Variabili e Metodi di classe
Interfacce e Package
File in Java

Variabili di classe: Static

- Una **variabile di classe** definisce un dato comune a tutti gli oggetti di una classe.

- Mentre le **variabili di istanza** hanno generalmente valori diversi in ogni oggetto creato, per le variabili di classe viene memorizzato **un unico valore** che è comune a tutti gli oggetti della classe.

- Le variabili di classe sono variabili **statiche**, cioè dichiarate con la parola chiave `static` :

```
static char separatore;
```

- Se una classe definisce una variabile `static` ciascun oggetto di quella classe avrà lo stesso valore per quella variabile.

Variabili di classe: Static

- Le variabili **`static`** servono per condividere valori comuni tra gli oggetti di una classe.
- Ad esempio:

```
class Prodotti {
   String nome;
   int prezzo;
   static int codiceProduttore;
}
```

- Il valore della variabile statica **`codiceProduttore`** è comune a tutti gli oggetti della classe **`Prodotti`**.
- Una variabile dichiarata **`static`** è memorizzata in un unica cella di memoria comune a tutti gli oggetti della classe.
- Una volta modificato il valore di una variabile **`static`** la modifica è valida per tutti gli oggetti della classe.

Variabili di classe: Static

- Generalmente le variabili di classe sono accedute al di fuori della classe premettendo il nome della classe :

```
Prodotti.codiceProduttore = 102 ;
```

- In questo modo si vuole intendere che la modifica riguarda tutti gli oggetti della classe.

- Si può anche usare pure l'object-id (ma non è consigliabile). Ad esempio sulla variabile **`codiceProduttore`** dopo la creazione:

```
Prodotti prod1 = new Prodotti();
```

posso scrivere in ambedue i modi

```
Prodotti.codiceProduttore = 102 ;
Prod1.codiceProduttore = 102;
```

- Si noti in ambedue i casi si modifica il campo **`codiceProduttore`** per tutti gli oggetti della classe **`Prodotti`** e non solo per **`prod1`**.

Metodi di classe

- Un **metodo di istanza** è un metodo che viene eseguito (chiamato) su un oggetto di una classe. L'oggetto è detto argomento implicito del metodo. Ad esempio:
  - `if (a.equals(c)) . . .`
  - `A = r1.area();`

  in questi due casi i metodi vengono invocati sugli oggetti **`a`** ed **`r1`**.

- Un **metodo di classe** è un metodo che agisce sulla classe e non su uno specifico oggetto. E quindi può essere usato senza legarlo ad un oggetto.

- Ad esempio:
  - `x= Math.sqrt(x);`
  - `s= String.valueOf(23);`

  in questi due casi i metodi vengono invocati non su oggetti ma usando il nome della classe. Non ci sono argomenti impliciti ma solo argomenti espliciti passati tramite i parametri attuali.

Metodi di classe : Static

- Un **metodo di classe** si definisce usando la parola chiave **`static`**.

  Ad esempio nel caso del metodo che calcola la media delle distanze:

```
class Distanze
{
  public static float valorMedio(float d[])
  {
   float dmedia;   // contiene il valore medio delle distanze
   float dsomma;   // contiene la somma dei valori delle distanze

   sommad=0.0;
   for (int i=0; i < d.length(); i++)
      sommad = sommad + d[i];
   dmedia = sommad/d.length;
   return dmedia;
  }
}
```

- Il metodo **`valorMedio`** è un metodo statico (o di classe) e verrà invocato usando il nome della classe nel modo seguente:

```
m = Distanze.valorMedio(d);
```

- Si noti che il metodo non accede alcuna variabile d'istanza di un oggetto.

Interfacce Java

- Le **intefacce** (`interface`) in Java sono uguali alle classi ma contengono metodi senza corpo e variabili statiche.

- Servono a definire le caratteristiche di una classe e l'elenco delle sue operazioni.

- Saranno le classi ad implementare i metodi delle interfacce:

```
interface Ordina { . . .}

class Dato implements Ordina { . . .}
```

Package Java

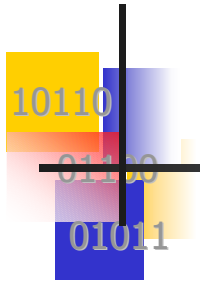
- I package raggruppano le classi e le interfacce in strutture gerarchiche.
- Una classe può specificare che fa parte di un package tramite le istruzioni:

```
package nomepackage1 ;
package nomepackage1.nomepackage2.nomepackage3;
```

- Le classi che appartengono a package diversi si possono usare con l'istruzione import

```
import nomepackage1.nomepackage2.nomeclasse.
```

- Il package **`java.lang`** è importato automaticamente in tutte le classi.



File in Java

- Un **file** è una struttura dati permanente che contiene un blocco logicamente contiguo di informazioni che risiede su disco.

- Un file può esistere prima dell'esecuzione di un programma oppure il programma lo può creare e può esistere dopo la fine dell'esecuzione del programma, se il programma dopo averlo usato non lo cancella.

- Un file ha un proprio nome definito da un pathname e risiede in una directory.

- In Java i file e le operazioni associate sono definiti nel package **`java.io`** che contiene molte classi (48) e 7 interfacce per la gestioni di file.

File in Java

- Una semplice classe è la classe **`File`**, altre classi importanti del package **`java.io`** sono

  `InputStream`, **`Reader`** : per effettuare operazioni di lettura sull' I/O

  `OutputStream`, **`Writer`** : per effettuare operazioni di scrittura sull'I/O

  `RandomAccessFile` : per effettuare operazioni di lettura e scrittura

InputStream	FileInputStream	DataInputStream → System.in → **`read`**
OutputStream	FileOutputStream	PrintStream → System.out → **`print`**, **`println`**
RandomAccessFile	Implementa le interfacce DataOutput e DataInput	

La classe File

- La classe **File** definisce oggetti file che vengono associati ai file fisici del file system e un insieme di metodi per creare, cancellare e gestire file.

- Non sono definiti metodi per la lettura e scrittura su file.

- Creazione/Apertura

```
File f = new File(pathname);
```

oppure

```
File f = new File(directory, pathname);
```

- Il pathname utilizza il carattere `/`. Ad esempio per creare o aprire un file *lettera* nella directory *C:|doc* si può usare:

```
File f = new File("C:/doc", "lettera")
```

La classe File

- Metodi della classe File:

`exists()`	verifica l'esistenza di un file. Esempio: **`f.exists()`**
`isFile()`	verifica se è un file normale. Esempio: **`f.isFile()`**
`isDirectory()`	verifica se è una directory.
`length()`	restituisce la lunghezza del file.
`renameTo(dest)`	cambia il nome del file.
`delete()`	cancella il file dal disco.
`getName()`	restituisce il nome del file.
`getPath()`	restituisce il pathname completo del file.

La classe File

- Nel caso in cui il file è una directory si possono usare i seguenti metodi:

- `list()`
  - restituisce un array di stringhe che corrispondono ai nomi dei file contenuti nella directory.
  - Esempio:
    - `String [] Listafile = f.list();`

- `mkdir()`
- crea una directory in base alle informazioni dell'oggetto su cui è invocata e ritorna un valore booleano che sarà true se l'operazione è stata effettuata.
-

- **N.B: La classe file non contiene operazioni di lettura e scrittura su file!**

La classe File : Esempio

- L'esempio seguente conta il numero di caratteri presenti in un file. Se il file è una directory, il programma mostra i nomi dei file che sono contenuti in essa.

```
import java.io.*;
public class ProvaFile1
{
  static public void main (String argv[])
  {
    if(argv.length != 1)
      System.err.println("Uso: java ProvaFile1 nomefile");
    else
      leggiFile(argv[0]);
  }
  . . . . . .
```

- Il programma assume che il nome del file viene passato tramite **`args`**.

La classe File : Esempio

```
 . . . . . .
static private void leggiFile(String filename)
{
  long dim = 0;
  File f = new File(filename);
  if (f.exists())
  {
   dim = f.length();
   System.out.println("dimensione " + filename + " = " + dim);
   if (f.isDirectory())
   {
    String lista[] = f.list();
    for (int i =0; i < lista.length; i++)
      System.out.println("nome file = " + filename + "/" + lista[i]);
   }
  }
 }  // fine metodo leggiFile
}   // fine classe
```