Tipi numerici

- `int`: interi, senza parti frazionarie

  `1, -4, 0`

- `double`: numeri floating-point (a precisione doppia)

  `0.5, -3.11111, 4.3E24, 1E-14`

- Un overflow numerico si ha se il risultato cade fuori dell'intervallo relativo al tipo numerico considerato

```
int n = 1000000;
System.out.println(n * n); // prints -727379968
```

- Java: 8 tipi primitivi, includendo 4 tipi intero e 2 tipi floating point

Tipi primitivi

Tipo	Descrizione	Dim.
int	Tipo intero, con intervallo -2,147,483,648 . . . 2,147,483,647	4 bytes
byte	Tipo che descrive un singolo byte, con intervallo -128 . . . 127	1 byte
short	Tipo intero breve, con intervallo -32768 . . . 32767	2 bytes
long	Tipo intero lungo, con intervallo -9,223,372,036,854,775,808 . . . -9,223,372,036,854,775,807	8 bytes
double	Il tipo floating-point a doppia precisione, con intervallo di circa $\pm 10^{308}$ e circa 15 cifre decimali significative	8 bytes
float	Tipo floating-point a singola precisione, con un intervallo di circa $\pm 10^{38}$ e circa 7 cifre decimali significative	4 bytes
char	Tipo carattere (codifica Unicode)	2 bytes
boolean	Tipo con due valori di verità: false e true	1 bit

Tipi numerici: tipi floating-point

- Errori di arrotondamento avvengono quando una conversione esatta tra due numeri non è possibile

```
double f = 4.35;
System.out.println(100 * f); // prints 434.99999999999994
```

- Java: è illegale assegnare un'espressione floating-point a una variabile intera

```
double balance = 13.75;
int dollars = balance; // Error
```

- Conversioni (cast): usate per convertire un valore in un tipo differente

```
int dollars = (int) balance; // OK
```

Le conversioni possono far perdere la parte decimale.

Continua

Tipi numerici: tipi floating point (continua)

`Math.round` converte un numero floating-point al più vicino intero

```
long rounded = Math.round(balance); // if balance is 13.75,
    then
                                    // rounded is set to 14
```

Syntassi 4.1 Conversione

```
(typeName) expression
```

Esempio:

```
(int) (balance * 100)
```

Scopo:

Convertire un’espressione ad un tipo differente.

Costanti: final

- Una variabile `final` è costante
- Una volta fissato il suo valore, questo non può essere modificato
- Costanti con nome rendono i programmi più semplici da leggere e manutenere
- Convenzione: usare solo maiuscole nel nome delle costanti

```
final double QUARTER_VALUE = 0.25;
final double DIME_VALUE = 0.1;
final double NICKEL_VALUE = 0.05;
final double PENNY_VALUE = 0.01;
payment = dollars + quarters * QUARTER_VALUE
   + dimes * DIME_VALUE + nickels * NICKEL_VALUE
   + pennies * PENNY_VALUE;
```

Costanti: static final

- Se i valori costanti sono necessari in parecchi metodi, li si dichiari insieme con le variabili di istanza di una classe e li si indichi come `static` e `final`
- Si dia alle costanti `static final` accesso pubblico per permettere ad altre classi di utilizzarle

```
public class Math
  {
     . . .
     public static final double E = 2.71828182845904523 54;
     public static final double PI =  3.14159265358979323846;
  }

  double circumference = Math.PI * diameter;
```

Operazioni aritmetiche

- `/` è l'operatore di divisione
- Se entrambi gli argomenti sono interi il risultato è intero. Il resto viene scartato.
- `7.0 / 4` produce `1.75`
  `7 / 4` produce `1`
- Il resto si può ottenere con `%` (detto "modulo")
  `7 % 4` produce `3`

La classe `Math`

- Classe `Math`: contiene metodi come `sqrt` e `pow`
- Per calcolare x^n, scrivere `Math.pow(x, n)`
- Comunque, per calcolare x^2 è molto più efficiente semplicemente calcolare `x * x`
- Per calcolare la radice quadrata di un numero, usare `Math.sqrt`; per esempio, `Math.sqrt(x)`

Metodi di tipo matematico

Funzione	**Valore restituito**
Math.sqrt(x)	Radice quadrata
Math.pow(x, y)	Potenze x^y
Math.exp(x)	e^x
Math.log(x)	Logaritmo naturale
Math.sin(x), Math.cos(x), Math.tan(x)	Seno, coseno, tangente (x in radianti)
Math.round(x)	L'intero più vicino a x
Math.min(x, y), Math.max(x, y)	Minimo, massimo

Chiamare metodi statici

- Un metodo `static` non opera su oggetti
- I metodi statici operano su classi
- Convenzione per la denominazione: i nomi delle classi iniziano con una lettera maiuscola, quelli degli oggetti con una lettera minuscola

```
Math
System.out
```

Stringhe

- Una stringa è una sequenza di caratteri
- Le stringhe sono oggetti della classe String
- Costanti stringa:
  `"Hello, World!"`
- Variabili stringa:
  `String message = "Hello, World!";`
- Lunghezza di una stringa:
  `int n = message.length();`
- Stringa vuota: `""`

Concatenazione

- Si può usare l'operatore +:

```
String name = "Dave";
String message = "Hello, " + name; // message is "Hello,
   Dave"
```

- Se uno degli argomenti dell'operatore `+` è una stringa, l'altro è convertito automaticamente a stringa

```
String a = "Agent";
int n = 7;
String bond = a + n; // bond is "Agent7"
```

Concatenazione in operazioni di visualizzazione

- Utile per ridurre il numero di istruzioni

```
System.out.print("The total is ");
System.out.println(total);
```

rispetto a

```
System.out.println("The total is " + total);
```

Conversione tra stringhe e numeri

- Convertire ad un numero:

```
int n = Integer.parseInt(str);
double x = Double.parseDouble(x);
```

- Convertire a stringa:

```
String str = "" + n;
str = Integer.toString(n);
```

Sottostringhe

- ```
  String greeting = "Hello, World!";
  String sub = greeting.substring(0, 5); // sub is "Hello"
  ```
- Fornire la posizione iniziale e quella successiva alla finale
- La prima posizione è `0`

Figure 3 String Positions

Continua
  ```

Sottostringhe (continua)

La lunghezza delle sottostringhe è la differenza tra i valori forniti

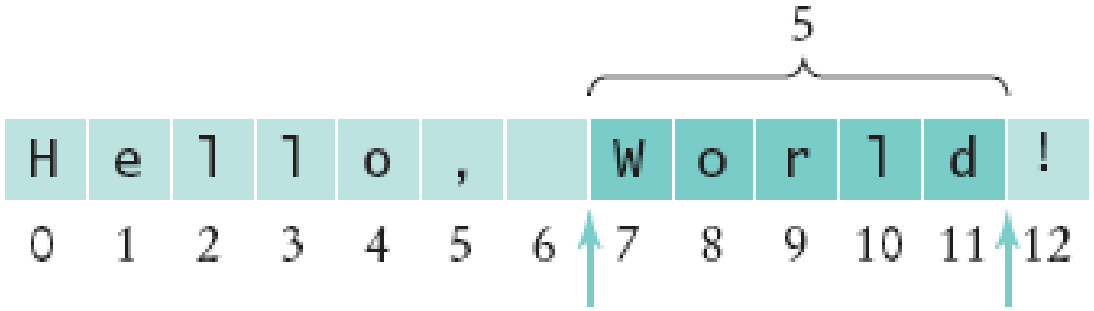

Figure 4 Extracting a Substring

String sub = greeting.substring(7, 12);

La sottostringa sub è lunga 12-7=5 caratteri

Alfabeti internazionali

A German Keyboard

Alfabeti internazionali

The Thai Alphabet

Leggere l’input

- System.in fornisce pochissimi strumenti – può leggere soltanto un byte per volta
- In Java 5.0 è stata aggiunta la classe Scanner per leggere l’input da tastiera in modo adatto
- ```
  Scanner in = new Scanner(System.in);
  System.out.print("Enter quantity:");
  int quantity = in.nextInt();
  ```
- `nextDouble` legge un double
- `nextLine` legge una linea (finché l’utente preme Enter)
- `nextWord` legge una parola (fino ad uno spazio bianco)
  ```

Leggere l’input con un box di dialogo

An Input Dialog Box

Leggere l’input con un box di dialogo

- `String input = JOptionPane.showInputDialog(`*`prompt`*`)`
- Convertire stringhe a numeri se necessario:

  ```
  int count = Integer.parseInt(input);
  ```

- Le conversioni lanciano un’eccezione se l’utente non fornisce un numero
- Si aggiunga `System.exit(0)` al metodo main di qualunque programma che impiega `JOptionPane`

Visualizzare l'output con un box di dialogo

- `JOptionPane.showMessageDialog(`*`parent`*`, `*`message`*`)`
- *parent* è il frame nel quale il box di dialogo è visualizzato; può valere anche *null*.
- *message* è il messaggio visualizzato
- Ci sono anche versioni con ulteriori parametri