Instituto de Computação
Bacharelado em Ciência da Computação
Disciplina: Técnicas de Programação Avançada
Lista de exercícios para avaliação - 2014.2

Data de Entrega: 14 de Outubro

1) Construa um diagrama de classes em UML para uma livraria online. O sistema deve atender os seguintes requisitos:

- A livraria deve vender livros, mantidos em um catálogo, através de pedidos feitos pela Internet.
- Os livros devem ser armazenados em um estoque mantido por diferentes fornecedores.
- Todo usuário do sistema deve poder adicionar livros em um carrinho de compras *online*, antes do pagamento (*checkout*).
- O usuário deve poder remover itens do carrinho de compras.
- O usuário deve poder cancelar pedidos feitos antes de serem enviados.
- O usuário deve poder pagar por boleto ou por cartão de crédito.
- O usuário deve poder criar uma conta de cliente, de modo que o sistema registre os detalhes do usuário, incluindo nome, endereço, cartão de crédito e etc.
- O sistema deve manter uma lista de contas em sua base de dados.
- Quando um usuário logar, ele dever ter sua senha conferida com uma lista de senhas armazenadas no sistema.
- O usuário deve ser capaz de buscar por livros através de vários métodos de busca (por título, autor, palavras chave e categoria e ver detalhes do livro).

2) Escreva um método genérico que ordene listas genéricas, que implementam a interface List do Java, de acordo com um critério de ordenação arbitrário. Obs.: **Não utilize o método sort da classe utilitária Collections. Utilize apenas as operações da interface.**

3) Implemente uma árvore binária de busca genérica com iteradores para percorrimento em ordem, pré-ordem, pós-ordem e largura.

4) A linguagem Lisp manipula estruturas simbólicas denominadas SExpressions. Uma SExpression é um átomo, isto é um dado considerado indivisível ou então é uma estrutura composta, formada pela sequencia "(", uma SExpression s1, ".", uma segunda SExpression s2 e ")".

Tais expressões simbólicas podem ser interpretadas, armazenadas e manipuladas na forma de uma lista genérica (ou lista de listas), onde cada nó da lista contém duas referências, que podem apontar para um nó ou para um átomo. Exemplos de listas válidas em Lisp são:

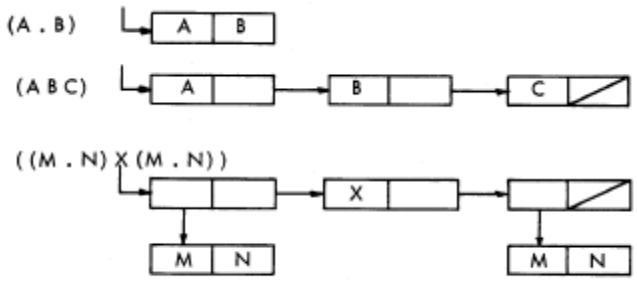

Figura obtida de [1]

Escreva uma classe genérica LispList<T> onde os átomos podem ser do tipo genérico T com as seguintes operações:

cons - retorna uma nova lista a partir da concatenação de um par de elementos que podem ser dois átomos, um átomo e uma lista ou duas listas existentes.

car - retorna o primeiro elemento da lista (cabeça).

cdr - retorna o restante da lista, excluindo o primeiro elemento (cauda).

Obs.: as operações car e cdr têm comportamento indefinido quando aplicadas sobre átomos. Lembre que as operações podem receber qualquer tipo de lista que seja de um subtipo de T.

[1] John McCarthy, Paul W. Abrahams, Daniel J. Edwards, Timothy P.Hart and Michael I. Levin.Lisp 1.5 Programmer's Manual – The Computation Center and Research Laboratory of Eletronics – Massachusetts Institute of Technology. http://www.softwarepreservation.org/projects/LISP/book/LISP%201.5%20Programmers%20Manual.pdf em 03 de junho de 2013.

5) Implemente uma lista circular como uma especialização (customização) de AbstractList<E>

6) Implemente o algoritmo de ordenação HeapSort que atue sobre uma coleção e que use um critério de ordenação que possa ser configurável para os elementos da coleção.

7) Explique como é implementado o método toArrays() da classe ArrayList()

8) Considere um conjunto documentos na forma de arquivos texto. Escreva um programa que receba um conjunto de documentos e determine o grau de similaridade entre cada um deles. Cada documento deve disponibilizar um modo de medir sua similaridade quando confrontado a outro de mesma natureza. Cada documento é caracterizado por um perfil (*profile*) e os perfis devem ser capazes de distinguir documentos diferentes. Existem inúmeras formas de se definir perfis.Uma possibilidade é usar a frequência dos elementos no documento. Um exemplo de perfil é construído com base no conceito de frequência de *k-grams*. Um *k-gram* é uma substring de tamanho *k* na string que compõe o documento. Para construir um histograma de *k-grams* é necessário associar um número a uma string (*hashing*). Perfis podem ser comparados através de um métrica que meça a distância entre os vetores unitários associados ao histograma sendo duas estratégias bem simples as baseadas na métrica euclidiana e no produto escalar. Análise como sua solução responderia em face as seguintes mudanças:

    a) for preciso trabalhar com outros Documentos;

    b) for permitido usar diferentes formas de descrever os documentos ;

    c) for preciso usar métricas diferentes para comparar as similaridades entre os perfis;

    d) for permitida diferentes visualizações dos resultados na interface gráfica

    e) as configurações puderem ser feitas dinamicamente em tempo de execução