Aula Prática 10

Objetivos

Utilização de Tipos de Dados Genéricos.
Utilização de Genéricos em Coleções.

Problema 10.1

Tendo como base o trabalho desenvolvido na alínea 9.3 do guião anterior, transforme as classes VectorPessoas e ListaPessoas em classes de utilização mais genérica. Isto é, pretendemos utilizar estas coleções com diferentes tipos de dados.

a) A classe VectorPessoas deve passar a chamar-se VectorGeneric e a classe ListaPessoas ListaGeneric. Os métodos incluídos passarão a chamar-se:

```
boolean addElem(T elem)
boolean removeElem (T elem)
int totalElem ()
```

Aproveite para utilizar também a interface Iterator<T> de JAVA.

b) Implemente na classe principal (`TestGeneric`) os seguintes métodos genéricos:

```
double sumArea (...);
// Deve aceitar uma Lista de (sub)tipos Figura e retornar o
somatório das suas áreas

void printSet (...);
// Deve aceitar um iterador (MyIterator) e imprimir todos
os elementos
```

c) Teste as classes e métodos desenvolvidos com um código do tipo:

```
public abstract class TestGeneric {
   public static void main(String[] args) {

        VectorGeneric<Pessoa> vp = new VectorGeneric<Pessoa>();
        for (int i=0; i<10; i++)
              vp.addElem(new Pessoa("Bebé no Vector "+i,
                                    1000+i, Data.today()));
        Iterator<Pessoa> vec = vp.iterator();

        printSet(vp.iterator());

        ListaGeneric<Pessoa> lp = new ListaGeneric<Pessoa>();
        while ( vec.hasNext() )
              lp.addElem( vec.next() );

        Iterator<Pessoa> lista = lp.iterator();
        while ( lista.hasNext() )
              System.out.println( lista.next() );

        ListaGeneric<Figura> figList = new ListaGeneric<Figura>();
        figList.addElem(new Circulo (1,3, 1));
        figList.addElem(new Quadrado(3,4, 2));
```

```
        figList.addElem(new Rectangulo(1,2, 2,5));

        printSet(figList.iterator());

        System.out.println("Soma da Area de Lista de Figuras: " +
                                sumArea(figList));

        // Partindo do principio que Quadrado extends Rectangulo:
        ListaGeneric< Rectangulo > quadList =
                               new ListaGeneric<Rectangulo>();
        quadList.addElem(new Quadrado(3,4, 2));
        quadList.addElem(new Rectangulo(1,2, 2,5));

       System.out.println("Soma da Area de Lista de Quadrados: " +
                                sumArea(quadList));
    }
}
```

Problema 10.2

Relembre o conceito de árvores binárias de pesquisa (Binary Search Trees – BST) abordado em programação II. Complete o código abaixo de forma a construir uma BST genérica.

```
import java.util.Iterator;
import java.lang.Comparable;
..
public class BinarySearchTree<T extends Comparable<? super T>>
            implements Iterable<T> {
// o elemento do tipo T deve ser comparável para efectuar pesquisas
// mas como pode herdar a implementação de compareTo() é mais correcto
// usar <? super T>

      private static class BSTNode<T> {
            T element;
            BSTNode<T> left;
            BSTNode<T> right;

            BSTNode(T theElement) {
                  element = theElement;
                  left = right = null;
            }
      }
      private BSTNode<T> root;
      private int numNodes;

      public BinarySearchTree() {
            root = null;
            numNodes = 0;
      }

      public void insert(T value) {
            root = insert(value, root);
      }

      public void remove(T value) {
            root = remove(value, root);
      }

      public boolean contains(T value) {
            return valueOf(find(value, root)) != null;
      }
      public Iterator<T> iterator() {
      ...
```