Laboratórios de Informática III (LI3)

LEI – 2º ano – 2º semestre – Universidade do Minho – 2012/2013

F. Mário Martins, João Miguel Fernandes, João Luís Sobral

Projecto de Java: Rede de Autores de artigos científicos

1.- Introdução.

O projecto de Java da disciplina de LI3 tem por objectivo fundamental ajudar à consolidação experimental dos conhecimentos teóricos e práticos adquiridos na disciplina de Programação Orientada pelos Objectos.

O projecto de Java de LI3 está neste ano lectivo muito condicionado pelo tempo despendido no projecto de C da disciplina, e, em consequência, pelo pouco tempo disponível para a sua realização.

Assim sendo, não será solicitada aos alunos a aprendizagem de novas construções da linguagem Java, procurando-se até capitalizar o mais possível do trabalho e dos resultados obtidos no projecto de C, e na utilização de parte desses conhecimentos para a criação de uma estruturação de dados em Java baseada na utilização das colecções e interfaces de JCF ("Java Collections Framework"), que permita a inserção e a realização de consultas interactivas de informações relativas a autores de artigos cientícos publicados no DBLP (tema de base do projecto de C).

2.- Pré-Requisitos.

O projecto de Java tem como pressuposto a existência de um ficheiro de artigos científicos obtido a partir do projecto de C, no qual cada linha representa os autores de uma dada publicação científica apresentada numa conferência ou numa revista no ano indicado.

Cada linha do ficheiro designado por publicx.txt conterá os nomes dos autores de tal publicação separados por vírgulas, sendo terminada pelo ano da respectiva publicação. Para todos os efeitos, deve considerar-se que todas as linhas foram "bem formadas" pelo programa que as criou, ou seja, os nomes dos autores estão correctos, não existem caracteres especiais entre estes a não ser as vírgulas e espaços, o ano poderá ser correctamente convertido para um inteiro (caso não o seja a linha deverá ser aceite e o ano considerado 0) e a linha é terminada pelo carácter de fim de linha do sistema onde foi criada.

Apresentam-se em seguida alguns exemplos destas linhas:

Kim-Sang, Hu Xiang, Alan C. Carter, 2001
John Kalmus, Gustavo Paz, João C. Lopes, 1991
K. Dix, Joana Sousa, Rui Sá, 2013

Em termos estritamente numéricos, teremos portanto um ficheiro com um número garantidamente elevado de linhas, certamente mais de 123.000, correspondendo a um número total de nomes a ler da ordem dos 330.000, dos quais cerca de 150.000 serão nomes distintos.

3.-Requisitos do programa a desenvolver.

Pretende-se desenvolver um programa em Java que seja capaz de, antes de mais, ler e armazenar numa estrutura de dados em memória tais informações, para que sobre a mesma possam ser realizadas diversas consultas ("queries"), algumas estatísticas e alguns testes de "performance".

Tomando por referência o trabalho anterior em C, em que se tornou necessário criar uma Rede de Autores global, ou seja, contendo todos os autores de todos os artigos, pretende-se agora criar uma estrutura em que, a cada ano para o qual existam artigos, se pretende associar a respectiva rede de autores e co-autores, sem prejuízo de podermos, ainda assim, colocar questões globais, ou seja, referentes a todos os anos ou até indiferentemente do ano.

O desenho e a implementação do código do programa deverão ter em atenção que o mesmo deverá ter dois grandes grupos de funcionalidades (correspondentes a 70% do valor do trabalho):

a) Leitura de dados de memória secundária e população das estruturas de dados em memória central; Gravação da estrutura de dados em memória em ficheiros de objectos;
b) Queries: operações de consulta sobre a estrutura de dados;

Complementarmente, equivalendo a 30% do valor do trabalho, pretende-se que seja desenvolvido um conjunto de módulos (ou de pequenos programas) independentes da aplicação anterior que realizem a funcionalidade designada por:

c) Medidas de performance do código e das estruturas de dados.

3.1.- Leitura, população das estruturas e persistência de dados.

O programa deverá poder ler o ficheiro publicx.txt a qualquer momento e carregar os respectivos dados para memória. Na primeira execução do programa a realização desta operação é obrigatória.

A qualquer momento deverá estar disponível uma opção que permita ao utilizador gravar toda a estrutura de dados de forma persistente usando ObjectStreams, criando o ficheiro publicx.obj.

A qualquer momento também deverá o utilizador poder carregar os dados a partir de uma ObjectStream de nome dado, repopulando assim toda a informação da estrutura de dados até então existente em memória.

O arranque da aplicação deve também poder fazer-se inicializando a estrutura de dados a partir de uma ObjectStream sobre o ficheiro publicx.obj.

O programa deverá também poder ler um outro qualquer ficheiro de texto do formato indicado mas cujo nome é dado pelo utilizador.

3.2.- Estatística e queries interactivas.

Sendo inúmeras as possíveis informações a extrair da estrutura de dados, elas deverão ser agrupadas para o utilizador da seguinte forma:

1.- Consultas estatísticas.

1.1.- Apresenta ao utilizador os dados referentes ao último ficheiro de texto lido, designadamente, nome do ficheiro, número total de artigos, número total de nomes lidos, número total de nomes distintos e intervalo fechado dos anos dos artigos lidos (cf. Anos = [1971 a 2012]).

1.2.- Apresenta ao utilizador os números gerais respeitantes aos dados actuais na estrutura, designadamente: número total de autores, número total de artigos de um único autor, número total de autores que apenas publicaram a solo (sem co-autores) e número total de autores que nunca publicaram a solo;

1.3.- Apresenta uma tabela com o número total de publicações registadas em cada ano a considerar, por ordem crescente do ano em questão.

2.- Consultas interactivas.

2.1.- Consultas indexadas por ano ou anos.

Dado um intervalo fechado de anos (os limites podem ser iguais, cf. [1999,1999]), estas consultas deverão determinar:

a) O TOP X de número de publicações (ordem crescente dos nomes) sendo X dado pelo utilizador;
b) O TOP X de co-autorias, ou seja, os X pares de autores e respectivo número de artigos, com mais artigos publicados em co-autoria, sendo X dado pelo utilizador (ordem decrescente do número de artigos);
c) A listagem, por ordem alfabética crescente, de todos os co-autores comuns aos autores de uma dada lista de nomes a introduzir pelo utilizador;
d) A listagem dos nomes dos autores que publicaram artigos em todos os anos do intervalo de anos dado (não fazer para um único ano, mas apenas para intervalos).

2.- Consultas globais especiais.

a) Contar o número de linhas em duplicado no ficheiro **publicx.txt** (o ficheiro deverá ser relido no momento da realização desta consulta);

b) Tabela com todos os autores e respectivas redes de co-autores, mas apenas para associações em que o número de co-autores seja inferior a um número X dado (deve ser devolvida a estrutura completa que satisfaz os requisitos);

A criação das consultas deve ser realizada de forma suficientemente estruturada de modo a que se torne simples alterar o identificador e texto da mesma no menu de consultas e invocar o método associado respectivo.

3.3.- Medidas de performance (30%).

Pretende-se realizar alguns testes de "performance" meramente experimentalistas e apenas com o intuito de introduzir a ideia, ainda não muito interiorizada por razões óbvias, de que os programas são entidades que possuem alguns atributos que são mensuráveis e cuja análise pode ser útil. Estes testes podem e devem ser realizados fora do contexto do programa anteriormente desenvolvido (não farão parte da funcionalidade do programa) usando as partes do código e as estruturas de dados necessárias para cada teste.

Para a realização destes testes sugere-se o uso do método **long System.nanoTime()** (ver exemplo básico de utilização colocado no BB).
Assim, como requisito complementar do projecto e valendo 20% da nota final do mesmo, pede-se aos alunos que realizem os seguintes testes e apresentem os resultados dos mesmos da forma mais simples e ilustrativa possível:

1.- Tempos de leitura (sem parsing) do ficheiro de base, **publicx.txt**, usando as classes **Scanner()** e **BufferedReader()**; Criar em seguida os ficheiros **publicx_x4.txt** e **publicx-x8.txt**, respectivamente com 4x e 8x o número de linhas do original (usar copy & paste), e fazer novas medições de tempos; Realizar os mesmos teste mas incluindo o tempo de parsing.

2.- Sendo para todos evidente que a estrutura de dados do projecto se vai basear na utilização de colecções do tipo **Map<K, V>**, bem como na utilização de colecções que são do tipo **Set<E>** e **List<E>**, comparar as "performances" de alguns dos queries mais complexos que foram pedidos.
Para tal, use **HashMap<>** onde usou **TreeMap<>** ou vice-versa, use **HashSet<>** ou **LinkedHashSet<>** onde usou **TreeSet<>** e vice-versa, e use **Vector<>** onde usou **ArrayList<>**.

Nota: *Estes testes aparentemente complexos e trabalhosos não o são de facto dado que a substituição de uma colecção por outra do mesmo tipo preserva a linguagem, ou seja, por terem a mesma API os métodos são os mesmos. Assim, apenas há que realizar find/replace nas declarações.*
Por outro lado, se programou os seus métodos usando interfaces e não classes concretas para definir os tipos dos parâmetros de entrada e dos resultados nada será alterado.

Afine os testes para o ficheiro **publicx.txt** e depois execute-os para os restantes ficheiros criados no ponto 1. acima.

4.- Apresentação do projecto e Relatório.

O projecto será submetido por via electrónica sob a forma de uma pasta em formato zip/rar do projecto em BlueJ. Serão igualmente aceites projectos de outros IDEs como NetBeans e Eclipse. O código Java avaliado será o código que foi submetido.

O relatório do projecto de Java deverá ter a estrutura já conhecida, sendo de salientar agora a importância do diagrama de classes, do desenho da estrutura de dados usada e da (eventual) apresentação dos resultados dos testes.

O relatório será entregue aquando da apresentação presencial do projecto e servirá de guião para a avaliação do mesmo.

F. Mário Martins