INTERFACCE

Un'interfaccia è una raccolta di metodi astratti.
Una classe implementa un'interfaccia, ereditando così i metodi astratti dell'interfaccia.

Un'interfaccia non è una classe.

Scrivere un interfaccia è simile a scrivere una classe, ma sono due concetti differenti.
Una classe descrive gli attributi e i comportamenti di un insieme di oggetti.
Un'interfaccia contiene i comportamenti che una classe implementa.
A meno che la classe che implementa l'interfaccia sia una classe astratta, nella classe utilizzatrice devono essere definiti tutti i metodi dell'interfaccia.
Un'interfaccia **è simile** a una classe perché :

- Un'interfaccia può contenere un numero qualsiasi di metodi.
- Un'interfaccia è scritta in un file con estensione .java, con il nome dell'interfaccia corrispondente al nome del file.
- Il bytecode di una interfaccia è in un file .class.
- Le Interfacce sono contenute in Package.

Tuttavia, l'interfaccia è **diversa** da una classe per diversi motivi:

- Non è possibile creare un'istanza di un'interfaccia.
- Un'interfaccia non contiene costruttori.
- Tutti i metodi di un'interfaccia sono astratti.
- Un'interfaccia non può contenere variabili di istanza. Gli unici campi che possono essere definiti in un'interfaccia devono essere dichiarati sia STATIC che FINAL
- L'interfaccia non deriva da una classe; è implementata da una classe.
- Un'interfaccia può estendere interfacce multiple.

Dichiarazione di Interfacce:

La parola chiave **interface** viene utilizzata per dichiarare una interfaccia. Ecco un semplice esempio per dichiarare una interfaccia:

Esempio:

questo esempio illustra l'incapsulamento:

```
import java.lang.*;


public interface Esempio
{
   //Any number of final, static fields
   //Any number of abstract method declarations\
}
```

Le Interfacce hanno le seguenti proprietà:

- Un'interfaccia è implicitamente astratta. Non è necessario utilizzare la parola chiave **abstract** quando si dichiara una interfaccia.
- Ogni metodo in un'interfaccia è anche implicitamente astratto.
- I Metodi in una interfaccia sono implicitamente pubblic.

Esempio:

/ * Nome file: Animai I.java * /

```
/* File name : Animali.java */
interface Animali {

   public void mangia();
   public void si_muove();
```

```
}
```

Implementazione Interfacce:

Quando una classe implementa un'interfaccia DEVE eseguire i comportamenti specifici dell'interfaccia. Se una classe non esegue tutti i comportamenti dell'interfaccia, la classe deve essere dichiarata ABSTRACT

Una classe utilizza la parola IMPLEMENTS per implementare un'interfaccia.

```
  / * Nome file: MammalInt.java * /
public class Mammiferi  implements Animali {

    public void mangia () {
       System.out.println ("mangia Mammifero");
    }

  public void si_muove () {
       System.out.println ("viaggia Mammifero ");
    }

   int pubblic  noOfLegs () {
       return 0;
    }


public static void main(String args[]){
      MammalInt m = new MammalInt();
      m.eat();
      m.travel();
   }}
```

Ciò dovrebbe produrre il seguente risultato:

```
mangia Mammifero
viaggia Mammifero
```

l'overriding di metodi definiti nelle interfacce segue diverse regole:

- Eccezioni controllate non devono essere dichiarati sui metodi di implementazione diversi da quelli dichiarati dal metodo di interfaccia o sottoclassi di quelle dichiarate dal metodo di interfaccia.
- La firma del metodo di interfaccia e lo stesso tipo di ritorno o sottotipo dovrebbe essere mantenuta nell' overriding dei metodi.
- Una classe di implementazione può essere astratta e se i metodi di interfaccia in modo non devono essere attuate.

Implementando un' interfaccia ci sono diverse regole:

- Una classe può implementare più interfacce alla volta.
- Una classe può estendere solo una classe, ma implementare molte interfacce.
- Un'interfaccia può estendere un'altra interfaccia, in modo simile al modo in cui una classe può estendere un'altra classe.

Estensione di Interfacce:

Un'interfaccia può estendere un'altra interfaccia, in modo simile al modo in cui una classe può estendere un'altra classe. La parola ***extends*** viene utilizzato per estendere un'interfaccia, e l'interfaccia figlia eredita i metodi dell'interfaccia padre.

```
//Nome file: Sports.java
public interface Sports
{
```

```
   public void setHomeTeam(String name);
   public void setVisitingTeam(String name);
}

//Nomefile: Football.java
public interface Football extends Sports
{
   public void homeTeamScored(int points);
   public void visitingTeamScored(int points);
   public void endOfQuarter(int quarter);
}

// Nomefile: Hockey.java
public interface Hockey extends Sports
{
   public void homeGoalScored();
   public void visitingGoalScored();
   public void endOfPeriod(int period);
   public void overtimePeriod(int ot);
}
```

L'interfaccia Hockey ha quattro metodi, ma eredita due da Sport; in tal modo, una classe che implementa Hockey deve attuare tutti i sei metodi. Allo stesso modo, una classe che implementa Football ha bisogno di definire i tre metodi da calcio e due metodi di Sports.

Estensione Interfacce multiple:

Una classe Java può estendere solo una classe padre. L'Ereditarietà multipla non è permessa.
Le interfacce non sono classi, tuttavia, e un'interfaccia puo' estendere più di un'interfaccia padre.

La parola chiave extends viene utilizzata una sola volta, e le interfacce padri sono dichiarate in un elenco separato da virgole.

Ad esempio, se l'interfaccia Hockey estendesse sia Sport che Evento, sarebbe dichiarato come:

```
public interface Hockey extends Sports, Event
```

Interfacce Tagging:

L'uso più comune di estendere interfacce si verifica quando l'interfaccia madre non contiene metodi. Ad esempio, l'interfaccia MouseListener nel pacchetto java.awt.event estende java.util.EventListener, che è definito come:

```
package java.util;
public interface EventListener
{}
```

Un'interfaccia che non contiene metodi è detta interfaccia **tagging.**
Due sono gli scopi essenziali di usare interfacce tagging:
Creazione di un genitore comune: Come con l'interfaccia EventListener, che si estende per decine di altre interfacce API Java, è possibile utilizzare un interfaccia tagging per creare un genitore comune tra un gruppo di interfacce. Ad esempio, quando un interfaccia estende EventListener, JVM sa che questo particolare interfaccia sta per essere utilizzato in un scenario di delegazione di evento.
Aggiunta di un tipo di dati a una classe . Una classe che implementa un'interfaccia Tagging non deve definire tutti i metodi (in quanto l'interfaccia non ha metodi definiti), ma la classe diventa un tipo di interfaccia attraverso il polimorfismo.