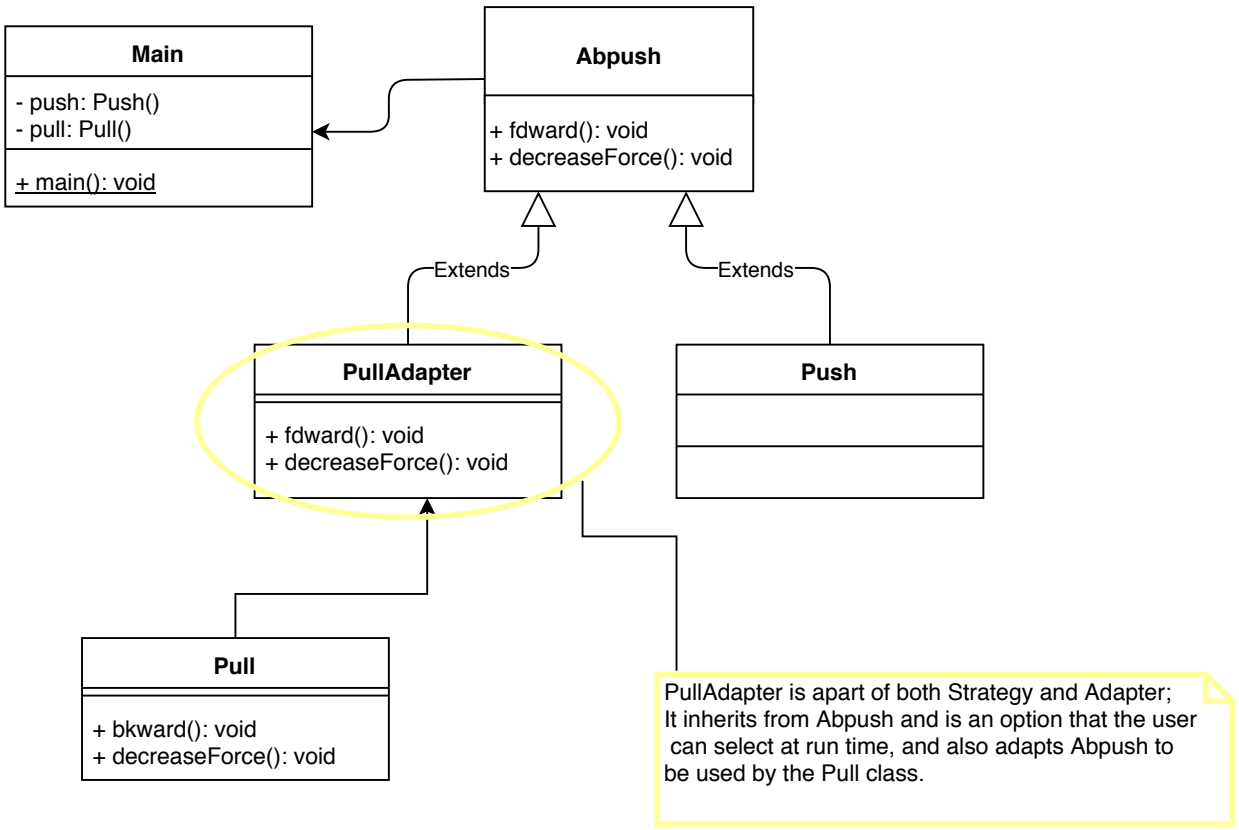
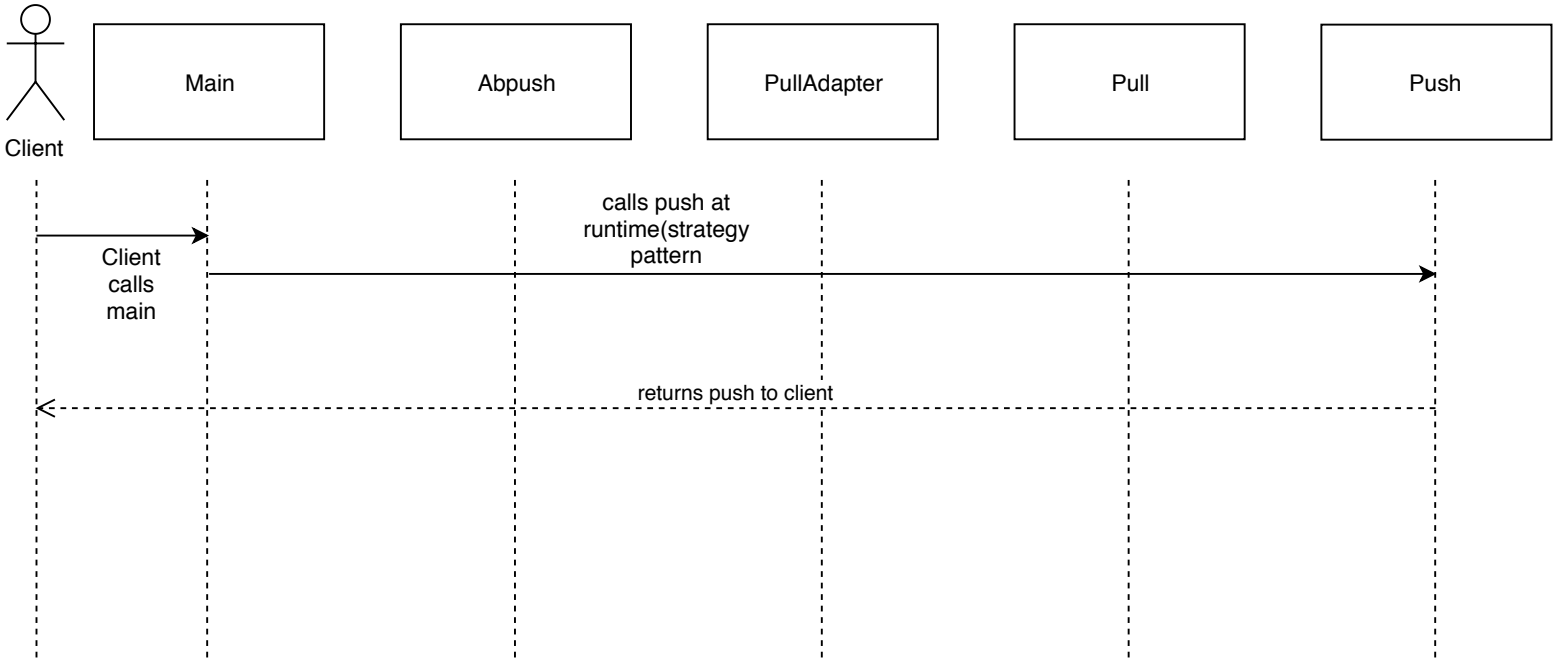
Ransom Bowman
Elijah Blake

Exercise 1a



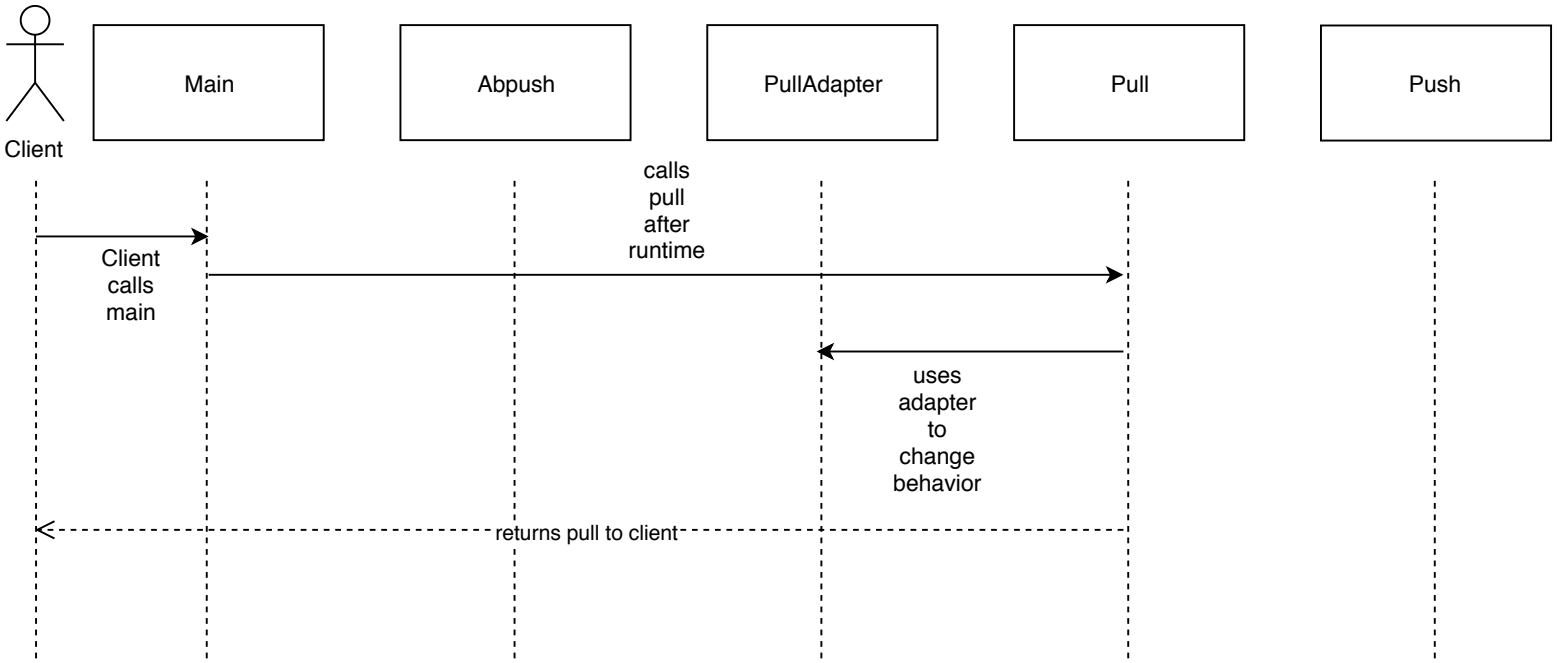
Exercise
1b

Strategy Pattern



Exercise
1b

Adapter Pattern



Exercise 2:

a.) There are 32 story points(Velocity). Since there were originally three people on the team, with two additions, there are five people on the new team. If each person were work 100% of the time, there would be 75 man days(15 days for each person), but since one of the people can only work 80% of the time, that person would only have 12 man days, summing a total of 72 man days.

Focus Factor = Actual Velocity / Available Man Days

Focus Factor = 32 Story Points / 72 Man Days = 0.71

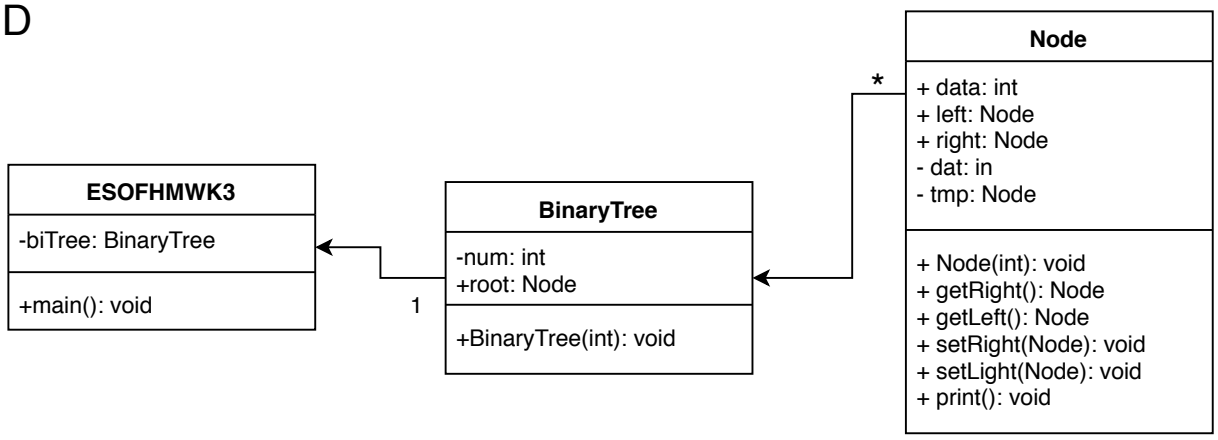
Estimated Velocity = Man Days * Focus Factor

Estimated Velocity = 72 Man Days * 0.71 = 51 Story Points

b.) For the brand-new team, the estimated focus factor would be 70%, since we don't know the true focus of all the members yet.

c.) Another way to estimate story points would be using the probability of flipping a coin, and make it proportional to the estimated story points. For example, A low number of story points would associate with one coin flip(2 for total possibilities). A really high number of story points might associate with 6 coin flips(64 for total possibilities). The same idea is with the Fibonacci sequence in that the higher the story points, there is more variety for error in how long a task will take in the workplace. I think my coin method is worse than the poker method because as mine exponentially grows, it can become really inaccurate because the numbers are so high. Poker uses addition and keeps everything realistic.

D



E

```
package esof.hmwk.pkg3;

/**
 *
 * @author ranso
 */
public class ESOFHMWK3 {

    public static void main(String[] args) {
        BinaryTree biTree = new BinaryTree(1); //creates Binary Tree and the root node
        biTree.root.setLeft(new Node(2)); //adds left node to root
        biTree.root.setRight(new Node(3)); //adds left node to root
        biTree.root.right.setRight(new Node(4)); //adds right node to root's right node

    }
}
```

----------Node

```
package esof.hmwk.pkg3;

/**
 *
 * @author ranso
 */
public class Node {
    int data;
    public Node left; //left node
    public Node right; //right node

    public Node(int dat){ data = dat;} //creates node
    public Node getRight(){ return right;} //return right child
    public Node getLeft(){ return left;} //return left child
    public void setRight(Node tmp){ right = tmp;} //sets right child to passed in node
    public void setLeft(Node tmp){ left = tmp;} //sets left child to passed in node
    public void print(){ System.out.println(data); //prints data for node being operated on

    }
}
```

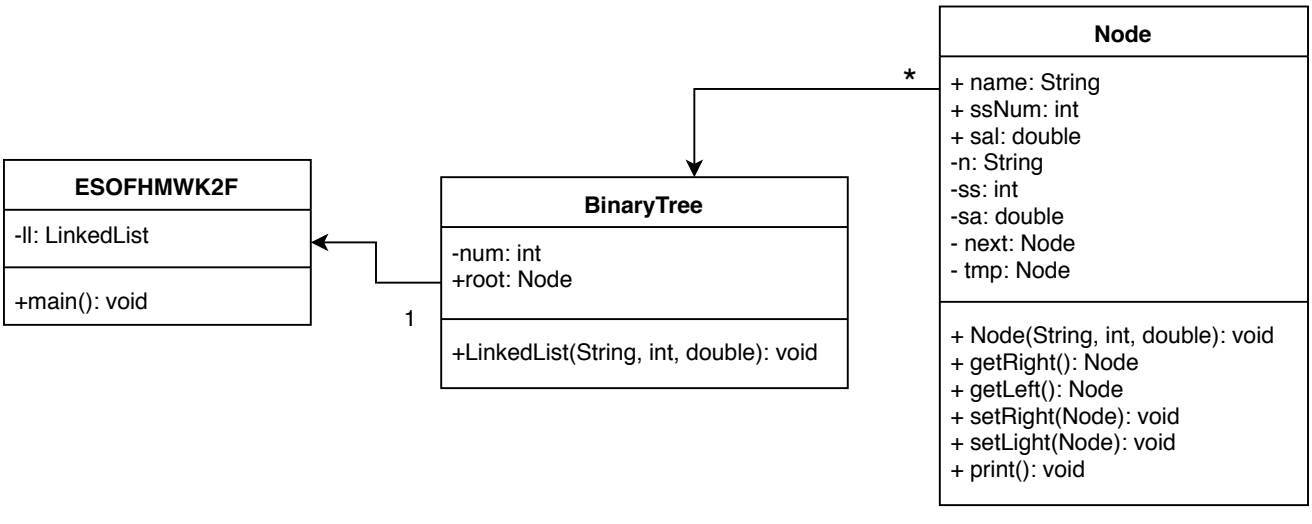
----------BinaryTree

```
package esof.hmwk.pkg3;

/**
 *
 * @author ranso
 */
public class BinaryTree { //Binary Tree
    public Node root; //first node in Binary Tree

    BinaryTree(int num){ //inititializes Binary Tree
        root = new Node(num); //creates new node in Binary Tree
    }
}
```

F



G

```
package esofhmwk2f;

/**
 *
 * @author ranso
 */
public class ESOFHMWK2F { //Typo. Supposed to be ESOFHMWK3F

    public static void main(String[] args) {
        LinkedList ll = new LinkedList("Ransom", 4838372, 15.00); //creates linked list

        ll.root.setNext(new Node("Elijah", 32456768, 55.00)); //creates second node in LL

        ll.root.next.setNext(new Node("Melbourne", 11111111, 35.00)); //creates third node in LL

        ll.root.print(); //prints first node
        ll.root.getNext().print(); //prints second node
        ll.root.next.getNext().print(); //prints third node
    }

}
```

----------Node

```
package esofhmwk2f;

/**
 *
 * @author ranso
 */
public class Node { //LinkedList Node
    String name;
    int ssNum;
    double sal;

    public Node next;

    public Node(String n, int ss, double sa){ //initializes node
        name = n;
        ssNum = ss;
        sal = sa;
    }

    public Node getNext(){ return next;} //returns the next node in the list
    public void setNext(Node tmp){ next = tmp;} //creates the next node in the list
    public void print(){ //prints name, ssNum, and sal for the node being operated on
        System.out.println(name + " l " + ssNum + " l " + sal);
    }
}
```

----------LinkedList

```
package esofhmwk2f;

/**
 *
 * @author ranso
 */
public class LinkedList { //LinkedList
    public Node root; //first node in LL

    LinkedList(String n, int ss, double sa){ //inititializes LL
        root = new Node(n, ss, sa); //creates new node in LL
    }
}
```