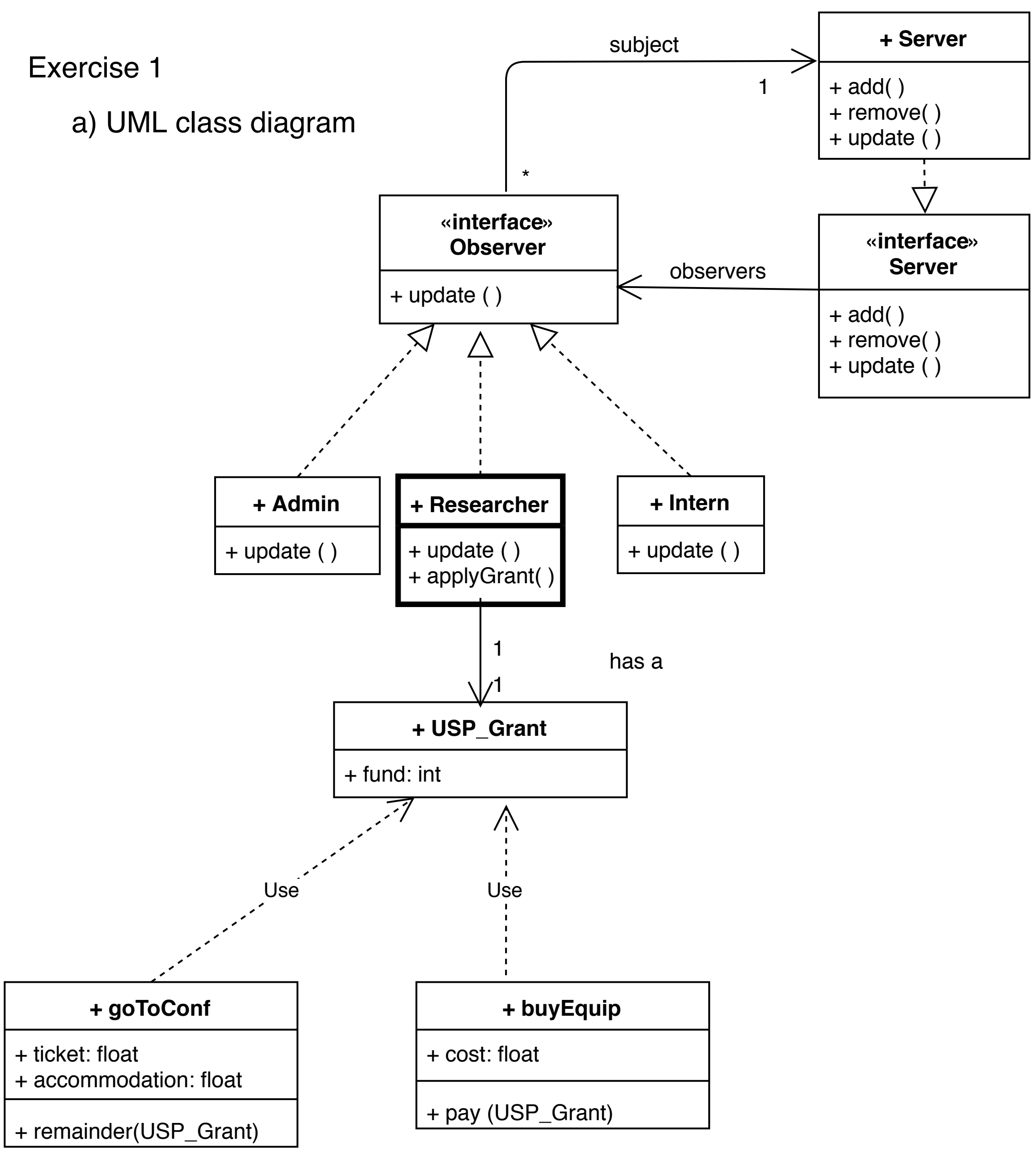
Maria Galliva, Norhan Abbas September 26, 2019

ESOF 322

Homework 3

Exercise 1
a) Draw a UML diagram that clearly shows the coupling. Please identify which class(es) participates in multiple patterns.

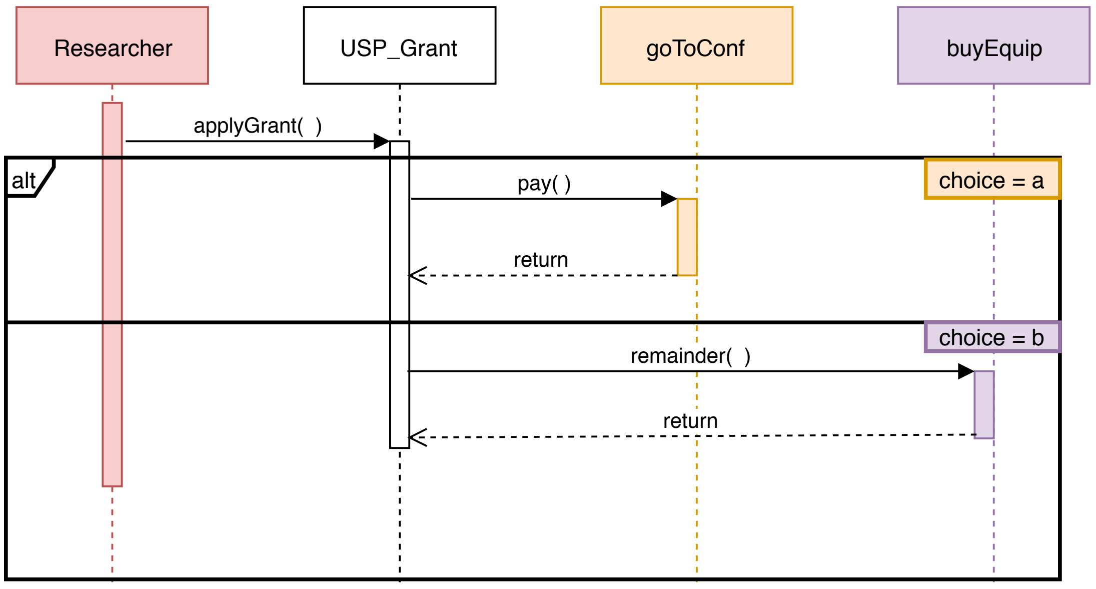
The class that participates in both design patterns, observer and strategy, is **Researcher** class



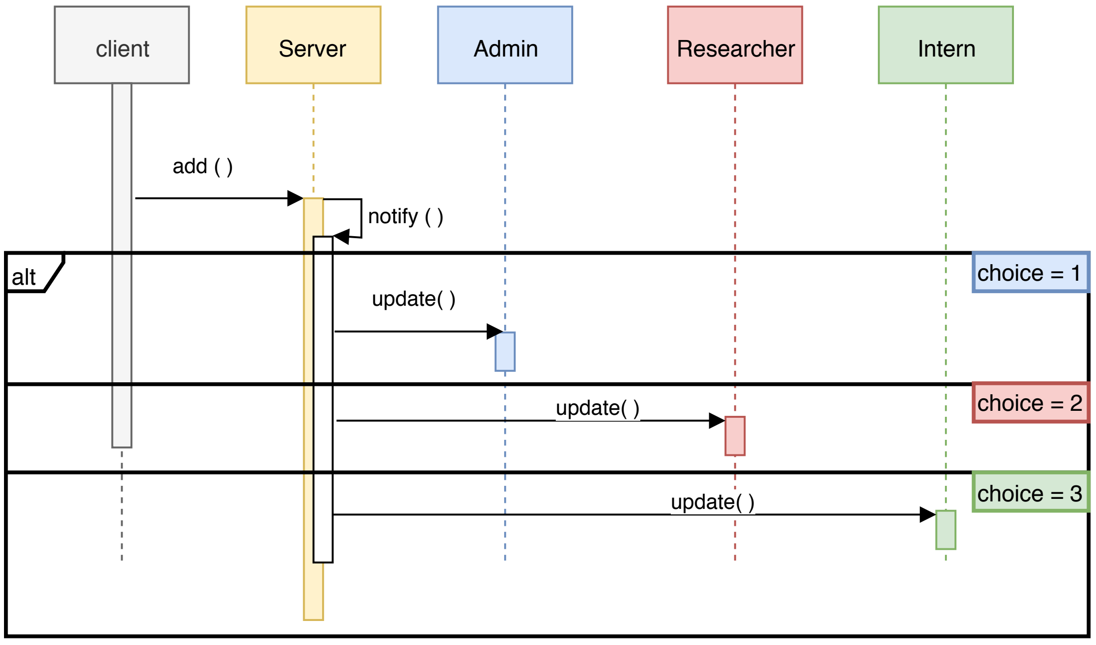
Even though the **Researcher** is a part of a company with Admin, Intern, Servers; however, the **Researcher** is still a part of MSU as well.

b) Draw a UML Sequence diagram where you demonstrate the behavior instance of the coupled class from the perspective of one pattern, then from the perspective of the other pattern.

From the other diagram's perspective



From the observe's diagram perspective



Exercise 2

a) Assume during your team's last sprint, that they completed 32 story points using a 3-person team working in sprints of 3 weeks for a total of 45-man days.

Calculate your team's estimated velocity for the next sprint if we still have 3-week sprints, but you now added 2 engineers to the team, and one of them can only work 80 % of the time.

---

completed story points = 32
no of team members = 3
no of man days = 45

the previous means
no of man-days/team member = $\frac{45}{3}$ = 15 man days/member
Focus factor of the 3-person team on the last project = $\frac{totalnoofstorypoints}{totalnoofavailablemandays} = \frac{32}{45} = 0.711 = 71.1\ \%$

for the new project
Estimated velocity of 4 members = $(\frac{noOfHours}{member} \times 4) \times$ their focus factor $= (15 \times 4) \times 0.71 = \boxed{42.6}$
Estimated velocity for one member, who can work up to 80% $= 15 \times 0.8 = \boxed{12}$

Total team's estimated velocity (5 members) $= 42.6 + 12 = \boxed{54.6}$

Another solution

average focus factor of the team $= \frac{(71\% \times 4)+(80\& \times 1)}{5} = 72.8\%$
total estimated velocity = available man days $\times$ average focus factor $= (15 \times 5) \times 72.8\% = \boxed{54.6}$

b) How would you estimate a focus factor for a brand-new team?

In quick calculations, it could be empirically considered 70%

In real-life situations, the team members could be asked to estimate how many story points they think their project needed. Having the estimated velocity, story points (poker cards), and knowing how many available days they had, the focus factor could be calculated.

Focus factor = $\frac{EstimatedVelocity}{AvailableMan-days}$

---

c) We looked at using poker using semi-Fibonnaci sequences to estimate story points. Think of another way to estimate story points and explain it. Is it better or worse than poker?

We can use poker cards but with incentives. If the team finished the project early, they'd be awarded an amount of Belgium chocolates equal to the difference units between when they thought they'd finish and when they actually finished, in pounds.

If they finished late, the team members totally should lose weight equal to the difference units between when they thought they'd finish and when they actually finished, in pounds.

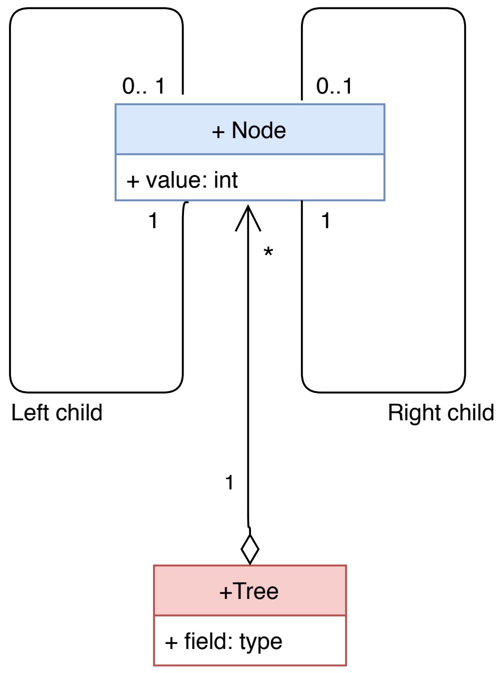
Having incentives could motivate the team to work harder.

Another solution

Also, we can use self-report method; ask the most experienced engineer on the team to estimate the story points of the assigned project.

Self-report could be tricky as if the engineer was super experienced at what s/he's doing, s/he'd estimate the story points from his/her own perspective depending on the skills s/he had. Yet, self-report wastes less time as the poker strategy could be really time consuming if there team is huge.

d) Draw a UML class diagram of a binary tree. Each node contains an integer.



We did choose aggregation for the Binary tree as if we deleted the Tree, we'd still have the nodes. No particular need to do this; yet, we could just keep the nodes around in case we'd decided to sort them in an ArrayList/List/array.

e) Provide the corresponding object-oriented code that implements your binary tree design from part d.

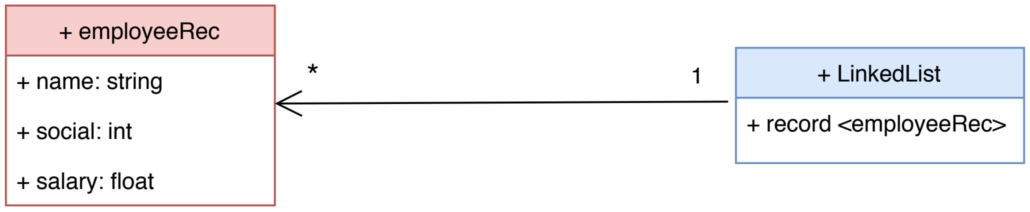
```
import java.util.ArrayList;

public class Node {
   int value;
   Node leftChild;
   Node rightChild;
   }

public class Tree( ){
   Node root;
   public Tree( ){
      Node root = new Node( );
   }
}
```

f) Draw a UML class diagram of a linked list that contains Employee records as data. An employee record has a name, a social security number, and a salary.

LinkedList class should have been named ”ourList” in the diagram, SORRY for that!



g) Provide the corresponding object-oriented code that implements your linked list from part f.

```
import java.util.LinkedList;

public class employeeRed {
   String name;
   int social;
   float salary;
   }

public class ourList {
   emploeeRec R;
   linkedList <emploeeRec> ( ) ourList = new linkedList< >( );
   ourList.add (R);
}
```