Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Programação Orientada a Objetos
Professor: Marco Antonio M. Carvalho

Lista de Exercícios 12 – Genéricos e Coleções

Instruções

- *Todos os exercícios que envolvem programas devem ser resolvidos através de programas em linguagem **Java**;*
- *Na solução dos exercícios, devem ser utilizados os conceitos listados no cabeçalho desta lista;*
- *Para cada exercício, deve ser criados arquivos com nomes "Nome_ListaX_ExercicioY.java", em que Nome denota o nome do aluno, **X** denota o número da lista de exercícios, Y denota o número do exercício;*
- *Os arquivos fonte deverão ser entregues através do Moodle, sem zipar;*
- *Códigos que não compilem serão zerados;*
- *Códigos copiados ou tentativas de trapaça acarretam em perda total da lista de exercícios;*
- *Eventuais dúvidas podem ser sanadas com o professor.*

1. Crie uma versão genérica simplificada do método *isEqualTo*, que compara seus dois argumentos com o método *equals* e retorna *true* se forem iguais e *false* caso contrário. Teste o método genérico com diversos tipos, incluindo *Object*.

2. Crie uma classe genérica *Par* que possui dois parâmetros – *P* e *S* – cada um representando o tipo do primeiro e segundo elementos do par, respectivamente. Adicione *getters* e *setters* para os dois elementos.

3. Crie uma função genérica *localizar*(), que recebe como parâmetros um vetor, um elemento do mesmo tipo que os elementos do vetor e um inteiro informando o tamanho do vetor. A função deve tentar localizar o elemento representado pelo segundo parâmetro no vetor. Caso o elemento seja encontrado, a função deve retornar o índice do elemento, caso contrário, deve retornar -1.

4. Crie uma classe genérica *Vetor*, que recebe como parâmetros o número e o tipo de elementos do vetor. Nesta classe devem ser criados um construtor para alocar o vetor, um método para retornar a quantidade de elementos no vetor (chamado *getSize*()) e outro para adicionar um elemento (chamado *add*()). **Não utilize a classe *Vector***.

5. Escreva uma versão genérica do método *bubbleSort*. Escreva um driver que utiliza o método para vetores e *Integer* e *Float*. Dica: Utilize **< T extends Comparable<T> >** na seção de parâmetros de tipo, para habilitar a utilização do método *compareTo* para comparar dois objetos genéricos.

6. Crie um programa que lê uma série de nomes e os armazena em uma *LinkedList*. Não armazene nomes repetidos. O programa ainda deve permitir que o usuário localize nomes armazenados.

7. Modifique o exemplo *WordTypeCount.java* para contar as ocorrências de cada letra, ao invés de cada palavra.

8. Crie um programa que determina e imprime o número de palavras duplicadas em uma sentença usando coleções. Trate maiúsculas e minúsculas sem diferenciação, e ignore a pontuação.

9. Crie um programa que utilize um *StringTokenizer* para separar uma linha de texto informada pelo usuários e armazene cada *token* em um *TreeSet*, que depois deve ser impresso.

10. Modifique o exemplo *PriorityQueueTest.java* para que o maior número tenha a maior prioridade.