Programma Java base (JavaSE) – Fondamenti I

Bibliografia essenziale

Programmazione di base e avanzata con Java – Valter Savitch (it)

Java 9 Guida allo sviluppo in ambienti Windows, macOS e GNU/Linux - Pellegrino Principe (it)

Manuale di Java – Claudio de Sio – Hoepli Informatica (it)

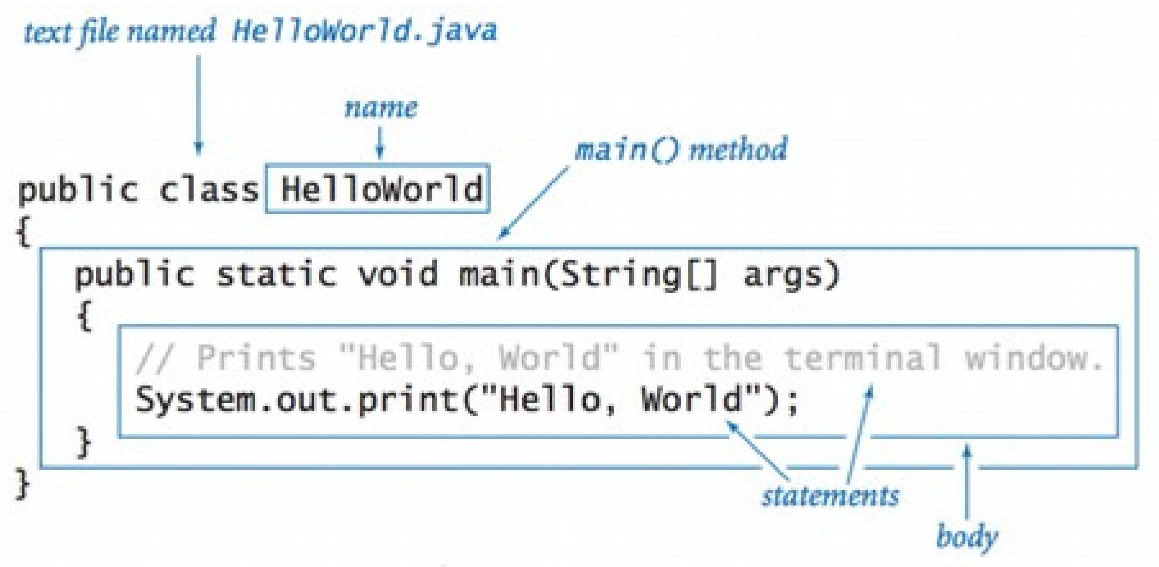
Concetti di informatica e fondamenti di Java - Cay Horstmann (it)

Corso Java – html.it (it)

Java SE Documentation – Oracle/Sun (en)

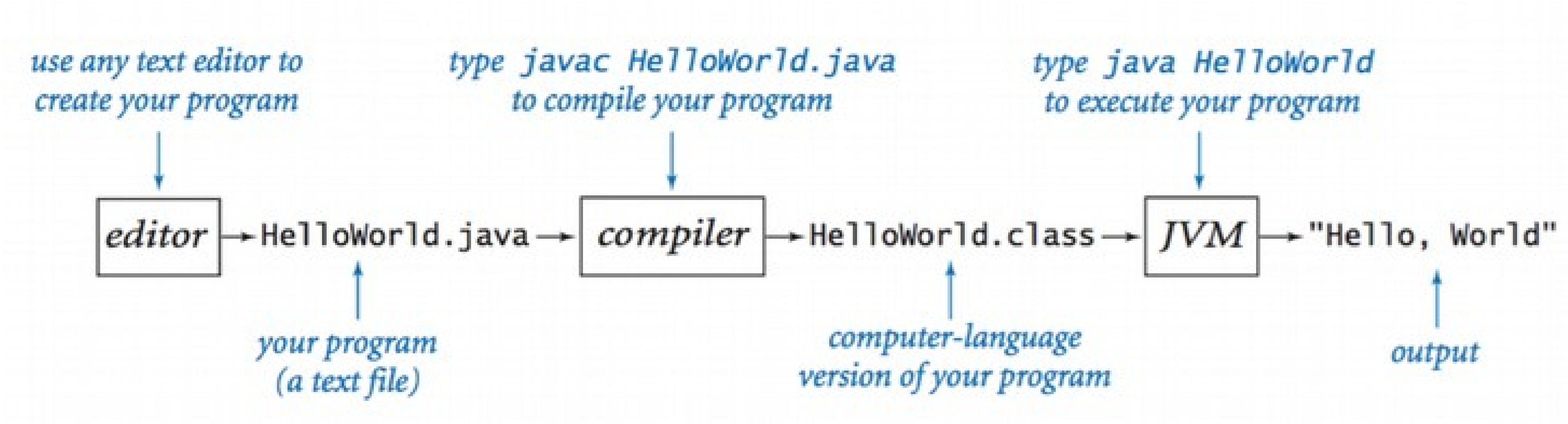
OCA Java Study Guide – Oracle (en)

Scrivere HelloWorld con Java



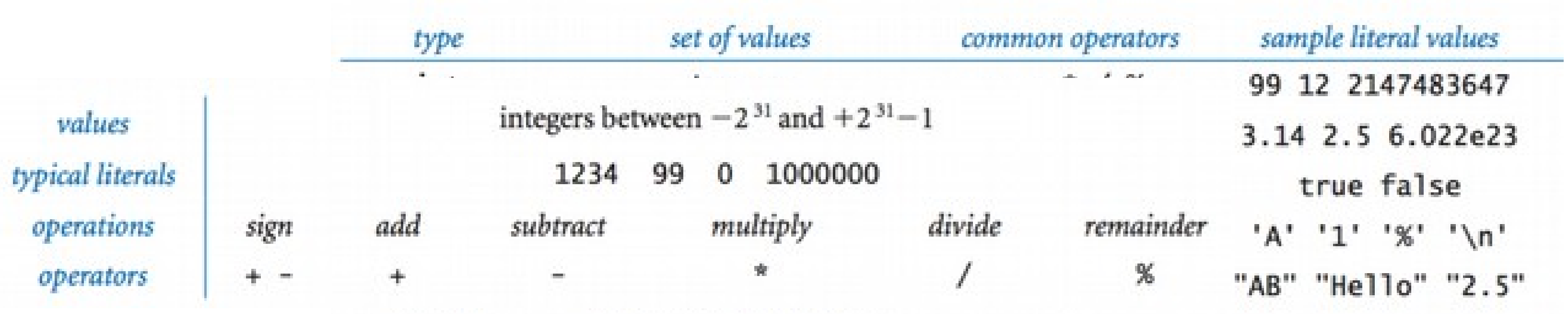
```
public class HelloWorld
{
   public static void main(String[] args)
   {
      // Prints "Hello, World" in the terminal window.
      System.out.print("Hello, World");
   }
}
```

Il processo di compilazione



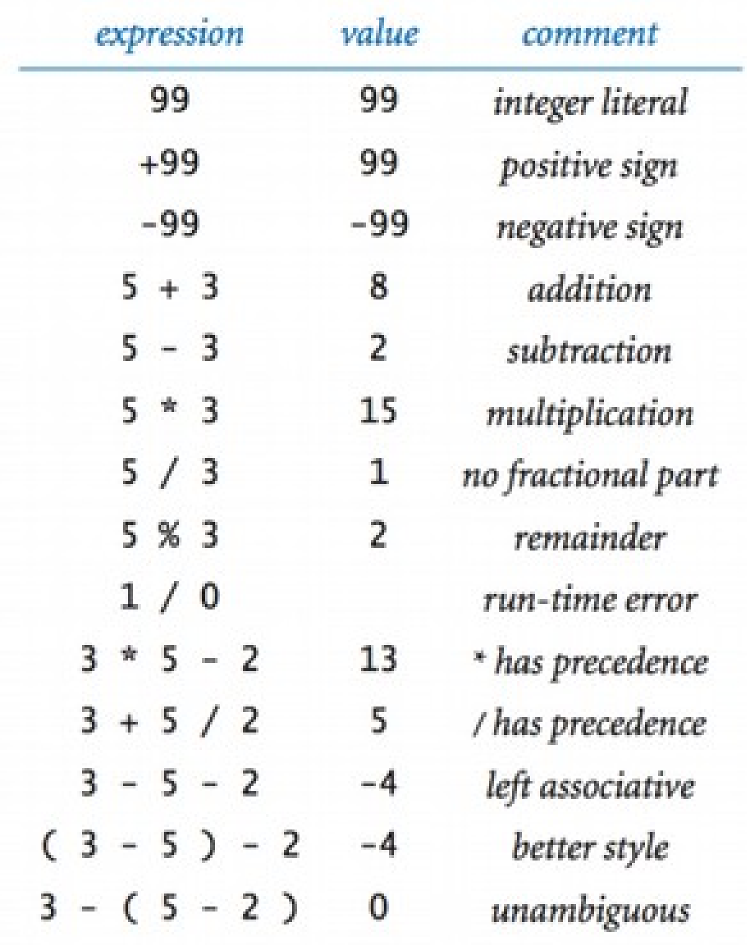
use any text editor to create your program
type javac HelloWorld.java to compile your program
type java HelloWorld to execute your program
editor
HelloWorld.java
compiler
HelloWorld.class
JVM
"Hello, World"
your program (a text file)
computer-language version of your program
output

Identificatori, tipi di dati



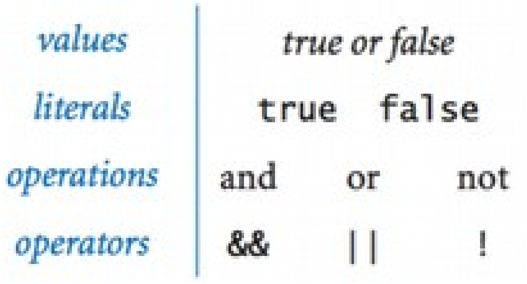
type	set of values	common operators	sample literal values
			99 12 2147483647
			3.14 2.5 6.022e23
			true false
			'A' '1' '%' '\n'
			"AB" "Hello" "2.5"

values	integers between -2^{31} and $+2^{31}-1$					
typical literals	1234 99 0 1000000					
operations	*sign*	*add*	*subtract*	*multiply*	*divide*	*remainder*
operators	+ -	+	-	*	/	%



expression	*value*	*comment*
99	99	*integer literal*
+99	99	*positive sign*
-99	-99	*negative sign*
5 + 3	8	*addition*
5 - 3	2	*subtraction*
5 * 3	15	*multiplication*
5 / 3	1	*no fractional part*
5 % 3	2	*remainder*
1 / 0		*run-time error*
3 * 5 - 2	13	** has precedence*
3 + 5 / 2	5	*/ has precedence*
3 - 5 - 2	-4	*left associative*
(3 - 5) - 2	-4	*better style*
3 - (5 - 2)	0	*unambiguous*

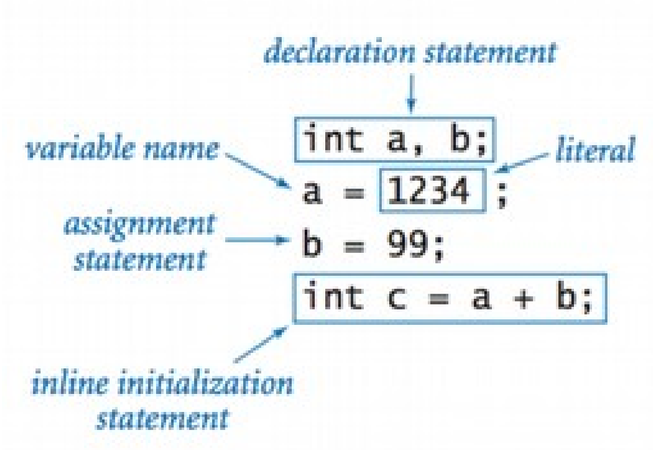
Booleans



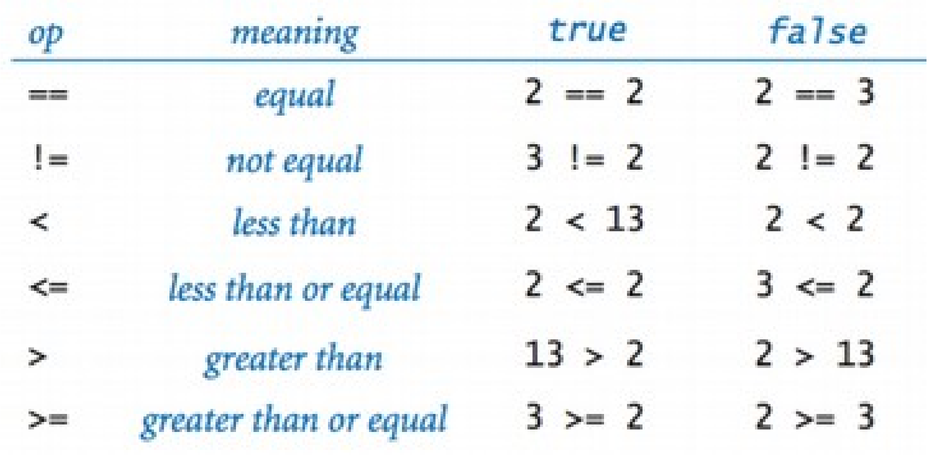
values	*true or false*		
literals	true false		
operations	and	or	not
operators	&&	||	!

Operatori e gestione del flusso di esecuzione

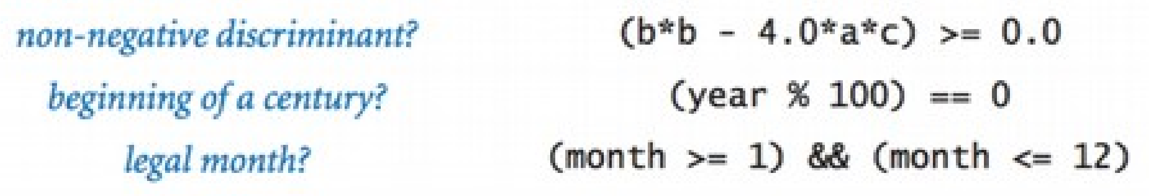
Operatore d'assegnazione



```
int a, b;
a = 1234 ;
b = 99;
int c = a + b;
```

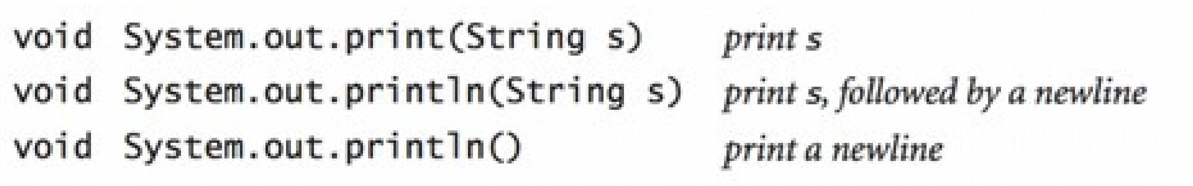


op	*meaning*	true	false
==	*equal*	2 == 2	2 == 3
!=	*not equal*	3 != 2	2 != 2
<	*less than*	2 < 13	2 < 2
<=	*less than or equal*	2 <= 2	3 <= 2
>	*greater than*	13 > 2	2 > 13
>=	*greater than or equal*	3 >= 2	2 >= 3



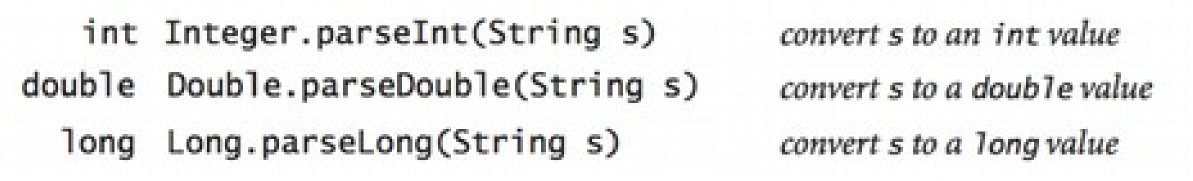
non-negative discriminant?	(b*b - 4.0*a*c) >= 0.0
beginning of a century?	(year % 100) == 0
legal month?	(month >= 1) && (month <= 12)

Sttampare a video



void	System.out.print(String s)	*print s*
void	System.out.println(String s)	*print s, followed by a newline*
void	System.out.println()	*print a newline*

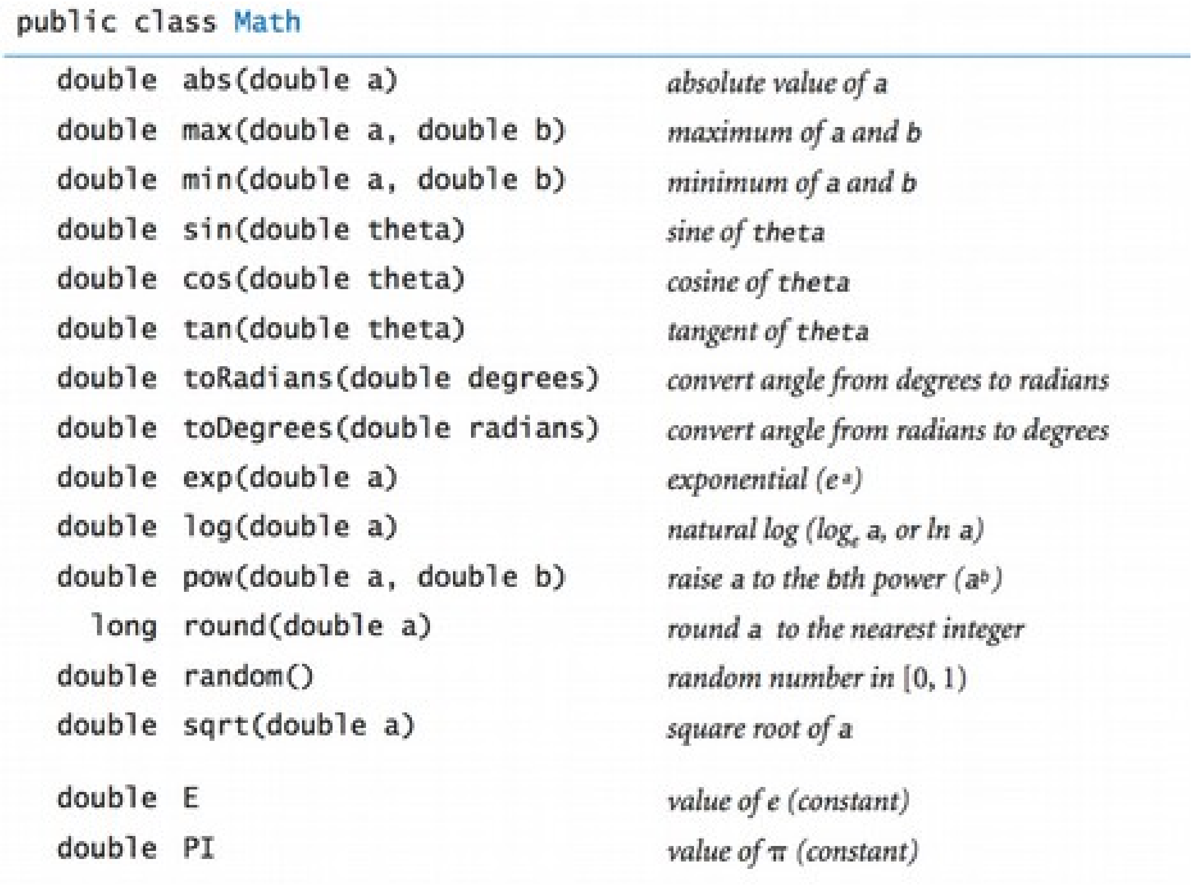
Trasformare argomenti di tipo String.



int	Integer.parseInt(String s)	*convert s to an* int *value*
double	Double.parseDouble(String s)	*convert s to a* double *value*
long	Long.parseLong(String s)	*convert s to a* long *value*

Concatenazione di stringhe con +

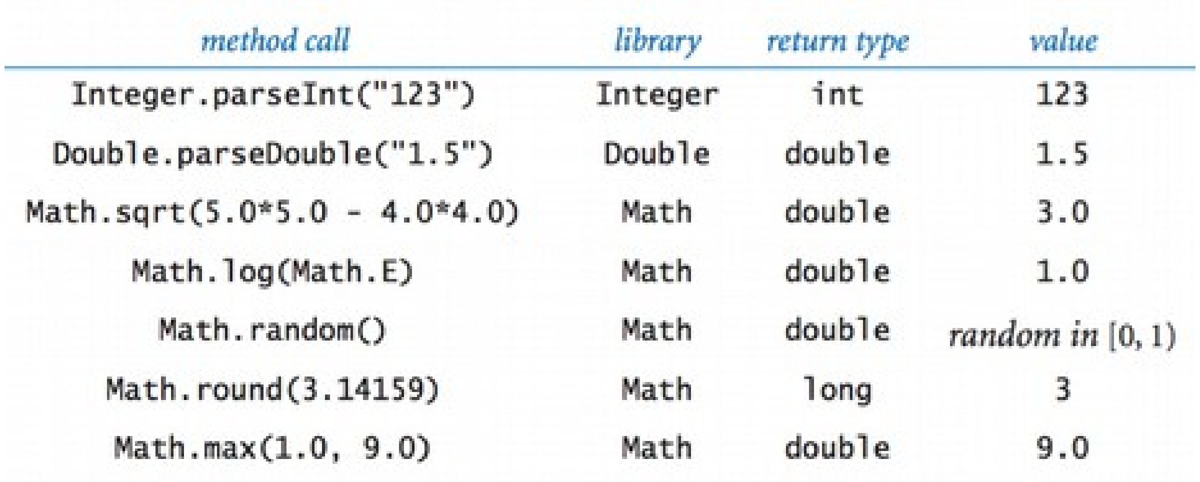
Math library.



public class Math

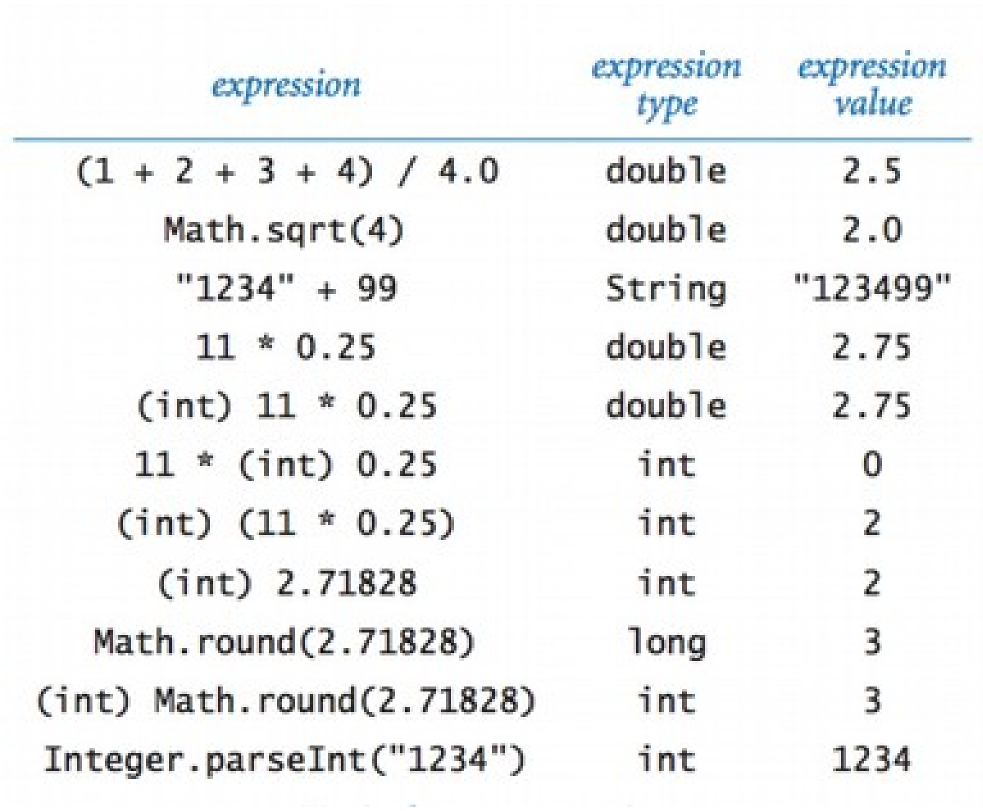
double	abs(double a)	*absolute value of* a
double	max(double a, double b)	*maximum of* a *and* b
double	min(double a, double b)	*minimum of* a *and* b
double	sin(double theta)	*sine of* theta
double	cos(double theta)	*cosine of* theta
double	tan(double theta)	*tangent of* theta
double	toRadians(double degrees)	*convert angle from degrees to radians*
double	toDegrees(double radians)	*convert angle from radians to degrees*
double	exp(double a)	*exponential* (e^a)
double	log(double a)	*natural log* ($\log_e a$, *or* ln a)
double	pow(double a, double b)	*raise* a *to the* b*th power* (a^b)
long	round(double a)	*round* a *to the nearest integer*
double	random()	*random number in* [0, 1)
double	sqrt(double a)	*square root of* a
double	E	*value of* e *(constant)*
double	PI	*value of* π *(constant)*

Java library calls.



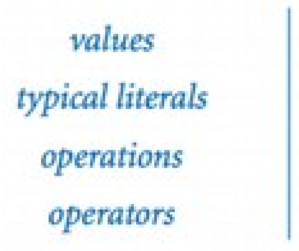
method call	*library*	*return type*	*value*
Integer.parseInt("123")	Integer	int	123
Double.parseDouble("1.5")	Double	double	1.5
Math.sqrt(5.0*5.0 - 4.0*4.0)	Math	double	3.0
Math.log(Math.E)	Math	double	1.0
Math.random()	Math	double	*random in* [0, 1)
Math.round(3.14159)	Math	long	3
Math.max(1.0, 9.0)	Math	double	9.0

Type conversion.



expression	*expression type*	*expression value*
`(1 + 2 + 3 + 4) / 4.0`	`double`	`2.5`
`Math.sqrt(4)`	`double`	`2.0`
`"1234" + 99`	`String`	`"123499"`
`11 * 0.25`	`double`	`2.75`
`(int) 11 * 0.25`	`double`	`2.75`
`11 * (int) 0.25`	`int`	`0`
`(int) (11 * 0.25)`	`int`	`2`
`(int) 2.71828`	`int`	`2`
`Math.round(2.71828)`	`long`	`3`
`(int) Math.round(2.71828)`	`int`	`3`
`Integer.parseInt("1234")`	`int`	`1234`

If and if-else statements.

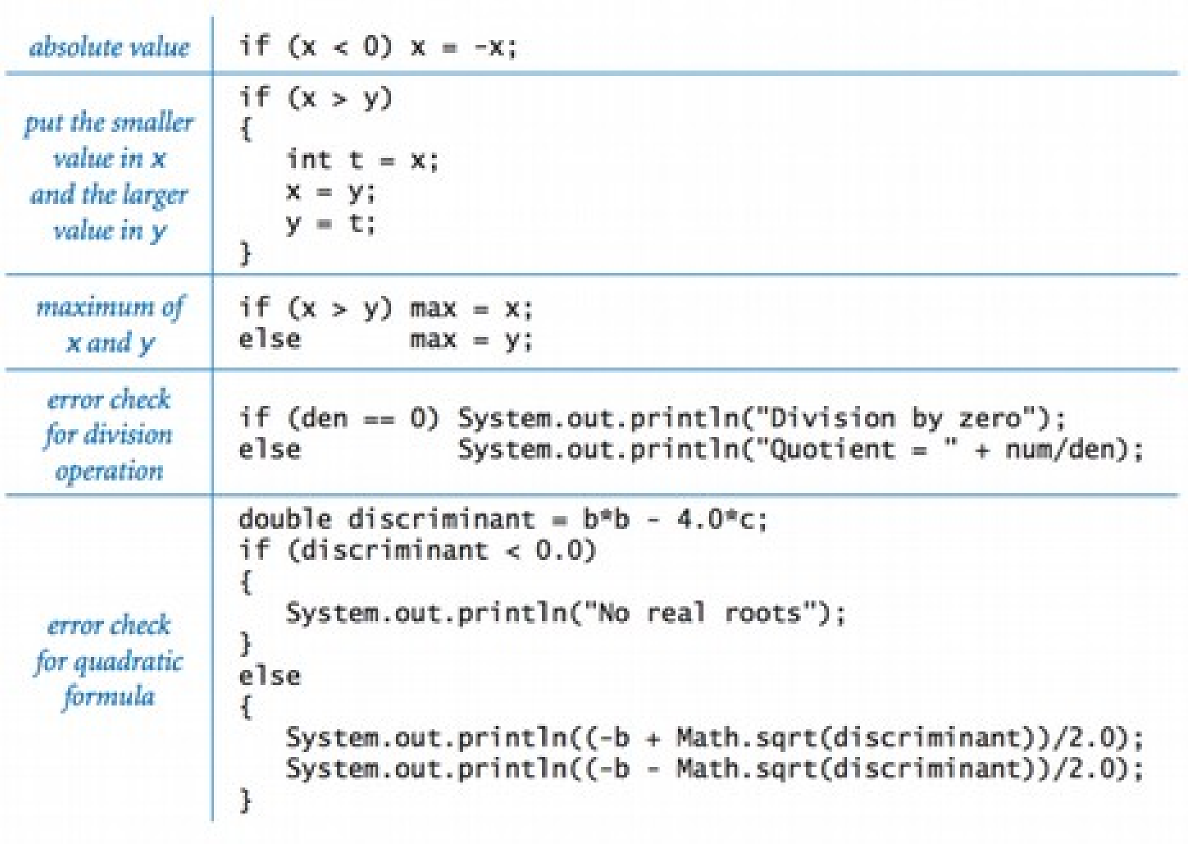


values

typical literals

operations

operators



absolute value	`if (x < 0) x = -x;`

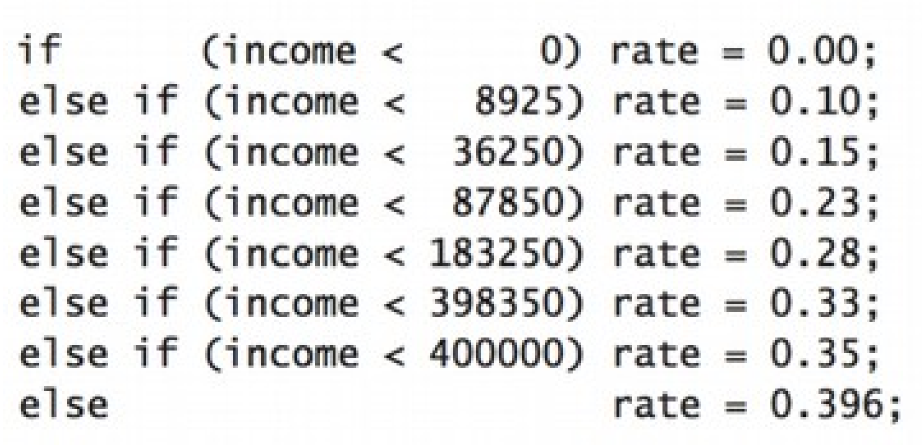
put the smaller value in x and the larger value in y	`if (x > y)`
	`{`
	`   int t = x;`
	`   x = y;`
	`   y = t;`
	`}`

maximum of x and y	`if (x > y) max = x;`
	`else       max = y;`

error check for division operation	`if (den == 0) System.out.println("Division by zero");`
	`else          System.out.println("Quotient = " + num/den);`

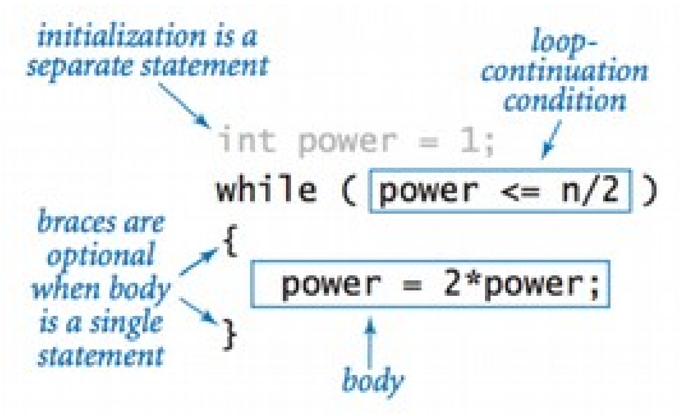
error check for quadratic formula	`double discriminant = b*b - 4.0*c;`
	`if (discriminant < 0.0)`
	`{`
	`   System.out.println("No real roots");`
	`}`
	`else`
	`{`
	`   System.out.println((-b + Math.sqrt(discriminant))/2.0);`
	`   System.out.println((-b - Math.sqrt(discriminant))/2.0);`
	`}`

Nested if-else statement.



```
if      (income <      0) rate = 0.00;
else if (income <   8925) rate = 0.10;
else if (income <  36250) rate = 0.15;
else if (income <  87850) rate = 0.23;
else if (income < 183250) rate = 0.28;
else if (income < 398350) rate = 0.33;
else if (income < 400000) rate = 0.35;
else                      rate = 0.396;
```

While loops.



```
int power = 1;
while ( power <= n/2 )
{
   power = 2*power;
}
```

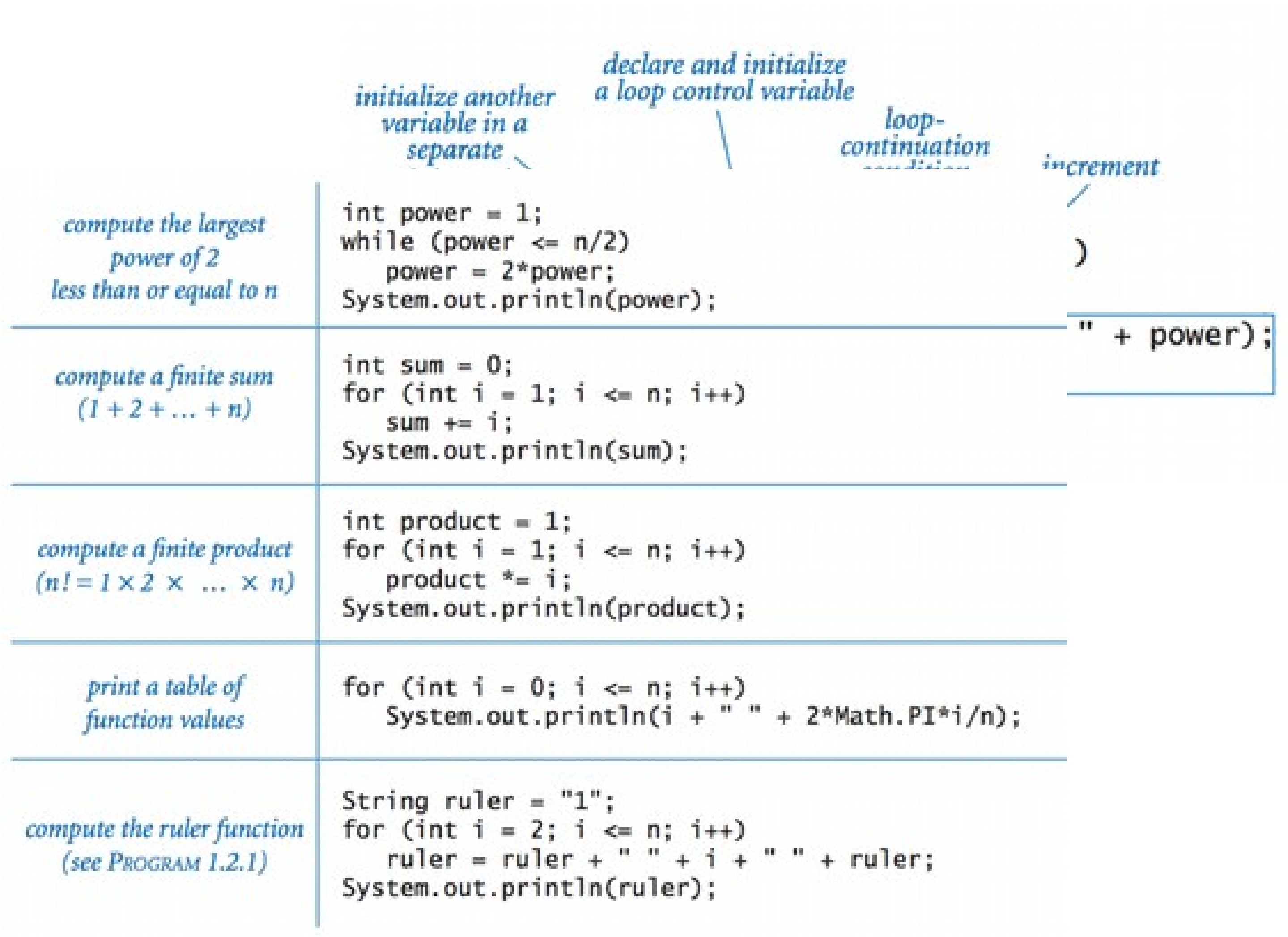
initialization is a separate statement

loop-continuation condition

braces are optional when body is a single statement

body

For loops.


initialize another variable in a separate
declare and initialize a loop control variable
loop-continuation
increment
)
" + power);

compute the largest power of 2 less than or equal to n	`int power = 1;`
	`while (power <= n/2)`
	`   power = 2*power;`
	`System.out.println(power);`

compute a finite sum (1 + 2 + … + n)	`int sum = 0;`
	`for (int i = 1; i <= n; i++)`
	`   sum += i;`
	`System.out.println(sum);`

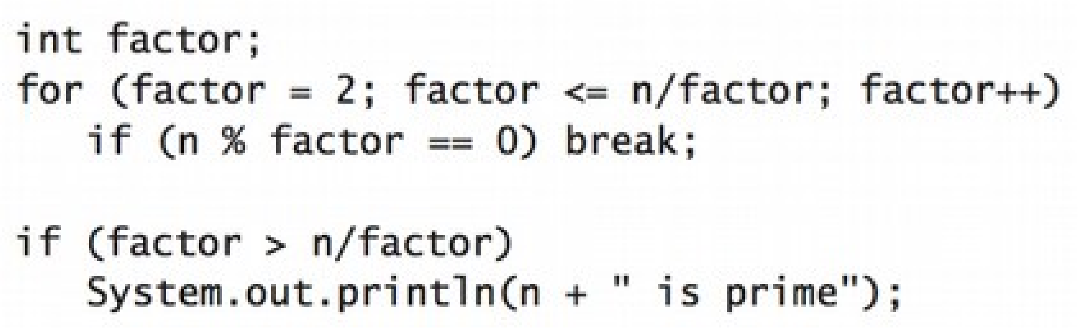
compute a finite product (n! = 1 × 2 × … × n)	`int product = 1;`
	`for (int i = 1; i <= n; i++)`
	`   product *= i;`
	`System.out.println(product);`

print a table of function values	`for (int i = 0; i <= n; i++)`
	`   System.out.println(i + " " + 2*Math.PI*i/n);`

compute the ruler function (see Program 1.2.1)	`String ruler = "1";`
	`for (int i = 2; i <= n; i++)`
	`   ruler = ruler + " " + i + " " + ruler;`
	`System.out.println(ruler);`

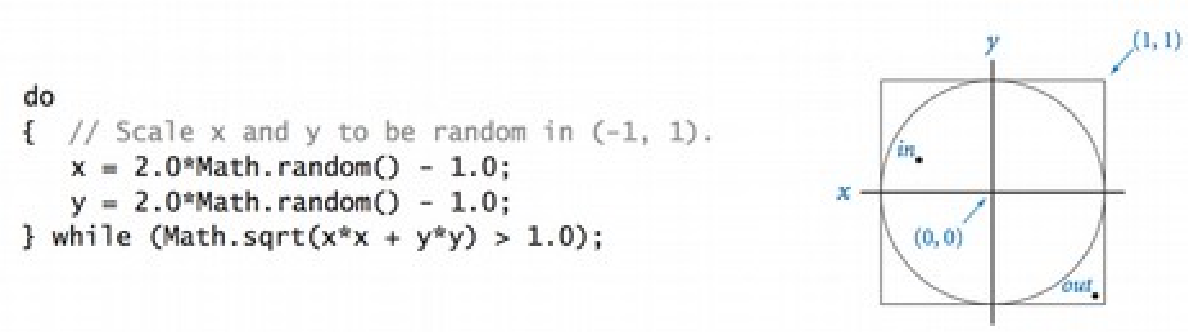
Fermare il ciclo

Break statement.



```
int factor;
for (factor = 2; factor <= n/factor; factor++)
   if (n % factor == 0) break;

if (factor > n/factor)
   System.out.println(n + " is prime");
```

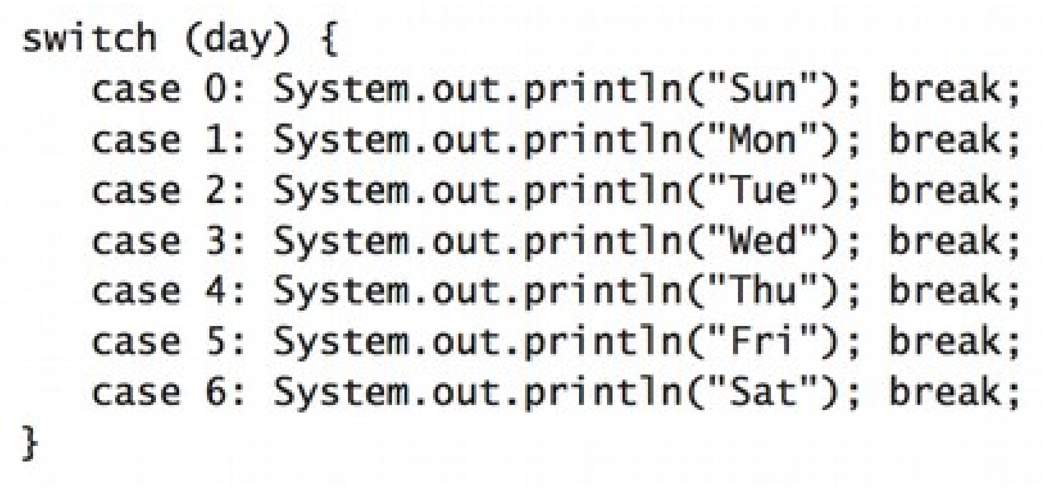
Do-while loop.



```
do
{  // Scale x and y to be random in (-1, 1).
   x = 2.0*Math.random() - 1.0;
   y = 2.0*Math.random() - 1.0;
} while (Math.sqrt(x*x + y*y) > 1.0);
```

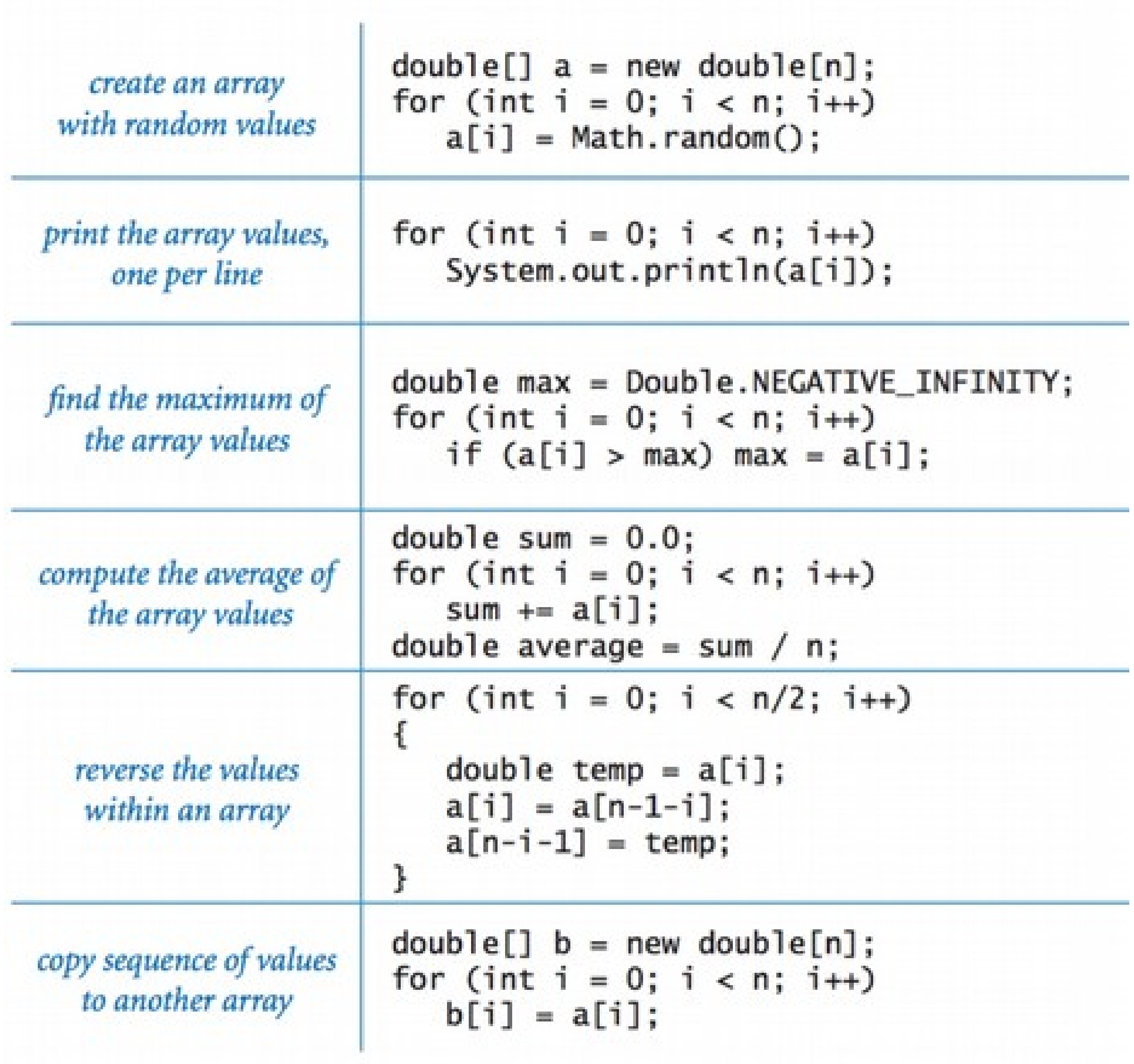
y
(1, 1)
in
x
(0, 0)
out

Switch statement.



```
switch (day) {
   case 0: System.out.println("Sun"); break;
   case 1: System.out.println("Mon"); break;
   case 2: System.out.println("Tue"); break;
   case 3: System.out.println("Wed"); break;
   case 4: System.out.println("Thu"); break;
   case 5: System.out.println("Fri"); break;
   case 6: System.out.println("Sat"); break;
}
```

Typical array-processing code.



create an array with random values	`double[] a = new double[n];`
	`for (int i = 0; i < n; i++)`
	`   a[i] = Math.random();`

print the array values, one per line	`for (int i = 0; i < n; i++)`
	`   System.out.println(a[i]);`

find the maximum of the array values	`double max = Double.NEGATIVE_INFINITY;`
	`for (int i = 0; i < n; i++)`
	`   if (a[i] > max) max = a[i];`

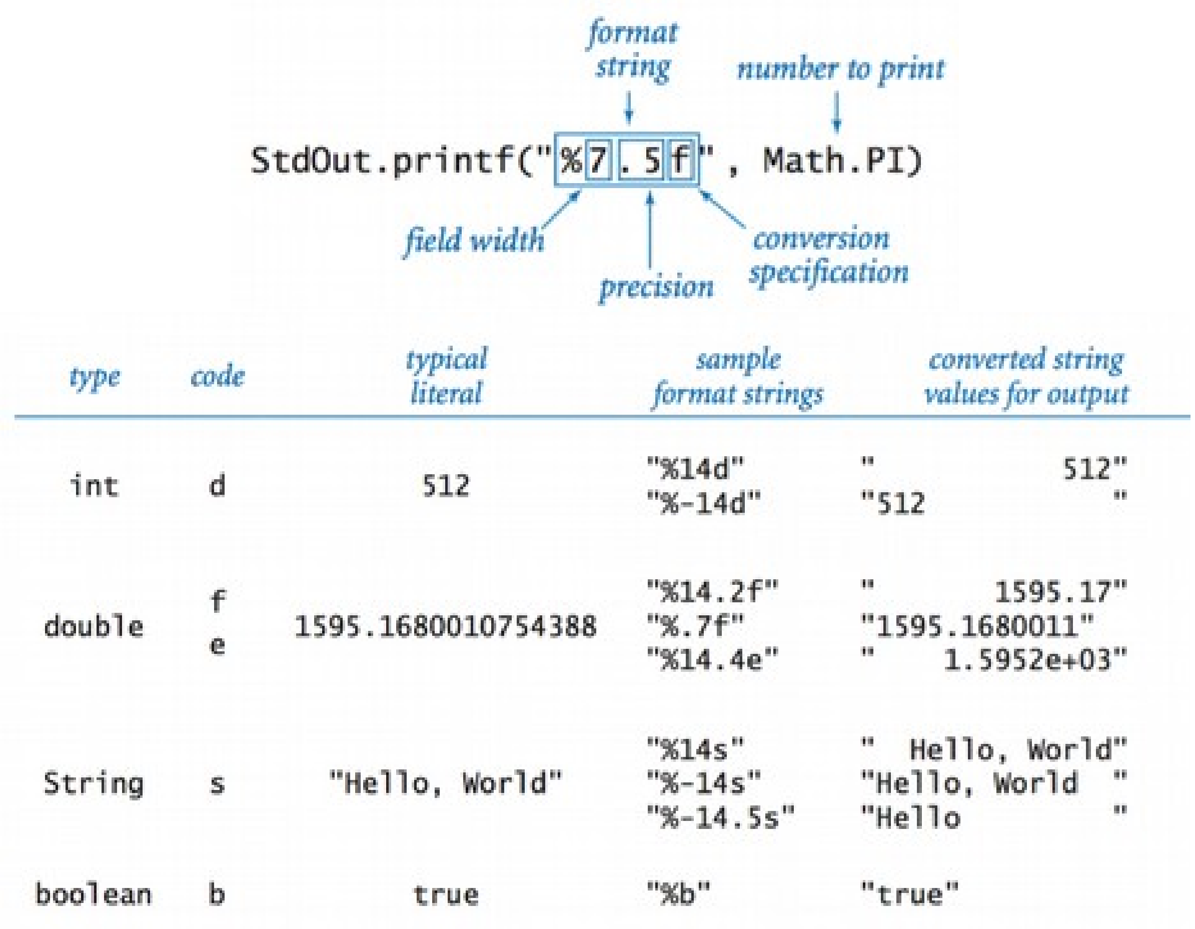
compute the average of the array values	`double sum = 0.0;`
	`for (int i = 0; i < n; i++)`
	`   sum += a[i];`
	`double average = sum / n;`

reverse the values within an array	`for (int i = 0; i < n/2; i++)`
	`{`
	`   double temp = a[i];`
	`   a[i] = a[n-1-i];`
	`   a[n-i-1] = temp;`
	`}`

copy sequence of values to another array	`double[] b = new double[n];`
	`for (int i = 0; i < n; i++)`
	`   b[i] = a[i];`

La classe String

La documentazione della libreria standard di Java



```
StdOut.printf("%7.5f", Math.PI)
```

type	code	typical literal	sample format strings	converted string values for output
int	d	512	"%14d" "%-14d"	" 512" "512 "
double	f e	1595.1680010754388	"%14.2f" "%.7f" "%14.4e"	" 1595.17" "1595.1680011" " 1.5952e+03"
String	s	"Hello, World"	"%14s" "%-14s" "%-14.5s"	" Hello, World" "Hello, World " "Hello "
boolean	b	true	"%b"	"true"

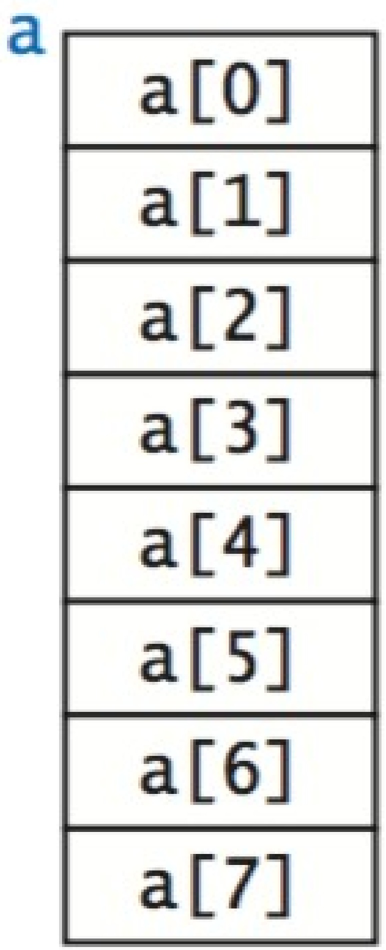
Gli array in Java

Dichiarazione

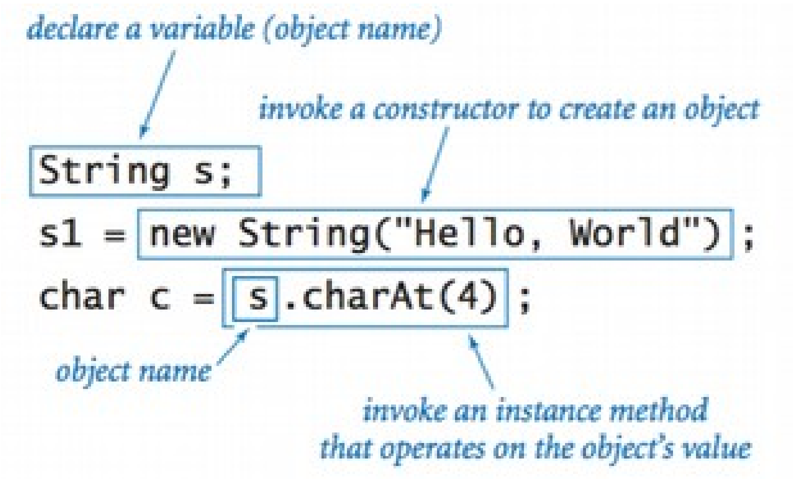
Creazione

Inizializzazione

```
String personOne = “Mauro";
String personTwo = “Paolo";


//inizializzazione standard
String[] array = new String[2];
//la dimensione non cambia più: length è una proprietà

//inizializzazione breve
String[] array = {personOne, personTwo};

for (String person : array) {
    System.out.print(“person:” + person);
}

String mauro = array[0];
```



Usare gli oggetti.



```
String s;
s1 = new String("Hello, World");
char c = s.charAt(4);
```

declare a variable (object name)
invoke a constructor to create an object
object name
invoke an instance method
that operates on the object's value

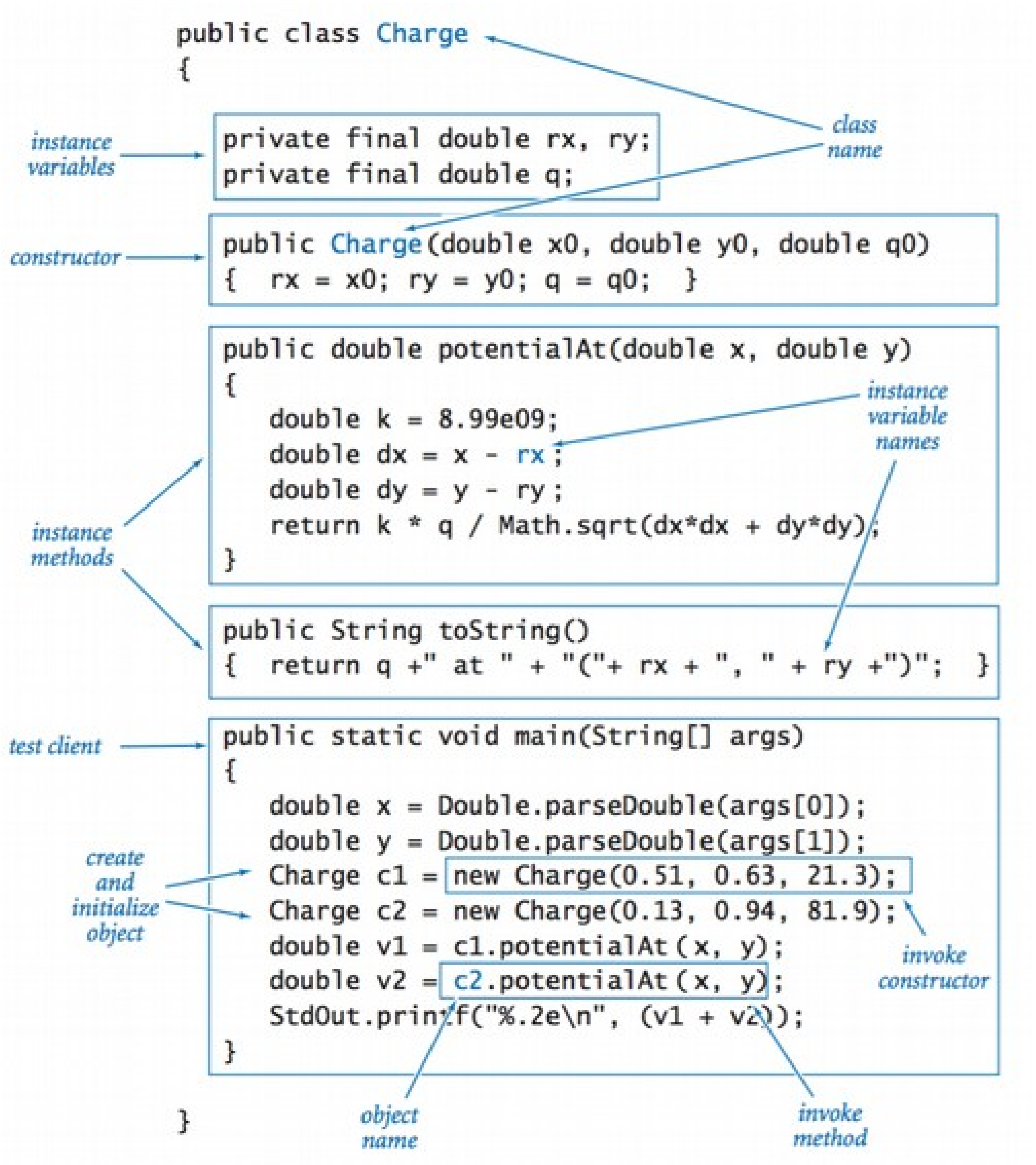
Istanziare gli oggetti.

Variabili di istanza

Costruttori

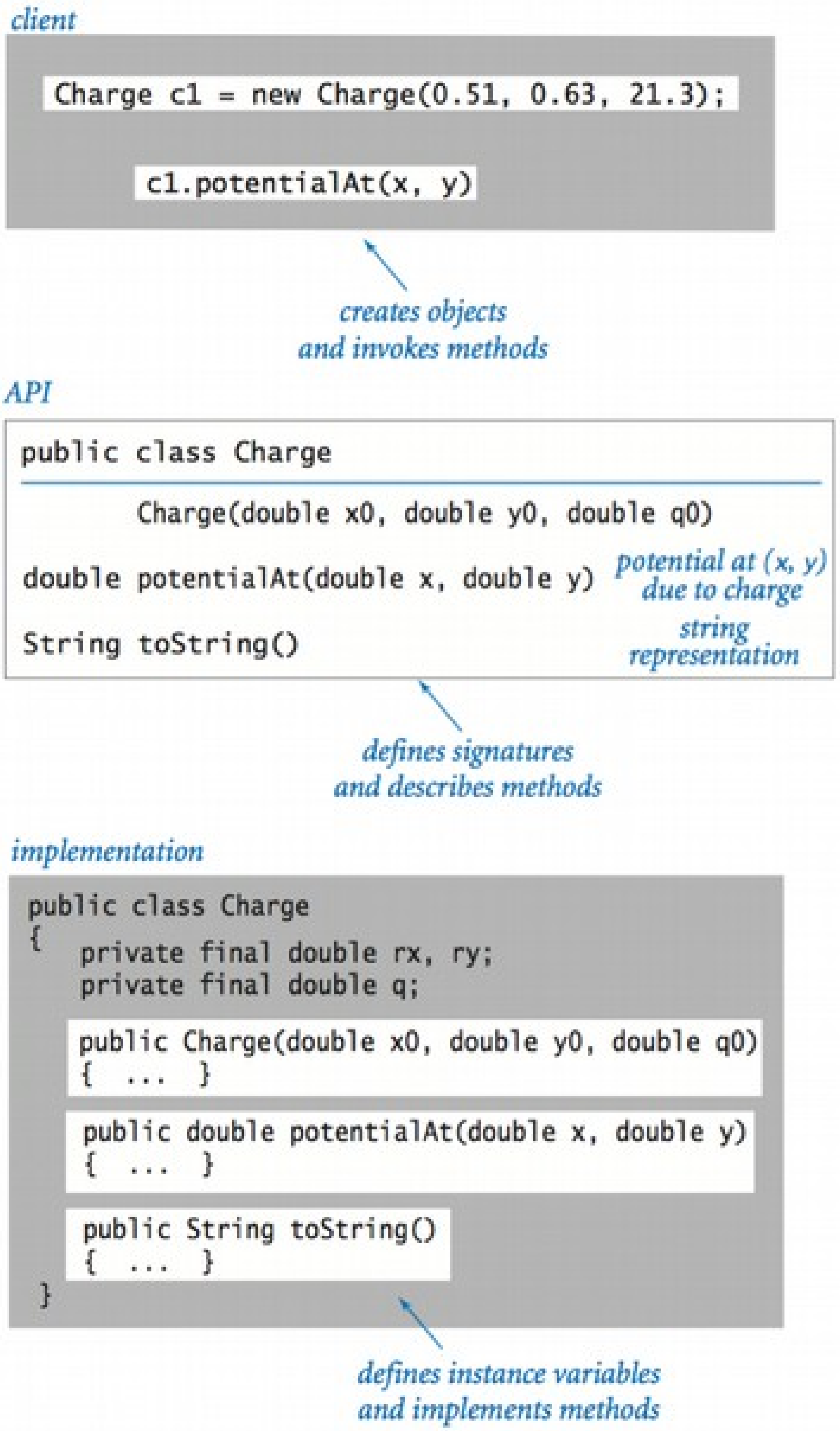
metodi

Classi



```
public class Charge
{
   private final double rx, ry;
   private final double q;

   public Charge(double x0, double y0, double q0)
   {  rx = x0; ry = y0; q = q0;  }

   public double potentialAt(double x, double y)
   {
      double k = 8.99e09;
      double dx = x - rx;
      double dy = y - ry;
      return k * q / Math.sqrt(dx*dx + dy*dy);
   }

   public String toString()
   {  return q +" at " + "("+ rx + ", " + ry +")";  }

   public static void main(String[] args)
   {
      double x = Double.parseDouble(args[0]);
      double y = Double.parseDouble(args[1]);
      Charge c1 = new Charge(0.51, 0.63, 21.3);
      Charge c2 = new Charge(0.13, 0.94, 81.9);
      double v1 = c1.potentialAt(x, y);
      double v2 = c2.potentialAt(x, y);
      StdOut.printf("%.2e\n", (v1 + v2));
   }

}
```

Object-oriented libraries.

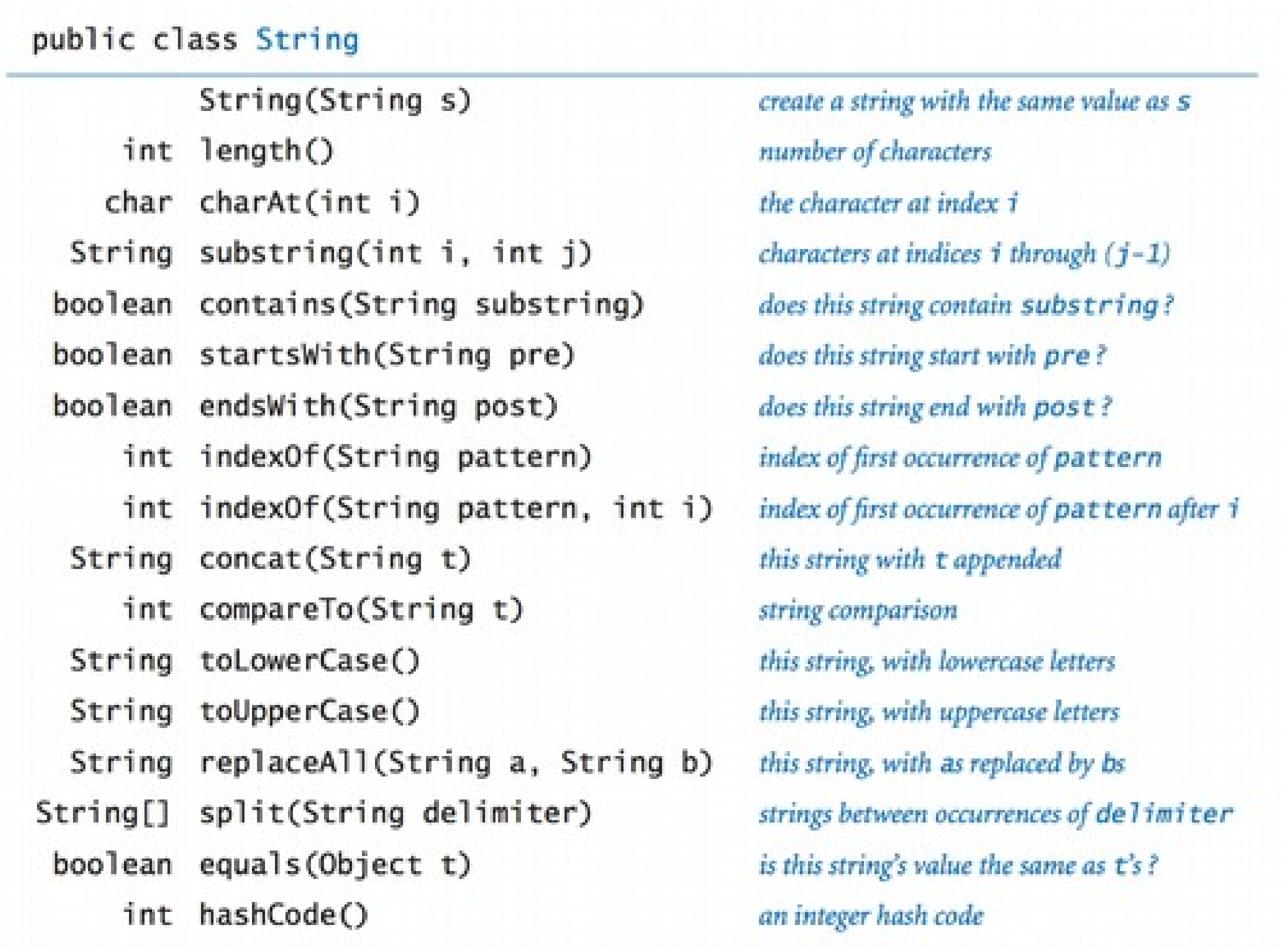


```
Charge c1 = new Charge(0.51, 0.63, 21.3);

c1.potentialAt(x, y)
```

```
public class Charge
----------------------------------------------------
        Charge(double x0, double y0, double q0)

double potentialAt(double x, double y)   potential at (x, y) due to charge

String toString()                        string representation
```

```
public class Charge
{
   private final double rx, ry;
   private final double q;

   public Charge(double x0, double y0, double q0)
   {  ...  }

   public double potentialAt(double x, double y)
   {  ...  }

   public String toString()
   {  ...  }
}
```

Java's String data type.



public class String

	String(String s)	*create a string with the same value as* s
int	length()	*number of characters*
char	charAt(int i)	*the character at index* i
String	substring(int i, int j)	*characters at indices* i *through* (j-1)
boolean	contains(String substring)	*does this string contain* substring*?*
boolean	startsWith(String pre)	*does this string start with* pre*?*
boolean	endsWith(String post)	*does this string end with* post*?*
int	indexOf(String pattern)	*index of first occurrence of* pattern
int	indexOf(String pattern, int i)	*index of first occurrence of* pattern *after* i
String	concat(String t)	*this string with* t *appended*
int	compareTo(String t)	*string comparison*
String	toLowerCase()	*this string, with lowercase letters*
String	toUpperCase()	*this string, with uppercase letters*
String	replaceAll(String a, String b)	*this string, with* a*s replaced by* b*s*
String[]	split(String delimiter)	*strings between occurrences of* delimiter
boolean	equals(Object t)	*is this string's value the same as* t*'s?*
int	hashCode()	*an integer hash code*

Programmazione ad oggetti utilizzando Java: polimorfismo

Overload

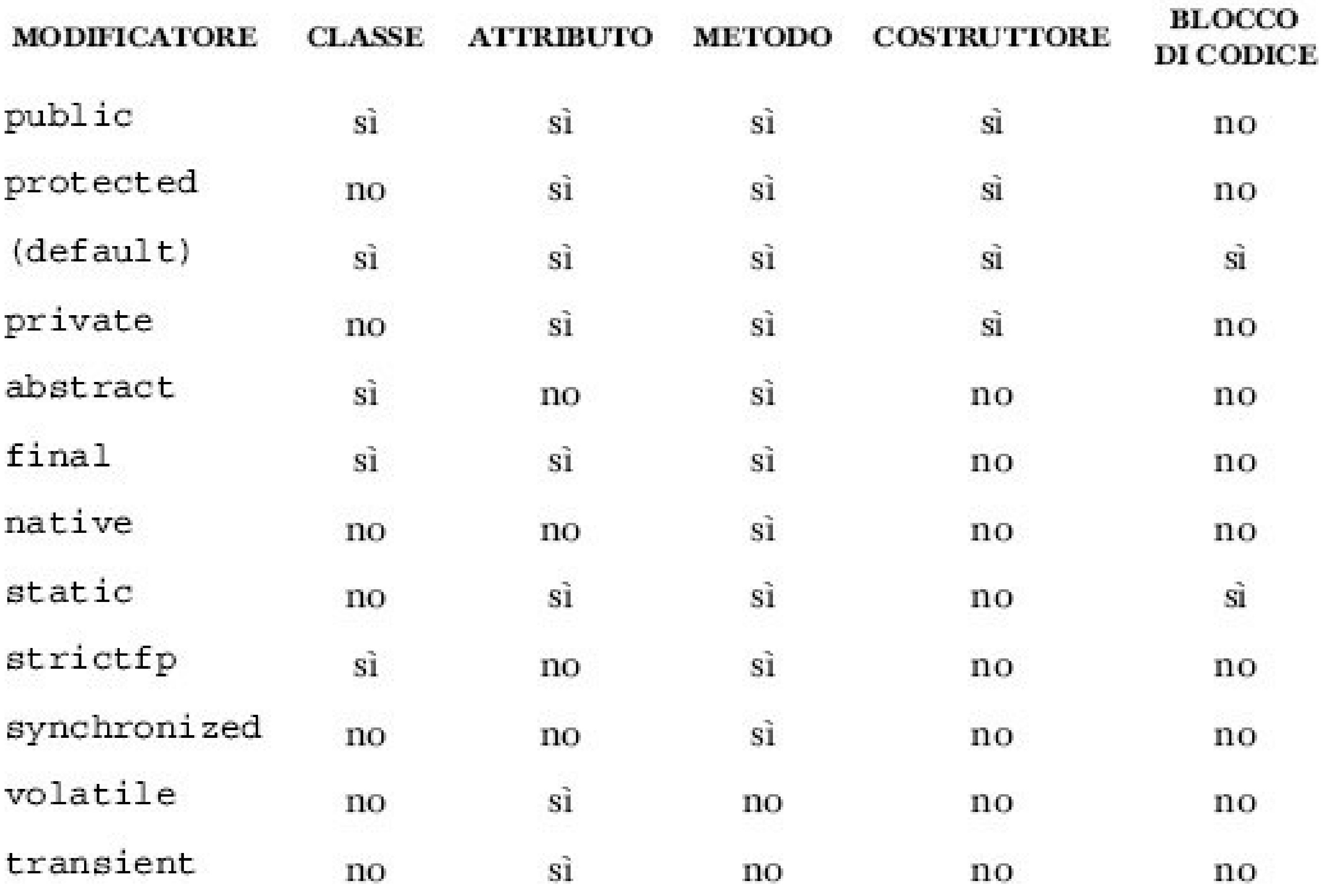
L’overload che consente di definire in una stessa classe più metodi aventi lo stesso nome, ma che differiscano nella firma, cioè nella sequenza dei tipi dei parametri formali.

E’ compito del compilatore determinare quale dei metodi “overloadati” dovrà essere

invocato, in base al numero e al tipo dei parametri attuali.

```
public class OperazioniSuNumeri {
    public int somma(int x, int y) {
        return x + y;
    }
    public float somma(float x, float y) {
        return x + y;
    }
}
```

Modificatori, package e interfacce



MODIFICATORE	CLASSE	ATTRIBUTO	METODO	COSTRUTTORE	BLOCCO DI CODICE
`public`	sì	sì	sì	sì	no
`protected`	no	sì	sì	sì	no
`(default)`	sì	sì	sì	sì	sì
`private`	no	sì	sì	sì	no
`abstract`	sì	no	sì	no	no
`final`	sì	sì	sì	no	no
`native`	no	no	sì	no	no
`static`	no	sì	sì	no	sì
`strictfp`	sì	no	sì	no	no
`synchronized`	no	no	sì	no	no
`volatile`	no	sì	no	no	no
`transient`	no	sì	no	no	no

- **Il modificatore final**
    - una variabile dichiarata **final** diventa una **costante**;
    - un metodo dichiarato **final** non può essere riscritto in una sottoclasse (non è possibile applicare l’**override**);
    - una classe dichiarata **final** non può essere **estesa**.
- **Il modificatore static**
    - **“condiviso da tutte le istanze della classe”, oppure “della classe”.**
    - **Metodi statici**
        - p.es metodo sqrt() della classe Math
    - Variabili statiche (di classe)
        - Una variabile **statica**, essendo condivisa da tutte le istanze della classe, assumerà <u>lo stesso valore per ogni oggetto di una classe</u>. se un’istanza modifica la variabile statica, essa risulterà modificata anche relativamente all’altra istanza. P. es. Il record di un gioco, o un contatore globale.