

# Pila.h versione 6

```
struct Pila {  
    private:  
        int size;  
        int defaultGrowthSize;  
        int marker;  
        int * contenuto;  
        void cresci(int increment);  
    public:  
        Pila(int initialSize) ;  
        Pila();  
        ~Pila() ;  
        void copy(Pila * to) ;  
        void inserisci(int k) ;  
        int estrai() ;  
        void stampaStato() ;  
};
```

```
class Pila {  
    private:  
        int size;  
        int defaultGrowthSize;  
        int marker;  
        int * contenuto;  
        void cresci(int increment);  
    public:  
        Pila(int initialSize) ;  
        Pila();  
        ~Pila() ;  
        void copy(Pila * to) ;  
        void inserisci(int k) ;  
        int estrai() ;  
        void stampaStato() ;  
};
```

# La Pila in Java - 1

oopC+

```
package strutture;
public class Pila {
    int size;
    int defaultGrowthSize;
    int marker;
    int contenuto[];

    Pila(int initialSize) {
        size=initialSize;
        defaultGrowthSize=initialSize;
        marker=0;
        contenuto=new int[initialSize];
    }
}
```

in C++

```
class Pila {
    private:
        int size;
        int defaultGrowthSize;
        int marker;
        int * contenuto;
```

in C++

```
Pila::Pila(int initialSize) {
    size=initialSize;
    defaultGrowthSize=initialSize;
    marker=0;
    contenuto=new int[initialSize];
}
```

# La Pila in Java - 2

```
private void cresci() {  
    size+=defaultGrowthSize;  
    int temp[] = new int[size];  
    for (int k=0; k<marker; k++)  
        temp[k]=contenuto[k];  
    this.contenuto=temp;  
}
```

in C++

```
void cresci();  
{  
    this->size+=defaultGrowthSize;  
    int * temp=new int[this->size];  
    //int temp[] = new int[size];  
    for (int k=0; k<s->marker; k++) {  
        temp[k]=this->contenuto[k];  
    }  
    delete [] (this->contenuto);  
    this->contenuto=temp;  
}
```

# La Pila in Java - 3

```
void inserisci(int k) {  
    if (marker==size)  
        cresci(defaultGrowthSize);  
    contenuto[marker]=k;  
    marker++;  
}
```

in C++

```
void inserisci(int k) {  
    if (marker==size)  
        cresci(defaultGrowthSize);  
    contenuto[marker]=k;  
    marker++;  
}
```

# La Pila in Java - 4

oopC+

```
int estrai() {  
    assert(marker>0) : "Estrazione da un pila vuota!";  
    return contenuto[--marker];  
}
```

java -ea Pila

```
int estrai() {           in C++  
    assert(marker>0);  
    return contenuto[--(marker)];  
}
```

```
java.lang.AssertionError: Estrazione da un pila vuota!  
    at pila.Pila.estrai(Pila.java:22)  
    at pila.Pila.main(Pila.java:39)
```

```
int estrai() {  
    if (marker==0) {  
        System.out.println(  
            "Non posso estrarre da una pila vuota");  
        System.exit(1);  
    }  
    return contenuto[--marker];  
}
```

# La Pila in Java - 5

```
public static void main(String args[]) {  
    Pila s=new Pila(5);  
    for (int k=0;k<10;k++)  
        s.inserisci(k);  
    for (int k=0;k<12;k++)  
        System.out.println(s.estrai());  
}  
}
```

```
int main() {  
    Pila * s=new Pila(5);  
    for (int k=0; k<10;k++)  
        s->inserisci(k);  
    for (int k=0; k<12;k++)  
        cout<<s->estrai()<<endl;  
}
```

# Using System.arraycopy()

```
System.arraycopy(  
    Object src, int src_position,  
    Object dst, int dst_position, int length  
) ;
```

Copies the specified source array, beginning at the specified position, to the specified position of the destination array.

# La Pila in Java – 2-alt

```
private void cresci() {  
    size+=defaultGrowthSize;  
    int temp[ ]=new int[size];  
    System.arraycopy(contenuto, 0, temp, 0, marker-1);  
    contenuto=temp;  
}
```

## Tipi di dato derivati (reference data)

■ Java, come tutti i linguaggi OO, permette di definire **NUOVI TIPI DI DATO** (classi).

■ Alcuni tipi di dato (classi) sono predefinite:  
■ ad esempio le stringhe. (**String**)

tipo

identificatore

Operatore  
di creazione

costruttore

■ Point punto = new Point(10,10);

### ■ No Structures or Unions

◆ Java does not support C struct or union types. Note, however, that a class is essentially the same thing as a struct, but with more features. And you can simulate the important features of a union by subclassing.

“Java non ha i puntatori”

Ma è vero?

```
Point punto = new Point(10,10);
```

l'identificatore di un oggetto (“punto”) sembra proprio un puntatore!

Quel che Java non ha è  
l'aritmetica dei puntatori

## Confronto dell'operatore new

in C++: **Point \* punto = new Point(10,10);**

in Java: **Point punto = new Point(10,10);**

punto.x di Java      equivale a      punto->x del C++

In Java gli oggetti sono accessibili  
SOLO per referenza

# memory management

La gestione (dinamica) della memoria e' automatica, tramite la creazione (operatore new ) e la distruzione (garbage collection) di oggetti.

GC interviene quando serve memoria.

GC elimina gli oggetti per i quali non vi sono piu' riferimenti attivi.

GC puo' essere attivato su richiesta esplicita: `System.gc()`

# memory management - costruttori

Operazioni da eseguirsi alla nascita di un oggetto vanno definite nel metodo “costruttore”.

Ogni classe deve avere **uno (o più)** costruttori.

I costruttori possono differire per numero e tipo di parametri.

Es.:

```
Pila() {  
    size=100; ...  
}
```

```
Pila(int size) {  
    this.size=size  
}
```

# memory management - distruttori

Operazioni da associarsi con l' eliminazione di un oggetto possono essere definite nel metodo “distruttore” `finalize()` (opzionale)

NOTA: il metodo `finalize` POTREBBE NON ESSERE CHIAMATO DAL SISTEMA (es. se il programma finisce prima...)

Per essere certi che vengano chiamati i metodi `finalize`, occorre chiamare la

`System.runFinalization()` subito DOPO la `System.gc()`

# System agisce come libreria

```
System.out.println(...);  
System.gc();  
System.runFinalization();  
System.exit(int status);  
System.arraycopy(Object src, int srcPos, Object dest, int destPos, int  
length);  
long System.currentTimeMillis();
```

# Class String

java.lang

## Class String

[java.lang.Object](#)

|

+--[java.lang.String](#)

### All Implemented Interfaces:

[CharSequence](#), [Comparable](#), [Serializable](#)

---

```
public final class String
extends Object
implements Serializable, Comparable, CharSequence
```

The `String` class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because `String` objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

# Class String

## Constructor Summary

### `String()`

Initializes a newly created `String` object so that it represents an empty character sequence.

### `String(byte[] bytes)`

Constructs a new `String` by decoding the specified array of bytes using the platform's default charset.

### `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 charset name or that use the platform's default charset.

### `String(byte[] bytes, int offset, int length)`

Constructs a new `String` by decoding the specified subarray of bytes using the platform's default charset.

### `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 charset name or that use the platform's default charset.

### `String(byte[] bytes, int offset, int length, String charsetName)`

Constructs a new `String` by decoding the specified subarray of bytes using the specified charset.

### `String(byte[] bytes, String charsetName)`

Constructs a new `String` by decoding the specified array of bytes using the specified charset.

### `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.

## Class String

## Method Summary

char	<a href="#">charAt(int index)</a>	Returns the character at the specified index.
int	<a href="#">compareTo(Object o)</a>	Compares this String to another Object.
int	<a href="#">compareTo(String anotherString)</a>	Compares two strings lexicographically.
int	<a href="#">compareToIgnoreCase(String str)</a>	Compares two strings lexicographically, ignoring case differences.
<a href="#">String</a>	<a href="#">concat(String str)</a>	Concatenates the specified string to the end of this string.
boolean	<a href="#">contentEquals(StringBuffer sb)</a>	Returns true if and only if this String represents the same sequence of characters as the specified <a href="#">StringBuffer</a> .
static <a href="#">String</a>	<a href="#">copyValueOf(char[] data)</a>	Returns a String that represents the character sequence in the array specified.
static <a href="#">String</a>	<a href="#">copyValueOf(char[] data, int offset, int count)</a>	Returns a String that represents the character sequence in the array specified.
boolean	<a href="#">endsWith(String suffix)</a>	Tests if this string ends with the specified suffix.
boolean	<a href="#">equals(Object anObject)</a>	Compares this string to the specified object.
boolean	<a href="#">equalsIgnoreCase(String anotherString)</a>	Compares this String to another String, ignoring case considerations.
byte[]	<a href="#">getBytes()</a>	Encodes this String into a sequence of bytes using the platform's default charset, storing the result into a new byte array.
void	<a href="#">getBytes(int srcBegin, int srcEnd, byte[] dst, int dstBegin)</a>	<b>Deprecated.</b> This method does not properly convert characters into bytes. As of JDK 1.1, the preferred way to do this is via the <a href="#">getBytes()</a> method, which uses the platform's default charset.

# Class String

## Constructor Summary

### `String()`

Initializes a newly created `String` object so that it represents an empty character sequence.

### `String(byte[] bytes)`

Constructs a new `String` by decoding the specified array of bytes using the platform's default charset.

### `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 charset name or that use the platform's default charset.

### `String(byte[] bytes, int offset, int length)`

Constructs a new `String` by decoding the specified subarray of bytes using the platform's default charset.

### `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 charset name or that use the platform's default charset.

### `String(byte[] bytes, int offset, int length, String charsetName)`

Constructs a new `String` by decoding the specified subarray of bytes using the specified charset.

### `String(byte[] bytes, String charsetName)`

Constructs a new `String` by decoding the specified array of bytes using the specified charset.

### `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.

# Class String

## Method Detail

### length

```
public int length()
```

Returns the length of this string. The length is equal to the number of 16-bit Unicode characters in the string.

#### Specified by:

[length](#) in interface [CharSequence](#)

#### Returns:

the length of the sequence of characters represented by this object.

---

### charAt

```
public char charAt(int index)
```

Returns the character at the specified index. An index ranges from 0 to `length() - 1`. The first character of the sequence is at index 0, the next at index 1, and so on, as for array indexing.

#### Specified by:

[charAt](#) in interface [CharSequence](#)

#### Parameters:

index - the index of the character.

#### Returns:

the character at the specified index of this string. The first character is at index 0.

#### Throws:

[IndexOutOfBoundsException](#) - if the index argument is negative or not less than the length of this string.

# String

- ◆ Per trasformare il contenuto di una stringa in un intero:  
**int Integer.parseInt(String s)**
  
- ◆ Per trasformare il contenuto di una stringa in un float:  
**float Float.parseFloat(String s)**

Esercizio:  
Costruite una Coda analoga alla Pila