Universidade Federal de Campina Grande – UFCG

Centro de Engenharia Elétrica e Informática – CEEI

Departamento de Sistemas e Computação – DSC

Professores: Gustavo Soares (turma 1)
Kyller Gorgônio (turma 2)
Livia Campos (turma 3)
Raquel Lopes (turma 4)

Disciplina: Laboratório de Programação 2
Período: 2013.2

Laboratório 05

Neste laboratório iremos exercitar os conceitos de encapsulamento e ocultação da informação por meio da criação de classes simples em Java.

Instruções

Este laboratório terá duração de 2 aulas e deverá ser entregue de acordo com as instruções abaixo:

- Data de entrega para todas as turmas: 02/12/2013 até às 23:59h
- Crie o projeto lab05 no eclipse e programe todos os exercícios dentro do pacote lp2.lab05;
- Salve o pacote criado, com as classes implementadas, em um arquivo zip chamado **lab05-turmaX-Lab05-<seuNome>.zip** (ou **.tgz**) e o submeta através do formulario:
  - http://goo.gl/BdYmlE
- Certifique-se de que seus programas não têm erros de compilação;

Antes da definição de cada classe escreva o seu nome em comentário (/*Aluno: <seu nome>*/).

Algumas observações sobre documentação e estilo de programação seguem.

Sobre uso de comentários

As duas formas de comentários mais utilizadas para facilitar o entendimento do código contido nos programas em Java são:

`//`	para uma linha de comentário
`/* ... */`	para um bloco de comentário

Não abuse de comentários dentro dos seus programas, isso não parece um bom sinal. É melhor valorizar os identificadores das variáveis, métodos e classes utilizados. Use comentários internos para esclarecer algum procedimento mais complexo realizado no seu programa;
LEMBRETE: insira um cabeçalho em todos os seus programas identificando o autor do roteiro.

Noções sobre JAVADOC

Existe um utilitário **javadoc** que analisa arquivos-fonte procurando por comentários especiais `/** … */`. Ele gera um arquivo HTML no mesmo formato que a documentação da API Java para os arquivos-fonte analisados. Na verdade, a documentação da API de Java, por exemplo, é o resultado do **javadoc** aplicado aos arquivos-fonte dos seus pacotes.

Cada comentário de documentação automática `/** … */` contém um texto livre, seguido por comandos. Um comando (ou marca) começa com um `@`, como *`@author`* ou *`@param`*. A primeira sentença do texto livre deve ser um enunciado resumido. No texto livre, caso necessário, pode-se utilizar marcadores HTML como ... para itálico, ... para negrito, etc.

O utilitário **javadoc** extrai informações sobre várias partes do seu programa, como por exemplo: pacote, classe pública, interface pública, método público ou protegido, variável ou constante pública ou protegida. **Na aula de hoje estaremos mais interessados em comentários javadoc de "classes" apenas**.

O comentário de uma classe precisa ser colocado depois dos comandos *import* e imediatamente antes da definição *class*. Exemplos de marcas suportadas seguem:
`@author` nome, para indicar o nome do autor da classe.
`@version` texto, para descrever a versão do programa, incluindo datas de criação e de última modificação.

Abaixo segue um exemplo de comentário javadoc para uma classe:

```
/**
   Uma classe para formatar números que seque as convenções de <tt>printf</tt>.
Todas as opções de <tt> printf</tt> são suportadas.
@version 1.01  25 de Junho de 2013
@author Cay Horstmann
@see  "Kernighan and Ritchie. A Linguagem de Programação C, 2ª. Edição."
*/
```

Existem também os comentários de "métodos", que precisam preceder imediatamente a assinatura do método que eles descrevem. Além das marcas de propósito geral, pode-se usar as seguintes marcas:
`@param` variável descrição, para adicionar uma entrada à seção de parâmetros do método atual. A descrição pode ocupar várias linhas e pode usar marcas HTML.
`@return` descrição, para descrever o valor retornado pelo método.

Eis um exemplo de comentário de método:

```
 /**
   Converte um double para string..
   @param x  O número a formatar
   @return  A string formatada
*/
```

Como Extrair Comentários **javadoc**:

i. Vá para o diretório que contém os arquivos-fonte que você quer documentar.
ii. Execute o comando: javadoc –d *diretorioDoc* *.java, para colocar no diretório *diretorioDoc* toda a documentação referente aos seus arquivos-fonte.
iii. Você pode utilizar outras opções ao chamar o **javadoc**, como por exemplo –version –author para incluir os texto de @author e @version (por default, não são incluídos).
iv. Você também pode gerar o javadoc a partir do eclipse: Project > Generate Javadoc

Criando minha primeira classe

Neste lab iremos explorar um pouco do mundo da matemática. No caso, queremos trabalhar com séries matemáticas.

Passo1: Considere uma progressão aritmética com o primeiro termo (a_1) e a razão (r) conhecidos. Podemos, então, gerar os "n" termos seguintes dessa progressão, sabendo que o i-esimo é dado por:

$$a_n = a_1 + (n-1) \cdot r,$$

Implemente uma classe ProgessaoAritmetica.java a partir da documentação fornecida aqui. Lembre-se de usar corretamente os conceitos de encapsulamento e ocultação da informação apresentados em sala.

Para verificar as funcionalidades da sua classe use a classe ExplorandoOMundoDasSeries.java.

Passo2: A partir do conhecimento obtido no passo1, escreva uma classe Fibonacci.java capaz de gerar os termos da conhecida série de Fibonacci.

Passo3: Modifique a classe ExplorandoOMundoDasSeries.java para usar as suas classes ProgressaoAritmetica e Fibonacci. O programa oferece as seguintes opções:

1. Para criar uma progressao aritmetica;
2. Para criar uma serie de Fibonacci;
3. Para ver o termo "n" da sua PA;
4. Para ver o termo "n" da serie de Fibonacci;
5. Para mostrar os "n" primeiros termos da sua PA e da serie de Fibonacci;
6. Para sair.

OBS1. Apenas uma serie de cada tipo pode estar ativa. Caso o usuario já tenha criado uma serie (opções 1 e 2) e novamente deseje criar uma serie de mesmo tipo, substitua a serie anterior pela nova.

OBS2. Uma serie só pode ser manipulada se ela tiver sido previamente criada. Caso o usuário digite a opção 3, por exemplo, sem ter criado a PA usando a opção 1, informe ao usuário que ele nao pode ver o termo de uma PA que não existe e sugira que ele crie a PA antes. Isto deve ocorrer para a série de Fibonacci também;

OBS3. Não esqueça de gerar documentação javadoc para as classes implementadas.

Para exercitar mais

1. Deseja-se criar um sistema de votação eletrônica e para isso será necessário representar os eleitores. Cada eleitor possui nome, cpf e título eleitoral. Note que, um eleitor é uma pessoa cujo título só pode ser concedido quando ela estiver apta a votar (idade maior ou igual a 16 anos). Deve ser possível conferir a idade do eleitor/pessoa e suas informações de identificação. Implemente uma classe em Java para representar os eleitores nesse sistema de votação eletrônica.
2. Agora que vocês já sabem criar suas próprias classes, experimentem voltar ao problema do jogo da velha e resolvê-lo usando Orientação a Objetos (OO). Implemente, uma ou mais classes para representar uma versão OO do jogo da velha. Uma dessas classes, a ser chamada de SistemaDeJogoDaVelha.java, deve conter o método main, que representará a solução do jogo a partir de um conjunto de objetos que serão usados por meio de chamadas aos seus métodos. Seu objetivo nesse momento é criar, pelo menos, uma classe além dessa.

Lembrem-se das regras do jogo da velha:

- Os jogadores não poderão jogar em posições que já foram "ocupadas em jogadas anteriores".
- O mesmo jogador não pode jogar duas vezes seguidas.
- Se um dos jogadores ganhar a partida ela deve ser finalizada, mesmo que o tabuleiro ainda não se encontre completamente preenchido.
- Se ao "ocupar" todas as casas do tabuleiro, nenhum dos jogadores ganhar o jogo deve ser decretado como empate.
- O tabuleiro deve ser exibido no início do jogo para que se tenha certeza que ele está vazio.
- A cada jogada o tabuleiro deve ser exibido para mostrar quais posições encontram-se ocupadas e por qual dos jogadores.