Tecniche di Progettazione: Design Patterns

Esercitazione

Bridge

The following code skeleton defines a class hierarchy for Queue and Stack.

```
abstract class Dispatcher {
    object get() {/*return the first object*/}
    void pop() {/*remove the first object*/}
    abstract void put(Object o); /*add o to the data structure*/
}
class Queue extends Dispatcher{
    void put(Object o); /*append o after the last object of the queue*/
}
class Stack extends Dispatcher{
    void put(Object o); /*insert o before the first object of the stack*/
}
```

Bridge

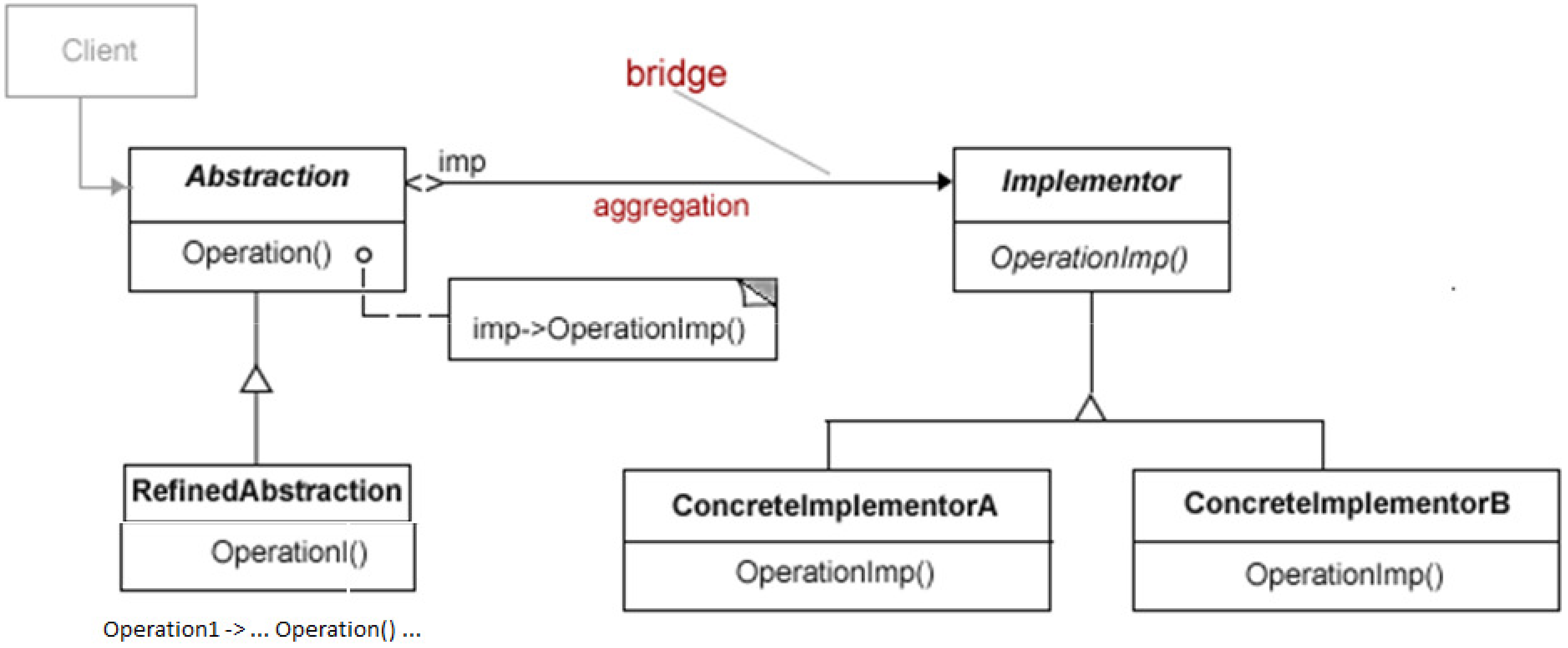
Use the Bridge pattern to implement the above class hierarchy.

You need to use Java ArrayList as the implementation.

You need to write the following code

- Code for get() and pop() methods and any additional code of Dispatcher class
- Code for put() method and any additional code of Queue class
- Code for put() method and any additional code of Stack class

Bridge Pattern structure



Problema mal posto

- Put() dovrebbe essere definita in termini di pop() e get().

- Come si modifica il caso di studio per applicare Bridge??
- Estendo Dispatcher con pick(){get();pop();}

- Similmente con le papere: ShowDuck extends Duck{
- Public void show {quack(); fly(); quack();}
- A dx le strategie di volo e quack.

Adapter (contrived exmple)

- Using object Adapter pattern to implement the above interfaces. You need to adapt Java ArrayList class. Note that you need to write three adapter classes:
  - DispatcherAdapter implements Dispatcher
  - QueueAdapter extends DisplatcherAdapter implements Queue
  - StackAdapter extends DispatcherAdapter implements Stack

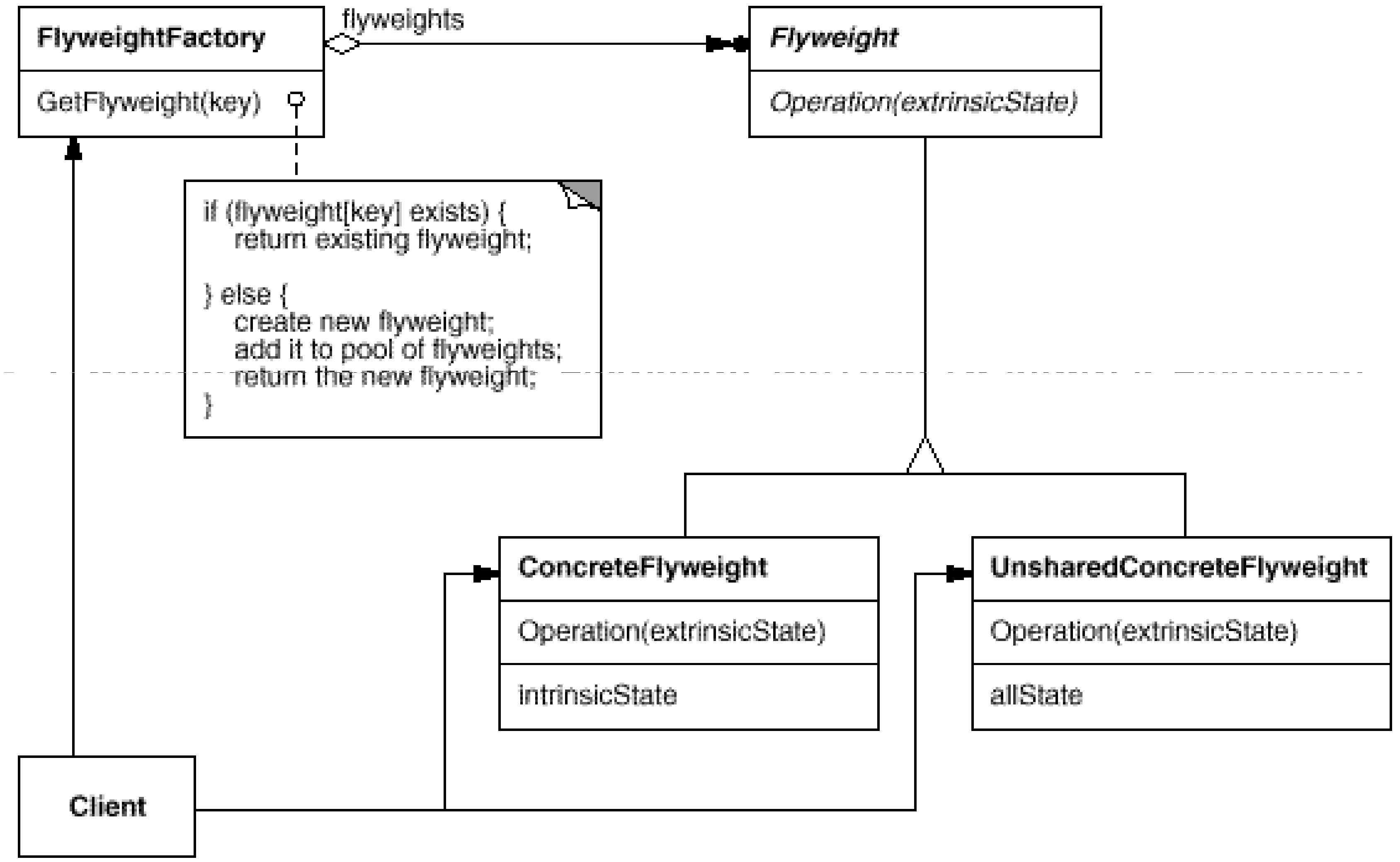
Iterators

- The given code (IteratorDoublyLinked folder) defines non-circularly double linked lists. Using Iterator pattern, write two external iterators for the double linked lists.
  - One iterates every from beginning to end, the other from end to beginning.
- Assume the iterator classes can access any member of DoubleLinkedList and Cell classes, but modification is not allowed.
- Remember:
  - The Iterator interface has two methods: hasNext(), next()
  - the collection must implement Iterator createIterator()
- Solution: folder IteratorDoublyLinked

Flyweight

- In Flyweight pattern, a Flyweight object has intrinsic state that cannot be changed. This also means that a Flyweight object cannot have any public-accessible *set()* method to set a new value for some instance variable of the object.
-
- Consider a variation of Flyweight pattern to allow a Flyweight object to have set() methods. When a set() method of a Flyweight object is called, the object becomes a non-Flyweight, non-shared object. This idea is similar to copy-on-write.

Flyweight


flyweights
FlyweightFactory
GetFlyweight(key)
Flyweight
Operation(extrinsicState)
if (flyweight[key] exists) {
return existing flyweight;
} else {
create new flyweight;
add it to pool of flyweights;
return the new flyweight;
}
ConcreteFlyweight
Operation(extrinsicState)
intrinsicState
UnsharedConcreteFlyweight
Operation(extrinsicState)
allState
Client

Solution

- See folder FlyweighjtBlueTree