Linguaggi di Programmazione

a.a. 14/15

docente: Gabriele Fici

gabriele.fici@unipa.it

5 - Stringhe

5 - Stringhe

- In Java c'è la classe di sistema `String`, che permette di creare stringhe di testo
- Le stringhe sono dunque oggetti, istanze della classe `String`
- Tuttavia, per creare una stringa *non è necessario* usare l'operatore `new`; si può usare un letterale stringa, cioè una stringa racchiusa tra virgolette

Esempio:

```
String a = "Pippo";
// equivale a:
// String a = new String("Pippo");
```

5 - Stringhe

- Le stringhe in Java usano caratteri in formato *Unicode* (estensione di ASCII)
- Ad esempio, per stampare "è" useremo il codice Unicode `\u00E9`
- Trovate i codici Unicode di tutti i simboli su www.unicode.org/charts/

Esempio:

```
String nome = "Jos\u00E9";   //stampa "Josè"
```

5 - Stringhe

- L'operatore + permette di concatenare due stringhe, oppure una stringa e un numero (che viene convertito in stringa)

Esempio:

```
String a = "Pippo";
String nome = "Nome: " + a;
```

Esempio:

```
int j = 7;
String indirizzo = "Interno " + j;
                        // vale "Interno 7"
```

5 - Stringhe

- Le Stringhe sono oggetti immutabili, una volta creati non possono essere modificati
- La JVM gestisce un "pool di stringhe" per risparmiare memoria; stringhe uguali verranno viste come lo stesso oggetto, a patto che siano state inizializzate con letterali
- Tuttavia le stringhe inizializzate col `new` non entrano a far parte del pool

```
String s = "pippo";
String t = "pippo";
String u = new String("pippo");

System.out.println(s==t);   //stampa true
System.out.println(t==u);   //stampa false
```

5 - Stringhe

- Un altro modo per forzare due stringhe uguali a essere oggetti diversi è assegnare il letterale in due passi

Esempio:

```
String s = "pippo";
String v = "pip"; v += "po";

System.out.println(s==v);  //stampa false
```

5 - Stringhe

- Per confrontare il contenuto di due stringhe si usa il metodo `equals`, che ritorna un boolean

Esempio:

```
String s = "ab", t = "a" + "b";

boolean c = s.equals(t);    // c vale true
```

- Per confrontare il contenuto di due stringhe senza tenere conto di maiuscole e minuscole si usa il metodo `equalsIgnoreCase`

5 - Stringhe

- Per confrontare l'ordine lessicografico di due stringhe si usa il metodo metodo `compareTo`, che ritorna la differenza tra le due prime lettere diverse, o zero se le stringhe sono uguali
- Se il valore di ritorno è negativo, la prima stringa è più piccola, se è positivo la prima stringa è più grande

Esempio:

```
String s = "pippo", t = "pippq";

System.out.println(s.compareTo(t));
// stampa -2
```

- Analogamente esiste il metodo `compareToIgnoreCase`

5 - Stringhe

- Per leggere una stringa in input, occorre usare uno *Scanner*, che si occuperà di prendere i dati da console
- La classe `Scanner` è nel package `java.util`, che dovremo importare all'inizio del file

Esempio:

```
import java.util.*; //importa il segmento java.util
...

Scanner pippo = new Scanner(System.in);
String s = pippo.next();
```

Metterà in `s` la stringa digitata dall'utente (terminata da un carattere spaziatore o invio)

5 - Stringhe

- Per prendere in input un'intera riga, si usa il metodo `nextLine()`
- Per prendere in input un intero, si usa invece il metodo `nextInt();`
- Analogamente per gli altri tipi: `nextBoolean()`, `nextByte()`, `nextShort()`, `nextLong()`, `nextFloat()`, `nextDouble()`

Esempio:

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

double x = dbl_in.nextDouble();
```

5 - Stringhe

- Per avere la lunghezza di una stringa si usa il metodo `length`, che restituisce un `int` e non ha parametri

Esempio:

```
String a = "Pippo";

int j = a.length();     // j vale 5
```

5 - Stringhe

- Il metodo `substring` restituisce una sottostringa di una stringa specificando il primo carattere (incluso) e l'ultimo carattere (escluso)
- Attenzione: il primo indice è 0
- Omettere il secondo parametro equivale a prendere tutti i caratteri fino alla fine della stringa

Esempio:

```
String s = "Ciao a tutti!";

String t = s.substring(1,4);   // t vale "iao"
String v = s.substring(10);    // v vale "ti!"
```

5 - Stringhe

- I metodi `toLowerCase` e `toUpperCase` trasformano i caratteri di una stringa tutti in minuscolo e tutti in maiuscolo, rispettivamente
- Si usano senza parametri e restituiscono una stringa

Esempio:

```
String s = "Ciao";

String s_low = s.toLowerCase(); //s_low è "ciao"
String s_upp = s.toUpperCase(); //s_upp è "CIAO"
```

5 - Stringhe

- Il metodo `trim` restituisce la stringa senza eventuali spazi presenti all'inizio e alla fine

Esempio:

```
String s = "  Ciao ciao! ";

System.out.println(s.trim());
//stampa "Ciao ciao!"
```

5 - Stringhe

- Il metodo `charAt` restituisce il carattere che si trova in una data posizione di una stringa
- Se il parametro è una posizione che non esiste si solleva un'eccezione (vedremo in seguito come si gestiscono queste eccezioni)

Esempio:

```
String s = "Ciao!";

System.out.println(s.charAt(4)); //stampa "!"
```

5 - Stringhe

- Il metodo `indexOf` restituisce la posizione della prima occorrenza di un carattere, oppure -1 se questo non è presente
- Il metodo `lastIndexOf` è analogo ma con l'ultima occorrenza

Esempio:

```
String s = "pippo";

System.out.println(s.indexOf('p')); //stampa 0
System.out.println(s.indexOf('q')); //stampa -1
System.out.println(s.lastIndexOf('p')); //stampa 3
```

5 - Stringhe

- Il metodo `replaceFirst` permette di sostituire la prima occorrenza di una sequenza di caratteri con un'altra
- Ha due parametri: la sequenza da sostituire e quella con cui sostituirla; restituisce la stringa modificata

Esempio:

```
String s = "Ciao!";

String t = s.replaceFirst( "!" , "..." );
// N.B. s non viene modificata!

System.out.println(t);     //stampa "Ciao..."
```

5 - Stringhe

- Analogamente, il metodo `replaceAll` permette di sostituire tutte le occorrenze di una sequenza di caratteri con un'altra
- Ha due parametri: la sequenza da sostituire e quella con cui sostituirla; restituisce la stringa modificata

Esempio:

```
String s = "Ciao!!!";

s = s.replaceAll( "!" , ".");
System.out.println(s);   //stampa "Ciao..."

// N.B. s è ora un riferimento alla stringa
 "Ciao..." e non più alla stringa "Ciao!!!"
```

5 - Stringhe

- Il metodo (statico) `String.valueOf` permette di trasformare un tipo primitivo in una stringa

Esempio:

```
int num = 1005609;

String snum = String.valueOf(num);
snum = snum.replaceAll( "0" , "X");
System.out.println(snum);     //stampa "1XX56X9"
```