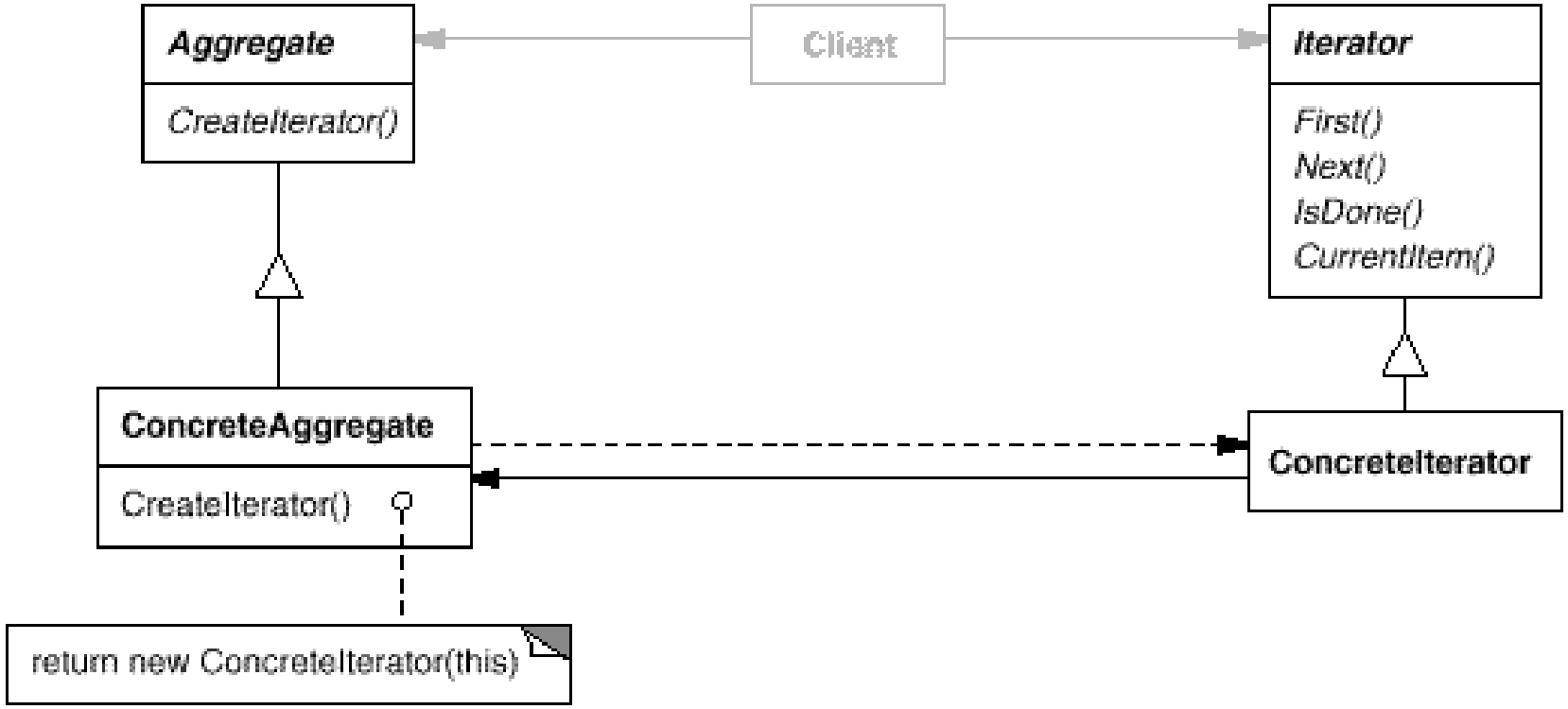
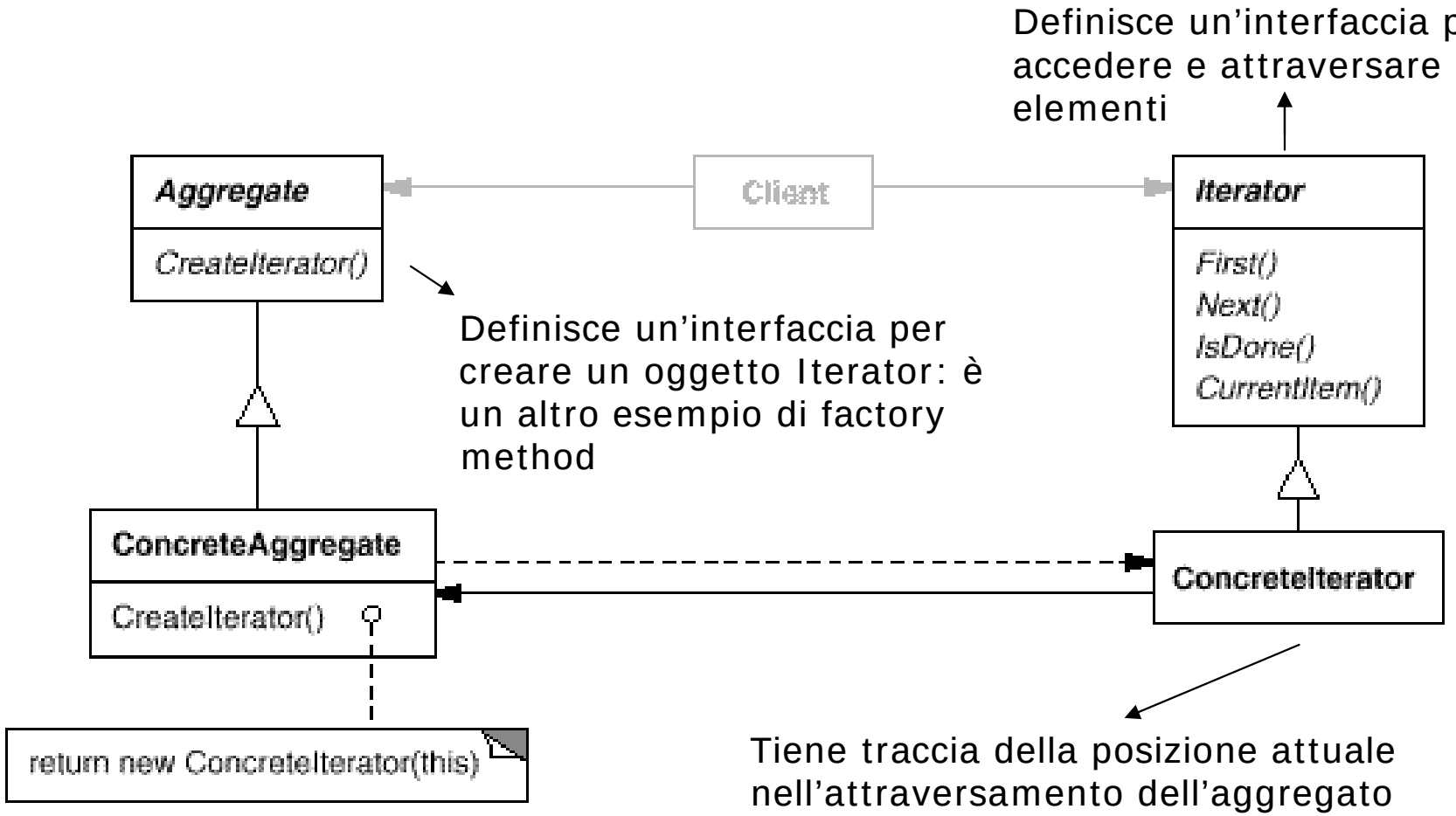
Iterator (object / behavioral)

- Fornisce un modo per accedere agli elementi di un oggetto aggregato in maniera **sequenziale**, senza esportarne la rappresentazione interna
- Idea: togliere le funzionalità di accesso e attraversamento dall'aggregato e inserirle all'interno di un oggetto **iterator**
- Un oggetto iterator è responsabile di tener traccia dell'elemento corrente
  - Conosce quali elementi sono già stati attraversati
  - E' possibile utilizzare contemporaneamente più iteratori sulla stessa struttura dati
- Per es., data una lista, l'iterator:
  - Fornisce l'accesso ai suoi elementi senza esportare la sua interfaccia
  - Evita di appesantire l'interfaccia della lista con operazioni per i diversi tipi di attraversamento (ad. es. reverse o filtrato)
  - Svincola il codice del client dall'implementazione della str. dati

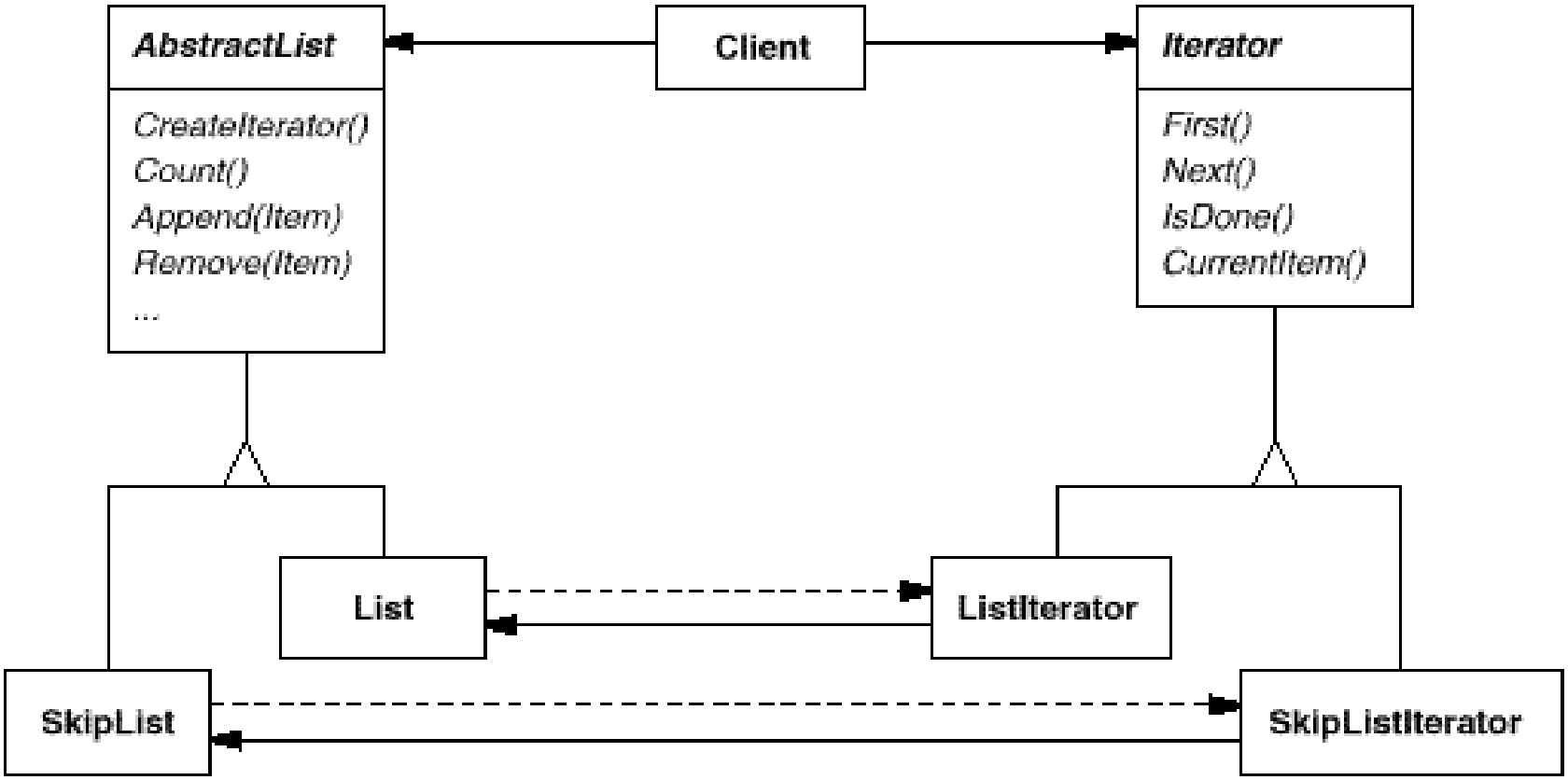
Iterator (ii)


Aggregate
CreateIterator()
Client
Iterator
First()
Next()
IsDone()
CurrentItem()
ConcreteAggregate
CreateIterator()
ConcreteIterator
return new ConcreteIterator(this)

Iterator (iii)



Iterator – Esempio



Iterator (iv)

- E’ possibile definire più *concrete iterator* che visitano lo stesso contenitore con politiche diverse
  - Approccio adatto soprattutto per gli alberi
- Ogni iterator deve conoscere i dettagli della classe che visita
- La STL del linguaggio C++ fa ampio uso del pattern iterator