PC II - 2013.2 / 2ª Lista de Exercícios

Em todos os exercícios a aluno deve atender aos requisitos enunciados. Métodos e variáveis auxiliares podem ser criadas e usadas, desde que pertinentes. O aluno deve necessariamente empregar e explorar as características de orientação a objetos do Java:

- Encapsulamento (incluindo modificadores de acesso),
- Herança (de classe e interface) e polimorfismo;
- Sobrecarga de métodos;
- Tratamento e geração de exceções;
- Uso das classes básicas (Object, por exemplo);
- Classes / pacotes

1º Exercício

A classe **Frase** (do 2º exercício da 1ª lista) deve ser modificada para que faça parte de um pacote chamado *br.uerj.ime.lp2.texto*.

Crie neste pacote a exceção verificada *ArquivoVazioEx* que deve ser inicializada com a mensagem "Esse arquivo não contém texto" (ou seja, o construtor deve receber esta string).

Implemente, como membro do pacote *br.uerj.ime.lp2.texto*, a classe *TextoArquivo*, que estende a classe *Frase* e possui os seguintes métodos:

- Um construtor que recebe como parâmetro o nome de um arquivo e abre esse arquivo para leitura;
- *leArquivo*: efetua a leitura de um arquivo e retorna o texto lido, ou lança a exceção *ArquivoVazioEx* caso o arquivo não contenha texto;
- *escreveArquivo*: recebe como argumentos duas strings, uma contendo o nome de um arquivo e a outra contendo um texto. O método deve criar (ou abrir) o arquivo e salvar nele o texto recebido (se o arquivo já existir, o mesmo será sobrescrito);
- *caracteresEspeciais*: recebe como argumento um objeto da classe String, conta e retorna o número de caracteres especiais (como '.' ou ';') existentes nessa String.

Crie uma classe para o programa principal que exiba para o usuário a opção de entrar com o nome de um arquivo ou encerrar o programa. Se o usuário entrar com o nome de um arquivo, o programa deve criar um objeto da classe *TextoArquivo* com o nome de arquivo informado pelo usuário, ler o texto contido nele e contar o número de vogais, consoantes, espaços e caracteres especiais, além de totalizar o número de caracteres contidos no texto. Essas informações devem ser exibidas na tela e também salvas em um arquivo chamado `estatisticas<nomeArquivo>` onde `nomeArquivo` é o nome do arquivo de onde foi lido o texto. O programa só deve ser encerrado quando o usuário selecionar a opção "sair" do menu.

2º Exercício – Banco de Dados de Empregados

A) Crie a exceção não-verificada *EmpregadoNaoCadastradoEx*, que exibe a mensagem "Não há empregado cadastrado com esse código".

B) - Crie uma classe *EmpregadoDB* que implemente a interface BD:

```
public interface BD
{
   public void cadastraEmpregado(Empregado emp);
   public Hashtable getCadastroEmpregado ();
   public void salvaArquivo(Hashtable h, String nomeArq) throws IOException;
   public void leArquivo(String nomeArq) throws FileNotFoundException,
                                                IOException, ClassNotFoundException;
}
```

- o método *cadastraEmpregado* recebe um objeto (*Empregado* ou *EmpregadoContratado*, classes do exercício 4 da lista 1) e o armazena na *Hashtable* utilizando o código como chave.
- *getCadastroEmpregado*: retorna a *Hashtable* que contém o cadastro dos empregados.
- o método *salvaArquivo* recebe como parâmetros um objeto da classe *HashTable* e um objeto da classe *String* contendo o nome do arquivo onde o primeiro parâmetro será salvo.
- *leArquivo*: recebe como parâmetros um objeto da classe *String* contendo o nome do arquivo a ser lido (como o construtor definido abaixo, mas que pode ser invocado a qualquer hora).

A classe *EmpregadoDB* deve ter um campo para o "banco de dados" de empregados na forma de um objeto da classe *Hashtable* e dois construtores:

- um que inicialize o campo (ou seja, crie uma nova instância do objeto *Hastable*);
- um que carregue o banco de dados de empregados, previamente salvo, de um arquivo.

EmpregadoDB deve possuir também um método *getEmpregado* que recebe um código, localiza o objeto correspondente na *Hashtable* e retorna esse objeto (que pode ser da classe *Empregado* ou *EmpregadoContratado*. Cabe a você " sacar" que tipo de objeto o método deverá retornar - polimorfismo)

C) Implementar a classe do programa principal com um menu com as seguintes opções:

- abrir arquivo: solicita ao usuário o nome de um arquivo e carrega a *Hashtable* contida nele, ou exibe uma mensagem e pede um novo nome de arquivo em caso de erro;
- novo cadastro: que cria o objeto da classe *EmpregadoDB* iniciando uma nova *Hastable* (e não lendo a mesma de um arquivo);
- salvar arquivo: solicita ao usuário o nome de um arquivo e salva nele a *Hashtable* contendo o cadastro de empregados.
- cadastrar empregado: lê todos os dados do *Empregado* ou *EmpregadoContratado*, cria o objeto correspondente e o armazena no cadastro. Obs.: crie um sub-menu perguntando se o usuário quer cadastrar um ou outro.
- consultar cadastro: recebe do usuário um código de empregado e exibe na tela os dados do empregado ou uma mensagem caso não haja empregado cadastrado com aquele código.

3º Exercício – Applet Conta Letras

Disponibilize o programa de contagem de letras em uma frase (use a versão da Lista 1 mesmo) em um *Applet*.

Utilize campos de entrada de dados, botões, rótulos (labels), etc..

Faça a previsão de mensagens de erro (em janelas pop-up ou campos de texto).

O *Applet* deve ser acessado pela Internet, através da URL: snarf.ime.uerj.br/~lp2_gXY

Para isso você vai precisar criar um diretório ~/public_html e um arquivo index.html com as permissões e o conteúdo adequado, além das classes em Java.