Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – Campina Grande
Professor: Katyusco de Farias Santos
Curso: Engenharia de Computação
Disciplina: [Laboratório de] Programação Orientada a Objetos

Lista02 – Mini projetos em OO

Observações: Todos os mini projetos da nossa disciplina deverão ser armazenados no repositório SEU do github (o mesmo já utilizado para armazenar os exercícios da disciplina).

Cada mini projeto deverá ter um diretório no SEU github. O nome do diretório deve ser o nome do respectivo mini projeto.

Mini Projeto: Jogo_da_Velha

Deve ser implementado um Jogo da Velha que siga as seguintes regras:

- O tabuleiro é uma matriz de três linhas por três colunas;
- Deve ser jogado por dois jogadores;
- Antes de iniciar o jogo cada jogador deve informar o seu identificador, um nome, por exemplo jose01. Este identificador deve ser usado para armazenar num arquivo todos os seus jogos inclusive indicando se ganhou, perdeu ou empatou;
- Cada jogador tem uma marcação para o tabuleiro, um círculo (O) e um xis (X);
- A interface de marcação do tabuleiro pode ser a caractere via terminal ou através de interface gráfica;
- Os jogadores jogam alternadamente, uma marcação por vez, numa lacuna que esteja vazia;
- A cada jogo finalizado, deve-se armazenar para os dois participantes qual foi o status (ganhou, perdeu, empatou) da sua participação.
- A cada início de jogo, caso o identificador do jogador já exista, deve aparecer o seu perfil de desempenho no jogo, por exemplo jose01: 30 participações: 15 vitórias, 7 derrotas, 8 empates.
- O jogador com pior desempenho entre os dois participantes inicia o jogo. O pior desempenho dado em função do menor número de vitórias, maior número de derrotas, maior número de empates
- Segue um exemplo de sua execução

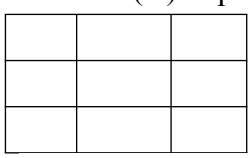
  $> java jogo_da_velha

  jogador O (círculo): jose01

  jogador X (Xis): maria03

  jose01 (O) : 30 participações: 15 vitórias, 7 derrotas, 8 empates

  maria03 (X): 0 participações:



  maria03 deve iniciar o jogo!!!

maria03, informe sua jogada:

Mini Projeto: Contador_Conteudo_Arquivos_Txt

Este programa deve ser capaz de receber como entrada um lista de arquivos em formato txt, como por exemplo livros, e para cada um deles contar de forma paralela suas respectivas quantidades de linhas, palavras, caracteres em branco (espaços e tabulação), vogais, consoantes, dígitos, e caracteres especiais (os demais caracteres que não se inserem nos anteriores).

- Ao fim de sua execução o programa deve gerar um relatório indicando tanto os totais por tipo de contagem por arquivo txt, como também a totalização de contagem por tipo para todos os arquivos txt;
- Como obrigatoriamente deve-se fazer uso de threads para a contagem por arquivo txt, o programa deverá indicar também quanto tempo, em segundos, foi gasto por cada thread para realizar sua respectiva contagem;
- A ordem de apresentação do relatório de contagem por arquivo deve obedecer a ordem de finalização das *threads*. Logo, a thread mais rápida é que apresentará o primeiro relatório e assim sucessivamente até a última *thread* finalizar. Só depois da última thread finalizar é que a totalização geral deverá ser exibida;
- Segue um exemplo de sua execução:

$> java contador_conteudo_arquivos_txt arq01.txt arq02.txt arq03.txt arq04.txt

arq03 processado em 4seg:

linhas: 300, palavras: 800, caracteres em branco: 100, vogais: 2000, consoantes: 4000, dígitos: 10, caracteres especiais: 500

---

arq02 processado em 8seg:

linhas: 600, palavras: 1600, caracteres em branco: 200, vogais: 4000, consoantes: 8000, dígitos: 20, caracteres especiais: 1000

...

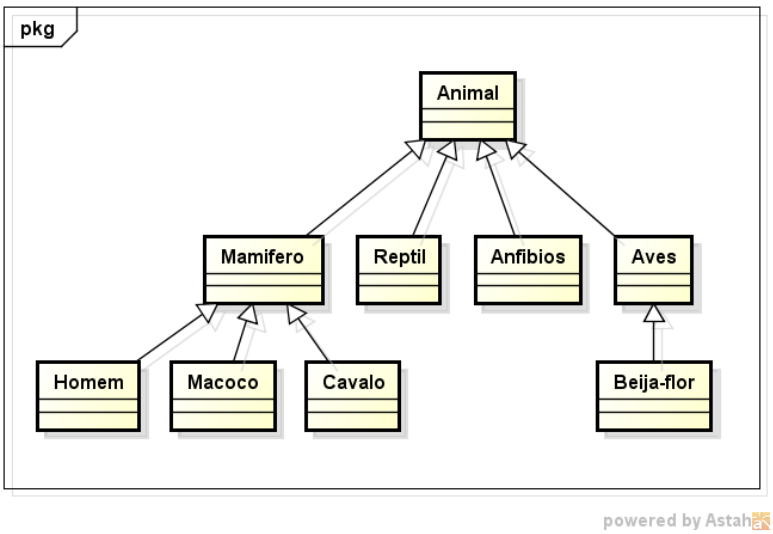
...

arquivos processados em 30 seg:

linhas: 22600, palavras: 11600, caracteres em branco: 1200, vogais: 41000, consoantes: 83000, dígitos: 2330, caracteres especiais: 14000

Mini Projeto: Base de Dados de Classificação de Animais

O diagrama abaixo deve ser utilizado como base para modelagem de um sistema que gerencie uma base de dados de classificação de animais.



- Deve-se evoluir o sistema para permitir que mantendo a mesma estrutura hieráquica de herança seja possível adicionar comportamentos aos animais potencialmente domesticáveis. Por exemplo, Homem e Cavalo são domesticáveis. Também deve-se adicionar comportamentos específicos os animais quando carnívoros ou herbívoros.
- Uma classe Sistema deve ser a responsável por gerenciar a criação, remoção, atualização e armazenamento de quaisquer animais que devam pertencer a sua base de dados. A classe Sistema deverá fazer uso de uma das classes da API Collection de Java para implementar sua base de dados. Basicamente, a classe Sistema deverá conter pelo menos os métodos:

  - adicionaAnimal(Animal);

  - removeAnimal(Animal);

  - atualizaAnimal(Animal);

  - ListaAnimaisArmazenados();
- Enquanto o Sistema estiver em execução a sua base de dados estará armazenda na memória RAM. Implemente uma solução, utilizando a biblioteca de IO(Input e Output) e de Exceções de Java, para que a base de dados possa ser persistida (armazenada numa memória não volátil) no sistema de arquivo atendendo as seguintes exigências:

  - Apenas antes do Sistema ser finalizado, ou seja antes de fechar sua execução corrente, a base de dados deve ser armazenda em arquivo;

  - Apenas antes do Sistema ser iniciado, o conteúdo arquivo que contém os dados salvos da última execução deve ser lido para a base de dados da memória RAM.

  - Checked Excetions devem ser tratadas no método em que elas eventualmente forem utilizadas;

  - Unchecked Excetions independentemente de onde ocorram em relação a pilha de execução de métodos, devem ser tratadas no método public static void main().
- O Sistema deve permitir que o seu operador possa a partir do teclado realizar as seguintes operações:

  - Inserir novos animais;

  - Remover animais existente;

- Alterar algum atributo de um animal existente;
- Solicitar um relatório de todos os animais cadastrados no Sistema.

Mini Projeto: Púlpito para Desktop

Consiste em construir um programa que possa encaminhar mensagens para todas as outras instâncias do mesmo programa que rodam num mesmo computador.

- Cada instância do programa deve ter um único usuário que deve ser diferente de todos os usuários já existentes nas demais instâncias em execução;
- Cada instância do programa executa sobre sua própria JVM.
- O nome do usuário deve ser passado como parâmetro para durante a chamada da instância do programa;
- Todas as mensagens enviadas por quaisquer das instâncias do programa deverão chegar a todas as demais instâncias do programa.