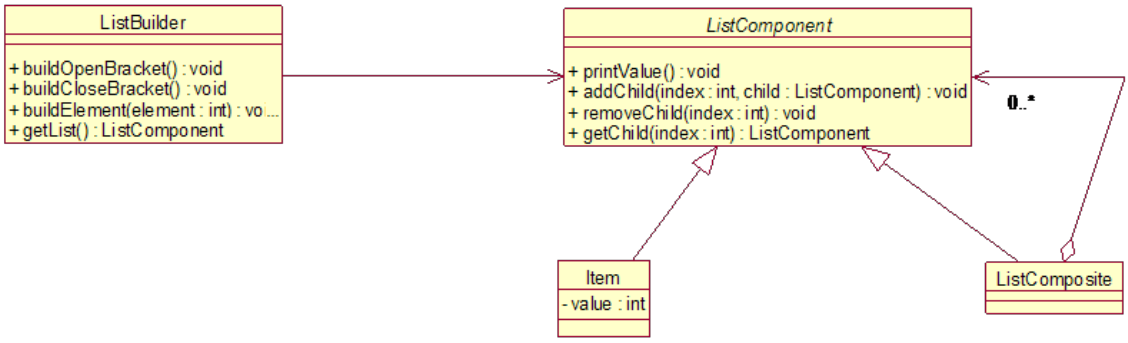
**Question#1**

Suppose that a list can contain items and other lists. Therefore, the composite pattern can perfectly used to implement such a list. For example, the list: [1 [2 3] [4 5] [6 4]] is composed of an item 1 and three lists: [2 3], [4 5] and [6 4].

In this program you will use Composite pattern and Builder pattern to implement a list structure. The following UML class diagram shows the program structure:




The program will take an input in the form: [1 [2 3] [4 5] [6 4]] and store it using a *ListComponent*. The program should create a *ListComposite* and add a child of type *Item* for 1. Then it will add another child of type *ListComposite* for [2 3], in which it will add two *Items* 2 and 3, and so on.

Item's printValue() method only prints out its value.

The following is a skeleton code for the main program:

```
ListBuilder builder = new ListBuilder();
read in input list into a string and tokenize it.
for each token in the input string
   if (token == "[") builder.buildOpenBracket();
   else if (token == "]" builder.buildCloseBracket();
   else if (token == number) builder.buildElement(number);
ListComponent list = builder.getList();
list.printValue();
```

**Question#2:**

**Proxy pattern:** write the following simple program to implement 3 protection proxies. The real subject is defined as following

```
class Text {
private String content;
void setContent(String newContent) {this.content = newContent;}
String getContent() {return content;}
}
```

Write proxies that can be set up at construction time to protect its real subject by allowing to read only, write only, or read/write (3 options) the content of the real subject.

Write a main program to test your proxy objects with different protection options.

**Question#3:**

Write a simple class called ArrayListSubject that maintains an arraylist of objects. It provides two operations:
void append(Object obj); //append the object to the end of the arraylist
void delete(Object obj); //remove the object from the arraylist

Use the observer pattern to define an observer that simply prints “an item being deleted” when the delete operation is called on ArrayListSubject.

For all these questions, test your code with some inputs. You have to submit the following to the blackboard:

1-Source code.
2-Screenshots of the output.