

# Tipi di dati fondamentali

---

# Numeri

Ogni valore in Java è: un riferimento a un oggetto oppure appartiene ad uno degli otto tipi primitivi

Sei tipi primitivi sono numerici:

- Quattro rappresentano numeri interi
- due tipi rappresentano numeri in virgola mobile

# Numeri

Ogni valore in Java è: un riferimento a un oggetto oppure appartiene ad uno degli otto tipi primitivi

Sei tipi primitivi sono numerici:

- Quattro rappresentano numeri interi
- due tipi rappresentano numeri in virgola mobile

Tipo	Descrizione	DIM
int	Tipo intero con intervallo -2,147,483,648 (Integer.MIN_VALUE) . . . 2,147,483,647 (Integer.MAX_VALUE) circa 2 miliardi	4 bytes
byte	Tipo che descrive un singolo byte con intervallo -128 . . . 127	1 byte
short	Tipo intero corto 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	Tipo in virgola mobile a doppia precisione con intervallo $\pm 10^{308}$ e circa 15 cifre significative	8 bytes
float	Tipo in virgola mobile a singola precisione con intervallo $\pm 10^{38}$ e circa 7 cifre significative	4 bytes
char	Tipo che rappresenta caratteri codificati secondo lo schema Unicode	2 Bytes
boolean	Tipo per i due valori logici true e false	1 bit

# Numeri

# Numeri

A thick yellow horizontal bar spans the width of the slide, with a vertical yellow bar extending downwards from its right end.



Un numero che appare nel tuo codice

Se ha un decimale, è in virgola mobile

In caso contrario, è un intero

# Numeri

**Table 2** Number Literals in Java

Number	Type	Comment
6	int	An integer has no fractional part.
-6	int	Integers can be negative.
0	int	Zero is an integer.
0.5	double	A number with a fractional part has type double.
1.0	double	An integer with a fractional part .0 has type double.
1E6	double	A number in exponential notation: $1 \times 10^6$ or 1000000. Numbers in exponential notation always have type double.
2.96E-2	double	Negative exponent: $2.96 \times 10^{-2} = 2.96 / 100 = 0.0296$
100000L	long	The L suffix indicates a long literal.
 100,000		<b>Error:</b> Do not use a comma as a decimal separator.
100_000	int	You can use underscores in number literals.
 3 1/2		<b>Error:</b> Do not use fractions; use decimal notation: 3.5

# Numeri

Generalmente si usa un int per i numeri interi

L'overflow si verifica quando il risultato di un calcolo supera l'intervallo per il tipo di numero. Esempio

```
int n = 1000000;  
System.out.println(n * n); // Prints -727379968  
// which is clearly wrong
```

Il risultato è maggiore del più grande int

Il risultato viene troncato per adattarsi a un int

Non viene fornito alcun avviso

Soluzione: usa invece lungo

Generalmente non si ha overflow con il tipo di dati double

# Errori di arrotondamento

Errori di arrotondamento si verificano quando non è possibile una rappresentazione esatta di un numero a virgola mobile.

I numeri in virgola mobile hanno una precisione limitata.

Non tutti i valori possono essere rappresentati con precisione e possono verificarsi errori di arrotondamento. Esempio

```
double f = 4.35;
```

```
System.out.println(100 * f); // Prints 434.99999999999994
```



# Costanti:

Usa nomi simbolici per tutti i valori, anche quelli che sembrano ovvi. Una variabile final è una costante

Una volta impostato il suo valore, non può essere modificato

Le costanti denominate semplificano la lettura e la manutenzione dei programmi. Convenzione: utilizzare nomi in maiuscolo per le 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:

Se sono necessari valori costanti in diversi metodi,

- Dichiarali insieme alle variabili di istanza di una classe
- Aggiungi static e final

La parola riservata static significa che la costante appartiene alla classe Dare accesso pubblico alle costanti final static per consentire ad altre classi di usarle:

## Dichiarazione delle costanti nella classe Math

```
public class Math
{
    . . .
    public static final double E = 2.7182818284590452354;
    public static final double PI = 3.14159265358979323846;
}
double circumference = Math.PI * diameter;
```

# Costanti:

*Syntax* Declared in a method: `final typeName variableName = expression;`

Declared in a class: `accessSpecifier static final typeName variableName = expression;`

Declared in a method

`final double NICKEL_VALUE = 0.05;`

The final reserved word indicates that this value cannot be modified.

Use uppercase letters for constants.

`public static final double LITERS_PER_GALLON = 3.785;`

Declared in a class

# Vedi esercizio CashRegisterCostanti

```
1 /**
2  A cash register totals up sales and computes change due.
3  */
4  public class CashRegisterCostanti
5  {
    public static final double QUARTER_VALUE = 0.25;
    public static final double DIME_VALUE = 0.1;
    public static final double NICKEL_VALUE = 0.05;
    public static final double PENNY_VALUE = 0.01;
```

10

.....

# Esercizio 1

Quale delle seguenti inizializzazioni non è corretta e perché?

- `int dollari = 100,0;`
- `double saldo = 100;`

# Esercizio 2

Qual è la differenza tra queste due righe di codice?

```
final double CM_PER_INCH = 2.54;  
public static final double CM_PER_INCH = 2.54;
```

# Esercizio 3

Cosa c'è di sbagliato?

```
double diameter = . . . ;  
double circumference = 3.14 * diameter;
```

# Operatori aritmetici

Quattro operatori di base:

addizione: +

sottrazione: -

moltiplicazione: \*

divisione: /

Espressione: combinazione di variabili, letterali, operatori e/o chiamate di metodo

$(a + b) / 2$



# Operatori aritmetici

Le parentesi controllano l'ordine di computazione:

$$(a + b) / 2$$

La moltiplicazione e la divisione hanno una precedenza maggiore rispetto all'addizione e alla sottrazione

$$a + b / 2$$

La combinazione di numeri interi e valori in virgola mobile in un'espressione aritmetica produce un valore in virgola mobile

`7 + 4.0` is the floating-point value `11.0`

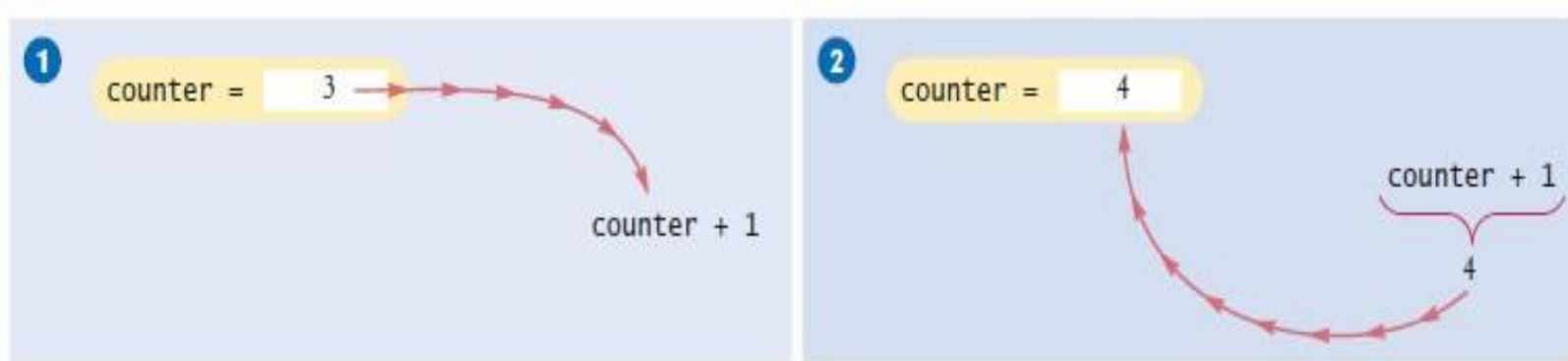
# Incrementare e decrementare

L'operatore ++ aggiunge 1 a una variabile (incrementa)

contatore++; // Aggiunge 1 alla variabile counter

L'operatore -- sottrae 1 dalla variabile (decrementa)

contatore--; // Sottrae 1 dal contatore



# Divisione intera e resto

La divisione funziona come ci si aspetterebbe, purché almeno uno dei numeri sia un numero in virgola mobile.

Esempio: tutti i seguenti valgono 1.75

$7.0 / 4.0$

$7 / 4.0$

$7.0 / 4$

Se entrambi i numeri sono interi, il risultato è un intero. Il resto viene scartato  $7/4$  è uguale ad 1

Usa l'operatore % per ottenere il resto con (pronunciato "modulus", "modulo" o "mod")

$7\% 4$  è 3

# Divisione intera e resto

Per determinare il valore in dollari e centesimi di 1729 pennies

Ottieni i dollari attraverso una divisione intera per 100

```
int dollars = pennies / 100; // Sets dollars to 17
```

ottieni il resto con l'operatore %

```
int cents = pennies % 100; // Sets cents to 29
```

La divisione intera e l'operatore % producono i valori in dollari e cent di un salvadanaio pieno di pennies.

# Divisione intera e resto

Table 3 Integer Division and Remainder

Expression (where $n = 1729$ )	Value	Comment
$n \% 10$	9	$n \% 10$ is always the last digit of $n$ .
$n / 10$	172	This is always $n$ without the last digit.
$n \% 100$	29	The last two digits of $n$ .
$n / 10.0$	172.9	Because 10.0 is a floating-point number, the fractional part is not discarded.
$-n \% 10$	-9	Because the first argument is negative, the remainder is also negative.
$n \% 2$	1	$n \% 2$ is 0 if $n$ is even, 1 or -1 if $n$ is odd.

# Potenze e radici

La classe Math contiene i metodi sqrt e pow per calcolare radici quadrate e potenze

- Per calcolare la radice quadrata di un numero, utilizzare Math.sqrt; ad esempio, Math.sqrt(x)
- Per calcolare  $x^n$ , scrivi Math.pow(x, n)
- Per calcolare  $x^2$  è significativamente più efficiente calcolare  $x * x$

Per Java,

$$b \times \left(1 + \frac{r}{100}\right)^n$$

può essere rappresentato come:

```
b * Math.pow(1 + r / 100, n)
```

# Potenze e radici

Per Java,

$$b \times \left(1 + \frac{r}{100}\right)^n$$

può essere rappresentato come:

$$\begin{array}{c} b * \text{Math.pow}(1 + r / 100, n) \\ \underbrace{\hspace{1.5cm}}_{\frac{r}{100}} \\ \underbrace{\hspace{1.5cm}}_{1 + \frac{r}{100}} \\ \underbrace{\hspace{1.5cm}}_{\left(1 + \frac{r}{100}\right)^n} \\ \underbrace{\hspace{1.5cm}}_{b \times \left(1 + \frac{r}{100}\right)^n} \end{array}$$

# Metodi della classe Math

Table 4 Mathematical Methods			
Method	Returns	Method	Returns
<code>Math.sqrt(x)</code>	Square root of $x$ ( $\geq 0$ )	<code>Math.abs(x)</code>	Absolute value $ x $
<code>Math.pow(x, y)</code>	$x^y$ ( $x > 0$ , or $x = 0$ and $y > 0$ , or $x < 0$ and $y$ is an integer)	<code>Math.max(x, y)</code>	The larger of $x$ and $y$
<code>Math.sin(x)</code>	Sine of $x$ ( $x$ in radians)	<code>Math.min(x, y)</code>	The smaller of $x$ and $y$
<code>Math.cos(x)</code>	Cosine of $x$	<code>Math.exp(x)</code>	$e^x$
<code>Math.tan(x)</code>	Tangent of $x$	<code>Math.log(x)</code>	Natural log ( $\ln(x)$ , $x > 0$ )
<code>Math.round(x)</code>	Closest integer to $x$ (as a long)	<code>Math.log10(x)</code>	Decimal log ( $\log_{10}(x)$ , $x > 0$ )
<code>Math.ceil(x)</code>	Smallest integer $\geq x$ (as a double)	<code>Math.floor(x)</code>	Largest integer $\leq x$ (as a double)
<code>Math.toRadians(x)</code>	Convert $x$ degrees to radians (i.e., returns $x \cdot \pi/180$ )	<code>Math.toDegrees(x)</code>	Convert $x$ radians to degrees (i.e., returns $x \cdot 180/\pi$ )



# Conversione di numeri in virgola mobile in numeri interi - Cast

Il compilatore non consente l'assegnazione di un double a un int perché è potenzialmente pericoloso

La parte frazionaria è persa

La grandezza potrebbe essere troppo grande

Questo è un errore

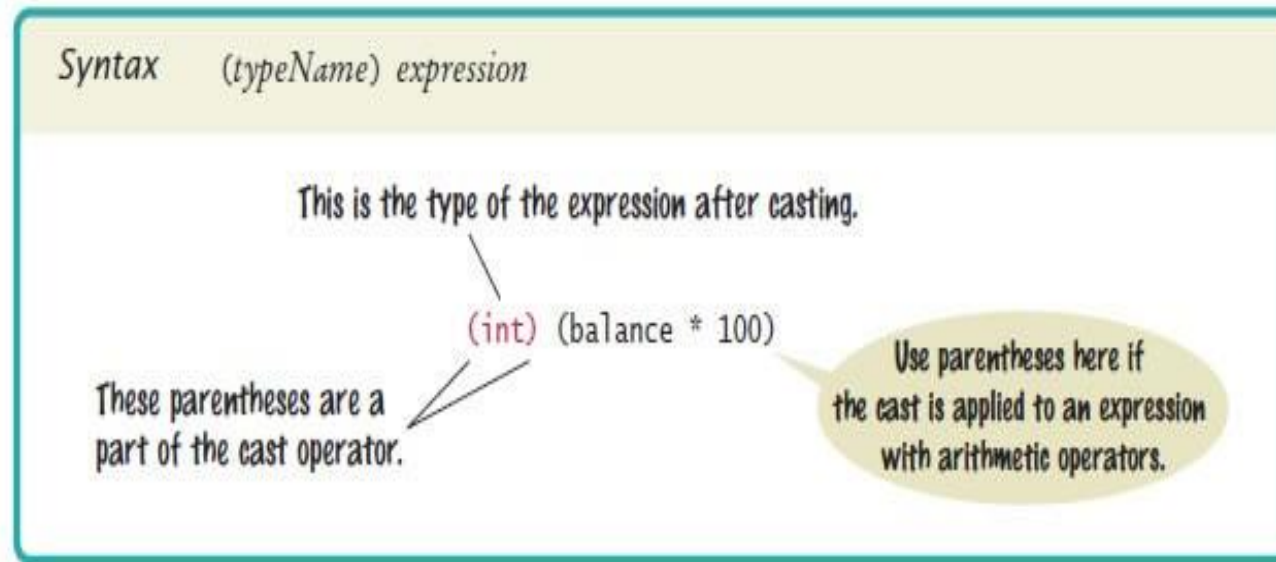
```
double balance = total + tax;  
int dollars = balance; // Error: Cannot assign double to int
```

Utilizzare l'operatore cast (int) per convertire un valore a virgola mobile di conversione in un numero intero.

```
double balance = total + tax;  
int dollars = (int) balance;
```

# Conversione di numeri in virgola mobile in numeri interi - Cast

- Cast scarta parte frazionaria
- Usa un cast (typeName) per convertire un valore in un tipo diverso



# Round

- `Math.round` converte un numero in virgola mobile nell'intero più vicino:

```
long rounded = Math.round(balance);
```

- Se il saldo è 13,75, l'arrotondamento è impostato a 14.

# Espressioni aritmetiche

Table 5 Arithmetic Expressions		
Mathematical Expression	Java Expression	Comments
$\frac{x + y}{2}$	<code>(x + y) / 2</code>	The parentheses are required; <code>x + y / 2</code> computes $x + \frac{y}{2}$ .
$\frac{xy}{2}$	<code>x * y / 2</code>	Parentheses are not required; operators with the same precedence are evaluated left to right.
$\left(1 + \frac{r}{100}\right)^n$	<code>Math.pow(1 + r / 100, n)</code>	Use <code>Math.pow(x, n)</code> to compute $x^n$ .
$\sqrt{a^2 + b^2}$	<code>Math.sqrt(a * a + b * b)</code>	<code>a * a</code> is simpler than <code>Math.pow(a, 2)</code> .
$\frac{i + j + k}{3}$	<code>(i + j + k) / 3.0</code>	If <i>i</i> , <i>j</i> , and <i>k</i> are integers, using a denominator of 3.0 forces floating-point division.
$\pi$	<code>Math.PI</code>	<code>Math.PI</code> is a constant declared in the <code>Math</code> class.

# Esercizio 1

Un conto in banca guadagna interessi una volta all'anno.

In Java, come si calcola l'interesse guadagnato nel primo anno?

Supponiamo che le variabili `percent` e `balance` di tipo `double` siano già state dichiarate.

## Esercizio 2

In Java, come si calcola la lunghezza del lato di un quadrato la cui area è memorizzata nell'area variabile?

## Esercizio 2

In Java, come si calcola la lunghezza del lato di un quadrato la cui area è memorizzata nell'area variabile?

# Esercizio 3

Volume sfera:

$$V = \frac{4}{3}\pi r^3$$

Se il raggio è dato da un raggio variabile di tipo double, scrivi un'espressione Java per il volume.

`double volume= 4*Math.PI*Math.pow(raggio,3)/3; double!!`

`double volume=(double ) (4/3)* Math.PI*Math.pow(raggio,3)`

`double volume= 4.0/3.0*Math.PI*Math.pow(raggio,3)`

`double volume= 4.0/3* Math.PI*Math.pow(raggio,3)`

`double volume=4/3* Math.PI*Math.pow(raggio,3) //ERRORE  
(1*Math.PI*Math.pow(raggio,3))`



# Richiamare metodi statici

- Impossibile chiamare un metodo su un tipo numero

```
double root = 2.sqrt(); // Error
```

- Bisogna utilizzare invece un metodo statico. Un metodo statico non opera su un oggetto:

```
double root = Math.sqrt(2); // Correct
```

- I metodi statici sono dichiarati all'interno delle classi
- Per chiamare un metodo statico:

The name of the class      The name of the static method

Math.sqrt(2)

# Leggere input da tastiera

- Per ottenere un oggetto di tipo Scanner:

```
Scanner in = new Scanner(System.in);
```

- Usare nextInt() per leggere interi

```
System.out.print("Please enter the number of bottles: ");  
int bottles = in.nextInt();
```

- Usare nextDouble() per leggere double

```
System.out.print("Enter price: ");  
double price = in.nextDouble();
```

- Per utilizzare la classe Scanner, si deve importarla inserendo quanto segue nella parte superiore del file del programma:

```
import java.util.Scanner;
```

# Leggere input da tastiera

Include this line so you can use the Scanner class.

```
import java.util.Scanner;
```

Create a Scanner object to read keyboard input.

```
Scanner in = new Scanner(System.in);
```

Don't use println here.

Display a prompt in the console window.

```
System.out.print("Please enter the number of bottles: ");
```

Define a variable to hold the input value.

```
int bottles = in.nextInt();
```

The program waits for user input, then places the input into the variable.

# Output formattato:

Utilizzare il metodo printf per specificare come devono essere formattati i valori.

Printf ti consente di stampare questo

Prezzo per litro: 1.22

Invece di questo Prezzo per litro: 1.215962441314554

Questo comando visualizza il prezzo con due cifre dopo la virgola:

```
System.out.printf("%.2f", price);
```

# Output formattato:

Puoi anche specificare una larghezza del campo

```
System.out.printf("%10.2f", price);
```

Stampa 10 caratteri:

Sei spazi seguiti dai quattro caratteri 1.22



```
System.out.printf("Price per liter:%10.2f", price);
```

```
Price per liter:  1.22
```

# Output formattato

**Table 6** Format Specifier Examples

Format String	Sample Output	Comments
"%d"	24	Use d with an integer.
"%5d"	24	Spaces are added so that the field width is 5.
"Quantity:%5d"	Quantity: 24	Characters inside a format string but outside a format specifier appear in the output.
"%f"	1.21997	Use f with a floating-point number.
"%.2f"	1.22	Prints two digits after the decimal point.
"%7.2f"	1.22	Spaces are added so that the field width is 7.
"%s"	Hello	Use s with a string.
"%d %.2f"	24 1.22	You can format multiple values at once.

# Vedi esercizio Volume.java

- Vedi esercizio  
Volume.java

```
3  /**
4   * This program prints the price per liter for a six-pack of cans and
5   * a two-liter bottle.
6   */
7  public class Volume
8  {
9      public static void main(String[] args)
10     {
11         // Read price per pack
12
13         Scanner in = new Scanner(System.in); //tastiera, tast, keyb, keyboard
14         System.out.print("Please enter the price for a six-pack: ");
15         double packPrice = in.nextDouble(); //legge un double
16         // Read price per bottle
17         System.out.print("Please enter the price for a two-liter bottle: ");
18         double bottlePrice = in.nextDouble(); //legge un double
19         //costanti
20         final double CANS_PER_PACK = 6;
21         final double CAN_VOLUME = 0.355; // 12 oz. = 0.355 l
22         final double BOTTLE_VOLUME = 2;
23         // Compute and print price per liter
24         double packPricePerLiter = packPrice / (CANS_PER_PACK * CAN_VOLUME);
25         double bottlePricePerLiter = bottlePrice / BOTTLE_VOLUME;
26
27         System.out.printf("Pack price per liter:  %8.2f", packPricePerLiter);
28         //spazio di 8 caratteri, e 2 sono sicuramente i decimali
29         System.out.println();
30
31         System.out.printf("Bottle price per liter: %8.2f", bottlePricePerLiter);
32         System.out.println();
33     }
34 }
```

# Problem solving

Passaggio molto importante per lo sviluppo di un algoritmo

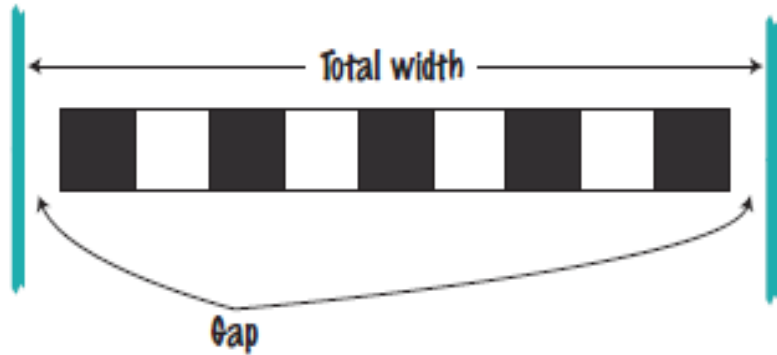
**Eseguire prima i calcoli a mano**

Scegli valori concreti per una situazione tipica da utilizzare in un calcolo manuale.

Problema: è necessario posizionare una fila di tessere bianche e nere lungo una parete. Il primo e l'ultimo sono neri. Calcola il numero di tessere necessarie e lo spazio che rimane vuoto a ciascuna estremità, dato lo spazio disponibile e la larghezza di ciascuna tessera.



# Problem solving



Usa i numeri seguenti

Larghezza totale: 100 pollici

Larghezza piastrella: 5 pollici

La prima tessera deve essere sempre nera, e poi aggiungiamo un certo numero di coppie bianco/nero

# Problem solving

La prima tessera occupa 5 pollici, lasciando 95 pollici da coprire a coppie.

Ogni coppia è larga 10 pollici.

Il numero di coppie necessarie è  $95 / 10 = 9,5$ .

Scartare la parte frazionaria.

Abbiamo bisogno di 9 coppie di tessere o 18 tessere, più la tessera nera iniziale => 19 tessere.

Le tessere misurano  $19 \times 5 = 95$  pollici

La distanza è  $100 - 19 \times 5 = 5$  pollici

Distribuire lo spazio su entrambe le estremità lo spazio è

$(100 - 19 \times 5) / 2 = 2,5$  pollici

# Problem solving

QUINDI si passa ad elaborare un algoritmo con valori arbitrari per la larghezza totale e la larghezza della piastrella.

Lo pseudocodice:

number of pairs = integer part of  $(\text{total width} - \text{tile width}) / (2 \times \text{tile width})$

number of tiles =  $1 + 2 \times \text{number of pairs}$

gap at each end =  $(\text{total width} - \text{number of tiles} \times \text{tile width}) / 2$

In Operazioni.java la soluzione

# Esercizio 1

Supponiamo che l'architetto specifichi un modello con piastrelle nere, grigie e bianche, come questa:



Anche in questo caso, la prima e l'ultima tessera dovrebbero essere nere. Come è necessario modificare l'algoritmo?

- Risposta: Ora ci sono gruppi di quattro tessere (grigio/bianco/grigio/nero) che seguono la tessera nera iniziale. **Pertanto, l'algoritmo è ora**  
$$\text{number of groups} = \text{integer part of } (\text{total width} - \text{tile width}) / (4 \times \text{tile width})$$
$$\text{number of tiles} = 1 + 4 \times \text{number of groups}$$

## Esercizio 2

Un robot ha bisogno di piastrellare un pavimento con piastrelle bianche e nere alternate. Sviluppa un algoritmo che dia il colore (0 per il nero, 1 per il bianco), dato il numero di riga e colonna. Inizia con valori specifici per la riga e la colonna, quindi generalizza.

	1	2	3	4
1	Black	White	Black	White
2	White	Black	White	Black
3	Black	White	Black	
4	White	Black		

# Esercizio 2

Risposta: la risposta dipende solo dal fatto che i numeri di riga e colonna siano pari o dispari, quindi prendiamo prima il resto dopo aver diviso per 2. Quindi possiamo enumerare tutte le risposte previste:

Rows%2	Columns%2	Color
0	0	0
0	1	1
1	0	1
1	1	0

Nelle prime tre voci della tabella, il colore è semplicemente la somma dei resti. Nella quarta voce, la somma sarebbe 2, ma vogliamo uno zero. Possiamo ottenerlo prendendo un'altra operazione resto:

```
color = ((row % 2) + (column % 2)) % 2
```

# Esercizio 3

Per una determinata automobile, i costi di riparazione e manutenzione nell'anno 1 sono stimati a \$ 100; nell'anno 10, a \$ 1.500.

Supponendo che il costo di riparazione aumenti della stessa quantità ogni anno, sviluppare uno pseudocodice per calcolare il costo di riparazione nell'anno 3 e quindi generalizzare all'anno  $n$

in nove anni i costi di riparazione sono aumentati di 1.400 dollari. Pertanto, l'aumento annuo è di  $\$ 1.400 / 9 \approx \$ 156$ . Il costo di riparazione nell'anno 3 sarebbe  $\$ 100 + 2 \times \$ 156 = \$412$ . Il costo di riparazione nell'anno  $n$  è  $\$ 100 + n \times \$ 156$ . Per evitare l'accumulo di errori di arrotondamento, in realtà è una buona idea utilizzare l'espressione originale che ha prodotto \$ 156, ovvero

`Repair cost in year n = 100 + n x 1400 / 9`

# Esercizio 4

La forma di una bottiglia è approssimata da due cilindri di raggio  $r_1$  e  $r_2$  e altezze  $h_1$  e  $h_2$ , uniti da una sezione conica di altezza  $h_3$ . Usando le formule per il volume di un cilindro,  $V = \pi r^2 h$ , e una sezione conica

$$V = \pi \frac{(r_1^2 + r_1 r_2 + r_2^2) h}{3}$$

sviluppare pseudocodice per calcolare il volume della bottiglia. Usando una bottiglia reale con volume noto come campione, fai un calcolo manuale del tuo pseudocodice.



# Esercizio 4

Misurare i volumi di una tipica bottiglia di vino  $r_1 = 3,6$ ,  $r_2 = 1,2$ ,  $h_1 = 15$ ,  $h_2 = 7$ ,  $h_3 = 6$  (tutti in centimetri).

QUINDI:

bottom volume= 610,73

Top volume= 31,67

Middle volume= 135,72

Total volume= 778,12

Il volume effettivo è di 750 ml, che è abbastanza vicino al nostro calcolo per dare la certezza che sia corretto.

Stringhe

# Stringhe

- Una stringa è una sequenza di caratteri.
- Puoi dichiarare variabili che contengono stringhe

```
String name = "Harry";
```

- Una variabile String è una variabile che può contenere una stringa
- I caratteri String sono sequenze di caratteri racchiusi tra virgolette

```
"Harry"
```

# Stringhe

- La lunghezza della stringa è il numero di caratteri nella stringa  
La lunghezza di "Harry" è 5
- Il metodo `length()` restituisce il numero di caratteri in una stringa  
`int n = nome.length();`
- Una stringa di lunghezza 0 è chiamata stringa vuota
  - Non contiene caratteri
  - È scritto: `""`

# Stringhe-concatenazione

- Concatenare le stringhe significa metterle insieme per formare una stringa più lunga
- Si usa l'operatore +:
  - Esempio:

```
String firstName = "Harry";  
String lastName = "Morgan";  
String name = firstName +lastName;
```

- Result: “HarryMorgan”
- Per separare con uno spazio vuoto:

```
String name = firstName + " " + lastName;
```

- Result: “Harry Morgan”

# Stringhe-concatenazione

- Concatenare le stringhe significa metterle insieme per formare una stringa più lunga
- Si usa l'operatore +:
  - Esempio:

```
String firstName = "Harry";  
String lastName = "Morgan";  
String name = firstName +lastName;
```

- Result: “HarryMorgan”
- Per separare con uno spazio vuoto:

```
String name = firstName + " " + lastName;
```

- Result: “Harry Morgan”

# Stringhe-concatenazione

- Se uno degli argomenti dell'operatore + è una stringa
- **L'altro è costretto a diventare una stringa:** entrambe le stringhe vengono quindi concatenate
- Esempio:

```
String jobTitle = "Agent";  int  
employeeId = 7;  
String bond = jobTitle + employeeId;
```

- Risultato: "Agent7"

# Input String

- Utilizzare il metodo `next()` della classe `Scanner` per leggere una stringa contenente una **singola parola**.

```
System.out.print("Please enter your name: ");  
String name = tast.next();
```

- Viene letta una sola parola.
- Si usa una seconda chiamata a `tast.next()` per ottenere una seconda parola



# Input String

- Utilizzare il metodo `next()` della classe `Scanner` per leggere una stringa contenente una **singola parola**.

```
System.out.print("Please enter your name: ");  
String name = tast.next();
```

- Viene letta una sola parola.
- Si usa una seconda chiamata a `tast.next()` per ottenere una seconda parola

# Sequenze escape

- Per includere una virgoletta in una stringa, precederla con una barra rovesciata ( \ )

```
"He said \"Hello\""
```

- Indica che le virgolette che seguono dovrebbero essere una parte della stringa e non segnare la fine della stringa
- È una sequenza di escape
- Per includere una barra rovesciata in una stringa, usa la sequenza di escape \\

```
"C:\\Temp\\Secret.txt"
```

# Sequenze escape

- Un carattere di nuova riga è indicato con la sequenza di escape `\n`
- Un carattere di nuova riga viene spesso aggiunto alla fine della stringa di formato quando si utilizza `System.out.printf`:

```
System.out.printf("Price: %10.2f\n", price);
```

# String e char

- Una stringa è una sequenza di caratteri Unicode.
- Un carattere è un valore del tipo char.
  - I caratteri hanno valori numerici
- I caratteri di tipo char sono delimitati da virgolette singole.
  - 'H' è un carattere. È un valore di tipo char
- Da non confondere con le stringhe
  - "H" è una stringa contenente un singolo carattere. È un valore di tipo String

# String e char

- Le posizioni delle stringhe vengono contate a partire da 0.

H	a	r	r	y
0	1	2	3	4

- Il numero di posizione dell'ultimo carattere è sempre uno in meno della lunghezza della stringa.
  - L'ultimo carattere della stringa "Harry» è alla posizione 4
  - Il metodo charAt restituisce un valore di carattere da una stringa
- Esempio

```
String name = "Harry";  
char start = name.charAt(0);  
char last = name.charAt(4);
```

- Imposta l'inizio al valore 'H' e l'ultimo al valore 'y'.

# Substring

Si utilizza il metodo `substring()` per estrarre una parte di una stringa.

La chiamata al metodo `str.substring(start, end)` restituisce una stringa composta dai caratteri nella stringa `str`, partendo dalla posizione iniziale, e contenente tutti i caratteri fino alla posizione `end` esclusa.

Esempio:

```
String greeting = "Hello, World!";  
String sub = greeting.substring(0, 5); // sub is "Hello"
```

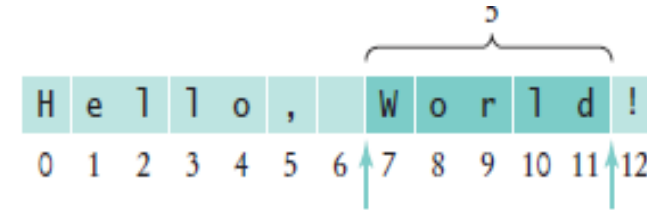
H	e	l	l	o	,		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11	12

# Subtring

Per estrarre la parola World

```
String sub2 = greeting.substring(7, 12);
```

La lunghezza della sottostringa è (end – start)



# Substring

Se si omette la posizione finale quando si chiama il metodo `substring()`, vengono copiati tutti i caratteri dalla posizione iniziale alla fine della stringa.

## Esempio

```
String tail = greeting.substring(7);  
// Copia tutti i valori dalla posizione 7 in poi
```

Risultato: Imposta `tail` a "World!".

Per creare una stringa di un carattere, presa dall'inizio di `first`  
`first.substring(0, 1)`

first =	R	o	d	o	l	f	o
	0	1	2	3	4	5	6
second =	S	a	l	l	y		
	0	1	2	3	4		
initials =	R	&	S				
	0	1	2				



# Esempio

```
1  import java.util.Scanner;
2
3  /**
4   * This program prints a pair of initials.
5   */
6  public class Initials
7  {
8      public static void main(String[] args)
9      {
10         Scanner in = new Scanner(System.in);
11
12         // Get the names of the couple
13
14         System.out.print("Enter your first name: ");
15         String first = in.next();
16         System.out.print("Enter your significant other's first name: ");
17         String second = in.next();
18
19         // Compute and display the inscription
20
21         String initials = first.substring(0, 1)
22             + "&" + second.substring(0, 1);
23         System.out.println(initials);
24     }
25 }
```

**Table 7 String Operations**

Statement	Result	Comment
<code>string str = "Ja"; str = str + "va";</code>	str is set to "Java"	When applied to strings, + denotes concatenation.
<code>System.out.println("Please" + " enter your name: ");</code>	Prints Please enter your name:	Use concatenation to break up strings that don't fit into one line.
<code>team = 49 + "ers"</code>	team is set to "49ers"	Because "ers" is a string, 49 is converted to a string.
<code>String first = in.next(); String last = in.next(); (User input: Harry Morgan)</code>	first contains "Harry" last contains "Morgan"	The next method places the next word into the string variable.
<code>String greeting = "H &amp; S"; int n = greeting.length();</code>	n is set to 5	Each space counts as one character.
<code>String str = "Sally"; char ch = str.charAt(1);</code>	ch is set to 'a'	This is a char value, not a String. Note that the initial position is 0.
<code>String str = "Sally"; String str2 = str.substring(1, 4);</code>	str2 is set to "all"	Extracts the substring starting at position 1 and ending before position 4.
<code>String str = "Sally"; String str2 = str.substring(1);</code>	str2 is set to "ally"	If you omit the end position, all characters from the position until the end of the string are included.
<code>String str = "Sally"; String str2 = str.substring(1, 2);</code>	str2 is set to "a"	Extracts a String of length 1; contrast with <code>str.charAt(1)</code> .
<code>String last = str.substring( str.length() - 1);</code>	last is set to the string containing the last character in str	The last character has position <code>str.length() - 1</code> .

# Cosa restituisce?

```
String str = "Harry";  
int n = str.length();  
String mystery=str.substring(0, 1)+str.substring(n-1, n);  
System.out.println(mystery);
```