APIs de Classes e de Interfaces de JAVA6

APIs DAS PRINCIPAIS CLASSES E INTERFACES JAVA6 USADAS NOS PROJECTOS

F. Mário Martins

DI/UM – V1.1 - 2008

Input

```
/* Métodos de Classe */

public static String lerString()
public static int lerInt()
public static double lerDouble()
public static float lerFloat()
public static boolean lerBoolean()
public static short lerShort()
```

Utilização:

```
palavra = Input.lerString();
valor = Input.lerInt();
temperatura = Input.lerDouble();
taxa = Input.lerFloat();
terminar = Input.lerBoolean();
idade = Input.lerShort();
```

---

public java.util.Scanner;

```
/* Construtores */

public Scanner(File f)
public Scanner(InputStream is) // new Scanner(System.in)

/* Métodos de Instância */

String next()
String nextLine()
tipoSimples nextTipoSimples() // s.nextInt(); s.nextDouble();
void close()
boolean hasNext()
boolean hasxNextTipoSimples()
```

---

public class java.lang.String (instâncias imutáveis)

Constantes: “” “abcd” “Uma linha\n”
“Exemplo\t\tFinal\n\n”
Concatenação: “abc” + “25”; “Luís tem“ + 6 + “ anos\n”

/* Construtores */

public String()
public String(String s)

/* Métodos de Instância */

public String valueOf(tipo_simples val)
public char charAt(int index)
public int length()
public int compareTo(String s)
public String concat(String s)
public boolean contains(String s)
public boolean endsWith(String s)
public int indexOf(char c)
public int indexOf(String s, int i)
public int lastIndexOf(String s)
public String substring(int inic, int fim)
public String toUpperCase()
public String toLowerCase()
public String trim()
public String replace(char velho, char novo)
public boolean equals(String str)
public boolean equalsIgnoreCase(String str)
public char[] toCharArray()

public class java.lang.StringBuilder
public class java.lang.StringBuffer

/* Construtores */

StringBuilder()
StringBuilder(int dim)
StringBuilder(String str)

/* Métodos de Instância */

public StringBuilder append(Object o | *primitivo*)
public char charAt(int index)
public StringBuilder delete(int start, int end)
public StringBuilder deleteCharAt(int index)
public StringBuilder insert(int offset, Object | *primitivo*)
public int length() // ?? size()
public StringBuilder replace(int start, int end, String str)
public StringBuilder reverse()
public void setCharAt(int index, char c)
public String subString(int start)
public String subString(int start, int end)
public String toString()
public Object clone()

public java.lang.Math

/* Constantes de Classe */

public static PI
public static E // valores de PI e da base E

/* Métodos de Classe */

public static numérico abs(tipo_numérico val)
public static double sqrt(double val)
public static double pow(double base, double exp)
public static double random() // valor em [0.0 1.0[
public static *num* max (*num* val1, *num* val2)
public static *num* min (*num* val1, *num* val2)
public static int round(float val)
public static float round(double val)
public static double sin(double val)
public static double cos(double val)

public class java.util.GregorianCalendar

Nome dos Campos (a usar em parâmetros designados *campo*):
Calendar.YEAR
Calendar.MONTH (JANUARY = 0; FEBRUARY = 1; ...)
Calendar.DAY_OF_MONTH;
Calendar.HOUR_OF_DAY;
Calendar.MINUTE;
Calendar.SECOND;
Calendar.MILLISECOND;

```
/* Construtores */

GregorianCalendar()  // data actual do sistema
GregorianCalendar(int ano, int mes_1, int dia)


/* Métodos de Instância */

int get(int campo)
void set(int campo, int valor)
void set(int ano, int mes_1, int dia)
void set(int ano, int mes_1, int dia, int hora_dia, int min, int seg)
long getTimeInMillis()
boolean after(GregorianCalendar cal)
boolean before(GregorianCalendar cal)
boolean equals(Object o)
String toString()

/* Utilidades */

out.printf("%tT%n", agora)                          // 12:23:35
out.printf("%1$tY-%1$tm-%1$td%n", agora); // 2005-03-21
data.set(2007, 3, 12, 16, 35, 14);
int ano = data.get(Calendar.YEAR);
```

public class java.lang.Integer (⇔ para Double, Float, etc.)

/* Construtores */

Integer(String str) throws NumberFormatException
Integer(int value)

/* Constantes de Classe*/

public static final int MAX_VALUE;
public static final int MIN_VALUE;

/* Métodos de Classe */

public static Integer getInteger(String s)
public static int parseInt(String s)
throws NumberFormatException

public static Integer valueOf(String s)
throws NumberFormatException
public static Integer valueOf(*tipoSimples* val)

public int intValue();

// public static *numerico numerico*Value()

public static String toBinaryString(int i)
public static String toHexString(int i)
public static String toOctalString(int i)

public static int compareTo(Integer i)
public static String toString()
public static String toString(int i)

COLECÇÕES DE JAVA

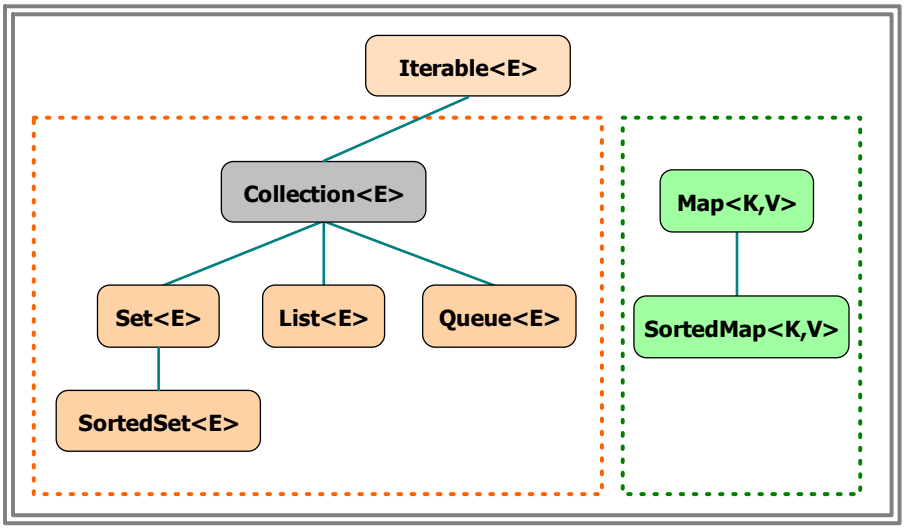

Figura – Tipos de Colecções em JCF

public interface Collection<E> = Set<E>

public abstract boolean add(E elem)
public abstract boolean addAll(Collection c)
public abstract void clear()
public abstract boolean contains(Object o)
public abstract boolean containsAll(Collection c)
public abstract boolean equals(Object o)
public abstract boolean isEmpty()
public abstract Iterator<E> iterator()
public abstract boolean remove(Object o)
public abstract boolean removeAll(Collection c)
public abstract boolean retainAll(Collection c)
public abstract int size()
public abstract Object[ ] to Array()

CLASSES IMPLEMENTADORAS DE List<E>

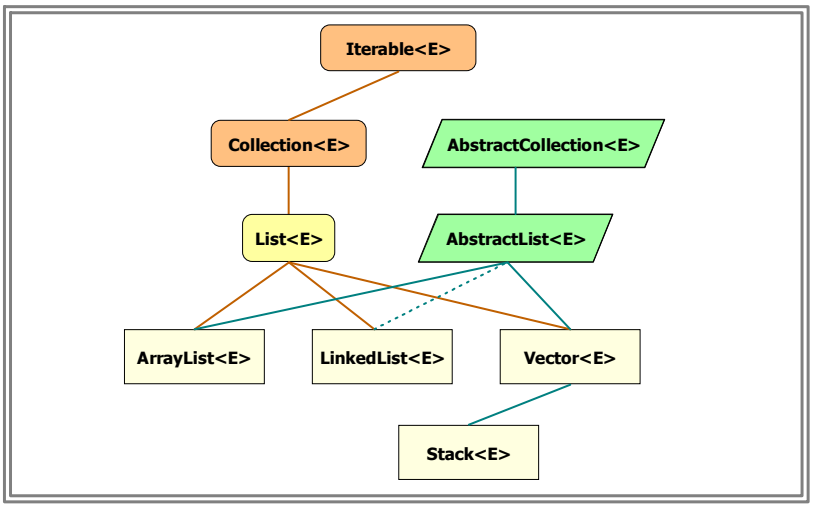

Figura – Classes que implementam List<E>

Categoria de Métodos	API de List<E>

Inserção de elementos	add(E o); add(int index, E o);
	addAll(*Collection*);
	addAll(int i, *Collection*);

Remoção de elementos	remove(Object o); remove(int index); removeAll(*Collection*); retainAll(*Collection*)

Consulta e comparação de conteúdos	E get(int index); int indexOf(Object o);
	int lastIndexOf(Object o);
	boolean contains(Object o);
	boolean isEmpty();
	boolean containsAll(*Collection*);
	int size();

Criação de Iteradores	Iterator<E> iterator(); ListIterator<E> listIterator(); ListIterator<E> listIterator(int index);
Modificação	set(int index, E elem); clear();
Subgrupo	List<E> sublist(int de, int ate);
Conversão	Object[] toArray();
Outros	boolean equals(Object o); boolean isEmpty();

public java.util.ArrayList

/* Construtores */

ArrayList<E>()
ArrayList<E>(*Collection* c)
ArrayList<E>(int capacidade)

/* Métodos de Instância */

public boolean add(E elem)
public boolean add(int index, E elem)
public boolean addAll(*Collection* c)
public boolean addAll(int index, *Collection* c)
public boolean contains(Object elem)
public boolean containsAll(*Collection* c)
public boolean equals(Object o)
public E get(int index)
public int indexOf(Object o)
public boolean isEmpty()
public Iterator<E> iterator()
public int lastIndexOf(Object o)
public Object remove(int index)
public boolean remove(Object o)
public boolean removeAll(*Collection* c)
public boolean retainAll(*Collection* c)
public E set(int index, Object o)
public int size()
public List<E> subList(int from, int to)
public ListIterator<E> listIterator()
public Object clone()

CLASSES IMPLEMENTADORAS DE Set<E>

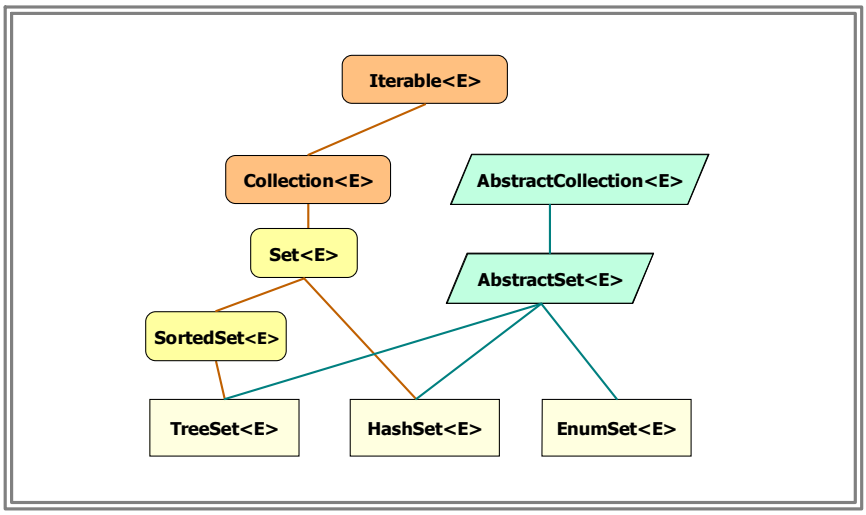

Figura - Classes que implementam Set<E>

Categoria de Métodos	API de Set<E>
Inserção de elementos	add(E o); addAll(Collection<? extends E);

Remoção de elementos	boolean remove(Object o);
	boolean removeAll(Collection<?> c);
	boolean retainAll(Collection<?> c)

Consulta e comparação	boolean contains(Object o);
	boolean isEmpty();
	boolean containsAll(Collection<?> c);
	int size();

Iteradores	Iterator<E> iterator();
Modificação	void clear();
Conversão	Object[] toArray();
Outros	boolean equals(Object o);

public class java.util. HashSet

/* Construtores */

HashSet<E>()
HashSet<E>(*Collection* c)
HashSet<E>(int dim)

/* Métodos de Instância */

public abstract boolean add(E elem)
public abstract boolean addAll(*Collection* c)
public abstract void clear()
public abstract boolean contains(Object o)
public abstract boolean containsAll(*Collection* c)
public abstract boolean equals(Object o)
public abstract boolean isEmpty()
public abstract Iterator<E> iterator()
public abstract boolean remove(Object o)
public abstract boolean removeAll(*Collection* c)
public abstract boolean retainAll(*Collection* c)
public abstract int size()
public abstract Object[ ] to Array()

public class java.util.TreeSet (métodos adicionais a Set<E>)

TreeSet<E>()
TreeSet<E>(*Comparator* c)
TreeSet<E>(*Collection* c)
TreeSet<E>(SortedSet<E> s)

E first()
SortedSet<E> headSet(E toElement)
E last()
SortedSet<E> subSet(E fromElem, E toKey)
SortedSet<E> tailSet(E fromElem)

CLASSES IMPLEMENTADORAS DE Map<K, V>

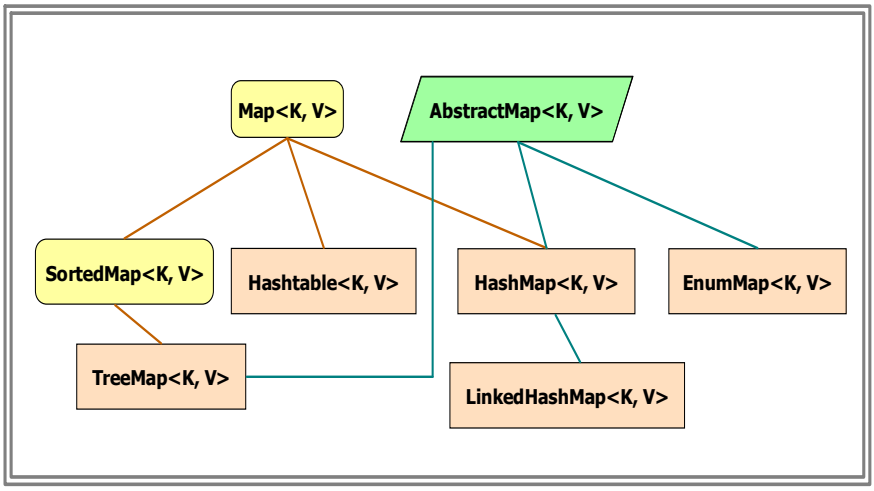

Figura – Implementações de Map<K, V>

Categoria de Métodos	API de Map<K,V>
Inserção de elementos	put(K k, V v); putAll(Map<? extends K, ? extends V> m);
Remoção de elementos	remove(Object k);

Consulta e comparação de conteúdos	V get(Object k);
	boolean containsKey(Object k);
	boolean isEmpty();
	boolean containsValue(Object v); int size();

Criação de Iteradores	Set<K> keySet(); Collection<V> values(); Set<Map.Entry<K, V>> entrySet();
Outros	boolean equals(Object o); Object clone()

public class java.util.HashMap

```
/* Construtores */

HashMap<K, V>()
HashMap<K, V>(int dim)
HashMap<K, V>(Map m)

/* Métodos de Instância */

public void clear()
public int size()
public boolean isEmpty()
public Set<Entry<K,V>> entrySet()
public Set<K> keySet()              // conjunto das chaves
public Collection<V> values()    // colecção dos valores
public boolean containsValue(Object value)
public boolean containsKey(Object key)
public V get(Object key)
public Object put(K key, V value)
public void putAll(Map m)
public Object remove(Object key)
public Object clone()
```

Categoria de Métodos	API de SortedMap<K,V>
Consulta	K lastKey(); K firstKey();
Criação de submapas	SortedMap<K,V> headMap(K ate); SortedMap<K,V> tailMap(K de); SortedMap<K,V> subMap(K de, K ate);

public class java.util.TreeMap (métodos adicionais a Map<K, V>)

/* Construtores */

TreeMap<K, V>()
TreeMap<K, V>(Comparator<? super K> comp)
TreeMap<K, V>(Map<? extends K, ? extends V> map)

/* Métodos de Instância */

K firstKey()
SortedMap<K, V> headMap(K toKey)
K lastKey()
SortedMap<K, V> subMap(K fromKey, K toKey)
SortedMap<K, V> tailMap(K fromKey)

TIPOS ENUMERADOS: java.lang.Enum<E>

```
public enum Cafe { Curto, Normal, Cheio }
public enum Linguagem { Pascal, Modula, Java, C, Csharp }
```

Tipos/Classe: Cafe, Linguagem
Constantes: Curto, Normal, Pascal, Java

```
Cafe c = Café.Curto; Linguagem ling = Linguagem.Java;

/* Métodos de Classe */

public static Enum[] values()   // Cafe[] lstCafes = Cafe.values()
public static String valueOf(constante E)

/* Métodos de Instância */

public String name()
public int ordinal()
public boolean equals(Object o)
public int compareTo(E o)
public String toString()
public Enum getDeclaringClass()
```

public java.util.EnumSet<E> = Set<E> + ...

/* Métodos de Classe – Construtores */

public static EnumSet<E> allOf(*Class Enum<E>*)
// EnumSet<Cafe> cfs = EnumSet.allOf(Cafe.class)
public static EnumSet<E> complementOf(EnumSet<E> es)
public static EnumSet<E> copyOf(EnumSet<E> es)
public static EnumSet<E> copyOf(Collection<E> c)
public static EnumSet<E> noneOf(EnumSet<E> es)
public static EnumSet<E> noneOf(*Class Enum<E>*)
public static EnumSet<E> of(E e)
public static EnumSet<E> of(E e1, E e2);
public static EnumSet<E> of(E e1, ..., E e5)
public static EnumSet<E> range(E prim, E ultimo)

---

public java.util.EnumMap<K, V> = Map<K, V> + ...

/* Métodos de Classe – Construtores */

public EnumMap<K, V>(Class<K> keyType)
public EnumMap<K, V>(*EnumMap* m)
public EnumMap<K, V>(*Map* m)