Unità C1 – Incapsulamento e tecniche OOP	**MODULO T1**

(A) CONOSCENZA TERMINOLOGICA

Dare una breve descrizione dei termini introdotti:

- Variabili di classe
- Variabili istanza
- Metodo di classe
- Metodologia OOA
- Metodologia OOD
- *Information hiding*
- Protezione dei dati
- Astrazione sui dati
- Astrazione sulle procedure
- Mascheramento dell'informazione
- Indipendenza degli oggetti
- Riutilizzo del software

(B) CONOSCENZA E COMPETENZA

Rispondere alle seguenti domande producendo anche qualche esempio

B1) Conoscenza

1. Cosa significa *protezione sui dati*?
2. Cosa significa che l'incapsulamento consente *astrazione sui dati*?
3. Cos'è *l'astrazione procedurale*?
4. Qual è l'utilità del *mascheramento dell'informazione*?
5. Cosa è la *riusabilità del software*?
6. Come si possono classificare le variabili?
7. In cosa differiscono i metodi **static** da quello non statici?
8. Cosa sono le metodologie di sviluppo **OOD** e **OOA**?

B2) Competenza

1. Come e perché l'incapsulamento offre *protezione sui dati*?
2. In quale situazione si presenta *l'astrazione procedurale*?
3. In quale modo l'incapsulamento consente la *riusabilità del software*?
4. In quali casi conviene dichiarare *attributi di classe*?

(C) ESERCIZI DI COMPRENSIONE

1. L'incapsulamento conferisce …… dei dati in quanto l'utente non può accedere direttamente ai membri ….., ma lo può fare solo attraverso i metodi …….
2. L'astrazione sui dati è una caratteristica che consente al programmatore di ignorare come i dati sono rappresentati …….. in memoria, ma di curare solo la loro rappresentazione ……. Per lo stesso motivo, possiamo ignorare anche l'occupazione in ……. dei dati di vario tipo.
3. Le variabili possono essere …….., quando sono dichiarate all'interno di un metodo e sono utilizzabili solo da questo, possono essere variabili …….., che indicano gli attributi di una classe, oppure possono essere variabili di classe, dette anche ……, che sono utilizzate come attributi della classe che hanno sempre lo stesso valore per tutti gli oggetti generati.
4. Nella Fase 1, detta **OOA** (……), occorre svolgere l'analisi del ……. e individuare le …… che servono a risolvere il problema; se possibile, si possono ……. classi preesistenti e già collaudate.
5. Associare le proposizioni di sinistra con le corrispondenti sulla destra:

1	Variabili di classe	**A**	Attributi
2	Variabili istanza	**B**	Visibilità solo in una funzione
3	Variabili locali	**C**	Visibilità in tutta l'applicazione
4	Variabili globali	**D**	Variabili statiche

6. Per ciascuna delle seguenti frasi, indicare se Vera o Falsa.

	Vero	**Falso**
Una variabile locale può essere usata in tutta l'applicazione		
Variabile di classe e variabile istanza sono la stessa cosa		
Una variabile di classe deve sempre essere inizializzata		
Un metodo può avere oggetti come parametri		
Un metodo può ritornare un oggetto		
Un metodo statico ha un parametro in meno di quello non statico		
Un metodo statico ha un parametro in più di quello non statico		

7. Associare le proposizioni di sinistra con quelle sulla destra, scrivendo la lettera corrispondente nelle caselle.

1	☐	Astrazione sui dati	**A**	Specificatori di accesso
2	☐	Protezione sui dati	**B**	Ignorare implementazione delle funzioni
3	☐	Astrazione procedurale	**C**	Oggetti usabili in contesti diversi
4	☐	Mascheramento delle informazioni	**D**	Considerare funzioni come scatole nere
5	☐	Indipendenza degli oggetti	**E**	Ignorare rappresentazione fisica
6	☐	Riutilizzo del software	**F**	Usare stesso software in altri contesti

(D) ESERCIZI DI APPLICAZIONE

1. Creare una classe *Angolo*, che preveda i metodi per:
   a. addizionare due angoli
   b. sottrarre due angolizione()
   c. moltiplicare un angolo per un numero
   d. dividere un angolo per un numero
2. Progettare una classe *Complex* che consenta di operare con numeri complessi, rappresentati da una coppia di numeri reali, fornendo le operazioni seguenti:
   a. addizione()
   b. sottrazione()
   c. moltiplicazione()
   d. divisione()
   e. modulo()
   f. coniugato()

   Mostrare come la classe progettata possa essere applicata al calcolo della radice di una equazione di primo grado a coefficienti complessi del tipo a*x + b =c
3. Progettare una classe *Razionale* che consenta di operare con frazioni, rappresentate da una coppia di numeri reali, fornendo le operazioni seguenti:
   a. addizione()
   b. sottrazione()
   c. moltiplicazione()
   d. divisione()
   e. uguali()
   f. setRazionale()
   g. getRazionale().

   Mostrare come la classe progettata possa essere applicata al calcolo della soluzione di un sistema lineare di due equazioni in due incognite, a coefficienti razionali.
4. Progettare una classe *Polinomio* che consenta di operare con polinomi di grado MAX, in una variabile a coefficienti interi, rappresentati dalla coppia (*grado*, *array dei coefficienti*), fornendo le seguenti operazioni:
   a. addizione()
   b. sottrazione()
   c. moltiplicazione()
   d. divisione()
   e. valore()
   f. setPolinomio()
   g. getPolinomio()
   h. getGrado()

   Mostrare come la classe progettata possa essere applicata al calcolo del polinomio P3, MCD tra 2 polinomi P1 e P2 dati da input.
5. Implementare la classe *Libro* con attributi e metodi relativi.
6. Implementare la classe *Punto*, che rappresenta un punto sul piano e fornire i metodi costruttori con parametri e senza parametri, il metodo Distanza() ed il metodo Pmedio().
7. Implementare la classe *Data*, che rappresenta una data con attributi giorno, mese ed anno. Fornire metodi per operare sulle date.