Libreria standard

Java possiede un'enorme libreria di classi standard organizzata in vari package che raccolgono le classi secondo un'organizzazione basata sul campo d'utilizzo.

I principali package sono:

- java.io contiene classi per realizzare l'input – output in Java
- java.awt contiene classi per realizzare interfacce grafiche
- java.net contiene classi per realizzare connessioni, come Socket
- java.applet contiene un'unica classe Applet che permette di realizzare applet
- java.util raccoglie classi d'utilità, come Date
- java.lang è il package che contiene le classi nucleo del linguaggio, come System e String.

Per utilizzare una classe della libreria, bisogna prima importarla. Supponiamo di voler utilizzare la classe Date del package java.util. Prima di dichiarare la classe in cui abbiamo intenzione di utilizzare Date dobbiamo scrivere:

```
import java.util.Date;
```

oppure, per importare tutte le classi del package java.util:

```
import java.util.*;
```

Di default in ogni file Java è importato automaticamente tutto il package java.lang, senza il quale non potremmo utilizzare classi fondamentali quali System e String.

Classe String

In Java le stringhe, a differenza della maggior parte dei linguaggi di programmazione, non sono array di caratteri (char), bensì oggetti, e in quanto oggetti, dovrebbero essere istanziate con la solita sintassi tramite la parola chiave new.

Per esempio:

```
String nome = new String("Mario Rossi");
```

Java però semplifica la vita del programmatore permettendogli di utilizzare le stringhe, come se si trattasse di un tipo di dato primitivo.

Per esempio,

```
String nome = "Mario Rossi";
```

Per assegnare un valore ad una stringa bisogna che esso sia compreso tra virgolette, a differenza dei caratteri per cui vengono utilizzati gli apici singoli.

Il fatto che String sia una classe ci garantisce una serie di metodi di utilità, semplici da utilizzare e sempre disponibili, per compiere operazioni con le stringhe.

Ad esempio toUpperCase(), toLowerCase(), trim() (restituisce la stringa senza gli spazi), equals(String).

La classe String è una classe particolare: un oggetto String è immutabile, ossia i metodi di cui sopra, infatti, non vanno a modificare l'oggetto su cui sono chiamati ma, semmai, ne restituiscono un altro.

```
String a = "claudio";
String b = a.toUpperCase();
System.out.println(a); // a rimane immutato
System.out.println(b); // b è la stringa maiuscola
```

produrrebbero il seguente output:

claudio

CLAUDIO

Per conoscere la classe String e tutte le altre classi, basta andare sul sito http://java.sun.com.

In Java gli array, in quanto collezioni costituite da vari elementi accessibili tramite indici interi, sono oggetti, e per utilizzarli bisogna dichiararli, crearli ed infine come per tutti gli oggetti inizializzarli.

```
char alfabeto [];  //dichiarazione di un array di char (tipo primitivo)
char [] alfabeto1; //ulteriore modo di dichiarare un array di char
Button bottoni [];//array di istanze di Button (classe del package java.awt)
Button [] bottoni; //ulteriore modo di dichiarare un array di Button
```

In pratica, per dichiarare un array, basta posporre o anteporre una coppia di parentesi quadre all'identificatore.

Un array è un oggetto speciale in Java e la **sintassi per l'istanziazione** è la seguente:

```
alfabeto = new char[21];
bottoni = new Button[3];
```

Al momento dell'istanza dell'array si deve dichiarare la dimensione, e tutti gli elementi dell'array vengono inizializzati automaticamente ai relativi valori nulli, e per inizializzare un array, bisogna inizializzarne ogni elemento singolarmente:

```
alfabeto [0] = 'a'; alfabeto [1] = 'b';. . . . . .alfabeto [20] = 'z';
bottoni [0] = new Button();bottoni [1] = new Button();bottoni [2] = new Button();
```

Comunque è possibile eseguire i tre passi con un'unica istruzione.

```
char alfabeto [] = {'a', 'b', 'c', 'd', 'e',…, 'z'};
Button bottoni [] = { new Button(), new Button(), new Button()} ;
```

La variabile length di un array, restituisce la sua dimensione.

```
X=alfabeto.length; // X varrà 21.
```

A differenza della maggior parte degli altri linguaggi, in Java un array bidimensionale non deve per forza essere rettangolare.

```
int arrayNonRettangolare [][] = new int[2][];
arrayNonRettangolare [0] = new int[2];arrayNonRettangolare [1] = new int[4];
arrayNonRettangolare [0][0] = 1;arrayNonRettangolare [0][1] = 2;
arrayNonRettangolare [1][0] = 1;. . . . . .
arrayNonRettangolare [1][3] = 10;
//in alternativa
int arrayNonRettangolare1 [][] = {
{1,2},
{1,0,0,0}};
```

Struttura di un programma in JAVA

Operatori e Gestione del flusso di esecuzione

Gli operatori che andremo ad analizzare si suddividono in:

- Operatori aritmetici. Comprendono somma, sottrazione, moltiplicazione, divisione intera, divisione con modulo ecc.
- Operatori di confronto. Operatori che permettono di verificare determinate condizioni, come ad esempio l'uguaglianza o la disuguaglianza.
- Operatori logici. Da utilizzare con le istruzioni condizionali ed iterative.

Oltre l'assegnamento = esistono i seguenti operatori aritmetici:

Operatore	Descrizione	Esempio
+	Somma o concatenazione di stringhe	a=a+1; a=b+”Ciao”;
-	Sottrazione	a=a-1;
*	Moltiplicazione	a=a*2;
/	Divisione	a=a/2;
%	Modulo	a=a%2;
+=	Somma ed assegnazione	a+=1;
-=	Sottrazione ed assegnazione	a-=1;
*=	Moltiplicazione ed assegnazione	a*=2;
/=	Divisione ed assegnazione	a/=2;
%=	Modulo ed assegnazione	a%=2;
++	Pre-Post incremento o di un’unità	++a; a++
--	Pre-Post decremento o di un’unità	--a; a--

Gli operatori di confronto restituiscono un valore true o false

Operatore	Descrizione	Applicabile
= =	Uguale ad	Tutti i tipi
!=	Diverso da	Tutti i tipi
<	Minore	Tipi Numerici
>	Maggiore	Tipi Numerici
<=	Minore uguale	Tipi Numerici
>=	Maggiore uguale	Tipi Numerici

La classe String gode di una singolare particolarità nell'utilizzo dell'operatore ==

```
String a = "Java";
String b = a;
String c = new String("Java");
System.out.println(a==b);
System.out.println(b==c);
/* produrrà il seguente output:
true
false
a e b puntano allo stesso oggetto, mentre per c, utilizzando new, verrà creato un
oggetto ex novo*/
System.out.println(a.equals(b));
System.out.println(b.equals(c));
/*produrrà il seguente output:
true
true*/
```

Gli operatori logico–booleani utilizzano solo operandi di tipo booleano ed il risultato è di tipo boolean:

Operatore	Descrizione
&	AND logico
!	NOT logico
|	OR logico
^	XOR logico
&&	Short circuit AND
||	Short circuit OR
&=	AND e assegnazione
|=	OR e assegnazione
^=	XOR e assegnazione

Le versioni short circuit (“corto circuito”) di AND ed OR fanno evitare il secondo controllo in caso di fallimento del primo.

In Java le istruzioni di controllo del flusso sono:

if else	if (espressione-booleana) {
	istruzione_1;
	.....;
	istruzione_k;
	} else {
	istruzione_k+1;
	.....;
	istruzione_k+n; }

L'operatore ternario	variabile = (espr-booleana) **?** espr1 **:** espr2**;**
while	[inizializzazione**;**] **while** (espr. booleana) **{** corpo**;** [aggiornamento iterazione**;**] **}**

For	**for** (inizializzazione**;** espr. booleana**;**aggiornamento)
	{istruzione_1**;**
	… istruzione_n**;}**
	for (variabile_temporanea : oggetto_iterabile) {
	corpo;
	}
	Es.
	int [] arr = {1,2,3,4,5,6,7,8,9};
	for (int tmp : arr) {
	System.out.println(tmp);}

do while	[inizializzazione;] do { corpo; [aggiornamento iterazione;] } while (espr. booleana);

Switch

```
switch (variabile di test) {
case valore_1: istruzione_1;break;
case valore_2: {
    istruzione_2;.......;
    istruzione_k;
    }break;
case valore_3:
case valore_4: { //blocchi di codice opzionale
        istruzione_k+1;
        ..........;
        istruzione_j;} break;
default: { //clausola default opzionale
         istruzione_j+1;
          ..........;
        istruzione_n; }
}
```

In Java come in C la parola chiave **break** è un comando capace di far terminare un qualsiasi ciclo, mentre la parola chiave **continue**, fa terminare non l'intero ciclo,ma solo l'iterazione corrente

```
//Il seguente di codice  stampa i primi dieci numeri interi:
int i = 0;
while (true) //ciclo infinito{
  if (i > 10) break;
  System.out.println(i);
  i++;}
i = 0;
Il seguente codice stampai primi dieci numeri, escluso il cinque:
do{
  i++;
  if (i == 5) continue;
System.out.println(i);}
while(i <= 10);
```

In Java sia break sia continue possono utilizzare etichette (label) per specificare, solo nel caso di cicli annidati, su quale ciclo devono essere applicati.

```
//Il seguente di codice  stampa i primi dieci numeri interi:
int  j = 1;
etichetta1: //possiamo dare un qualsiasi nome ad una label
while (true){
  while (true){
      if (j > 10) break etichetta1;
     System.out.println(j);
     j++;
  }
}
```

Una label può essere posizionata solo prima di un ciclo, non dove si vuole. In Java non esiste il comando goto.

Esercizio 3

Scrivere un programma con i seguenti requisiti.
Utilizzare una **classe Persona** che dichiara le variabili nome, cognome, età. Si dichiari inoltre un metodo dettagli() che restituisce in una stringa le informazioni sulla persona in questione.

Utilizzare una classe Principale che, nel metodo main(), istanzia due oggetti chiamati persona1 e persona2 della classe Persona, inizializzando per ognuno di essi i relativi campi.
Dichiarare un terzo riferimento (persona3) che punti ad uno degli oggetti già istanziati. Controllare che effettivamente persona3 punti allo oggetto voluto, stampando i campi di persona3.

Oppure fare queste operazioni nel metodo main all'interno della classe Persona.

Esercizio 4

Scrivere un programma con i seguenti requisiti.
Utilizzare una **classe Impiegato** che dichiara le variabili nome, cognome, salario. Si dichiari un metodo dettagli() che restituisce in una stringa le informazioni sulla persona in questione.
Si dichiari un metodo aumentasalario() che aumenti lo stipendio secondo una certa percentuale.

Utilizzare un metodo main nella classe Impiegato oppure una classe Principale oppure che, nel metodo main(), istanzia due oggetti chiamati impiegato1 e impiegato2 della classe Impiegato, inizializzando per ognuno di essi i relativi campi. Si aumenti il salario di impiegato1 del 10% e poi si verifichi se il salario di impiegato 1 è maggiore di quello di impegato2.