CLASSI ASTRATTE

L'ereditarietà porta riflettere sul rapporto fra *progetto* e *struttura*:

- **una classe può limitarsi a *definire solo l'interfaccia*…**
- **..lasciando "in bianco" uno o più metodi...**
- **...che verranno poi implementati dalle classi derivate:**

è una classe astratta

CLASSI ASTRATTE

Una classe astratta:

- **fattorizza, *dichiarandole*, operazioni comuni a tutte le sue sottoclassi…**
- **... ma non le *definisce* (implementa)**

In effetti,

- **non viene creata *per definire istanze*** (che non saprebbero come rispondere ai metodi "lasciati in bianco")
- **ma per derivarne altre classi**, che dettaglieranno i metodi qui solo dichiarati

CLASSI ASTRATTE: ESEMPIO

```
public abstract class Animale {
 public abstract String verso();
 public abstract String si_muove();
 public abstract String vive();
 ...
}
```

- **I metodi astratti *non hanno corpo***
- **Se *anche solo un metodo* è `abstract`, la classe deve essere qualificata `abstract`, altrimenti si ha ERRORE**

CLASSI ASTRATTE: PERCHÉ?

- **Moltissime entità che usiamo per descrivere il mondo *non sono reali***
- **sono *pure categorie concettuali,* ma sono comunque *molto utili!***

Esempio: gli animali

- **parlare di "animali" ci è molto utile, però *non esiste "il generico animale" !!***
- **nella realtà esistono solo *animali specifici***
- **ma la categoria concettuale "animale" ci fa molto comodo per "parlare del mondo"**

ESEMPIO: GLI ANIMALI

- **Tutti sappiamo cosa sia un animale:**
  - **ogni animale ha *un qualche verso***
  - **ogni animale si muove *in qualche modo***
  - **ogni animale vive *in un qualche ambiente***
- **...ma *proprio per questo*, *non esiste "il generico animale"!***
- **Ogni animale reale:**
  - **ha *uno specifico verso,***
  - **si muove *in uno specifico modo,***
  - **vive *in uno specifico ambiente.***
    (acqua, aria, terraferma, etc.)

ESEMPIO: GLI ANIMALI

- **Introdurre una classe `animale` è utile *proprio per fattorizzare gli aspetti comuni***
  - **tutti gli animali hanno un verso, si muovono in un qualche modo, ecc.**
- **Tali aspetti inducono una *classificazione* del mondo:**
  - ***animali acquatici, uccelli, mammiferi…***
  - **Peraltro, non esistono neanche generici uccelli, mammiferi, ecc… ma *disporre di queste categorie concettuali è assai utile*!**

CLASSI ASTRATTE: ESEMPIO

```
public abstract class Animale {
 public abstract String verso();
 public abstract String si_muove();
 public abstract String vive();
 ...
}
```

- **Esprime il fatto che ogni animale ha un verso, si muove, e vive da qualche parte..**
- ***...ma in generale non si può dire come***

CLASSI ASTRATTE: ESEMPIO

```
public abstract class AnimaleTerrestre
  extends Animale {
 public String vive() { // era abstract
   return "sulla terraferma"; }
 ...
}
```

- **Una classe derivata può definire uno o più metodi che erano astratti**
- **se *anche solo un metodo* rimane astratto, la classe derivata è comunque *astratta***
  (e deve essere qualificata come tale)

L'ESEMPIO COMPLETO

Ipotesi:

- **ogni animale ha un metodo `chi_sei()` che restituisce una stringa descrittiva**
- **ogni animale ha un metodo `mostra()` che lo stampa a video e che è *indipendente* dallo specifico animale** (si appoggia sugli altri metodi)
- **tutti gli animali si rappresentano nello stesso modo, tramite il loro *nome* e il *verso* che fanno**

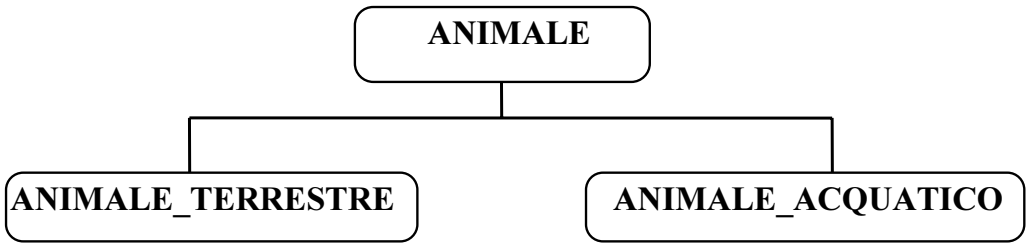
L'ESEMPIO COMPLETO

```
public abstract class Animale {
  private String nome;
  protected String verso;
  public Animale(String s) { nome=s; }
  public abstract String si_muove();
  public abstract String vive();
  public abstract String chi_sei();
  public void mostra() {
   System.out.println(nome + ", " +
     chi_sei() + ", " + verso +
     ", si muove " + si_muove() +
     " e vive " + vive() );  }
}
```

Costruttore necessario

L'ESEMPIO COMPLETO

Una possibile classificazione:



Sono ancora classi astratte:

- nulla si sa del movimento
- quindi è impossibile definire il metodo **`si_muove()`**

L'ESEMPIO COMPLETO

```
public abstract class AnimaleTerrestre
     extends Animale {
  public AnimaleTerrestre(String s) {
     super(s); }
  public String vive() {
     return "sulla terraferma"; }
  public String chi_sei() {
     return "un animale terrestre"; }
}
```

Due metodi astratti su tre sono ridefiniti,
ma ***uno è ancora astratto***
→ la classe è ancora astratta

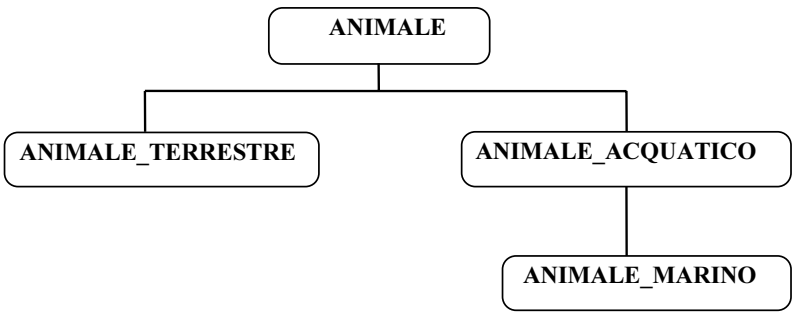
L'ESEMPIO COMPLETO

```
public abstract class AnimaleAcquatico
     extends Animale {
  public AnimaleAcquatico(String s) {
     super(s); }
  public String vive() {
     return "nell'acqua"; }
  public String chi_sei() {
     return "un animale acquatico"; }
}
```

Due metodi astratti su tre sono ridefiniti, ma ***uno è ancora astratto***
→ la classe è ancora astratta

L'ESEMPIO COMPLETO

Una possibile specializzazione:



Perché introdurre l'animale marino?

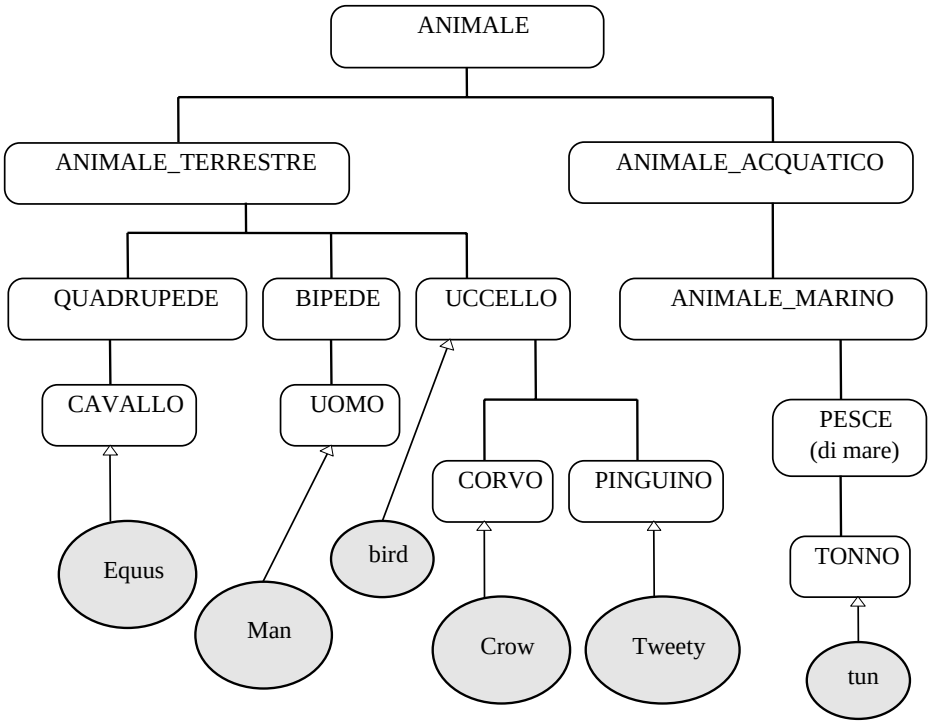
- non è correlato a verso, movimento, etc
- rispecchia semplicemente una realtà

L'ESEMPIO COMPLETO

```
public abstract class AnimaleMarino
     extends AnimaleAcquatico {
  public AnimaleMarino(String s) {
     super(s); }
  public String vive() {
     return "in mare"; }
  public String chi_sei() {
     return "un animale marino"; }
}
```

Specializza i metodi **`vive()`** e **`chi_sei()`**, ma ***non definisce l'altro***
→ la classe è ancora astratta

LA TASSONOMIA COMPLETA



LE CLASSI “CONCRETE”

```
public class PesceDiMare
     extends AnimaleMarino {
 public PesceDiMare(String s) {
     super(s);
     verso = "non fa versi"; }
 public String chi_sei() {
     return "un pesce (di mare)"; }
  public String si_muove() {
     return "nuotando"; }
}
```

Definisce l’ultimo metodo astratto rimasto,
`si_muove()` → la classe non è più astratta

LE CLASSI “CONCRETE”

```
public class Uccello
     extends AnimaleTerrestre {
  public Uccello(String s) {
     super(s);
     verso="cinguetta"; }
  public String si_muove() {
     return "volando"; }
  public String chi_sei() {
     return "un uccello";}
  public String vive() {
     return "in un nido su un albero";
  }
}
```

LE CLASSI “CONCRETE”

```
public class Bipede
     extends AnimaleTerrestre {
  public Bipede(String s) { super(s); }
  public String si_muove() {
     return "avanzando su 2 zampe";
  }
  public String chi_sei() {
     return "un animale con due zampe";
  }
}
```


LE CLASSI “CONCRETE”

```
public class Quadrupede
     extends AnimaleTerrestre {
  public Quadrupede(String s) {
     super(s); }
  public String si_muove() {
     return "avanzando su 4 zampe"; }
  public String chi_sei() {
     return "un animale con 4 zampe"; }
}
```

ALTRE CLASSI PIÙ SPECIFICHE

```
public class Cavallo extends Quadrupede {
  public Cavallo(String s) {
     super(s); verso = "nitrisce"; }
  public String chi_sei() {
     return "un cavallo"; }
}
public class Corvo extends Uccello {
  public Corvo(String s) {
     super(s); verso = "gracchia"; }
  public String chi_sei() {
     return "un corvo"; }
}
```


ALTRE CLASSI PIÙ SPECIFICHE

```
public class Uomo extends Bipede {
  public Uomo(String s) {
     super(s); verso = "parla"; }
  public String si_muove() {
     return "camminando su 2 gambe"; }
  public String chi_sei() {
     return "un homo sapiens"; }
  public String vive() {
     return "in condominio"; }
}
```

ALTRE CLASSI PIÙ SPECIFICHE

```
public class Pinguino extends Uccello {
  public Pinguino(String s) {
     super(s); verso = "non fa versi"; }
  public String chi_sei() {
     return "un pinguino"; }
  public String si_muove() {
     return "ma non sa volare"; }
}
public class Tonno extends PesceDiMare {
  public Tonno(String s) { super(s); }
  public String chi_sei() {
     return "un tonno"; }
}
```


UN MAIN… “mondo di animali”

```
public class MondoAnimale {
 public static void main(String args[]) {
  Cavallo  c = new Cavallo("Furia");
  Uomo     h = new Uomo("John");
  Corvo    w = new Corvo("Pippo");
  Tonno    t = new Tonno("Giorgio");
  Uccello  u = new Uccello("Gabbiano");
  Pinguino p = new Pinguino("Tweety");
  c.mostra();    h.mostra();
  w.mostra();    t.mostra();
  u.mostra();    p.mostra();
 }
}
```

UN MAIN… “mondo di animali”

.. e il suo output:

```
Furia, un cavallo, nitrisce, si muove
avanzando su 4 zampe e vive sulla
terraferma.

John, un homo sapiens, parla, si muove
camminando su 2 gambe e vive in un
condominio.

Pippo, un corvo, gracchia, si muove
volando e vive in un nido su un albero.

...
```


ALTERNATIVA: UNO “ZOO”

```
public class Zoo {
 public static void main(String args[]) {
  Animale fauna[] = new Animale[6];
  fauna[0] = new Cavallo("Furia");
  fauna[1] = new Uomo("John");
  fauna[2] = new Corvo("Pippo");
  fauna[3] = new Tonno("Giorgio");
  fauna[4] = new Uccello("Gabbiano");
  fauna[5] = new Pinguino("Tweety");
  for(int i=0; i<6; i++)
            fauna[i].mostra();
 }
}
```

POLIMORFISMO: per ogni animale viene invocato **lo specifico metodo `mostra()`**