Unità B2 – Significato e proprietà della OOP	**MODULO T1**

(A) CONOSCENZA TERMINOLOGICA

Dare una breve descrizione dei termini introdotti:

- Diagramma di sequenza
- Ciclo di vita di un oggetto
- Java Virtual Machine
- Garbage collector
- Binding statico e dinamico
- Polimorfismo
- Incapsulamento
- Ereditarietà
- Information hiding
- Derivazione
- Gerarchia di classi
- Overloading
- Overriding
- Sottoclasse o classe derivata

(B) CONOSCENZA E COMPETENZA

Rispondere alle seguenti domande producendo anche qualche esempio

B1) Conoscenza

1. Perché un'applicazione *object priented* consiste in una serie di scambi di messaggi tra oggetti?
2. Quali sono i principali vantaggi dell'*incapsulamento*?
3. Quali sono i principali vantaggi dell'*ereditarietà*?
4. Quali sono i principali vantaggi del *polimorfismo*?
5. In quali forme si può presentare il *polimorfismo*?

B2) Competenza

1. Come si disegna e come si interpreta un *diagramma di sequenza*?
2. Come si rappresenta e come si interpreta una *gerarchia di classi*?
3. Come si descrive un programma OOP visto come *processo*?
4. Come si descrive il *ciclo di vita di un oggetto*?
5. Come si descrive la *comunicazione tra oggetti*?

(C) ESERCIZI DI COMPRENSIONE

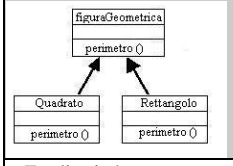
1. Un programma ad oggetti può essere visto come uno scambio di ……. tra oggetti. Distinguiamo un oggetto …….., che invia il messaggio, un oggetto …….. destinatario che lo riceve, e il messaggio stesso che in genere è rappresentato da un …….. con eventuali parametri.
2. La proprietà della OOP che consente di vedere un oggetto come un'entità autonoma, che può essere usata soltanto tramite la sua ……... prende il nome di ……
3. La proprietà della OOP che consente di ……. un oggetto software, eventualmente ridefinendolo o espandendolo, senza doverlo riscrivere, prende il nome di …….
4. La proprietà della OOP che consente di avere metodi con …….. nome, ma che svolgono compiti diversi, prende il nome di ……. . In particolare, se i metodi omonimi sono nella stessa classe, si parla di …….., mentre s si trovano in classi diverse della stessa ……., prende il nome di ……..
5. I principali vantaggi dell'incapsulamento sono: ……..dei dati, …….. sui dati, …… sulle procedure, …… …….. (detta anche mascheramento dell'informazione), indipendenza degli oggetti e riutilizzo del software.
6. I principali vantaggi dell'ereditarietà sono: possibilità di riutilizzo del ……., aggiunta di nuovo membri (…… del software) e …… del software.
7. Per rappresentare lo scambio di …… tra oggetti, si usa il diagramma di ……., in cui ogni messaggio è caratterizzato da un …….., che ne indica la progressione, dal nome del ……. richesto, eventualmente con parametri …… e una freccia che va dall'oggetto …… all'oggetto …… .
8. Durante l'esecuzione di un'applicazione, gli oggetti vengono …..….. e …….. mediante una tecnica di uso della memoria che si dice …… …….. Il periodo di tempo, durante l'esecuzione dell'applicazione, in cui l'oggetto viene …….. si dice ……..... dell'oggetto.
9. Quando un oggetto software riceve un messaggio, allora esegue l'operazione che gli viene richiesta, in modo parametrico rispetto ai dati specificati. Il processo si rappresenta con lo schema grafico seguente (**diagramma di sequenza**)
10. Associare la proprietà indicata a sinistra, con la frase scritta a destra, indicando la corrispondente lettera nella casella.

1		Ereditarietà	**A**	Classe derivata
2		Polimorfismo	**B**	Metodi omonimi nella stessa classe
3		Incapsulamento	**C**	Stesso codice usato in contesti diversi
4		Overloading	**D**	Metodi omonimi in classi diverse
5		Overriding	**E**	Information hiding

11. Indicare, per ciascuna delle seguenti affermazioni, se sia vera o falsa.

	Vero	**Falso**
In Java, l'allocazione dinamica della memoria spetta al programmatore		
Il binding statico è tipico dei linguaggi compilati		
Il binding dinamico è tipico dei linguaggi compilati		
L'*information hiding* è consentita dalla proprietà di incapsulamento		
L'*information hiding* è consentita dalla proprietà di ereditarietà		
Il polimorfismo consente la creazione di gerarchie di classi		
Il polimorfismo consente il riuso del software		

12. Quali proprietà presenta la gerarchia mostrata in figura?



	Vero	**Falso**
Ereditarietà		
Incapsulamento		
Polimorfismo		
Overloading		
Overriding		
Binding statico		
Binding dinamico		

13. Considerando la classe *Punto* seguente, individuare gli errori e correggerli:

```
class Point
{
    private int x, y;
    public …… . () {int = 0, int = 0 ;}
    public ……   ( int a, int b ) {setPoint ( …., …..); }
    public ……. getX() { return …..; }
    public ……. getY() { ……. y; }
    public …….setPoint (int a, int b )
    {
        ……..
        ……..
    }
}
```

(D) ESERCIZI DI APPLICAZIONE

1. Con riferimento alla classe *Angolo* dell'Unità precedente, mediante un apposito diagramma di sequenza, mostrare la sequenza di messaggi scambiati fra gli oggetti di un'applicazione che legga da input il valore di un angolo e ne stampi la sua conversione in radianti.
2. Con riferimento alla classe *Razionale* dell'Unità precedente, mediante un apposito diagramma di sequenza, mostrare la sequenza di messaggi scambiati fra gli oggetti di un'applicazione che acquisisca da input un oggetto di classe *Razionale* e ne stampi la sua conversione in numero reale.
3. Con riferimento alla classe *Polinomio* dell'Unità precedente, mediante un apposito diagramma di sequenza, mostrare la sequenza di messaggi scambiati fra gli oggetti di un'applicazione che acquisisca da input un oggetto di classe *Polinomio* e ne stampi il valore.
4. Progettare la classe *Conto*, che rappresenta un conto corrente bancario, con i metodi per effettuare versamenti, prelievi e saldo. Successivamente, progettare una classe *Bancomat* che consenta all'utente di interagire con la classe *Conto*.
5. (**Array di oggetti**) Generalizzare l'esercizio precedente, simulando la gestione di una serie di clienti della banca.

(E) ESERCITAZIONI PRATICHE

Esercitazione n. 1

Obiettivi: progettare una classe, rappresentare in UML gli attributi e i metodi, con interfacce e specificatori di accesso; derivare da questa, altre classi, sfruttando l'ereditarietà.
Problema: si vuole progettare una classe che rappresenti un telefono, considerando come attributi minimi *numeroTasti* e *Tariffa*, e come metodi *Squilla*(…) e *componiNumero*(..). Successivamente, derivare da questa classe le classi per rappresentare un telefono cellulare e cordless.

(E) ESERCITAZIONI PRATICHE

Esercitazione n. 2

Obiettivi: implementazione di classi, applicazione dell'ereditarietà, utilizzo dell'overriding.
Problema: progettare la classe *Point* ed ereditare da questa la classe *Circle*, con i relativi membri. Da *Circle* ereditare, successivamente, *Cylinder*. Testare in modo adeguato le classi realizzate.