Unità C2 – Ereditarietà e polimorfismo	**MODULO T1**

(A) CONOSCENZA TERMINOLOGICA

Dare una breve descrizione dei termini introdotti:

- Ereditarietà singola
- Ereditarietà multipla
- Classe derivata
- Ereditarietà
- Superclasse
- Sottoclasse
- Riusabilità
- Generalizzazione
- Specializzazione
- Aggregazione
- Scomposizione
- Upcasting
- Downcasting
- Polimorfismo

(B) CONOSCENZA E COMPETENZA

Rispondere alle seguenti domande producendo anche qualche esempio

B1) Conoscenza

1. Cosa vuol dire che l'ereditarietà consente un alto grado di *riuso del software*?
2. Cosa è una *gerarchie di classi*?
3. Cosa vogliono dire i termini *generalizzazione* e *specializzazione*, in una gerarchia di classi?
4. Cosa vuol dire *ereditarietà pubblica* e *privata*?
5. Quali sono i vantaggi tipici dell'*ereditsrietà*?
6. Cosa vuol dire che una classe *eredita* da una superclasse ?
7. In cosa consiste l'*overriding* ?
8. In cosa consiste l'estensione per *espansione* ?
9. Cosa è una *classe astratta* e qual è il suo utilizzo ?
10. In una gerarchia di classi, che relazione c'è tra una *superclasse* e le *sottoclassi* ?

B2) Competenza

1. Come si può, attraverso l'ereditarietà, ottenere il *riuso del software*?
2. Come si rappresenta una *gerarchie di classi*?
3. In quali casi può necessitare di usare l'*ereditarietà privata*?
4. Come si può fare in modo che il *downcasting* avvenga correttamente?
5. In quali casi può necessitare di usare *classi astratte*?
6. Perché e con quale simbologia si introducono i membri *protected* ?
7. In quali modi si può realizzare il *casting* tra le classi di una gerarchia ?

(C) ESERCIZI DI COMPRENSIONE

1. L'ereditarietà consente il ……. del software, in quanto è possibile ……. o ……. i membri e consente la creazione di gerarchie di ……..
2. In una ……. di classi, la creazione e la distruzione di oggetti segue un ordine preciso: in particolare, la creazione di un oggetto invoca prima il costruttore della classe ….. e poi quello della classe ……., mentre quando un oggetto viene distrutto, viene invocato prima il distruttore della classe …….. e poi quello della classe ……..
3. Il *casting* tra classi si distingue in ……, quando si tratta di conversione verso la superclasse e in …….., quando si tratta di conversione verso una sottoclasse. Mentre la conversione ……è sempre possibile, per il ……. occorre creare l'oggetto mediante il costruttore della classe ……. e poi effettuare la conversione.
4. Il polimorfismo si presenta come …… oppure come ……: nel primo caso si hanno più metodi omonimi, nella stessa classe, con firma ……., mentre nel secondo caso abbiamo più metodi omonimi in classi diverse della stessa …….. di classi.
5. Gli specificatori di accesso possono essere: …….. se si vogliono rendere pubblici i membri, ……. se li si vogliono rendere privati oppure …….. se devono essere privati, ma ereditabili.
6. Associare le parole chiave di sinistra con le corrispondenti frasi sulla destra:

1	☐	public	**A**	Massima visibilità
2	☐	private	**B**	Visibilità solo nella gerarchia di classi
3	☐	protected	**C**	Minima visibilità

7. Associare le parole chiave di sinistra con il corrispondente simbolo sulla destra:

1	☐	public	**A**	–
2	☐	private	**B**	#
3	☐	protected	**C**	+

8. Completare laseguentetabella, indicando, per ciascuna delle caratteristiche indicate, a quale proprietà della OOP fa riferimento:

	Incapsulamento	**Ereditarietà**	**Polimorfismo**
Overloading			
Overriding			
Gerarchia di classi			
Information hiding			
Downcasting			
Generalizzazione			
Upcasting			
Protezione dei dati			

(D) ESERCIZI DI APPLICAZIONE

1. (**Esercizio risolto**). Data la classe *MezziTrasporto*, derivare da questa le classi *VeicoloAMotore* e *Autobus*, indicandone i membri essenziali e disponendole in un'opportuna gerarchia.
   Si puo supporre che la classe MezziTrasporto abbia attributi velocità massima, tipo di trazione (animale, muscolare, motore) e numero di persone massimo trasportabili.
   La classe VeicoloAmotore, eredita tutti i membri di MezziTrasporto, ma prevede anche il consumo e il tipo di alimentazione (benzina, diesel, kerosene). In questo caso come metodi possiamo prevedere l'accensione e lo spegnimento e la possibilità di accelerare e frenare.
   Infine, la classe Autobus, aggiunge gli attributi relativi al costo del biglietto e al numero di posti sia in piedi che a sedere. I metodi da aggiungere serviranno per aprire e chiudere le porte e, ad esempio, il metodo per prenotare la fermata.
   Una possibile UML è mostrata in **Fig. 1**. La facile implementazione è lasciata al lettore.
2. Progettare e implementare la classe *ContoBancario*, contenente attributi *NumeroConto*, *Titolare*, *Saldo*, fornendo i metodi necessari. Successivamente, derivare da questa la classe *ContoBancarioFruttifero* che contiene come dati aggiuntivi *TassoInteresse* e *InteressiMaturati* e tutte le funzionalità per il calcolo degli interessi. Rappresentare la situazione gerarchica mediante UML.
3. Date le seguenti classi: *Veicolo*, *Bicicletta*, *Motocicletta*, *Automobile*, *VeicoloAMotore*:
   a. disporle in una gerarchia, rappresentandole mediante UML e prevedendo attributi e metodi opportuni;
   b. stabilire quali membri vanno estesi per ridefinizione;
   c. stabilire quali membri vanno estesi
   d. per espansione.
3. Date le classi *Televisore* e *Video*, rappresentarle mediante la UML, prevedendo gli attributi e i metodi opportuni, e disporle in modo gerarchico, evidenziando eventuali membri ridefiniti e/o espansi.
4. Definire la classe *Motore* con attributi e metodi opportuni e derivarne la classe *MotoreDiesel* ridefinendo e/o espandendo i membri. Rappresentare la situazione gerarchica mediante UML.
5. Date le classi *Triangolo*, *TriangoloIsoscele* e *TriangoloEquilatero*:.
   a. disporle in una gerarchia rappresentandole mediante UML e prevedendo attributi e metodi opportuni;
   b. stabilire quali membri vanno estese per ridefinizione;
   c. stabilire quali membri vanno estese per espansione.
6. Definire la classe *OrdinaProdotto* con attributi e metodi opportuni, e derivare da questa la classe *SpedizioneProdotto* aggiungendo gli attributi *nomeCorriere*, *dataSpedizione* e *dataConsegna*. Derivare la classe *Fattura* con attributo *aliquotaIVA* ed un metodo per la stampa dell'importo totale. Rappresentare la situazione gerarchica mediante UML.
7. Descrivere il grafo di gerarchia delle seguenti classi: Mammiferi, Pesci, Uccelli, Anfibi, Rettili, Canidi, Felidi, Vertebrati, Invertebrati.
8. Progettare la classe Quadrato, che rappresenti un quadrato nel piano cartesiano. Gli oggetti di questa clesse sono contraddistinti dalla misura del lato, dalla posizione e dal colore. Definire i metodi opportuni per la gestione degli oggetti di questa classe. In particolare, scrivere le istruzioni per:
   a. creare due quadrati di lato rispettivamente 5 e 9;
   b. spostare uno dei due quadrati in una posizione data;
   c. calcolare e stampare il perimetro e l'area delle due figure.

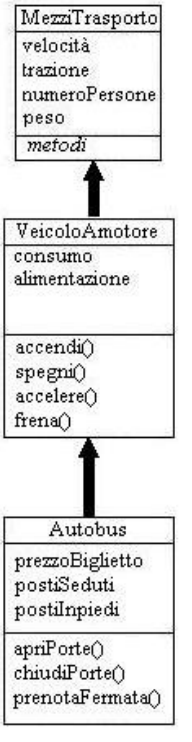

Fig.*1 UML della gerarchia MezziTrasporto*