Lezione 7

Caratteri e Stringhe

Lezione 8 - Obiettivi

Al termine di questa lezione saremo in grado di

- Dichiarare e manipolare dati di tipo char.
- Scrivere programmi per la manipolazione di stringhe utilizzando oggetti String e StringBuffer.
- Distinguere tra le classi String e StringBuffer e utilizzare la classe più appropriata al contesto.
- Distinguere tra tipi primitivi e tipi "reference" e mostrare come l'allocazione di memoria differisce nei due casi.
- Distinguere tra test di uguaglianza e test di equivalenza per oggetti della classe String.
- Mostrare, utilizzando diagrammi sullo stato della memoria, come gli oggetti sono passati ai metodi e restituiti dai metodi.

Caratteri

- In Java i singoli caratteri sono rappresentati utilizzando il tipo di dato char. Le costanti di questo tipo sono simboli racchiusi tra apici singoli, per esempio, 'a', 'X', e '5'.
- Per rappresentare i caratteri nei calcolatori sono stati pensati diversi schemi di codifica.
- Una schema di codifica per i caratteri molto diffuso è il codice ASCII (American Standard Code for Information Interchange), basato sull'alfabeto inglese.
- Per consentire la rappresentazione completa di alfabeti più ricchi di quello inglese, il consorzio Unicode ha promulgato l'Unicode Worldwide Character Standard, noto come Unicode.

Tabella ASCII (standard)

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
10	lf	vt	ff	cr	so	si	dle	dc1	dc2	dc3
20	cd4	nak	syn	etb	can	em	sub	esc	fs	gs
30	rs	us	sp	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	}	|	~	del		

Il carattere 'O' ha codice 79 (riga 70 + colonna 9 = 79).

Gestione di Caratteri

```
char ch1, ch2 = 'X';
```

Dichiarazione ed inizializzazione

```
messageBox.show("Il codice ASCII di X è " +
                (int) 'X' );

message.show("Il codice ASCII 88 corrisponde a" +
                (char)88 );
```

Conversione di tipo tra int e char.

```
'A' < 'c'
```

Questo confronto ritorna vero perchè il codice ASCII di 'A' è 65 mentre il codice di 'c' è 99.

 Introduction to Object-Oriented Programming with Java--Wu Chapter 8 - 5

Stringhe

- Una stringa è una sequenza di caratteri alfanumerici trattata come un singolo valore.
- La classe predefinita String è utilizzata per rappresentare le stringhe in Java.
- Gli oggetti String possono essere usati, ad esempio, per visuallizzare testi con messageBox:

```
messageBox.show( "Ciao, come stai?" );
```

 Introduction to Object-Oriented Programming with Java--Wu Chapter 8 - 6

Le Stringhe sono Oggetti in Java

- String è una classe nel package java.lang.
- Dato che String è una classe dobbiamo creare istanze di String per contenere e manipolare stringhe in Java. Come tutti gli oggetti, sono necessarie una dichiarazione e una creazione per usare un oggetto String:

```
String name1;
name1 = new String( "Latte" );
```

Normalmente, usando un'abbreviazione, trattiamo gli oggetti String come se fossero tipi di dato primitivi:

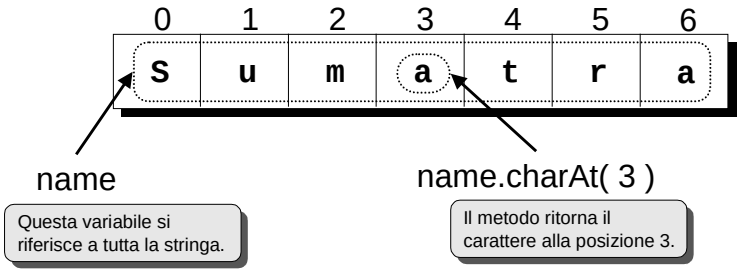
```
String name1;
name1 = "Latte";
```

Questi comandi sono equivalenti.

Accesso ai Caratteri

- I singoli caratteri in un oggetto String sono accessibili con il metodo charAt.

```
String name = "Sumatra";
```



Questa variabile si riferisce a tutta la stringa.

Il metodo ritorna il carattere alla posizione 3.

Determinare la Dimensione

- Il numero di caratteri in una stringa si ottiene con il metodo length.

```
String name = “Sumatra”,
       str1 = “one”,
       str2 = “”,
       str3;
```

```
name.length( );   ⟶ 7
str1.length( );   ⟶ 3
str2.length( );   ⟶ 0
str3.length( );   ⟶ Errore!
```

str3 non corrisponde ad alcun oggetto.

Esempio: Conteggio di Vocali

```
char     letter;
String   name               = inputBox.getString(“Nome?");
int      numberOfCharacters = name.length();
int      vowelCount         = 0;

for (int i = 0; i < numberOfCharacters; i++) {
    letter = name.charAt(i);

    if (    letter == 'a' || letter == 'A' ||
            letter == 'e' || letter == 'E' ||
            letter == 'i' || letter == 'I' ||
            letter == 'o' || letter == 'O' ||
            letter == 'u' || letter == 'U'
                                              ) {
        vowelCount++;
    }
}
messageBox.show(name + ", il tuo nome ha " + vowelCount + " vocali");
```

Queste istruzioni contano il numero di vocali nella stringa in ingresso (name).

Esempio: Conteggio di Parole

```
String sentence           = inputBox.getString("Dammi una frase:");
int    numberOfCharacters  = sentence.length();
int    index              = 0;
int    wordCount          = 0;

while (index < numberOfCharacters ) {

    //ignora gli spazi bianchi
    while (sentence.charAt(index) == ' ') {
         index++;
    }

    //trova la fine della parola
    while (sentence.charAt(index) != ' ') {
         index++;
    }

    wordCount++;  //un'altra parola trovata, incremento del contatore
}
```

Problema:
Il ciclo più interno potrebbe assegnare ad index un valore uguale a numberOfCharacters!

Esempio: Conteggio di Parole - 2

```
String sentence           = inputBox.getString("Dammi una frase:");
int    numberOfCharacters  = sentence.length();
int    index              = 0;
int    wordCount          = 0;

while (index < numberOfCharacters ) {

    //ignora gli spazi bianchi
    while (index < numberOfCharacters && sentence.charAt(index) == ' ') {
         index++;
    }

    //trova la fine della parola
    while (index < numberOfCharacters && sentence.charAt(index) != ' ') {
         index++;
    }

    wordCount++;  //un'altra parola, incrementa il contatore
}
```

Problema:
wordCount è maggiore di uno rispetto al conteggio corretto se la frase finisce con uno o più spazi.

Esempio: Conteggio di 'Java'

```
int         javaCount      = 0;
boolean     repeat         = true;
String      word;

while ( repeat ) {

    word = inputBox.getString("Prossima parola:");

    if ( word.equals("STOP") )     {
         repeat = false;
    }
    else if ( word.equalsIgnoreCase("Java") ) {
         javaCount++;
    }

}
```

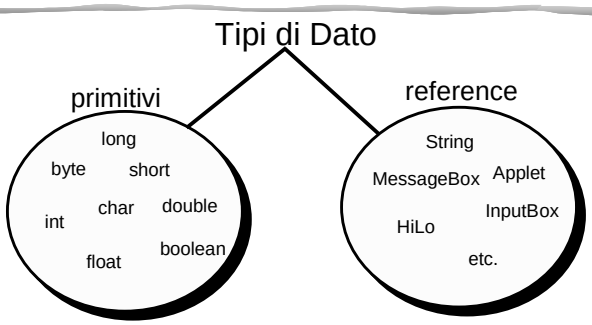
Continua a leggere le parole e conta quante volte la parola Java appare nel testo in ingresso.

Notare il confronto tra stringhe che non utilizza l'operatore ===.

Altri Metodi della Classe String

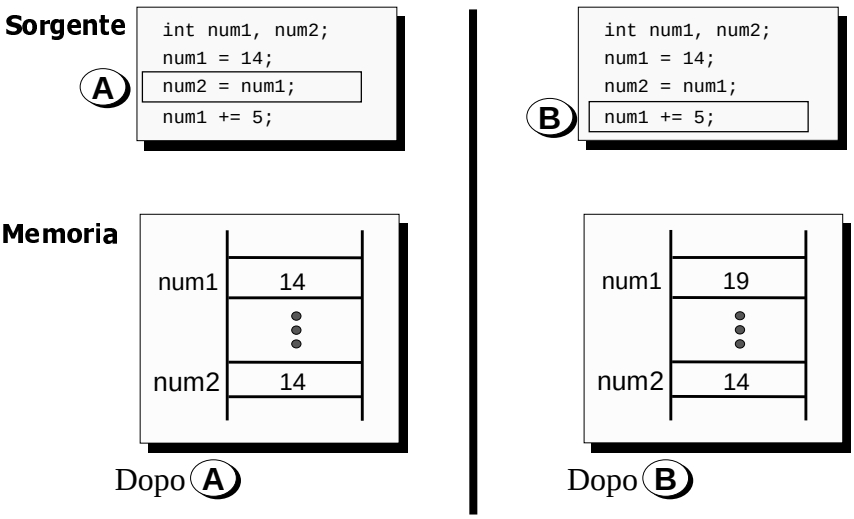
Metodi	Significato
compareTo	Comparazione tra stringhe. `str1.compareTo( str2 )`
substring	Estrae una sottostringa da una stringa. `str1.substring( 1, 4 )`
trim	Rimuove gli spazi all'inizio e alla fine. `str1.trim( )`
valueOf	Converte un valore di tipo int, float, ecc. in una stringa. `String.valueOf( 123.4565 )`
startsWith	Ritorna true se una stringa comncia con un dato prefisso. `str1.startsWith( str2 )`
endsWith	Ritorna true se una stringa termina con un dato suffisso. `str1.endsWith( str2 )`

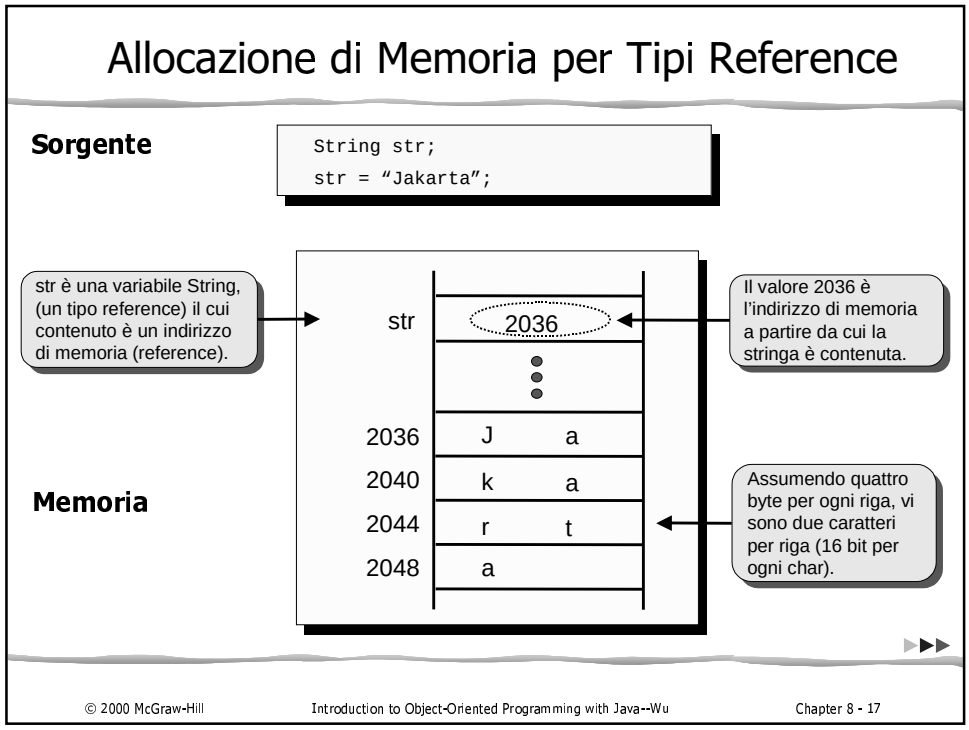
Tipi Primitivi e Tipi "Reference"

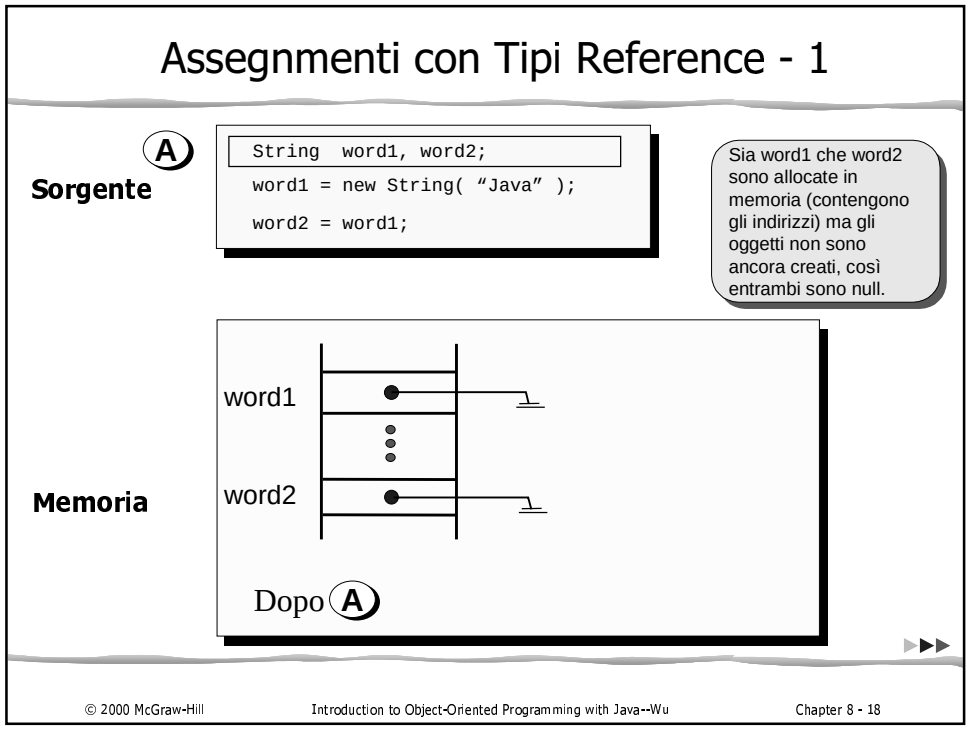


- Vi sono due tipi di dato: primitivi e reference.
- I tipi di dato non numerici char e boolean e tutti i tipi di dato numerici sono primitivi.
- Tutti gli oggetti sono di tipo reference.

Assegnamento con Tipi Primitivi




Allocazione di Memoria per Tipi Reference
Sorgente
String str;
str = “Jakarta”;
str è una variabile String, (un tipo reference) il cui contenuto è un indirizzo di memoria (reference).
str
2036
Il valore 2036 è l’indirizzo di memoria a partire da cui la stringa è contenuta.
Memoria
2036 J a
2040 k a
2044 r t
2048 a
Assumendo quattro byte per ogni riga, vi sono due caratteri per riga (16 bit per ogni char).
© 2000 McGraw-Hill
Introduction to Object-Oriented Programming with Java--Wu
Chapter 8 - 17


Assegnmenti con Tipi Reference - 1
A
Sorgente
String word1, word2;
word1 = new String(“Java”);
word2 = word1;
Sia word1 che word2 sono allocate in memoria (contengono gli indirizzi) ma gli oggetti non sono ancora creati, così entrambi sono null.
word1
word2
Memoria
Dopo A
© 2000 McGraw-Hill
Introduction to Object-Oriented Programming with Java--Wu
Chapter 8 - 18

Assegnamenti con Tipi References - 2

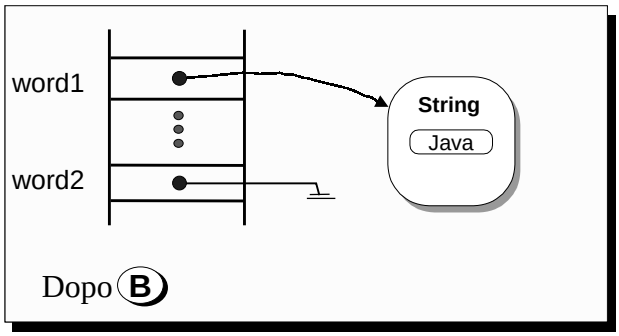
Sorgente

B

```
String  word1, word2;
word1 = new String( "Java" );
word2 = word1;
```

Un oggetto String viene creato e assegnato a word1, quindi word1 contiene l'indirizzo di questo oggetto.

Memoria



Assegnamenti con Tipi Reference - 3

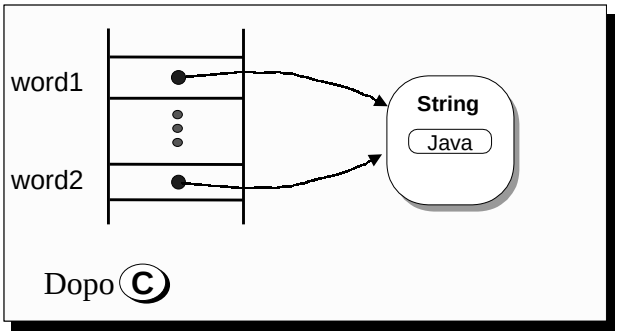
Sorgente

C

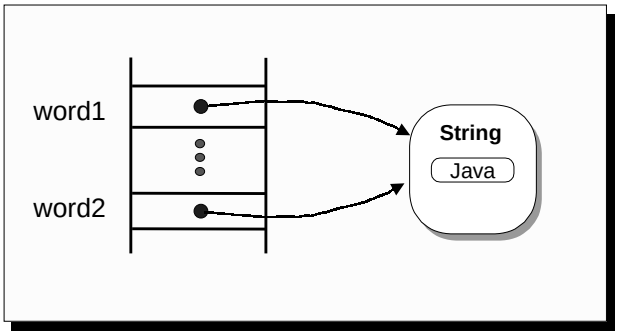
```
String  word1, word2;
word1 = new String( "Java" );
word2 = word1;
```

Il contenuto di word1, che è un indirizzo, è assegnato a word2, e quindi word2 si riferisce allo stesso oggetto di word1.

Memoria



Uguaglianza (==) vs. equals – Caso 1



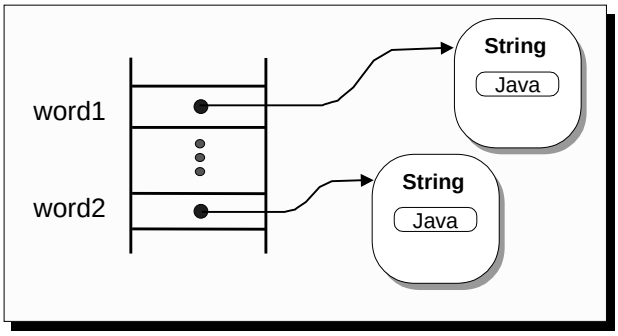
`word1 == word2` → true

`word1.equals( word2 )` → true

word1 e word2 puntano allo stesso oggetto.

 Introduction to Object-Oriented Programming with Java--Wu Chapter 8 - 21

Uguaglianza (==) vs. equals - Case 2



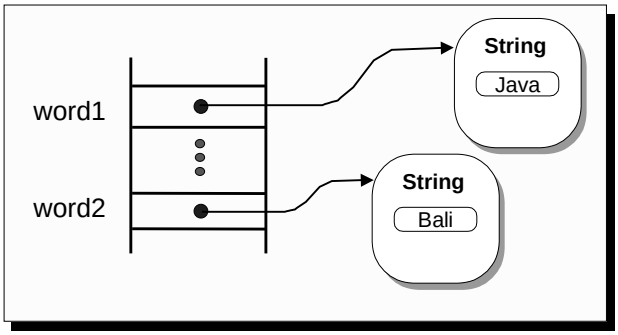
`word1 == word2` → false

`word1.equals( word2 )` → true

word1 e word2 puntano ad oggetti diversi con uguale contenuto.

 Introduction to Object-Oriented Programming with Java--Wu Chapter 8 - 22

Uguaglianza (==) vs. equals - Case 3



`word1 == word2` → false

`word1.equals( word2 )` → false

word1 e word2 puntano ad oggetti diversi con diversi contenuti.

StringBuffer

- Un oggetto String è immutabile.
- Si possono leggere i singoli caratteri, ma non è possibile modificare il contenuto di un oggetto String.
  - Nota: I metodi della classe String, come toUpperCase and substring, non modificano la stringa iniziale, bensì ne ritornano una nuova.
- Java adotta questa restrizione per poter allocare efficientemente gli oggetti String in memoria.
- Vi sono casi in cui manipolare direttamente i contenuti di una stringa è più conveniente.
  - Manipolazione in questo caso significa compiere operazioni come rimpiazzare un carattere, aggiungere una stringa ad un altra, cancellare una porzione di una stringa, ...

Esempio di StringBuffer - 1

- Rimpiazzare tutte le vocali in una frase con'X'.

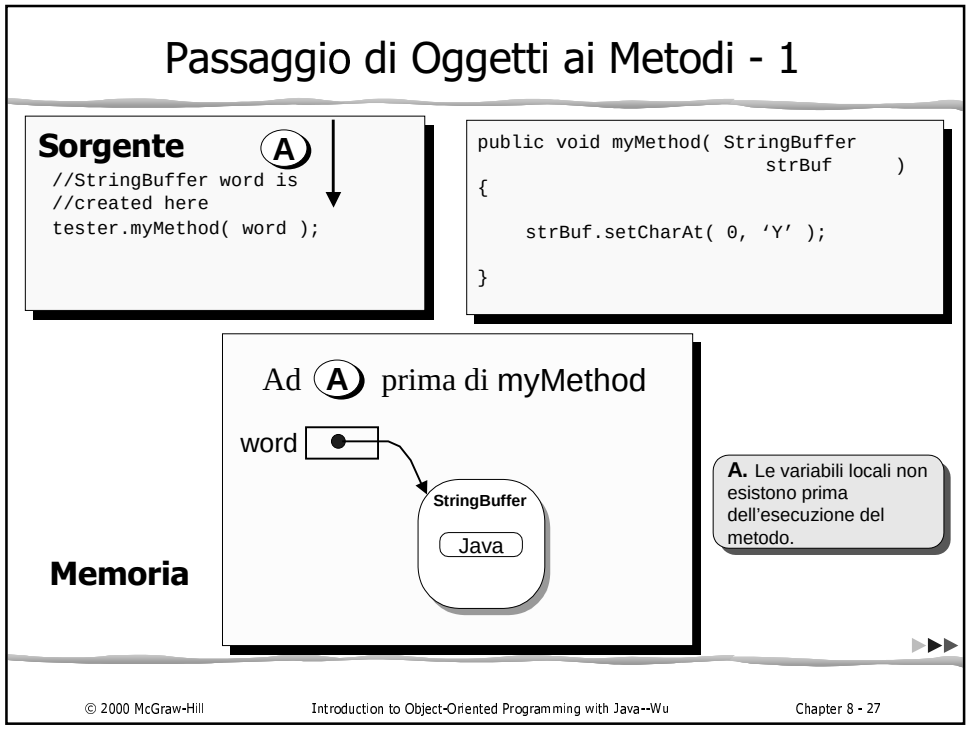
```
char         letter;
String       inSentence         = inputBox.getString("Frase:");
StringBuffer tempStringBuffer   = new StringBuffer(inSentence);
int          numberOfCharacters = tempStringBuffer.length();

for (int index = 0; index < numberOfCharacters; index++) {

    letter = tempStringBuffer.charAt(index);

    if ( letter == 'a' || letter == 'A' || letter == 'e' || letter == 'E' ||
         letter == 'i' || letter == 'I' || letter == 'o' || letter == 'O' ||
         letter == 'u' || letter == 'U'  ) {
         tempStringBuffer.setCharAt(index,'X');
    }
}
messageBox.show( tempStringBuffer );
```

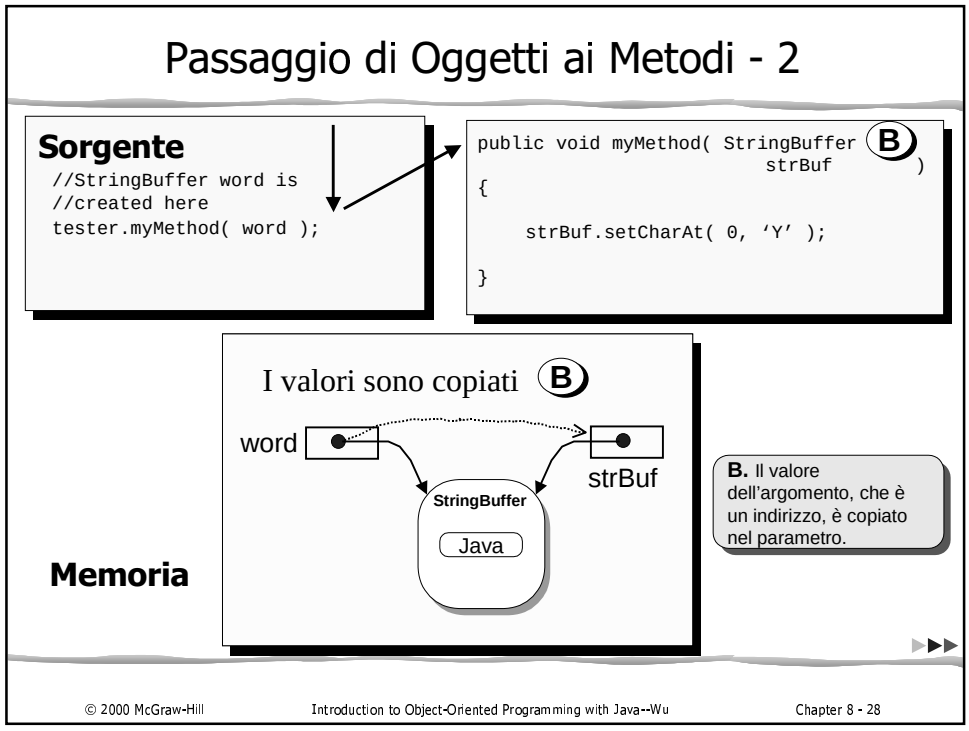

Esempio di StringBuffer - 2

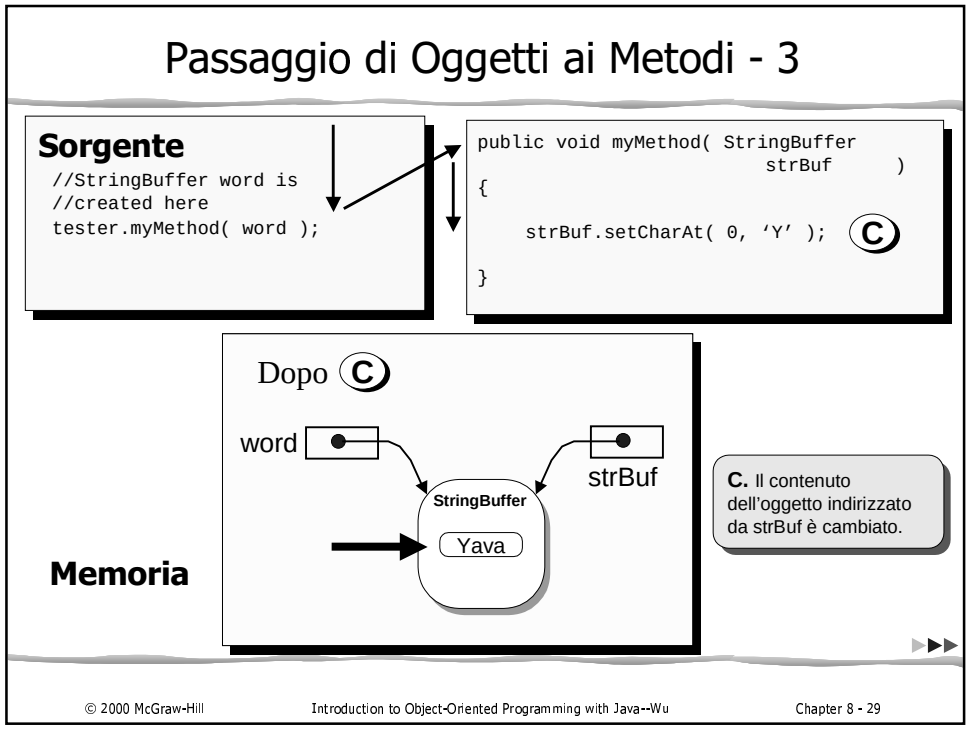
- Crea frasi con parole che hanno un numero pari di lettere. Si ferma quando la parola in ingresso è STOP.

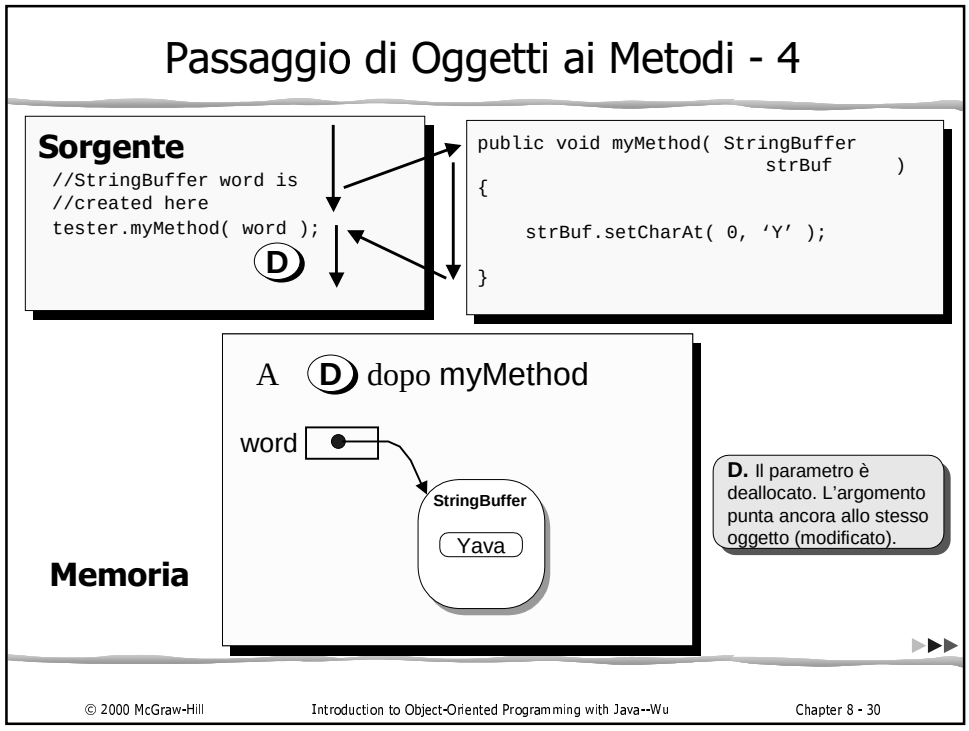
```
boolean          repeat = true;
String           word;

StringBuffer tempStringBuffer = new StringBuffer("");

while ( repeat ) {

    word = inputBox.getString("Prossima parola:");

    if ( word.equals("STOP") ) {
         repeat = false;
    }
    else if ( word.length() % 2 == 0 ) {

         tempStringBuffer.append(word + " ");
    }
}
```

Aggiunge la parola e uno spazio a tempStringBuffer.


Passaggio di Oggetti ai Metodi - 1
Sorgente
A
//StringBuffer word is
//created here
tester.myMethod(word);
public void myMethod(StringBuffer
strBuf)
{
strBuf.setCharAt(0, 'Y');
}
Ad A prima di myMethod
word
StringBuffer
Java
Memoria
A. Le variabili locali non esistono prima dell'esecuzione del metodo.
© 2000 McGraw-Hill
Introduction to Object-Oriented Programming with Java--Wu
Chapter 8 - 27


Passaggio di Oggetti ai Metodi - 2
Sorgente
//StringBuffer word is
//created here
tester.myMethod(word);
public void myMethod(StringBuffer B
strBuf)
{
strBuf.setCharAt(0, 'Y');
}
I valori sono copiati B
word
strBuf
StringBuffer
Java
Memoria
B. Il valore dell'argomento, che è un indirizzo, è copiato nel parametro.
© 2000 McGraw-Hill
Introduction to Object-Oriented Programming with Java--Wu
Chapter 8 - 28


Passaggio di Oggetti ai Metodi - 3
Sorgente
//StringBuffer word is
//created here
tester.myMethod(word);
public void myMethod(StringBuffer
strBuf)
{
strBuf.setCharAt(0, 'Y');
C
}
Dopo C
word
strBuf
StringBuffer
Yava
Memoria
C. Il contenuto dell'oggetto indirizzato da strBuf è cambiato.
© 2000 McGraw-Hill
Introduction to Object-Oriented Programming with Java--Wu
Chapter 8 - 29


Passaggio di Oggetti ai Metodi - 4
Sorgente
//StringBuffer word is
//created here
tester.myMethod(word);
D
public void myMethod(StringBuffer
strBuf)
{
strBuf.setCharAt(0, 'Y');
}
A D dopo myMethod
word
StringBuffer
Yava
Memoria
D. Il parametro è deallocato. L'argomento punta ancora allo stesso oggetto (modificato).
© 2000 McGraw-Hill
Introduction to Object-Oriented Programming with Java--Wu
Chapter 8 - 30

Ritornare un Oggetto da un Metodo

- "Passare un oggetto ad un metodo come argomento" significa passare l'indirizzo dell'oggetto al metodo.
- I lucidi precedenti illustrano gli effetti del passaggio di oggetti a un metodo.
- La stesse regole si applicano quando si "ritorna un oggetto da un metodo". Significa che l'indirizzo dell'oggetto viene ritornato dal metodo.

Esempio: Eggy-Peggy

- Problema

  Scrivere un'applicazione per il gioco di parole Eggy-Peggy. Il programma converte una stringa data dall'utente in una nuova stringa, piazzando la parola "egg" in fronte a tutte le vocali della stringa.

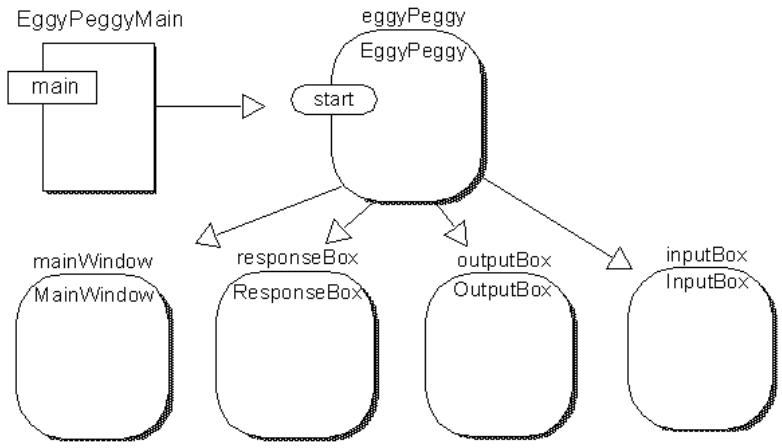
- Compiti principali

```
while ( l'utente vuole giocare ) {
    Task 1: richiedere una stringa all'utente;
    Task 2: generare una nuova stringa eggy-peggy;
    Task 3: visualizzare il risultato;
}
```


Eggy-Peggy – Progetto

Design Document: EggyPeggy Class	Purpose
`EggyPeggyMain`	The main class of the program.
`EggyPeggy`	The top-level control object that manages other objects in the program.
`InputBox`	An InputBox object is used to get a string from the player. This class is from javabook.
`OutputBox`	An OutputBox object is used for the input string and the eggy-peggy string.
`ResponseBox`	A ResponseBox object is used to get the player's instruction to play another game or not.
`String`	String objects are used for storing strings.
`StringBuffer`	A StringBuffer object is used during the conversion process.
`MainWindow`	The main window of the application.

Eggy-Peggy – Diagramma



Gioco Eggy-Peggy – Passi di Sviluppo

1. Iniziare con lo scheletro del programma. Definire le classi EggyPeggyMain e EggyPeggy.
2. Aggiungere le istruzioni di input e di output.
3. Aggiungere le istruzioni per il gioco Eggy-Peggy.
4. Finalizzare il programma.

The End

