String
- value: char[] - hash: int - serialVersionUID = -6849794470754667710 : long - serialPersistentFields : ObjectStreamField[]
+ String() + String (original : String) + String (value: char[]) + String (value: char[] , offset : int, count : int) + String(codepoints: int[], offset : int, count : int) + String(ascii :byte, hibyte : int, , offset : int, count : int) + String (ascii : int[], hibyte : int) + checkBounds(bytes: byte[], offset :int, length : int) + String(bytes : byte[], offset : int ,length : int, charsetName : String) + String(bytes: byte[], offset :int, length :int, charset : Charset) + String (bytes : byte[], charsetName : String) + String (bytes : : byte[], charset : Charset) + String(bytes: byte[]) + String(buffer : StringBuffer) + String (builder: StringBuilder) + String(value: char[],share : Boolean) + Length(): int + isEmpty() : boolean + charAt(index : int): char + codepointAt(index : int) : int + codepointBefore(index : int) : int + codepointCount(beginIndex : int, endIndex : int) : int + offsetByCodePoints(index : int, codepointsOffset : int) : int + getChars(dst :char[],dstBegin : int): void + getChars(srcBegin : int, srcEnd int, dst :char[], dstBegin: int) : void + getBytes(charsetName: String): byte[] + getBytes(charset: Charset) + getBytes() + equals(anObject: Object) : boolean + contentEquals(sb: StringBuffer): boolean + contentEquals(cs: CharSequence): boolean + equalsIgnoreCase(anotherString: String): boolean + compareTo(anotherString: String) + compareToIgnoreCase(str: String) + regionMatches(toffset: int, other: String, offset: int, len: int): boolean + startsWith(prefix: String, toffset: int) : boolean + endsWith(suffix: String) : boolean + hashCode():int

+ indexOf(str: String) :int
+ lastIndexOf(ch :int, fromIndex: int) :int
+ lastIndexOf(str: String, fromIndex: int) :int
- nonSyncContentEquals(sb: AbstractStringBuilder):int
- indexOfSupplementary(ch :int, fromIndex: int):int
+ static copyValueOf(data:char[]):String
+ static valueOf(d: double) :String
+ replaceFirst(regex: String, replacement: String) :String
+ replaceAll(regex: String, replacement: String) :String
split(regex: String) :String[]
+ contains(s: CharSequence) : boolean
+ toCharArray():char[]
+ toLowerCase():String
+ toUpperCase():String
+ toString():String
+ trim():String

System
+ final static in = null : InputStream + final static out = null :PrintStream + final static err = null : PrintStream - static volatile security = null : SecurityManager - static props : Properties - static lineSeparator : String
+ static load(filename : String):void + static loadLibrary(libname:String):void - static initializeSystemClass():void - static setJavaLangAccess():void + static getProperties():Properties + static lineSeparator():String + static getProperty(key : String):String + static clearProperty(key:String):String - static checkKey(key:String):void + static exit(status: int) + static gc():void + static runFinalization() :void + static load(String filename):void + static loadLibrary(String libname): void - static initializeSystemClass() : void - static setJavaLangAccess() : void + static setProperty(String key, String value): String

PrintStream
- final autoFlush: boolean ; - trouble = false :boolean ; - formatter :Formatter ; - textOut: BufferedWriter ; - charOut: OutputStreamWriter ;
- static toCharset(csn: String) : Charset - PrintStream(autoFlush: boolean , out:OutputStream): void - PrintStream(autoFlush:boolean , out: OutputStream , charset:Charset): void - PrintStream(autoFlush:boolean, charset: Charset , out:OutputStream): void + PrintStream(out:OutputStream): void + PrintStream(out: OutputStream, autoFlush: boolean): void + PrintStream(out: OutputStream, autoFlush: boolean , encoding: String): void + PrintStream(fileName: String): void + PrintStream(fileName:String , csn: String): void + PrintStream(file:File): void - void ensureOpen(): void + flush() : void + close(): void - write(buf: char[]):void - write(s:String): void + write(buf:byte[], off: int , len: int): void - newLine(): void + print(obj:Object): void + println(x:char[]): void + println(x:String): void # setError():void # clearError(): void + println(x:Object): void + PrintStream printf(format: String , args: Object ...): void + PrintStream printf(l :Locale, format: String , args: Object ...): void + PrintStream format(format:String , args:Object ...): void + PrintStream format(l :Locale, format: String , args: Object ...) : void + PrintStream append(csq:CharSequence): void + PrintStream append(csq: CharSequence, start: int , end:int): void + PrintStream append(c:char) : void