Lezione 3

Trattamento dell'Informazione Numerica

Lezione 3 - Obiettivi

Al termine di questa lezione saremo in grado di:

- Attribuire correttamente i tipi (interi, reali, ...) ai dati numerici.
- Scrivere espressioni aritmetiche in Java.
- Valutare espressioni aritmetiche utilizzando le regole di precedenza.
- Descrivere come funziona l'allocazione di memoria per gli oggetti e per i dati numerici.
- Scrivere espressioni matematiche utilizzando i metodi della classe standard Math.
- Scrivere programmi interattivi utilizzando le classi InputBox e OutputBox del package javabook.
- Applicare tecniche di sviluppo incrementale nella scrittura di programmi.

Manipolazione di Espressioni Numeriche

- In Java, l'istruzione per addizionare due quantità x e y è

```
x + y
```

- Tuttavia, prima di effettuare l'addizione è necessario stabilire il *tipo* delle quantità in gioco. Se x e y sono numeri interi, è necessaria la seguente *dichiarazione*

```
int x, y;
```

oppure

```
int x;
int y;
```

Variabili

- Al momento della dichiarazione viene riservata una locazione di memoria per contenere i valori x e y.
- x e y sono *variabili*. Una variabile ha tre proprietà:
  - Una locazione di memoria che contiene il valore,
  - il tipo di dato contenuto nella locazione di memoria, e
  - il nome usato per riferirsi alla locazione di memoria.
- Esempi di dichiarazione di variabili:

```
int x;
int v, w, y;
```

Tipi per Dati Numerici

- In Java ci sono sei tipi per dati numerici: byte, short, int, long, float, e double.
- Esempi di dichiarazione di variabili numeriche:

```
int     i, j, k;
float   numberOne, numberTwo;
long    bigInteger;
double  bigNumber;
```

- Dichiarando una variabile è possibile inizializzarla. Per esempio si possono inizializzare le variabili intere count ed height, rispettivamente, a 10 e 34 come segue:

```
int count = 10, height = 34;
```


Precisione e Dimensioni dei Tipi di Dato

I sei tipi di dato differiscono nella precisione e nella dimensione dei valori che possono contenere.

Data Type	Content	Default Value†	Minimum Value	Maximum Value
byte	Integer	0	−128	127
short	Integer	0	−32768	32767
int	Integer	0	−2147483648	2147483647
long	Integer	0	−9223372036854775808	9223372036854775807
float	Real	0.0	−3.40282347E+38††	3.40282347E+38
double	Real	0.0	−1.7976931348623157OE+308	1.7976931348623157OE+308

†. No default value is assigned to a local variable. A local variable is explained on page 162.
††. The character E indicates a number is expressed in scientific notation. This notation is explained on page 99.

Istruzioni di Assegnamento

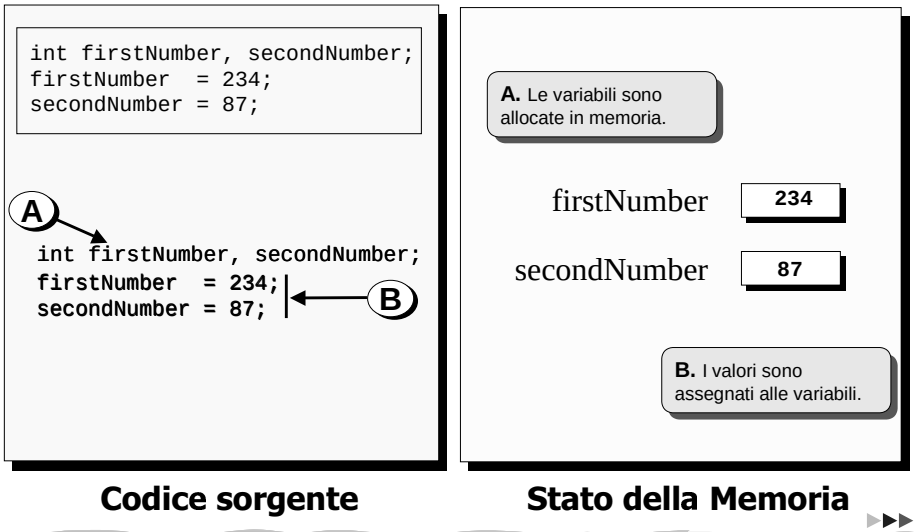
- Per assegnare un valore ad una variabile si utilizza un'*istruzione di assegnamento*.
- La sintassi è la seguente

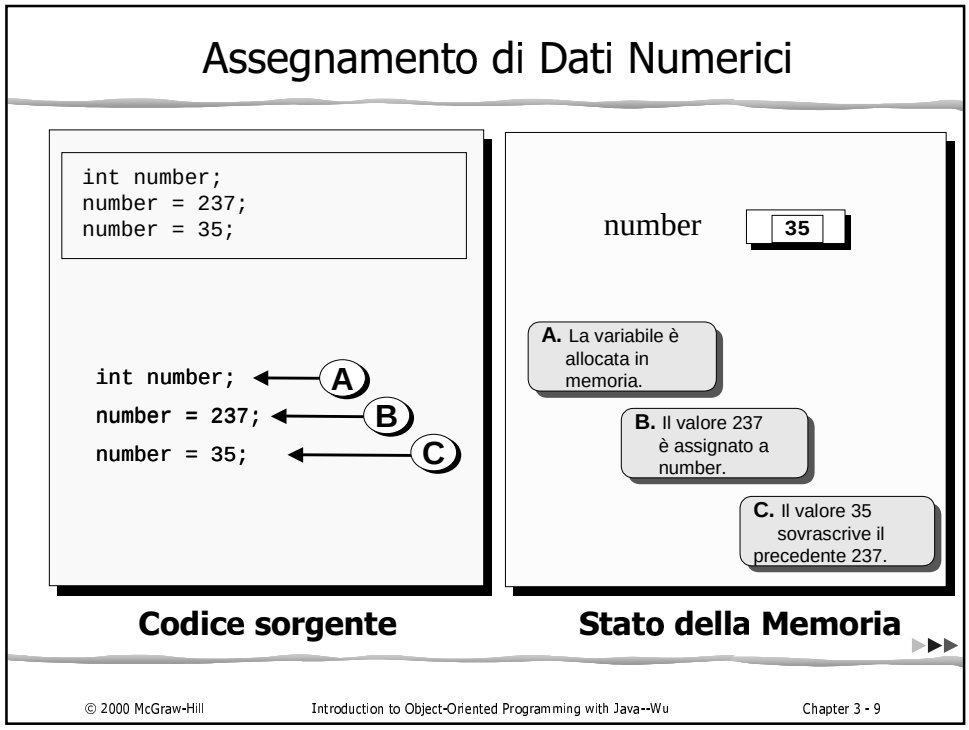
```
<variabile> = <espressione> ;
```

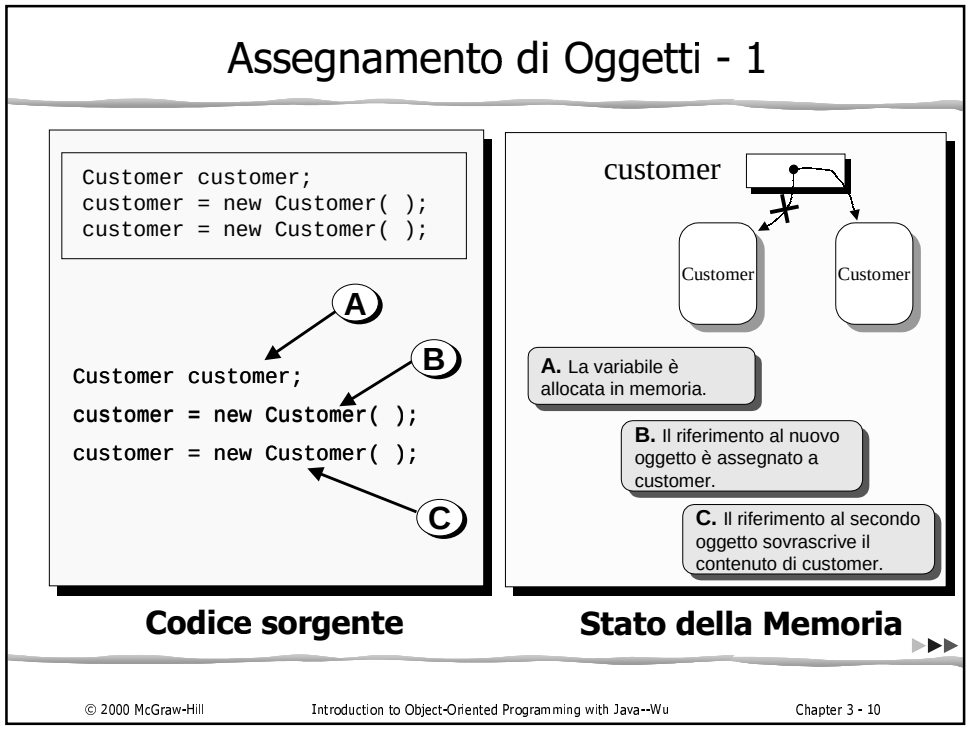
- Esempi:

```
sum = firstNumber + secondNumber;
avg = (one + two + three) / 3.0;
```

Dichiarazione ed Assegnamento




Assegnamento di Dati Numerici
int number;
number = 237;
number = 35;
int number;
number = 237;
number = 35;
A
B
C
number
35
A. La variabile è allocata in memoria.
B. Il valore 237 è assegnato a number.
C. Il valore 35 sovrascrive il precedente 237.
Codice sorgente
Stato della Memoria
© 2000 McGraw-Hill
Introduction to Object-Oriented Programming with Java--Wu
Chapter 3 - 9

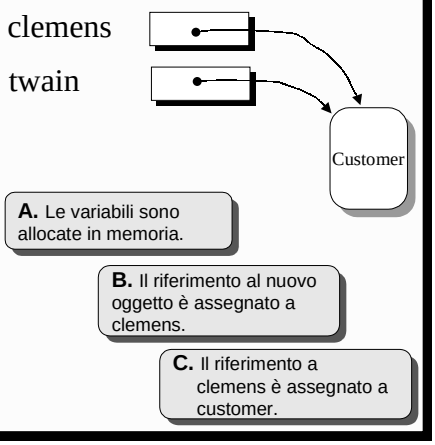

Assegnamento di Oggetti - 1
Customer customer;
customer = new Customer( );
customer = new Customer( );
Customer customer;
customer = new Customer( );
customer = new Customer( );
A
B
C
customer
Customer
Customer
A. La variabile è allocata in memoria.
B. Il riferimento al nuovo oggetto è assegnato a customer.
C. Il riferimento al secondo oggetto sovrascrive il contenuto di customer.
Codice sorgente
Stato della Memoria
© 2000 McGraw-Hill
Introduction to Object-Oriented Programming with Java--Wu
Chapter 3 - 10

Assegnamento di Oggetti - 2

```
Customer clemens, twain;
clemens = new Customer( );
twain   = clemens;
```

```
Customer clemens, twain;
clemens = new Customer( );
twain   = clemens;
```

A, B, C



Codice Sorgente **Stato della Memoria**

Operatori Aritmetici

- La tabella riassume gli operatori aritmetici a disposizione in Java.

Operation	Java Operator	Example	Value (x=10, y=7, z =2.5)
Addition	+	x + y	17
Subtraction	-	x - y	3
Multiplication	*	x * y	70
Division	/	x / y	1
		x / z	4.0
Modulo division (remainder)	%	x % y	3

In una divisione tra interi la parte frazionaria è troncata.

Espressioni Aritmetiche

- Come viene valutata l'espressione seguente?

```
x + 3 * y
```

  Risposta: x è addizionato a 3y.

- L'ordine di valutazione delle espressioni segue rigorosamente delle *regole di precedenza.*
- Un operatore con precedenza maggiore viene valutato prima di quelli con precedenza minore. Se due operatori hanno la stessa precedenza, la valutazione procede in genere da sinistra a destra.

Regole di Precedenza

Order	Group	Operator	Rule
High	subexpression	( )	Subexpressions are evaluated first. If parentheses are nested, the innermost subexpression is evaluated first. If two or more pairs of parentheses are on the same level, then they are evaluated from left to right.
	unary operator	-, +	Unary minuses and pluses are evaluated second.
	multiplicative operator	*, /, %	Multiplicative operators are evaluated third. If two or more multiplicative operators are in an expression, then they are evaluated from left to right.
Low	additive operator	+, -	Additive operators are evaluated last. If two or more additive operators are in an expression, then they are evaluated from left to right.

Conversione di Tipo

- Se **x** è un **float** e **y** è un **int**, qual'è il tipo della seguente espressione?

```
x * y
```

  Risposta: il tipo risultante è **float**.

- L'espressione sopra è una *espressione mista*.
- I tipi degli operandi nelle espressioni miste sono convertiti in modo da assicurare che il tipo dell'espressione sarà lo stesso dell'operando il cui tipo richiede la precisione (dimensione) maggiore.

Conversione Esplicita di Tipo

- Invece di affidarsi alle regole per la conversione implicita è possibile convertire esplicitamente un'espressione da un tipo ad un altro con la seguente istruzione:

```
( <tipo di dato> ) <espressione>
```

- Esempio

```
(float) x / 3
```

Converte **x** in un **float** e poi esegue la divisione.

```
(int) (x / y * 3.0)
```

Converte il risultato dell'espressione **x / y * 3.0** in un **int**.

Ancora sulla Conversione di Tipo

- Si consideri la seguente espressione:

```
double x = 3 + 5;
```

- Il risultato di 3 + 5 è di tipo **int**. Però, dato che la variabile **x** è un **double**, il valore 8 (tipo **int**) è convertito a 8.0 (tipo **double**) prima di venire assegnato ad **x**.
- Si noti che il viceversa non è consentito.

```
int x = 3.5;
```

Un valore non può essere assegnato ad una variabile che ha minore precisione.

Costanti

- Il valore di una variabile può cambiare nel tempo. Se un valore rimane invariato, si utilizzano le *costanti*.

```
final double PI               = 3.14159;
final int    MONTH_IN_YEAR    = 12;
final short  FARADAY_CONSTANT = 23060;
```

La parola chiave **final** è utilizzata per dichiarare costanti.

Queste sono costanti, dette anche *costanti nominative*.

Queste sono *costanti letterali*.

La Classe Math

- La classe Math del package java.lang include molte funzioni matematiche di uso comune, tra cui seno, coseno, tangente, radice quadrata, esponenziale, e altre.
- La formula matematica $\left|\sin \frac{\pi}{4} x\right|$

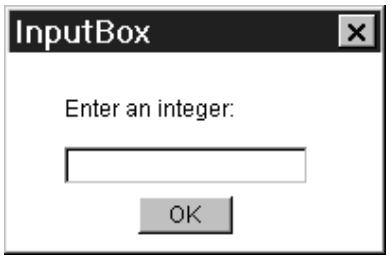
  è espressa in Java come

```
Math.abs( Math.sin( Math.PI / 4.0) * x )
```


InputBox

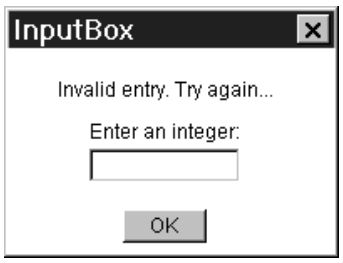
- La classe InputBox consente di chiedere all'utente di inserire dati di un certo tipo.

```
InputBox inputBox;
inputBox = new InputBox( mainWindow );
inputBox.getInteger( );
```



InputBox – Messaggi di Errore

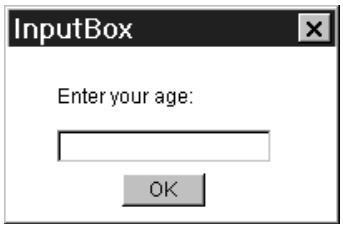
- Se viene inserito un valore di tipo inadeguato, si ottiene un messaggio di errore.



InputBox - Personalizzazione

- Il programmatore può decidere il messaggio.

```
inputBox.getInteger("Enter your age:");
```



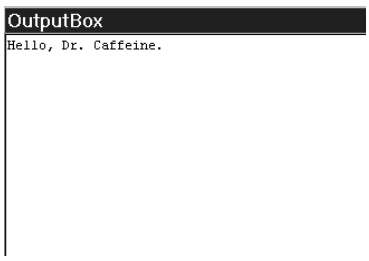
InputBox - Metodi

CLASS: InputBox		
Method	**Argument**	**Description**
`getDouble`	`<none>` or `text`	Allows the user to enter a double number. The InputBox dialog object will not close until the user enters a valid double number. If there is no argument, then the default prompt Enter a Double is displayed in the dialog. If a text value is passed as the argument, then it is used as a prompt in the dialog.
`getFloat`	`<none>` or `text`	Allows the user to enter a float number. The InputBox dialog object will not close until the user enters a valid float number. If there is no argument, then the default prompt Enter a Float is displayed in the dialog. If a text value is passed as the argument, then it is used as a prompt in the dialog.
`getInteger`	`<none>` or `text`	Allows the user to enter an integer, a number without a decimal point. The InputBox dialog object will not close until the user enters a valid integer. If there is no argument, then the default prompt Enter an Integer is displayed in the dialog. If a text value is passed as the argument, then it is used as a prompt in the dialog.

OutputBox

- La classe OutputBox consente di visualizzare dati.

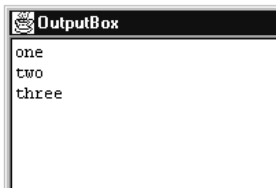
```
OutputBox outputBox;
outputBox = new OutputBox( mainWindow );
outputBox.print("Hello, Dr. Caffeine" );
```



OutputBox - printLine

- A differenza di MessageBox, OutputBox consente di visualizzare più linee di testo.

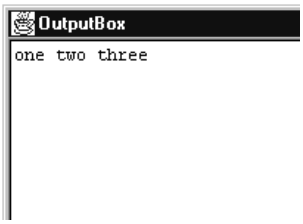
```
outputBox.printLine("one" );
outputBox.printLine("two" );
outputBox.printLine("three" );
```



OutputBox - print

- A differenza di printLine non inserisce un ritorno a capo tra un testo e l'altro.

```
outputBox.print("one " );
outputBox.print("two " );
outputBox.print("three" );
```



OutputBox - Metodi

CLASS: OutputBox		
Method	**Argument**	**Description**
`print`	number or text	Prints out the number or text passed as an argument in the dialog. Printing will continue from the end of currently displayed output.
`printLine`	number or text	Same as the print method, but the line is skipped after the output so the next output will continue from the next line.
`skipLine`	integer	Skips N lines where N is an integer passed as an argument.
`saveToFile`	filename	Saves the contents of an OutputBox to a file whose name is passed as an argument. If the designated file already exists, then the current contents of the file are erased and replaced by the contents of the OutputBox.
`appendToFile`	filename	Appends the contents of an OutputBox to a file whose name is passed as an argument. If the designated file does not exist, then this method works like the saveToFile method.

Esempio: Calcolo delle Rate di un Prestito

- Problema

  Scrivere un programma che calcoli i pagamenti mensili e il pagamento totale per un prestito, dato l'ammontare, l'interesse annuale e la durata del prestito.

- Compiti principali
  1. Richiedere tre valori in ingresso
  2. Calcolare la rata e la somma delle rate
  3. Visualizzare i risultati

- Formula per il calcolo della rata

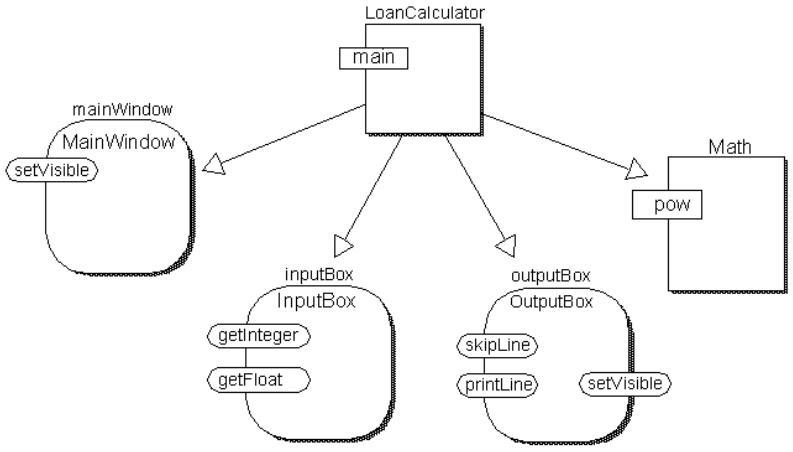
$$\text{Monthly Payment} = \frac{L \times R}{\left[1 - \left(\frac{1}{1+R}\right)^{N}\right]}$$

L – prestito
R – tasso mensile
N – numero di pagamenti

LoanCalculator – Progetto

Design Document: LoanCalculator Class	Purpose
`LoanCalculator`	The main class of the program.
`MainWindow`	The main frame window of the program. The title is set to Loan Calculator. This class is from javabook.
`InputBox`	An InputBox object is used to get three input values: loan amount, annual interest rate, and loan period. This class is from javabook.
`OutputBox`	An OutputBox object is used to display the input values and two computed results: monthly payment and total payment. This class is from javabook.
`Math`	The pow method is used to evaluate exponentiation in the formula for computing the monthly payment. This class is from java.lang. Note: You don't have to import java.lang. The classes in java.lang are available to a program without importing.

LoanCalculator – Diagramma



LoanCalculator – Passi dello Sviluppo

1. Iniziare con uno scheletro del programma.
2. Aggiungere il codice per inserire i dati.
3. Aggiungere il codice per visualizzare i risultati.
4. Aggiungere il codice per calcolare i pagamenti mensili e totali.