SKILL FACTORY

Scuola **E**sperienza **A**ggiornamento **L**avoro

www.skillfactory.it

SKILL FACTORY

Scuola Esperienza Aggiornamento Lavoro

JAVA – SVILUPPO APPLICAZIONI DESKTOP

Logica di programmazione

Autore: Mirko Onorato – Skill Factory Srl

www.skillfactory.it

Logica di programmazione

autore Mirko Onorato – Skill Factory Srl

1.1 – TECNICA DI BASE: I CONTATORI

I contatori sono variabili di tipo intero che verificano il numero di volte in cui viene eseguita un'operazione. Un contatore si aggiorna incrementando/decrementando di 1 o altro valore secondo la formula:

```
contatore=contatore+1 oppure contatore+=1 oppure contatore++
```

```
//Contatore nel ciclo while
public void contaFinoA10() {
int i=1; //contatore
while(i<=10) {
System.out.println(i);
i++; //incremento del contatore
}
}

//Contatore nel ciclo for
public void contaFinoA10() {
for(int i=1; i<=10; i++) { //contatore e suo incremento interni al ciclo for
System.out.println(i);
}
}
```

1.2 – TECNICA DI BASE: I TOTALIZZATORI

I totalizzatori servono per aggiungere o sottrarre al contenuto di una variabile un valore. La formula per creare un totalizzatore è simile a quella dei contatori, ma al posto dello step bisogna indicare il valore da aggiungere o sottrarre, come mostra l'esempio seguente:

```
totalizzatore=totalizzatore+valore oppure totalizzatore+=valore
```

Ammettiamo di volere in stampa i primi cinque multipli di 5, l'algoritmo sarà il seguente:

```
public void usoTotalizzatore() {
int totalizzatore=0;
int valore=5;
for(int i=1; i<=5; i++) {
totalizzatore=totalizzatore+valore;
System.out.println(totalizzatore);
}
}
```

```
5
10
15
20
25
```

1.3 – TECNICA DI BASE: IL CONCATENAMENTO

Il concatenamento è la tecnica usata per unire più stringhe tra loro. Una stringa è una sequenza di valori alfanumerici definiti dai sistemi di codifica ASCII o Unicode (in funzione del numero di caratteri necessari) che assegnano un numero univoco ad ogni carattere usato per la scrittura di testi. Date due o più stringhe, la concatenazione tra queste determinerà un'unica stringa:

```
String str1="Skill";
String str2="Factory";
String strConcat="Skill"+"Factory"; //concatenazione

System.out.print(strConcat); //in stampa leggeremo "Skill Factory«

//oppure

System.out.print("Skill"+"Factory"); //in stampa leggeremo "Skill Factory"
```

1.3 – TECNICA DI BASE: IL CONCATENAMENTO

Il concatenamento converte in stringa anche i tipi primitivi se quest'ultimi vengono sommati ad una stringa:

```
int a1=5;
int a2=10;
boolean b=false;
String c="Skill Factory";

System.out.print(c+b+a1+a2); //stampa "Skill Factoryfalse510
```

La variabile a1 perde la sua identità in quanto concatenandosi con una stringa diventa a sua volta una stringa; stesso discorso vale per la variabile a2. Mettendo tra parentesi a1 e a2 consentiamo prima la somma numerica dei relativi contenuti e poi la concatenazione della somma stessa:

```
System.out.print(c+b+(a1+a2)); //stampa "Skill Factoryfalse15
```

1.3 – TECNICA DI BASE: IL CONCATENAMENTO

Se i tipi primitivi precedono una concatenazione questi mantengono la loro identità:

```
System.out.print(1+2+"Skill Factory"); //stampa "3Skill Factory"

System.out.print(1+2+"Skill Factory"+2+1); //stampa "3Skill Factory21"

System.out.print(1+2+"Skill Factory"+(2+1)); //stampa "3Skill Factory3"
```

1.4 – TECNICA DI BASE: COME MANIPOLARE LE STRINGHE

La classe String regala molteplici funzionalità per la manipolazione delle stringhe; è possibile: effettuare conversioni da String ad un tipo primitivo (es. in un numero intero) come per le classi Wrapper; scomporre una stringa in un vettore di stringhe; scomporre una stringa in un vettore di char; rendere minuscoli o maiuscoli i caratteri di una stringa; confrontare stringhe; sostituire una o più parti della stringa; concatenare più stringhe; ecc. Elenchiamo di seguito alcuni dei metodi più usati:

-**equals(String str)**
Confronta la stringa contenuta in una variabile con un'altra.

```
String var="Skill Factory";
if(var.equals("Skill Factory")){
        System.out.print("Welcome!!!"); //avremo questo risultato
}
else{
        System.out.print("Le due stringhe non corrispondono");
}
```

1.4 – TECNICA DI BASE: COME MANIPOLARE LE STRINGHE

-equalsIgnoreCase(String str)

Java è un linguaggio «case sensitive», ovvero, fa distinzione tra caratteri minuscoli e maiuscoli. Stringhe come «SKILL», «skill» e «Skill» non sono uguali ma valori tra loro diversi. Il metodo equalIgnoreCase() si comporta come l'equals() ma rende sempre minuscoli tutti i caratteri della stringa contenuta nella variabile; torna utile per prevenire errori (ammettiamo di dover rispondere con un «si», digitando «Si» non avremmo il risultato atteso).

```
String var="SKILL FactoRY";
if(var.equalsIgnoreCase("Skill Factory")){
        System.out.print("Welcome!!!"); //avremo questo risultato
}
else{
        System.out.print("Le due stringhe non corrispondono");
}
```

1.4 – TECNICA DI BASE: COME MANIPOLARE LE STRINGHE

-**concat(String str)**
Concatena più stringhe in una.

```
String var1="Skill ";
String var2="Factory";
System.out.print(var1.concat(var2)); //stampa Skill Factory
```

-**substring(int indiceIniziale) o substring(int indiceIniziale, int indiceFinale)**
Il metodo ammette uno o due argomenti interi ed estrae una stringa (una substringa) da quella originale. Il singolo argomento indica la posizione del carattere a partire dal quale avviene l'estrazione; con due argomenti si indicano invece gli estremi dell'estrazione, ovvero, il primo indica la posizione del carattere da cui essa inizia mentre il secondo la posizione del carattere ***«escluso»*** dalla substringa, limite superiore dell'estrazione stessa.

```
String var="Skill Factory";
String subStr1=var.substring(0,5); //viene estratto «Skill»
String subStr2=var.substring(6); //viene estratto «Factory»
System.out.print(subStr1); //stampo «Skill»
System.out.print(subStr2); //stampo «Factory»
System.out.print(subStr1.concat(subStr2)); //stampo «Skill Factory»
```

1.4 – TECNICA DI BASE: COME MANIPOLARE LE STRINGHE

-contains(String str)

Verifica se una determinata sequenza di caratteri è presente nella stringa. Il metodo ritorna ***true*** se la sequenza è presente, altrimenti ***false***.

```
String var="Skill Factory";
if(var.contains("il")) {
System.out.println("Welcome!!!"); //l'if si attiva: «Welcome!!!»
}
```

-length(String str)

Il metodo ritorna la lunghezza della stringa, ovvero, il numero di caratteri che la compone.

```
System.out.print(var.length()); //stampa 13 (è compreso lo spazio)
```

-isEmpty(String str)

Il metodo ritorna ***true*** se la stringa ha lunghezza 0 (non ha nemmeno un carattere).

```
String var=""; //stringa vuota
if(var.isEmpty()) {
System.out.println("Welcome!!!"); //l'if si attiva: «Welcome!!!»
}
```

1.4 – TECNICA DI BASE: COME MANIPOLARE LE STRINGHE

-**split(String regex)**

Il metodo ritorna un array di stringhe i cui elementi sono le substringhe estratte da quella originale; per ottenere questi elementi bisogna impostare una regex (una regola) che si traduce in un elemento ricorrente della stringa originale.

```
String var="Skill,Factory,Community";
String[] array=var.split(","); //l'elemento ricorrente è la virgola
System.out.print(array[0]); //stampa «Skill»
System.out.print(array[1]); //stampa «Factory»
System.out.print(array[2]); //stampa «Community»
```

-**trim()**

Elimina gli spazi alle estremità di una stringa.

```
String var=" Skill Factory ";
System.out.println(var.trim()); //stampa «Skill Factory» senza spazi ai lati
```

1.4 – TECNICA DI BASE: COME MANIPOLARE LE STRINGHE

-**replace(char carattereVecchio, char carattereNuovo)**
Sostituisce il carattere di una stringa con un altro.

```
String var1="Skill Factory";
String var2=var1.replace('l', '1');
System.out.println(var2); //stampa «Ski11 Factory»
```

-**replaceAll(String substringaVecchia, String substringaNuova)**
Sostituisce una parte della stringa con un'altra.

```
String var1="Skill Factory";
String var2=var1.replaceAll("Factory", "Community");
System.out.println(var2); //stampa «Skill Community»
```

-**toLowerCase()**
Rende minuscoli tutti i caratteri della stringa.

```
String var1="SKILL FACTORY";
System.out.println(var1.toLowerCase()); //stampa «skill factory»
```

1.4 – TECNICA DI BASE: COME MANIPOLARE LE STRINGHE

-**toUpperCase()**
Rende maiuscoli tutti i caratteri della stringa.

```
String var1="skill factory";
System.out.println(var1.toUpperCase()); //stampa «SKILL FACTORY»
```

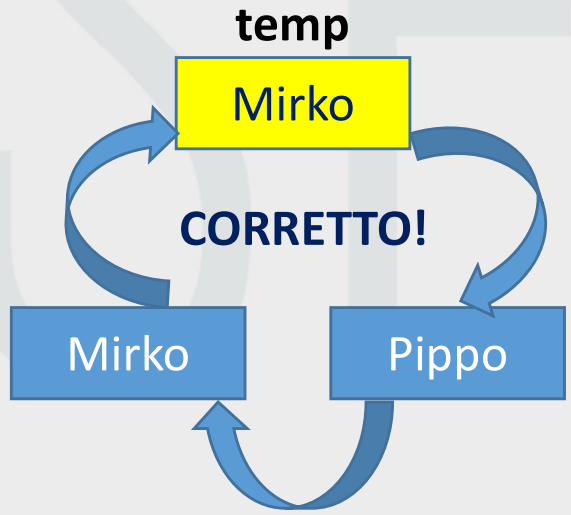
-**charAt(int indiceCarattere)**
Il metodo tratta una stringa come se fosse un array contenente dati di tipo char. Passando come argomento un indice recuperiamo il carattere corrispondente.

```
String var1="Skill";
System.out.print(var1.charAt(1)); //stampa «k»
```

1.5 – TECNICA DI BASE: COME INVERTIRE IL CONTENUTO DI DUE VARIABILI

Lo scambio o swap è una tecnica con la quale è possibile invertire il contenuto di due variabili. Lo swap non avviene direttamente in quanto occorre una ***variabile temporanea*** che conservi il valore della prima variabile; senza la variabile temporanea il contenuto della seconda variabile sovrascriverebbe quello della prima, ritrovandoci due variabili con lo stesso valore.


```
public void swap() {
String s1="Mirko";
String s2="Pippo";
String temp;
System.out.println("Prima dello swap:"+s1+"|"+s2);
temp=s1;
s1=s2;
s2=temp;
System.out.println("Dopo lo swap:"+s1+"|"+s2);
}
```



1.6 – TECNICA DI BASE: MINIMO E MASSIMO

```
public void minMax() {
Scanner input = new Scanner(System.in);
System.out.println("Inserire il primo valore numerico");
int a=Integer.parseInt(input.nextLine());
System.out.println("Inserire il secondo valore numerico");
int b=Integer.parseInt(input.nextLine());
if(a<b)
System.out.println(a+" minore di "+b);
else if(a==b)
System.out.println("Le due variabili contengono lo stesso valore");
else
System.out.println(a+" maggiore di "+b);
}
```

1.7 – TECNICA DI BASE: PARI E DISPARI

Un valore è pari se divisibile per 2 e se il resto è pari a zero; diversamente è dispari. Per valutare se un numero è pari possiamo utilizzare il modulo % (che rappresenta il resto) all'interno della condizione di una struttura condizionale.

```
public void pariDispari() {
Scanner input = new Scanner(System.in);
System.out.println("Inserire un valore numerico");
int n=Integer.parseInt(input.nextLine());
if(n%2==0)
System.out.println(n+" è un numero pari");
else
System.out.println(n+" è un numero dispari");
}
```

1.8 – TECNICA DI BASE: PRE-INCREMENTO E POST-INCREMENTO

Il pre-incremento aggiorna immediatamente il contenuto di una variabile numerica, quindi prima incrementa il valore e poi lo usa; il post-incremento prima usa il contenuto della variabile e poi lo incrementa. Stesso discorso vale per pre-decremento e post-decremento.

Pre-incremento: ++i
Post-incremento: i++
Pre-decremento: --i
Post-decremento: i--

Esempio 1:

```
int i=5;
System.out.println(--i); //4 -> prima decrementa, poi usa la variabile
```

Esempio 2:

```
int i=5;
System.out.println(i--); //5 -> prima usa la variabile, poi decrementa
```

Esempio 3:

```
int i=5;
System.out.println(i++); //5 -> prima usa la variabile, poi incrementa a 6
System.out.println(++i); //7 -> prima incrementa, poi usa la variabile
```


THANK
for watching
YOU
www.skillfactory.it