JAVA – Numeri e stringhe

Metodi Avanzati di Programmazione

Laurea Triennale in Informatica

Università degli Studi di Bari Aldo Moro

Docente: Pierpaolo Basile

Numbers

- JAVA mette a disposizione delle classi che rappresentano i tipi primitivi numerici
  - Byte, Short, Long, Integer, Float, Double
  - Ereditano tutte dalla classe Number
- Il compilatore converte automaticamente tra tipi primitivi e classi (boxing/unboxing)
- BigInteger/BigDecimal per calcolo ad alta precisione
- AtomicInteger/AtomicLong in applicazioni concorrenti

```
Integer ic = 3;
int a = ic + 2;
```

Numbers

Tre motivi per utilizzare le classi Number

1. Quanto l'argomento di un metodo deve essere un oggetto e non un tipo primitivo
2. Utilizzare le costanti static definite nella classi (tipo MAX_VALUE, MIN_VALUE)
3. Per conversioni tra tipo
   a) convertire String in numeri
   b) conversioni tra differenti tipi numerici

Metodi implementati da Number

- Converte il valore in un tipo primitivo
  - byte byteValue()
  - short shortValue()
  - int intValue()
  - long longValue()
  - float floatValue()
  - double doubleValue()
- Confronta Number con l'argomento
  - int compareTo(Byte anotherByte)
  - int compareTo(Double anotherDouble)
  - int compareTo(Float anotherFloat)
  - int compareTo(Integer anotherInteger)
  - int compareTo(Long anotherLong)
  - int compareTo(Short anotherShort)
- boolean equals(Object obj)

>0 più grande
=0 uguali
<0 più piccolo

Metodi per la conversione

- static Integer decode(String s): converte una stringa in Integer
- static int parseInt(String s): converte una stringa in int
- static int parseInt(String s, int radix): converte una stringa in int, (radix=sistema di numerazione 10, 2, 8, 16)
- String toString(): restituisce la stringa che rappresenta questo numero
- static String toString(int i): restituisce la stringa che rappresenta il numero i
- static Integer valueOf(int i): restituisce l'Integer che rappresenta il valore primitivo i
- static Integer valueOf(String s): simile a parseInt ma restituisce un Integer
- static Integer valueOf(String s, int radix): simile a parseInt

Stampa dei numeri

- Poiché ogni numero può essere convertito in String si possono utilizzare per la stampa dei numeri direttamente
  - System.out.print(String s)
  - System.out.println(String s)
- System.out è un oggetto PrintStream, possiamo usare i metodi *printf* e *format* (sono equivalenti)

format (printf)…

format(String format, Object... args)

Es.

```
System.out.format("The value of  the float
variable is %f, while the value of the integer
variable is %d, and the string is %s",
floatVar, intVar, stringVar);
```

- *format* specifica come gli oggetti *args* devono essere stampati

…format (printf)

Esempio

```
int i = 461012;
System.out.format("The value of i is: %d%n", i);
```

format converter e flag

Converter	Flag	Explanation
d		A decimal integer
f		A float
n		A new line character appropriate to the platform running the application. You should always use %n, rather than \n
tB		A date & time conversion—locale-specific full name of month
td, te		A date & time conversion—2-digit day of month. td has leading zeroes as needed, te does not
ty, tY		A date & time conversion—ty = 2-digit year, tY = 4-digit year
tl		A date & time conversion—hour in 12-hour clock.
tM		A date & time conversion—minutes in 2 digits, with leading zeroes as necessary.
tp		A date & time conversion—locale-specific am/pm (lower case).
tm		A date & time conversion—months in 2 digits, with leading zeroes as necessary.
tD		A date & time conversion—date as %tm%td%ty
	08	Eight characters in width, with leading zeroes as necessary.
	+	Includes sign, whether positive or negative.
	,	Includes locale-specific grouping characters.
	-	Left-justified..
	.3	Three places after decimal point.
	10.3	Ten characters in width, right justified, with three places after decimal point.

format (esempio)

```
import java.util.Locale;

public class TestFormat {
    public static void main(String[] args) {
      long n = 461012;
      System.out.format("%d%n", n);      //  -->  "461012"
      System.out.format("%08d%n", n);    //  -->  "00461012"
      System.out.format("%+8d%n", n);    //  -->  " +461012"
      System.out.format("%,8d%n", n);    // -->  " 461,012"
      System.out.format("%+,8d%n%n", n); //  -->  "+461,012"
      double pi = Math.PI;
      System.out.format("%f%n", pi);       // -->  "3.141593"
      System.out.format("%.3f%n", pi);     // -->  "3.142"
      System.out.format("%10.3f%n", pi);   // -->  "     3.142"
      System.out.format("%-10.3f%n", pi);  // -->  "3.142"
      System.out.format(Locale.FRANCE, "%-10.4f%n%n", pi); // -->  "3,1416"
   }
}
```

Math

- La classe *Math* fornisce strumenti più avanzati per le operazioni matematiche
- Costanti matematiche
  - Math.E, Math.PI
- Operazioni di base
  - abs (valore assoluto), round (arrotondamento), min, max (tra due numeri)
  - *double ceil(double d)*: l'intero più piccolo che è più grande o uguale a *d*
  - *double floor(double d)*: l'intero più grande che è più piccolo o uguale a *d*
  - *double rint(double d)*: l'intero più vicino a *d*

Math (esponenziali e logaritmi)

- double exp(double d): e^{d}
- double log(double d): ln(d)
- double pow(double base, double exponent): $base^{exponent}$
- double sqrt(double d): radice quadrata di d

Math (funzioni trigonometriche)

Passare come argomento un double che rappresenta l'angolo espresso in radianti

- *sin, cos, tan*
- *asin, acos, atan* (arcoseno, …)
- *double atan2(double y, double x)*: converte coordinate rettangolari (x, y) in coordinate polari (r, theta) e restituisce theta
- *double toDegrees(double d), double toRadians(double d)*: converte l'argomento in gradi o radianti

Numeri random

- double Math.random(): restituisce un numero random tra [0; 1)
  - si può moltiplicare il risultato per un intero per scalare il range *Math.random()*10*
- Per creare una serie di numeri random utilizzare la classe *java.util.Random*

STRING

La classe Character

- In genere i caratteri vanno gestiti con il tipo primitivo char (char a = ‘a’)
- Metodi statici della classe Character

boolean isLetter(char ch) boolean isDigit(char ch)	determina se il carattere è una lettera o una cifra
boolean isWhitespace(char ch)	determina se il carattere è un white space (spazio, accapo, tabulazione)
boolean isUpperCase(char ch) boolean isLowerCase(char ch)	determina se il caratter è in maiuscolo (UpperCase) o minuscolo (LowerCase)
char toUpperCase(char ch) char toLowerCase(char ch)	determina la maiuscola e la minuscola del corrispettivo carattere
toString(char ch)	restituisce una stringa che rappresenta un singolo carattere

String

- La classe `String` rappresenta una stringa di caratteri
- Un'istanza di `String` è **immutabile**: il suo valore non può essere modificato dopo la creazione
  - utilizzare `StringBuilder` o `StringBuffer`
- Ogni literal "sdjfkl skf dskj" è rappresentato in JAVA da un'istanza di `String`

String

String può essere visto come un array di char

```
char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.' };
String helloString = new String(helloArray);
System.out.println(helloString);
```

Il metodo charAt(int i) di String restituisce il carattere alla i-esima posizione

String (metodi)

- `length()`
  - restituisce la lunghezza della stringa in caratteri
- `contains(CharSequence s)`
  - restituisce true se contiene la sottostringa s
- `indexOf(String s)`
  - restituisce l'indice dal quale inizia la sottostringa s, -1 se non esiste la sottostringa
- `indexOf(String s, int i)`
  - inizia la ricerca della sottostringa a partire dall'*i*-esimo carattere

String (metodi)

- `replace(CharSequence s1, CharSequence s2)`
  - sostituisce la sequenza s1 con s2
- `replaceAll(String regex, String r)`
  - sostituisce tutte le sequenze che corrispondono all'espressione regolare regex con r
- `matches(String regex)`
  - restituisce true se la stringa corrisponde all'espressione regolare regex
- `split(String regex)`
  - restituisce un array di String dividendo dove c'è il match con regex

String (metodi)

- `startsWith, endsWith`
  - controlla se inizia/finisce con una particolare sequenza
- `equals(Object o)`
  - operatore di uguaglianza, da usare per confrontare due stringhe **NON USARE ==**
- `int compareTo(String str), int compareToIgnoreCase(String str)`
  - confronta due stringa in modo

String (metodi)

- `substring(int b, int e)`, `substring(int b)`
  - genera una sottostringa da *b* ad *e* (escluso), genera una sottostringa a partire da *b* (incluso)
- `trim()`
  - elimina white space dall'inizio e dalla fine
- `toLowerCase()`, `toUpperCase()`
  - conversione minuscolo/maiuscolo

String (esempio)

```
public class StringDemo {
    public static void main(String[] args) {
        String palindrome = "Dot saw I was Tod";
        int len = palindrome.length();
        char[] tempCharArray = new char[len];
        char[] charArray = new char[len];
        // put original string in an
        // array of chars
        for (int i = 0; i < len; i++) {
            tempCharArray[i] =
                palindrome.charAt(i);
        }
        // reverse array of chars
        for (int j = 0; j < len; j++) {
            charArray[j] =
                tempCharArray[len - 1 - j];
        }
        String reversePalindrome =
            new String(charArray);
        System.out.println(reversePalindrome);
    }
}
```

Inversione dei caratteri in una stringa

Conversione da stringhe a numeri

```
int i = Integer.parseInt(“42”);
float f = Float.parseFloat(“3.14”);
double d = Double.parseDouble(“4.32144”);
```

```
Float fo = Float.valueOf(“3.14”);
```

Conversione da numeri a stringhe

```
int i=3;
double d=3.4;
String s3 = Integer.toString(i);
String s4 = Double.toString(d);
```

```
int i=3;
String s1 = String.valueOf(i);
```

Esercizio

1. Prendere due stringhe passate come argomento al main
2. Convertirle in double
3. Effettuare la somma dei due double e stamparla a video
4. Effettuare l'elevamento a potenza di $arg1^{arg2}$ e stamparlo a video
5. Se il primo numero è minore di 100 genera un numero random tra 0 e 100 e stampare true se il primo numero è più piccolo del numero random, altrimenti stampare false

Espressioni regolari

- Un'Espressione regolare è una parola che denota un linguaggio regolare
  - può essere utilizzate per verificare se una parola (String) corrisponde alle regole di un linguaggio
  - Es. *a+* denota tutte le stringhe composte da sole *a*
- Nella javadoc della classe `Pattern` trovate una sintesi sulle espressioni regolari
- JAVA tutorial http://docs.oracle.com/javase/tutorial/essential/regex/

Espressioni regolari (caratteri)

x	The character *x*
****	The backslash character
\0*n*	The character with octal value 0*n* (0 <=*n*<=7)
\0*nn*	The character with octal value 0*nn* (0 <=*n*<=7)
\0*mnn*	The character with octal value 0*mnn* (0<=*m*<=3, 0<=*n*<=7)
\x*hh*	The character with hexadecimal value 0x*hh*
\u*hhhh*	The character with hexadecimal value 0x*hhhh*
\t	The tab character ('\u0009')
\n	The newline (line feed) character ('\u000A')
\r	The carriage-return character ('\u000D')
\f	The form-feed character ('\u000C')
\a	The alert (bell) character ('\u0007')
\e	The escape character ('\u001B')
\c*x*	The control character corresponding to *x*

Espressioni regolari (chars classes)

[abc] a, b, or c (simple class)

[^abc] Any character except a, b, or c (negation)

[a-zA-Z] a through z or A through Z, inclusive (range)

[a-d[m-p]] a through d, or m through p: [a-dm-p] (union)

[a-z&&[def]] d, e, or f (intersection)

[a-z&&[^bc]] a through z, except for b and c: [ad-z] (subtraction)

[a-z&&[^m-p]] a through z, and not m through p: [a-lq-z](subtraction)

Espressioni regolari (default classes)

. Any character (may or may not match line terminators)

\d A digit: [0-9]

\D A non-digit: [^0-9]

\s A whitespace character: [\t\n\x0B\f\r]

\S A non-whitespace character: [^\s]

\w A word character: [a-zA-Z_0-9]

\W A non-word character: [^\w]

Espressioni regolari (boundary)

^ The beginning of a line

$ The end of a line

\b A word boundary

\B A non-word boundary

\A The beginning of the input

\G The end of the previous match

\Z The end of the input but for the final terminator, if any

\z The end of the input

Espressioni regolari (greedy operator)

X? *X*, once or not at all

*X** *X*, zero or more times

X+ *X*, one or more times

X{*n*} *X*, exactly *n* times

X{*n*,} *X*, at least *n* times

X{*n*,*m*} *X*, at least *n* but not more than *m* times

Espressioni regolari (quotation)

\ Nothing, but quotes the following character

\Q Nothing, but quotes all characters until \E

\E Nothing, but ends quoting started by \Q

Espressioni regolari (logical operators)

XY *X* followed by *Y*

X|*Y* Either *X* or *Y*

(*X*) X, as a capturing group

Esercizio

1. Definire l'espressione regolare per riconoscere un indirizzo email
2. Prendere il primo argomento passato al main
3. Se l'argomento è un indirizzo email stampa true altrimenti stampa false

StringBuilder

- E’ una classe simile a String ma a lunghezza variabile
  - posso sempre aggiungere nuovi caratteri
- Metodi
  - length(): restituisce la lunghezza della stringa presente nel StringBuilder
  - capacity(): restituisce la capacità attuale di questo StringBuilder (>=length())

StringBuilder (costruittori)

Costruttore	Descrizione
StringBuilder()	Crea uno StringBuilder vuoto
StringBuilder(CharSequence cs)	Crea uno StringBuilder a partire da una sequenza di caratteri
StringBuilder(int initCapacity)	Crea uno StringBuilder con una capacità iniziale *initCapacity*
StringBuilder(String s)	Crea uno StringBuilder a partire da una stringa *s*

StringBuilder (metodi)…

- *StringBuilder append(…)*: aggiunge alla fine dello StringBuilder l'argomento (viene convertito in String anche i tipi primitivi)
- *StringBuilder delete(int start, int end), StringBuilder deleteCharAt(int index)*: cancella una porzione o un carattere
- *StringBuilder insert(int offset, …)*: inserisce l'argomento a partire dalla posizione *offset*

…StringBuilder (metodi)

- *StringBuilder replace(int start, int end, String s), void setCharAt(int index, char c)*: sostituisce una porzione o un singolo carattere
- *StringBuilder reverse()*: inverte l'ordine dei caratteri
- *String toString()*: restituisce una stringa che contiene la sequenza dei caratteri in StringBuilder

StringBuilder (esempio)

```
public class StringBuilderDemo {
    public static void main(String[] args) {
        String palindrome = "Dot saw I was Tod";
        StringBuilder sb = new StringBuilder(palindrome);
        sb.reverse();  // reverse it
        System.out.println(sb);
    }
}
```

Inversione di una stringa utilizzando StringBuilder

THE END


ONE DOES NOT SIMPLY
MUTATE A STRING IN JAVA
memegenerator.net