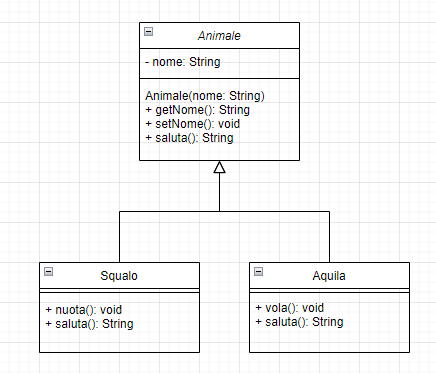
**Una gerarchia di animali**

****

Per **POLIMORFISMO** si intende:

Per **DYNAMIC BINDING** si intende:

Implementa in Java la gerarchia di classi mostrata nello schema UML riportato sopra. Riferendoti al codice che hai scritto, considera il seguente esempio (scrivi un main per provarlo):

*Animale a = new Squalo();*

*System.out.println(a.saluta());*

Nella sua dichiarazione la variabile **a** è di tipo **Animale** (tipo …………….. ), ma in fase di esecuzione ciò che si guarda non è il tipo dichiarato per la variabile, ma l’oggetto che è effettivamente referenziato da essa (tipo …………………). Eseguendo il codice ti accorgi che parte la variante di saluta() definita nella classe …………………. Dunque, possiamo utilizzare oggetti di classi derivate come fossero oggetti della classe base.

Considera ora il seguente esempio:

*Aquila v = new Aquila(“Giovanna”);*

*Animale a = v;*

*System.out.println(a.saluta());*

Il codice funziona/non funziona perché …………………………………………………………………………………..

Il tipo statico di **a** è: …………………………………………………………………………………………………………

Il tipo dinamico di **a** è: ………………………………………………………………………………………………………

Il metodo saluta() che viene eseguito è quello implementato nella classe: ………………………………………….

Considera ora il codice seguente:

*Aquila v = new Aquila(“Regina”);*

*Animale a = v;*

*a.vola();*

Il codice funziona/non funziona perché ………………………………………………………………………………...

Considera ora il codice seguente:

Animale a = new *Aquila*(“Regina”);

*Aquila* v = (*Aquila*)a;

v.vola();

Il tipo …………….. di **a** è Animale, il tipo ……………… di **a** è Aquila, perciò alla riga 2 è stato effettuato un casting esplicito, cioè abbiamo detto al compilatore che siamo sicuri che l’animale **a** è un’aquila, quindi, possiamo invocare sicuramente il metodo vola().

Infine, considera il seguente codice

Animale a = new Squalo();

Aquila q = (Aquila)a;

q.vola();

Il codice ti dà errore?

Se provi ad eseguirlo cosa succede? Perché?