/TIJ4.
2. Após a leitura mencionada no item 1, resolver os seguintes exercícios do Capítulo "*Everthing is an Object*": 1, 6, 8, 10, 12.
3. Implemente uma classe para a figura geométrica cubo. A classe deve possuir o atributo privado aresta e dois construtores: um contendo um parâmetro que inicializa o valor da aresta do cubo e outro sem parâmetro que inicializa a aresta com valor 10. Crie métodos *get* e *set* para obter e alterar o valor da aresta, respectivamente. Crie um método para obter o volume do cubo. Crie um método *main* para testar a classe. Esse método deve criar dois cubos e chamar cada método da classe para testar os cubos criados.
4. Implemente uma classe simples para manipulação de datas (não é necessário fazer a validação das datas). A classe deve possuir três atributos privados: dia, mês e ano, e três construtores: um contendo os parâmetros dia, mês e ano, outro contendo apenas os parâmetros dia e mês, e um terceiro apenas com um parâmetro dia. Os parâmetros não informados devem ser preenchidos com valores *default*. Mostre o uso da palavra-chave **this**. Crie métodos *get* e *set* para obter e alterar cada atributo da classe. Crie um método *main* para testar o uso da classe, criando, alterando e imprimindo algumas datas.

5. Considere um sistema de uma empresa que vende produtos de informática. Dentre as várias classes desse sistema existe uma classe que define o tipo de dado Produto. Implemente essa classe em Java considerando a seguinte descrição fornecida por um usuário do sistema:

   "Cada produto possui um código de identificação e uma descrição. A empresa necessita manter o controle sobre o preço de venda de seus produtos. É importante ter opções para definir o preço de cada produto e para conceder reajuste de preço de acordo com um percentual. A empresa também necessita manter o controle de estoque de cada produto. A efetuar uma venda, o estoque deve ser decrementado de acordo com a quantidade vendida (o estoque não pode ficar negativo), e ao efetuar uma compra, o estoque deve ser incrementado de acordo com a quantidade comprada."

   Mostre exemplos de uso da classe Produto, criando alguns produtos e chamando as operações definidas na classe.

6. Considere um sistema bancário. Dentre as várias classes desse sistema existe uma classe que define o tipo de dado Conta. Implemente essa classe em Java considerando a seguinte descrição fornecida por um usuário do sistema:

   "Cada conta possui um número de identificação e um saldo. É importante ter opções para saque e depósito em uma conta e para transferência entre contas. Em uma operação de saque, o saldo é decrementado de acordo com o valor sacado. Não é permitido sacar um valor maior do que o saldo. Em uma operação de depósito, o saldo é incrementado de acordo com o valor depositado. E em uma operação de transferência entre contas, o saldo da conta de origem é decrementado de acordo com o valor transferido, se o saldo for suficiente, e o saldo da conta de destino é incrementado de acordo com o valor transferido."

   Mostre exemplos de uso da classe Conta, criando algumas contas e chamando as operações definidas na classe.

7. Implemente uma classe chamada Ponto2D com funcionalidades de um ponto geométrico bi-dimensional. A classe deve possuir como atributos as coordenadas x e y, e três construtores: um sem parâmetros atribuindo coordenadas (0,0), um contendo as coordenadas (x,y) como parâmetro e outro contendo um ponto bi-dimensional como parâmetro. Defina métodos para atribuir e obter as coordenadas do ponto e um método para calcular a distância entre o ponto corrente e outro ponto passado como parâmetro. Crie um método main para testar a classe.

8. Implemente uma classe Triângulo usando como base a classe Ponto2D do exercício anterior. A classe Triângulo deve possuir três atributos do tipo Ponto2D (os vértices do triângulo) e um construtor que recebe três pontos como parâmetros. Defina os seguintes métodos para a classe:
   a) **`static boolean formaTriangulo (Ponto2D p1, Ponto2D p2, Ponto2D p3)`** – retorna True se os três pontos formam um triângulo e False, caso contrário. Em um triângulo, a soma de quaisquer dois lados é maior do que o terceiro lado.
   b) **`boolean equilatero()`** – retorna True se o triângulo é equilátero e False, caso contrário. Um triângulo equilátero possui os três lados iguais.
   c) **`boolean isósceles()`** – retorna True se o triângulo é isósceles e False, caso contrário. Um triângulo isósceles possui dois lados iguais.

d) **`boolean escaleno()`** – retorna True se o triângulo é escaleno e False, caso contrário. Um triângulo escaleno não possui nenhum lado igual a outro.
e) **`double perimetro()`** – retorna o perímetro do triângulo.
f) **`double area()`** – retorna a área do triângulo. Use a seguinte fórmula para cálculo da área, onde sp = semi-perímetro do triângulo:

sp = (lado1 + lado2 + lado3) / 2

área = raiz quadrada de (sp * (sp – lado1) * (sp – lado2) * (sp – lado3))

Crie um método **main** para testar o uso da classe.

9. Implemente em Java um sistema para controle de sua biblioteca pessoal. O sistema é formado pelas classes "Livro" e "Biblioteca" com as características descritas abaixo:

Classe: **Livro**

- Atributos: titulo (título do livro, tipo String), autores (autores do livro, tipo String) e editora (editora que publicou o livro, tipo String). Todos atributos privados;
- Método construtor para inicializar os atributos;
- Métodos *get* para obter cada um dos atributos;
- Métodos *set* para alterar cada um dos atributos.

Classe: **Biblioteca**

- Atributos: livros (lista de livros da biblioteca, tipo arrayList), numLivros (número de livros inseridos na lista, tipo int) e MAXLIV (número máximo de livros na lista, constante do tipo int com valor 50);
- Método para adicionar um livro na lista. Parâmetros: titulo, autores e editora;
- Método para adicionar um livro na lista. Parâmetro: objeto do tipo Livro;
- Método para excluir um livro da lista. Parâmetro: título do livro;
- Método para excluir um livro da lista. Parâmetro: objeto do tipo Livro;
- Método para retornar o livro cujo título é recebido como parâmetro. Retorna nulo se o livro não existir. Parâmetro: título do livro;
- Método para retornar a lista de livros.
- Método para retornar o número de livros da biblioteca.
- Método *main* para executar as seguintes ações, usando todos os métodos definidos nas classes:
  - Criar um objeto do tipo Biblioteca;
  - Adicionar quatro livros à biblioteca;
  - Imprimir os dados de um livro da biblioteca dado seu título, ou uma mensagem de erro se o livro não existir;
  - Imprimir a relação de todos os livros da biblioteca;
  - Excluir dois livros da biblioteca.

10. Idem ao exercício anterior, com as seguintes modificações:

- Na classe Livro, crie uma lista de autores ao invés de armazenar todos os autores em uma única string. Também modifique a informação sobre a editora que publicou o livro, passando agora a ter os seguintes dados sobre cada editora: CNPJ, nome, endereço e telefone;
- Crie uma classe de interface com o usuário com opções para cadastrar e excluir livros (e editoras) de uma biblioteca, pesquisar por um livro imprimindo seus dados e imprimir a relação de todos os livros.