CS1322 Final Exam Reference Packet

Fall 2002

Table of Contents

You do not need to bring a copy of this packet to the exam, one will be provided for you.

API reference pages were taken from the Java API and modified for this exam.

LLNode

```
public class LLNode {
  private Object data;
  private LLNode next;
  public LLNode() { data = null; next = null; }
  public LLNode(Object data) { this.data = data; next = null; }
  public LLNode(Object data, LLNode next) {
    this.data = data;
    this.next = next;
  }//LLNode(Object LLNode)
  public void setData(Object d) { data = d; }
  public void setNext(LLNode n) { next = n; }
  public Object getData() { return data; }
  public LLNode getNext() { return next; }
}//end class LLNode
```

BSTNode

```
public class BSTNode{
  private Comparable data;
  private BSTNode left;
  private BSTNode right;
  public BSTNode(Comparable d) {
    data=d;
    left=null; right=null;
  }
  public void setData(Comparable d) { data = d; }
  public Comparable getData() { return data; }
  public void setLeft(BSTNode l) { left = l; }
  public BSTNode getLeft() { return left; }
  public void setRight(BSTNode r) { right = r; }
  public BSTNode getRight() { return right; }
}//class BSTNode
```

HashNode

```
public class HashNode{
  private Object data;
  private Object key;
  private HashNode next;
  public HashNode(Object k, Object d){
    key=k;
    data=d;
  }
  public void setData(Object d){
    data=d;
  }
  public Object getData() {
    return data;
  }
  public void setKey(Object k){
    key=k;
  }
  public Object getKey() {
   return key;
  }
  public void setNext(HashNode n) {
   next=n;
  }
  public HashNode getNext() {
    return next;
  }
}//class HashNode
```

GraphNode

```
import java.util.*;
public class GraphNode {
  private Vector edges;
  private String myName;
  public GraphNode(String name) {
    edges=new Vector();
    myName=name;
  }//GraphNode(String)
  public void addEdge(GraphNode newAdjacency) {
    edges.add(newAdjacency);
  }//addEdge(GraphNode)
  public Vector getAdjacencies() {
    return(edges);
  }//getAdjacencies()
  public String toString() {
    return myName ;
  }//toString()
  public boolean equals(Object o) {
    if( o instanceof GraphNode )
      return myName.equals(((GraphNode) o).myName);
    return false;
  }//equals(Object)
}//class GraphNode
```

Enumeration API

All Known Subinterfaces:
NamingEnumeration

All Known Implementing Classes:
StringTokenizer

---

public interface **Enumeration**

Since:
JDK1.0
See Also:
`Iterator`, `SequenceInputStream`, `nextElement()`, `Hashtable`, `Hashtable.elements()`, `Hashtable.keys()`, `Vector`, `Vector.elements()`

Method Summary

`boolean`	**`hasMoreElements()`** Tests if this enumeration contains more elements.
`Object`	**`nextElement()`** Returns the next element of this enumeration if this enumeration object has at least one more element to provide.

StringTokenizer API

```
java.lang.Object
  |
  +--java.util.StringTokenizer
```

All Implemented Interfaces:
Enumeration

---

public class **StringTokenizer**
extends Object
implements Enumeration

Constructor Summary

`StringTokenizer``(String str)` Constructs a string tokenizer for the specified string.
`StringTokenizer``(String str, String delim)` Constructs a string tokenizer for the specified string.
`StringTokenizer``(String str, String delim, boolean returnDelims)` Constructs a string tokenizer for the specified string.

Method Summary

`int`	**`countTokens`**`()` Calculates the number of times that this tokenizer's `nextToken` method can be called before it generates an exception.
`boolean`	**`hasMoreElements`**`()` Returns the same value as the `hasMoreTokens` method.
`boolean`	**`hasMoreTokens`**`()` Tests if there are more tokens available from this tokenizer's string.
`Object`	**`nextElement`**`()` Returns the same value as the `nextToken` method, except that its declared return value is `Object` rather than `String`.
`String`	**`nextToken`**`()` Returns the next token from this string tokenizer.
`String`	**`nextToken`**`(String delim)` Returns the next token in this string tokenizer's string.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

String API

```
java.lang.Object
  |
  +--java.lang.String
```

All Implemented Interfaces:
Comparable, Serializable

public final class **String**
extends Object
implements Serializable, Comparable

Constructor Summary

`String``()` Initializes a newly created `String` object so that it represents an empty character sequence.
`String``(byte[] bytes)` Construct a new `String` by converting the specified array of bytes using the platform's default character encoding.
`String``(byte[] ascii, int hibyte)` **Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the `String` constructors that take a character-encoding name or that use the platform's default encoding.*
`String``(byte[] bytes, int offset, int length)` Construct a new `String` by converting the specified subarray of bytes using the platform's default character encoding.

String(byte[] ascii, int hibyte, int offset, int count) **Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the* `String` *constructors that take a character-encoding name or that use the platform's default encoding.*
String(byte[] bytes, int offset, int length, String enc) Construct a new `String` by converting the specified subarray of bytes using the specified character encoding.
String(byte[] bytes, String enc) Construct a new `String` by converting the specified array of bytes using the specified character encoding.
String(char[] value) Allocates a new `String` so that it represents the sequence of characters currently contained in the character array argument.
String(char[] value, int offset, int count) Allocates a new `String` that contains characters from a subarray of the character array argument.
String(String original) Initializes a newly created `String` object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.
String(StringBuffer buffer) Allocates a new string that contains the sequence of characters currently contained in the string buffer argument.

Method Summary

char	**charAt**(int index) Returns the character at the specified index.
int	**compareTo**(Object o) Compares this String to another Object.
int	**compareTo**(String anotherString) Compares two strings lexicographically.
int	**compareToIgnoreCase**(String str) Compares two strings lexicographically, ignoring case considerations.
String	**concat**(String str) Concatenates the specified string to the end of this string.
static String	**copyValueOf**(char[] data) Returns a String that is equivalent to the specified character array.

`static String`	**`copyValueOf`**`(char[] data, int offset, int count)` Returns a String that is equivalent to the specified character array.
`boolean`	**`endsWith`**`(String suffix)` Tests if this string ends with the specified suffix.
`boolean`	**`equals`**`(Object anObject)` Compares this string to the specified object.
`boolean`	**`equalsIgnoreCase`**`(String anotherString)` Compares this `String` to another `String`, ignoring case considerations.
`byte[]`	**`getBytes`**`()` Convert this `String` into bytes according to the platform's default character encoding, storing the result into a new byte array.
`void`	**`getBytes`**`(int srcBegin, int srcEnd, byte[] dst, int dstBegin)` **Deprecated.** *This method does not properly convert characters into bytes. As of JDK 1.1, the preferred way to do this is via the `getBytes(String enc)` method, which takes a character-encoding name, or the `getBytes()` method, which uses the platform's default encoding.*
`byte[]`	**`getBytes`**`(String enc)` Convert this `String` into bytes according to the specified character encoding, storing the result into a new byte array.
`void`	**`getChars`**`(int srcBegin, int srcEnd, char[] dst, int dstBegin)` Copies characters from this string into the destination character array.
`int`	**`hashCode`**`()` Returns a hashcode for this string.
`int`	**`indexOf`**`(int ch)` Returns the index within this string of the first occurrence of the specified character.
`int`	**`indexOf`**`(int ch, int fromIndex)` Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
`int`	**`indexOf`**`(String str)` Returns the index within this string of the first occurrence of the specified substring.
`int`	**`indexOf`**`(String str, int fromIndex)` Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
`String`	**`intern`**`()` Returns a canonical representation for the string object.

`int`	**`lastIndexOf`**`(int ch)` Returns the index within this string of the last occurrence of the specified character.
`int`	**`lastIndexOf`**`(int ch, int fromIndex)` Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.
`int`	**`lastIndexOf`**`(String str)` Returns the index within this string of the rightmost occurrence of the specified substring.
`int`	**`lastIndexOf`**`(String str, int fromIndex)` Returns the index within this string of the last occurrence of the specified substring.
`int`	**`length`**`()` Returns the length of this string.
`boolean`	**`regionMatches`**`(boolean ignoreCase, int toffset, String other, int ooffset, int len)` Tests if two string regions are equal.
`boolean`	**`regionMatches`**`(int toffset, String other, int ooffset, int len)` Tests if two string regions are equal.
`String`	**`replace`**`(char oldChar, char newChar)` Returns a new string resulting from replacing all occurrences of `oldChar` in this string with `newChar`.
`boolean`	**`startsWith`**`(String prefix)` Tests if this string starts with the specified prefix.
`boolean`	**`startsWith`**`(String prefix, int toffset)` Tests if this string starts with the specified prefix beginning a specified index.
`String`	**`substring`**`(int beginIndex)` Returns a new string that is a substring of this string.
`String`	**`substring`**`(int beginIndex, int endIndex)` Returns a new string that is a substring of this string.
`char[]`	**`toCharArray`**`()` Converts this string to a new character array.
`String`	**`toLowerCase`**`()` Converts all of the characters in this `String` to lower case using the rules of the default locale, which is returned by `Locale.getDefault`.

String	**toLowerCase**(Locale locale) Converts all of the characters in this String to lower case using the rules of the given Locale.
String	**toString**() This object (which is already a string!) is itself returned.
String	**toUpperCase**() Converts all of the characters in this String to upper case using the rules of the default locale, which is returned by Locale.getDefault.
String	**toUpperCase**(Locale locale) Converts all of the characters in this String to upper case using the rules of the given locale.
String	**trim**() Removes white space from both ends of this string.
static String	**valueOf**(boolean b) Returns the string representation of the boolean argument.
static String	**valueOf**(char c) Returns the string representation of the char argument.
static String	**valueOf**(char[] data) Returns the string representation of the char array argument.
static String	**valueOf**(char[] data, int offset, int count) Returns the string representation of a specific subarray of the char array argument.
static String	**valueOf**(double d) Returns the string representation of the double argument.
static String	**valueOf**(float f) Returns the string representation of the float argument.
static String	**valueOf**(int i) Returns the string representation of the int argument.
static String	**valueOf**(long l) Returns the string representation of the long argument.
static String	**valueOf**(Object obj) Returns the string representation of the Object argument.

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Vector API

```
java.lang.Object
  |
  +--java.util.AbstractCollection
        |
        +--java.util.AbstractList
              |
              +--java.util.Vector
```

All Implemented Interfaces:
Cloneable, Collection, List, Serializable
Direct Known Subclasses:
Stack

public class **Vector**
extends AbstractList
implements List, Cloneable, Serializable

Constructor Summary

`Vector``()` Constructs an empty vector so that its internal data array has size `10` and its standard capacity increment is zero.
`Vector``(Collection c)` Constructs a vector containing the elements of the specified collection, in the order they are returned by the collection's iterator.
`Vector``(int initialCapacity)` Constructs an empty vector with the specified initial capacity and with its capacity increment equal to zero.
`Vector``(int initialCapacity, int capacityIncrement)` Constructs an empty vector with the specified initial capacity and capacity increment.

Method Summary

`void`	**`add`**`(int index, Object element)` Inserts the specified element at the specified position in this Vector.
`boolean`	**`add`**`(Object o)` Appends the specified element to the end of this Vector.

boolean	**addAll**(Collection c) Appends all of the elements in the specified Collection to the end of this Vector, in the order that they are returned by the specified Collection's Iterator.
boolean	**addAll**(int index, Collection c) Inserts all of the elements in in the specified Collection into this Vector at the specified position.
void	**addElement**(Object obj) Adds the specified component to the end of this vector, increasing its size by one.
int	**capacity**() Returns the current capacity of this vector.
void	**clear**() Removes all of the elements from this Vector.
Object	**clone**() Returns a clone of this vector.
boolean	**contains**(Object elem) Tests if the specified object is a component in this vector.
boolean	**containsAll**(Collection c) Returns true if this Vector contains all of the elements in the specified Collection.
void	**copyInto**(Object[] anArray) Copies the components of this vector into the specified array.
Object	**elementAt**(int index) Returns the component at the specified index.
Enumeration	**elements**() Returns an enumeration of the components of this vector.
void	**ensureCapacity**(int minCapacity) Increases the capacity of this vector, if necessary, to ensure that it can hold at least the number of components specified by the minimum capacity argument.
boolean	**equals**(Object o) Compares the specified Object with this Vector for equality.
Object	**firstElement**() Returns the first component (the item at index 0) of this vector.
Object	**get**(int index) Returns the element at the specified position in this Vector.
int	**hashCode**() Returns the hash code value for this Vector.

int	**indexOf**(Object elem) Searches for the first occurence of the given argument, testing for equality using the equals method.
int	**indexOf**(Object elem, int index) Searches for the first occurence of the given argument, beginning the search at index, and testing for equality using the equals method.
void	**insertElementAt**(Object obj, int index) Inserts the specified object as a component in this vector at the specified index.
boolean	**isEmpty**() Tests if this vector has no components.
Object	**lastElement**() Returns the last component of the vector.
int	**lastIndexOf**(Object elem) Returns the index of the last occurrence of the specified object in this vector.
int	**lastIndexOf**(Object elem, int index) Searches backwards for the specified object, starting from the specified index, and returns an index to it.
Object	**remove**(int index) Removes the element at the specified position in this Vector.
boolean	**remove**(Object o) Removes the first occurrence of the specified element in this Vector If the Vector does not contain the element, it is unchanged.
boolean	**removeAll**(Collection c) Removes from this Vector all of its elements that are contained in the specified Collection.
void	**removeAllElements**() Removes all components from this vector and sets its size to zero.
boolean	**removeElement**(Object obj) Removes the first (lowest-indexed) occurrence of the argument from this vector.
void	**removeElementAt**(int index) Deletes the component at the specified index.
protected void	**removeRange**(int fromIndex, int toIndex) Removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive.
boolean	**retainAll**(Collection c) Retains only the elements in this Vector that are contained in the specified Collection.

Object	**set**(int index, Object element) Replaces the element at the specified position in this Vector with the specified element.
void	**setElementAt**(Object obj, int index) Sets the component at the specified index of this vector to be the specified object.
void	**setSize**(int newSize) Sets the size of this vector.
int	**size**() Returns the number of components in this vector.
List	**subList**(int fromIndex, int toIndex) Returns a view of the portion of this List between fromIndex, inclusive, and toIndex, exclusive.
Object[]	**toArray**() Returns an array containing all of the elements in this Vector in the correct order.
Object[]	**toArray**(Object[] a) Returns an array containing all of the elements in this Vector in the correct order.
String	**toString**() Returns a string representation of this Vector, containing the String representation of each element.
void	**trimToSize**() Trims the capacity of this vector to be the vector's current size.

Methods inherited from class java.util.AbstractList
iterator, listIterator, listIterator

Methods inherited from class java.lang.Object
finalize, getClass, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.util.List
iterator, listIterator, listIterator

Table of ASCII Character Values

0	nul	1	soh	2	stx	3	etx	4	eot	5	enq	6	ack	7	bel
8	bs	9	ht	10	nl	11	vt	12	np	13	cr	14	so	15	si
16	dle	17	dc1	18	dc2	19	dc3	20	dc4	21	nak	22	syn	23	etb
24	can	25	em	26	sub	27	esc	28	fs	29	gs	30	rs	31	us
32	sp	33	!	34	"	35	#	36	$	37	%	38	&	39	’
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	‘	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	del