Tipi di dato Fondamentali

Tipi numerici

- **`int`: interi, senza parte frazionaria**

```
1, -4, 0
```

- **`double`: numeri in virgola mobile (precisione doppia)**

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

Tipi numerici

- **Una computazione su tipi numerici può causare overflow.**

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

- **Java: 8 tipi primitivi, che includono quattro tipi interi e due in virgola mobile**

Tipi primitivi

Tipo	Range di valori	Dimensione
`int`	–2,147,483,648 . . . 2,147,483,647	4 bytes
`byte`	–128 . . . 127	1 byte
`short`	–32768 . . . 32767	2 bytes
`long`	9,223,372,036,854,775,808 . . . –9,223,372,036,854,775,807	8 bytes

Continua…

Tipi primitivi

Tipo	Descrizione dei valori	Dim
`double`	Virgola mobile, precisione doppia. Range $\pm 10^{308}$ e 15 cifre decimali	8 bytes
`float`	Virgola mobile, precisione singola. Range $\pm 10^{38}$ e 7 cifre decimali	4 bytes
`char`	Caratteri nel sistema Unicode	2 bytes
`boolean`	`false` e `true`	1 byte

Float e double

- **Attenzione agli errori derivanti dagli arrotondamenti**

```
double f = 4.35;
System.out.println(100 * f); // stampa 434.9999999999994
```

- **Non è ammesso assegnare un double (o float) ad un intero … non è C !**

```
double d = 13.75;
int i = d; // Errore
```

Continua…

Float e double

- **`Cast`: per convertire un double ad un int è quindi necessario un cast**

```
int i = (int) d; // OK
```

- **Cast equivale a troncamento.**
- **`Math.round` arrotondamento**
  - `static long round(double a)`
  - `static int round(float a)`

Domande

- In quale caso il cast `(long) x` dà un risultato diverso da `Math.round(x)`?
- Quale istruzione utilizzereste per arrotondare `x:double` al valore `int` più vicino?

Risposte

- Quando la parte frazionaria di `x` è ≥ 0.5
- `(int) Math.round(x)` se assumiano che `x` è minore di $2 \cdot 10^9$

Stringhe

- Le stringhe sono oggetti
- Stringhe costanti

```
"Hello, World!"
```

- Variabili di tipo stringa

```
String message = "Hello, World!";
```

- Lunghezza di una stringa

```
int n = message.length();
```

- La stringa vuota:

```
""
```

Concatenazione

- **Utilizziamo l'operatore +:**

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

- **Se uno degli argomenti dell'operatore è una stringa, l'altro argomento viene automaticamente convertito**

```
String a = "Agente";
int n = 7;
String bond = a + "00" + n; // bond =  "Agente007"
```

Concatenazione in output

- **Utile per ridurre il numero di istruzioni `System.out.print`:**

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

è equivalente a

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

Conversioni

- **Da stringhe a numeri:**

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

- **Da numeri a stringhe**

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

Confronti

- **s1 == s2**
  - true se e solo se s1 e s2 sono riferimenti alla stessa stringa (come per tutti i tipi reference)
- **s1.equals(s2)**
  - true se e solo se s1 e s2 sono stringhe uguali (case sensitive)
- **s1.equalsIgnoreCase(s2)**
  - come sopra ma case insensitive
- **s1.compareTo(s2)**
  - < 0 se s1 precede s2 in ordine lessicografico
  - = 0 se s1 e s2 sono uguali
  - > 0 se s1 segue s2 in ordine lessicografico

Continua…

Confronti

- **Attenzione**

```
“str” == “str”
```

true

```
new String(“str”) == new String(“str”)
```

false

```
“str” == new String(“str”)
```

false

- **Le chiamate a new creano sempre oggetti diversi, l’uso di costanti invece è ottimizzato**

Stringhe e array di caratteri

- **Le stringhe non sono array di caratteri**
- **Esistono conversioni**
  - String ⟶ char[]

```
String s = “str”
Char data[] = s.toCharArray()
```

  - char[] ⟶ String

```
char data[] = {‘s’, ‘t’, ‘r’ }
String s = new String(data)
```

Sottostringhe

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

- **`s.substring(i, j)`** restituisce la sottostringa di **`s`** dalla posizione **`i`** alla **`j-1`** (estremi inclusi)
- **`s.substring(i)`** restituisce la sottostringa di **`s`** da **`i`** (incluso) al termine di **`s`**
- **`s.indexOf(s1)`** restituisce l'indice della prima occorrenza di **`s1`** in **`s`**; **`-1`** se non ci sono occorrenze
- **`s.indexOf(s1,i)`** l'indice della prima occorrenza di **`s1`** in **`s`** da **`i`** (incluso); **`-1`** se non ci sono occorrenze

Continua…

Lettura da input

- **`System.in` (lo standard input) offre supporto minimo per la lettura**
  - read(): lettura di una byte
- **La classe `Scanner` di Java 5.0 permette di leggere da input in modo più strutturato**

```
Scanner in = new Scanner(System.in);
System.out.print("Enter quantity: ");
int quantity = in.nextInt();
```

Continua…

Lettura da input

Metodi utili della classe `Scanner`

- **`nextDouble()` legge un double**
- **`nextLine()` legge un linea**
  - la sequenza fino al primo newline
- **`nextWord()` legge una parola**
  - la sequenza fino al primo spazio/tab/newline

File stringTester.java

```
import java.util.Scanner;
class stringTester {
    /**
      Legge da input linee di testo ed estrae i campi delimitati da :
     */
    public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      String record, field;
      char delim = ':';         // il delimitatore
      int record_count = 1;
      while (in.hasNextLine()) {
          System.out.println("Record " + record_count++);
          record = in.nextLine();
          int begin, end, i; begin = 0;
          for (i=0; (end = record.indexOf(delim,begin)) >= 0; i++) {
            // end = indice della prossima occorrenza del delimiter
            field = record.substring(begin,end);
            begin = end + 1; // salta il delimitatore
            System.out.println("\tField" + i + ": " + field);
          }
          field = record.substring(begin);  // l'ultimo campo
          System.out.println("\tField" + i + ": " + field);
      }
    }
}
```

File `stringTester.java`

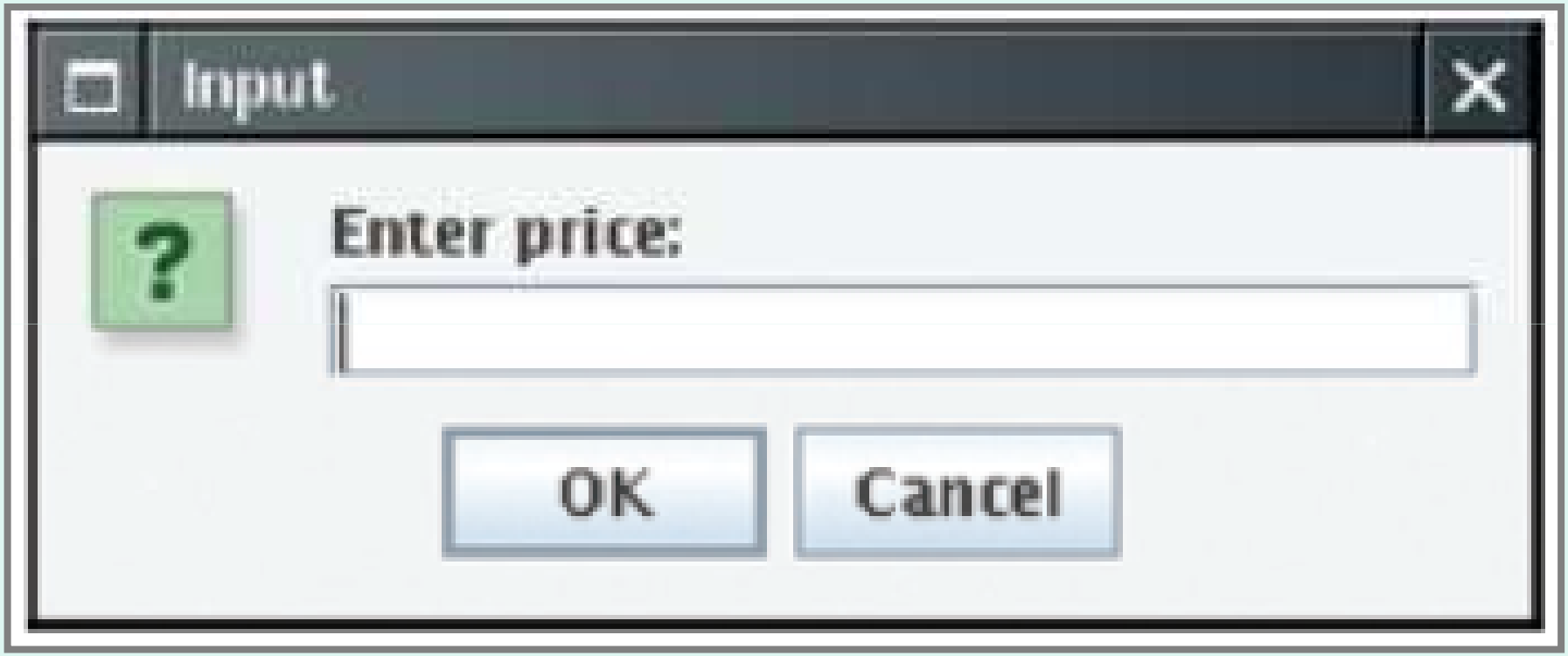
Input

```
Gianni:Rossi:via Verdi 123:Milano:Italia
Michele:Bugliesi:via Torino 155:Mestre:Italia:30173
```

Output

```
Record 1
 Field 1: Gianni
 Field 2: Rossi
 Field 3: via Verdi 123
 Field 4: Milano
 Field 5: Italia
Record 2
 ...
```

Input da un Dialog Box



Input da un Dialog Box

- ```
  String input = JOptionPane.showInputDialog(prompt)
  ```

- **Converti le stringhe in numeri se necessario:**

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

- **La conversione lancia una eccezione se l'utente non fornisce un numero**
- **Aggiungete `System.exit(0)` al metodo `main` di qualunque programma che usi `JOptionPane`**
```

Domande

15. Perché non possiamo leggere input direttamente da `System.in`?
16. Supponiamo `s` sia un oggetto di tipo `Scanner` che estrae dati da `System.in`, e consideriamo la chiamata

    ```
    String name = s.next();
    ```

    Quale è il valore della variabile `name` se l'utente dà in input `Hasan M. Jamil`?

Risposte

15. Possiamo, ma il tipo di `System.in` permette solo letture molto primitive (un byte alla volta)..
16. Il valore è `“Hasan".`