Iterator

CSPP 51023

Qiyu Wang

Intent

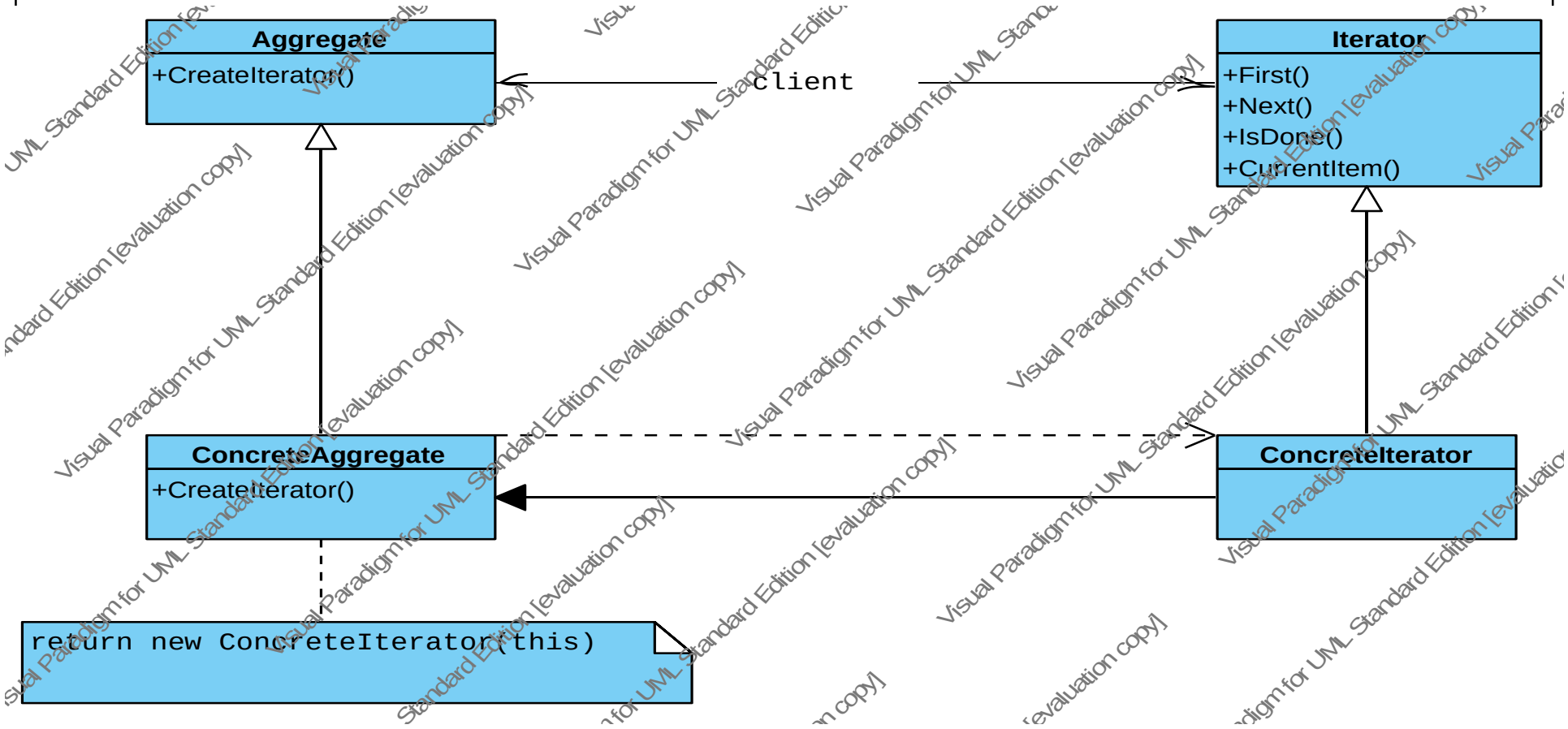
- Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation
- Also known as: Cursor

Applicability

- to access an aggregate object's contents without exposing its internal representation
- to support multiple traversals of aggregate objects
- to provide a uniform interface for traversing different aggregate structures(that is, to support polymorphic iteration)

Structure

- Participants: Iterator, ConcreteIterator, Aggregate, ConcreteAggregate



Code

```
class PrintNEmployee : public ListTraverser<Employee*>{
public :
     PrintNEmployees(List<Employee*>* aList, int n) :
          ListTraverser<Employee*>(alist),
          _total(n), _count(0) {   }

protected:
     bool ProcessItem(Employee* const&);
private:
     int_total:
     int_count;
};

Bool PrintNEmployees::ProcessItem (Employee* const& e){
    _count++;
    e->Print();
    return_count<_total;
}
```

Code

```
List<Employee*>* employees;
//…

PrintNEmployees pa(employee,10);
Pa.Traverse();

-----------------------------------------------------------------------------------------------------
ListIterator<Employee*> i(employees);
int count=0;

for(i.First(); !i.IsDone; i.Next()) {
   count++;
    i.CurrentItem() ->Print();

    if(count>=10){
     break;
     }
}
```

Code

- FilteringListTraverser

```
template <class Item>
class FilteringListTraverser{
public:
        FilteringListTraverser(List<Item>* aList);
        bool Traverse();
protected:
        virtual bool ProcessItem(const Item&) = 0;
        virtual bool TestItem(const Item&) = 0:
Private:
        ListIterator<Item> _iterator;
};

template <class Item>
void FilteringListTraverser<Item>::Traverse(){
for(
     _iterator.First();
     !_iterator.IsDone();
     _iterator.Next()
){
   if (TestItem(_iterator.CurrentItem()){
    result = ProcessItem(_iterator.CurrentItem());
              if(result==false){
              break;
        }
```

Concequence

- It supports variation in the traversal of an aggregate.
- Iterators simplify the Aggregate interface.
- More than one traversal can be pending on an aggregate.