Partial Example

Which participant is this?

class ? : public Expression
{

    public:
    ?() {
        left = right = 0;
    }

    virtual ~?() {
        //Remember the problems with deleting?
        if(left)
            delete left;
        if(right)
            delete right;

        left = right = 0;
    }

    virtual void setLeft(Expression* ex) {
        if(left)
            delete left;
        left = ex;
    }

    virtual void setRight(Expression* ex) {
        if(right)
            delete right;
        right = ex;
    }

    virtual void print () {
        left->print();
        cout<<getSymbol();
        right->print();
    }

    protected:
        Expression* left;
        Expression* right;
};
Which participant is this?

class ? : public ?
{
    public:
    
    virtual ?()
    {
        //What happens here?
    }

    virtual int evaluate()
    {
        if(left && right)
        {
            return left->evaluate() + right->evaluate();
        }
        else
        {
            return 0;
        }
    }

    protected:
    virtual char getSymbol()
    {
        return '+';
    }
};

Remember that the fact that this example could do simple sums and print objects was not the goal of the pattern. The goal of the pattern is actually to allow the construction/creation objects composed of similar objects. The other thing to consider is who is responsible for managing what memory...