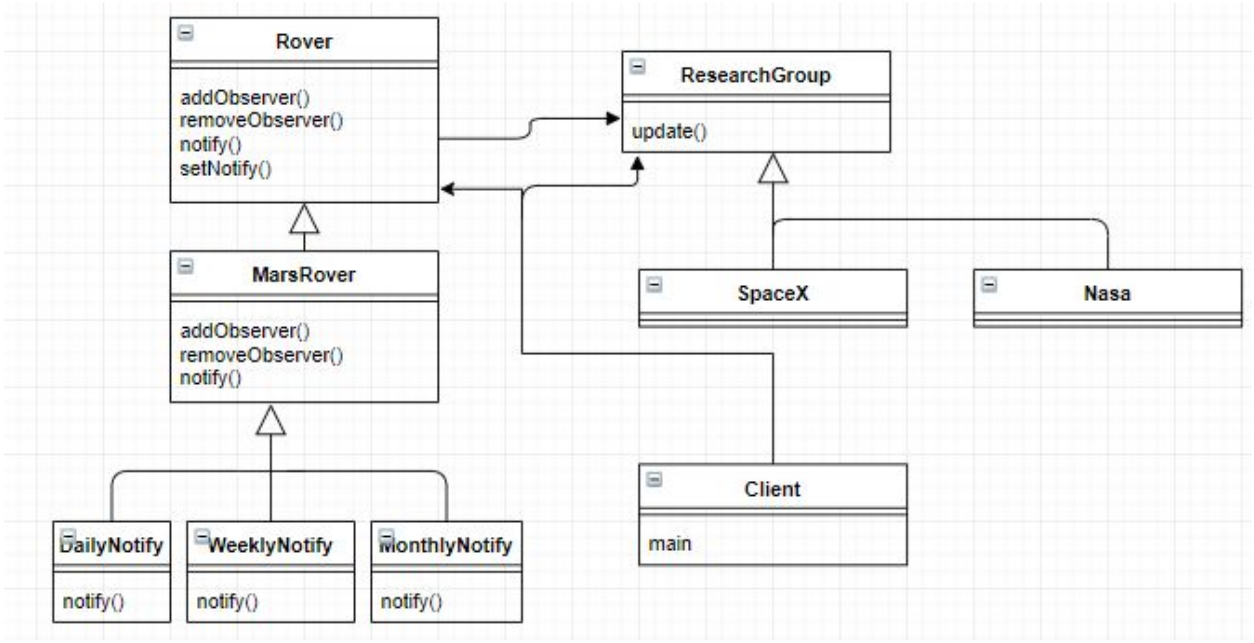
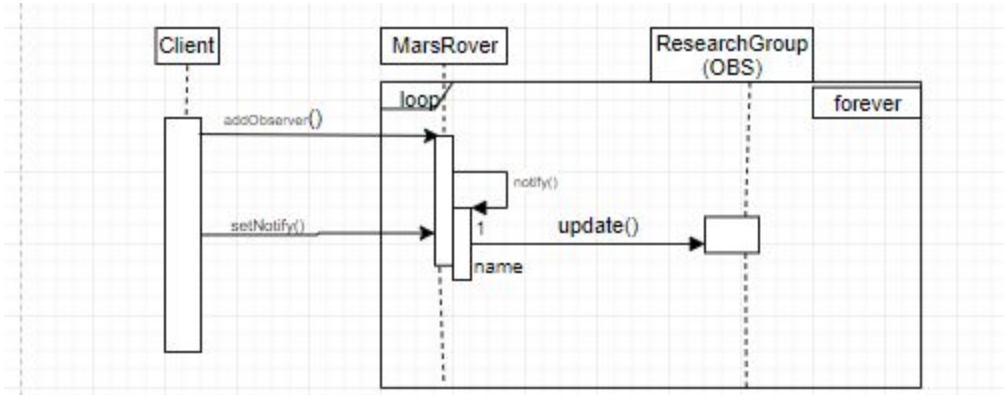
PART I

a)



b)



PART II

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 add two engineers to the team, and one of them can only work 80% of the time.

To find the focus factor of a 3-person team working in a 3 week sprint, you divide story points and total man days - 32/45 = .7111 or 71% productivity. To find a team's estimated velocity for a 5 person team if only one of the two new members can only work 80% of the time you need to find the total number of man days - 15*4= 60+(15*0.80) = 72-man days. To find the total number of story

points accomplished you take the total number of man days and take 71% of it - 72*.71 = 51 total story points will be completed.

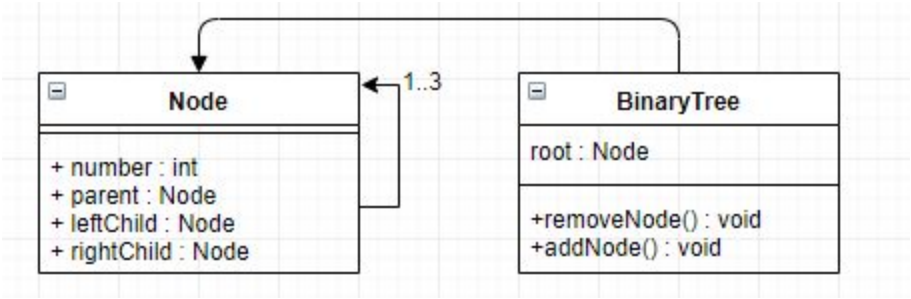
b) How would you estimate a focus factor for a brand-new team?

As mentioned in class it can be assumed that their will be as low a focus factor as 70% in any given company. It would also be possible to look at the performance in the last sprint.

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

Another way that you could estimate the semi-Fibonacci sequence is by using dominos by taking a domino and placing it on the table so that it is face down on the table and placing it into as many patterns as you can make with one domino, two dominos and so on you would wind up with a very similar sequence of numbers as the Fibonacci sequence meaning that this would be a good way of describing this phenomenon. I had the help of maths.surrey.ac.uk to solve this question.

d)



e)

```
public class Node {

  Int number;

  Node leftChild;

  Node rightChild;

  Node parent;


  Node(int val, Node parent) {

    this.number = number;

    rightChild = null;
```

```
        leftChild = null;  }
    //sets right node
    public void setR(Node r) {
        rightChild = r; }
    //sets left node
    public void setL(Node l) {
        leftChild = l; }
    //return right node
    public Node getR() {
        return rightChild; }
    //return left node
    public Node getL() {
        return leftChild; }
    public Node getParent()    {
        return parent; }
}
public class BinaryTree {
Node root;
public void insert(Node n){
            "insert algorithm here" }
}
```

f)