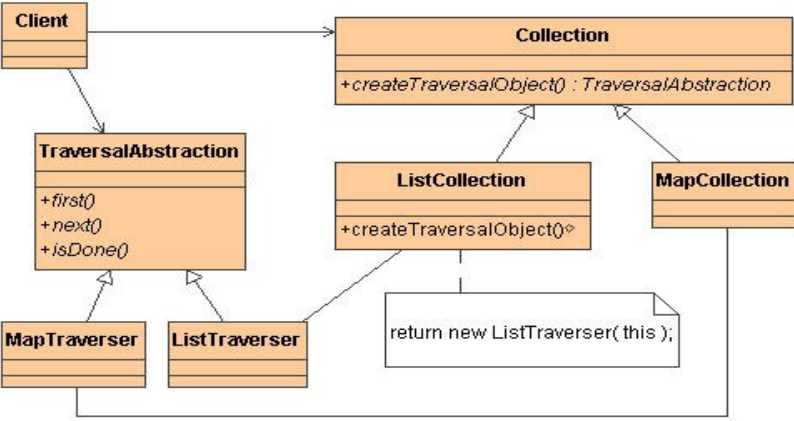
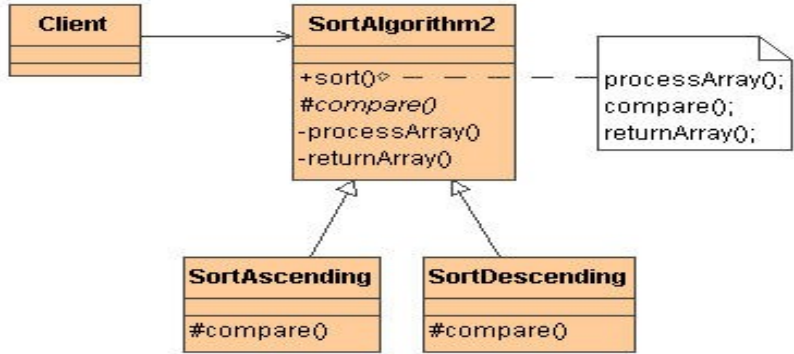
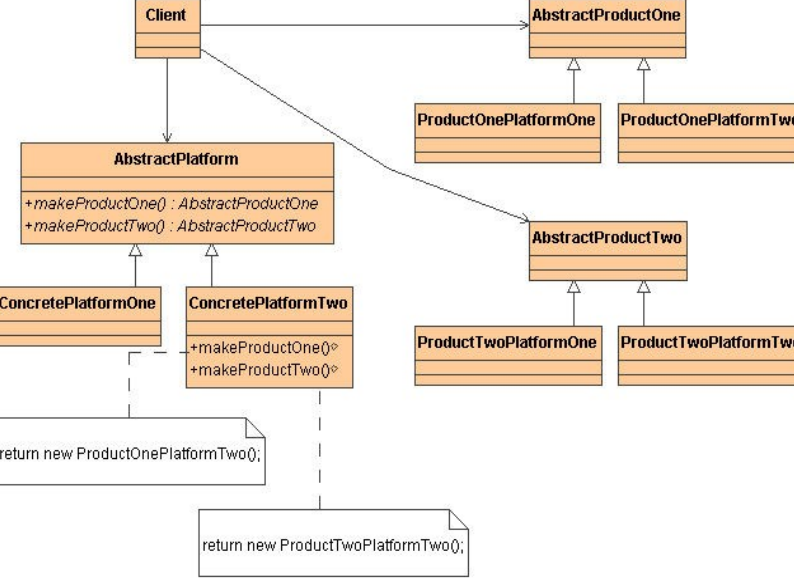


Testo e soluzioni

 <pre> classDiagram class Client class TraversalAbstraction { +first() +next() +isDone() } class Collection { +createTraversalObject() TraversalAbstraction } class ListCollection { +createTraversalObject() } class MapCollection class MapTraverser class ListTraverser Client --> Collection Client --> TraversalAbstraction TraversalAbstraction < -- MapTraverser TraversalAbstraction < -- ListTraverser Collection < -- ListCollection Collection < -- MapCollection ListCollection --> ListTraverser ListCollection --> MapCollection ListCollection --> ListCollection : return new ListTraverser(this); </pre> <p>n° 16 _</p>	<p>1) Punti 4/30</p> <p>A quali <i>pattern</i> si riferiscono i tre schemi a fianco riportati ?</p> <p>Ricavare la numerazione dalla lista che segue:</p> <ol style="list-style-type: none"> 1 Abstract Factory 2 Builder 3 Factory Method 4 Prototype 5 Singleton 6 Adapter 7 Bridge 8 Composite 9 Decorator 10 Facade 11 Flyweight 12 Proxy 13 Chain of Responsibility 14 Command 15 Interpreter 16 Iterator 17 Mediator 18 Memento 19 Observer 20 State 21 Strategy 22 Template Method 23 Visitor
 <pre> classDiagram class Client class SortAlgorithm2 { +sort() #compare() -processArray() -returnArray() } class SortAscending { #compare() } class SortDescending { #compare() } Client --> SortAlgorithm2 SortAlgorithm2 < -- SortAscending SortAlgorithm2 < -- SortDescending SortAlgorithm2 --> SortAlgorithm2 : processArray(); compare(); returnArray(); </pre> <p>n° 22 _</p>	
 <pre> classDiagram class Client class AbstractPlatform { +makeProductOne() AbstractProductOne +makeProductTwo() AbstractProductTwo } class ConcretePlatformOne class ConcretePlatformTwo { +makeProductOne() +makeProductTwo() } class AbstractProductOne class ProductOnePlatformOne class ProductOnePlatformTwo class AbstractProductTwo class ProductTwoPlatformOne class ProductTwoPlatformTwo Client --> AbstractPlatform Client --> AbstractProductOne Client --> AbstractProductTwo AbstractPlatform < -- ConcretePlatformOne AbstractPlatform < -- ConcretePlatformTwo AbstractProductOne < -- ProductOnePlatformOne AbstractProductOne < -- ProductOnePlatformTwo AbstractProductTwo < -- ProductTwoPlatformOne AbstractProductTwo < -- ProductTwoPlatformTwo ConcretePlatformOne --> ProductOnePlatformTwo : return new ProductOnePlatformTwo(); ConcretePlatformTwo --> ProductTwoPlatformTwo : return new ProductTwoPlatformTwo(); </pre> <p>N° 1 _</p>	

Testo e soluzioni

2) **Punti 2/30** "UML: which of the following statements is NOT true of a UML interface?"

- "An interface is a class which has no attributes or direct instances."
- "An interface can be represented by a circle."
- **"Interfaces can participate in associations that are navigable away from the interface."**
- "Interfaces can be realized by more than one class."

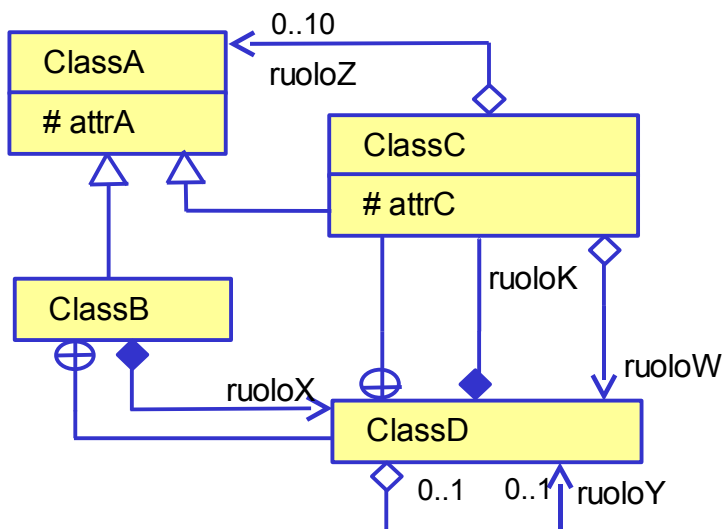
3) **Punti 2/30** "UML: on an activity diagram a fork is used when...?"

- **"Two or more concurrent paths can be followed."**
- "A decision must be made between two different paths."
- "When a number of incoming transitions must be united into a single outgoing transition."
- "To ensure that lines don't cross on the diagram."

4) **Punti 2/30** Patterns: the analysis patterns are used ?

- "To make the system design more efficient."
- "To improve the collaboration between system objects."
- "To impose control on the system objects."
- **"To identify typical solutions to business problems."**
- "Because the system lacks the control."
- "To create models of an accounting information system."

5) **Punti 8/30** Dire se è possibile tradurre il seguente schema UML delle classi in dichiarazioni in linguaggio Java o C++, e in caso affermativo, scrivere le dichiarazioni Java o C++ corrispondenti e creare una istanza di diagramma degli oggetti; in caso negativo, descrivere i difetti contenuti nel diagramma e suggerire delle possibili correzioni.



Testo e soluzioni

/ Nello schema presentato nel testo dell'elaborato vi sono alcuni piccoli difetti, ma tali da non precludere la possibilità di generare il codice relativo allo schema: manca il tipo degli attributi, mancano alcuni indicatori di visibilità, la composizione di "ruoloK" non ha indicatore di navigabilità. Si potrebbe pensare che lo schema è in fase di analisi, e non ancora una progettazione definitiva. Le dichiarazioni usando il linguaggio C++, che risulta essere più espressivo in questo caso, sono le seguenti:*

**/*

```
class ClassA
{
    protected:
        int attrA;           // si assume int in mancanza di specifiche
}; //ClassA

class ClassB: public ClassA // classe derivata, come da progetto (public è arbitrario)
{
    class ClassD // classe annidata, come da progetto
    {
        class ClassC: public ClassA
        { protected:
            int attrC;

            public: // l'assunzione della visibilità pubblica è arbitraria
                ClassA * ruoloZ[10];           // variabile da 0 a 10 oggetti (referenze)
                ClassD * ruoloW;           // aggregazione: referenza

        }; //ClassC

        ClassC ruoloK;           // composizione: valore
        ClassD * ruoloY;       // aggregazione: referenza

    }; //ClassD

    ClassD ruoloX; // composizione: valore

}; //ClassB
```

Testo e soluzioni

6) **Punti 3/30** Indicare le differenze (natura, finalità, collocazione) che intercorrono tra le attività di verifica e quelle di validazione

- **verifica:** risponde alla domanda: “*did I build the [system/product/item] right?*”, ovvero si occupa di come sia stata svolto un processo o anche solo una parte di esso; intende accertare che l'esecuzione di esso non abbia introdotto errori; la verifica si effettua in modo sia a priori (tramite la rigorosa applicazione di norme standard) che a posteriori (tramite l'attuazione di forme di verifica).
- **validazione:** risponde alla domanda: “*did I build the right [system/product/item]?*”, ovvero si occupa di cosa sia stato prodotto da un processo; intende accertare che il prodotto corrisponda alle attese; la validazione si effettua sempre all'uscita di un processo, in relazione ai suoi ingressi.

7) **Punti 3/30** Una azienda informatica asserisce di applicare ai suoi processi produttivi lo standard ISO/IEC 12207. Si tratta di una affermazione corretta? Spiegare brevemente la risposta.

L'affermazione è senz'altro scorretta. Lo standard in questione fissa le norme e i concetti generali che stanno alla base della definizione dei processi *software*. Questa visione generale non è direttamente applicabile. Per esserlo deve invece essere adattata (per istanziiazione) al contesto desiderato (dominio, azienda).

8) **Punti 3/30** Enumerare le fasi costitutive di un processo di sviluppo. Indicare la percentuale di impegno raccomandato per ciascuna in un progetto “normale”, insieme a una sintetica giustificazione di ciascun valore assegnato.

- Analisi (dei requisiti)
- Progettazione architetturale
- Progettazione di dettaglio
- Programmazione
- Verifica
- Validazione

Le percentuali sono a scelta (ragionata).

Da questa enumerazione sono escluse le attività provenienti da altri processi primari (p.es.: fornitura, manutenzione) e quelle implicite derivanti dall'istanziiazione di processi di supporto entro il processo di sviluppo (p.es.: documentazione).

9) **Punti 3/30** Fornire una definizione dei seguenti termini e ciò a cui essa corrisponde in un linguaggio di programmazione ad alto livello (specificando quale).

- **Modulo:** l'elemento atomico della progettazione (non della programmazione!); tipicamente una classe o un interfaccia.
 - **Unità:** un insieme coeso di moduli, appaltabili in realizzazione a un singolo programmatore; non ha sempre un corrispondente diretto in un linguaggio di programmazione.
 - **Componente:** un insieme di unità funzionalmente coese; in Java, rappresentabile da uno o più *package*.
-