Name ____________

1. Please draw the class diagram for the software described below.

The user-interface of this software contains a text-box and a few other frames. Users are expected to fill the text-box with a date (e.g., 5/17/2016). Once the text-box content is changed, the content in other frames will also change. Specifically, one frame will show the user’s schedule on that day, if available; another frame will show the important events (sports or political or whatever theme defined by the user) that happened or are scheduled to happen on that day; a third window will show the weather or weather prediction for that day. Since more window-style gadgets may be added to the user-interface, you want to make your design easy to extend.

Please include important attributes and method functions in your class diagram.

![Class Diagram](image)

Which of the following design patterns is most suitable here? ____________

a. observer
b. composite
c. interpreter
2. Please refactor the code below using “composite” design pattern. Please put both the modified code and the new class diagram on the next page(s).

```cpp
Class Employee {
    Private:
        String name;
        Int Salary;
    Public:
        ...
        String getName() {return name;}
        int getAnnualBudget() { return Salary; }
}

Class Group {
    Private:
        String name;
        Vector<Employee> DirectStaffs;
        Vector<Group> SubGroups;
    Public:
        ...
        String getName() {return name;}
        int getAnnualBudget();
}

int Group::getAnnualBudget(){
    int total=0;
    for(int i=0; i< DirectStaffs.size(); i++)
        total+= DirectStaffs[i].getAnnualBudget();
    for(int i=0; i<SubGroups.size(); i++)
        total+= SubGroups[i].getAnnualBudget();
    return total;
}
```
New code (pseudo code):

Class Unit {
    Private:
        String name;
    Public:
        ...
        String getName() {return name;}
        int getAnnualBudget()=0; //virtual function
}

Class Employee : public Unit {
    Private:
        Int Salary;
    Public:
        ...
        int getAnnualBudget() { return Salary;}
}

Class Group : public Unit {
    Private:
        Vector<Item> members;
    Public:
        ...
        int getAnnualBudget();
}

int Group::getAnnualBudget(){
    int total=0;
    for(int i=0; i< members.size(); i++)
        total+= members[i].getAnnualBudget();
    return total;
}
New Class Diagram

Unit
- name: String;
String getName();
int getAnnualBudget();

Employee
- Salary: int;
int getAnnualBudget();

Group
int getAnnualBudget();

members*
3. Please provide a class diagram based on interpreter design pattern for the software described below.

This is part of a calculator software. The input to this component is an algebraic expression tree, like the one shown below. The supported operators include: \( \times \) (multiplication), \(+\) (addition), and \(-\) (negative). “\( \times \)” and “\(+\)” each takes two operands; “\(-\)” takes one operand. The leaves in the trees are constant integers. The output of this component is the evaluation results of the corresponding algebraic expression. For example, the output for the expression tree below should be 9.